INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Development of a mass drug administration monitoring system for a Humanitarian NGO

RAFAEL ALMEIDA RIBEIRO
Setembro de 2025

POLITÉCNICO
DO PORTO

ISEP INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

Development of a mass drug administration monitoring system for a Humanitarian NGO

Rafael Almeida Ribeiro

A dissertation submitted in partial fulfillment of the requirements for the degree of Master of Science, Specialisation Area of Software Engineering

Advisor: Paulo Baltarejo Sousa
Supervisor: Cláudio Teixeira
Co-Supervisor: Pedro Vicente

Porto, September 20, 2025

Statement of Integrity

I hereby declare that I have conducted this academic work with integrity.

I have not plagiarised or applied any form of undue use of information or falsification of results along the process leading to its elaboration.

Therefore, the work presented in this document is original and authored by me, having not previously been used for any other end. The exceptions are explicitly recognised in the section "Ethical considerations" of the first chapter. This section also states how AI tools were used and for what purpose.

I further declare that I have fully acknowledged the Code of Ethical Conduct of P.PORTO.

ISEP, Porto, September 20, 2025

Rafael Almeida Ribeiro

Abstract

Mass drug administration is the process of distributing medication to people who are in areas with risk of neglected tropical diseases. The monitoring of this process is supported by the application of forms in the communities, which are then sent to the headquarters, where they are manually entered into governmental health information systems. This dissertation aims at finding a solution for automating the process, making it more efficient and less error-prone.

Optical Character Recognition (OCR) is a technology that converts different types of documents into searchable data. By reviewing the literature, it is possible to conclude that the application of OCR is feasible for accurate text extraction. The literature also suggests that AWS Textract is the tool with higher accuracy when extracting handwritten text.

A system composed of a Backend, an Android Application and a Backoffice web application was designed and implemented. This solution was evaluated with two experiments. Text extraction tests were performed using 8 example test forms filled with fake data. Different effectiveness metrics where calculated, resulting in mean values of 0.8%, 2.5% and 3.7% for the Character Error Rate (CER), Word Error Rate (WER) and Field Error Rate (FER), while a mean value of 96.2% was achieved for the Precision, Recall and F1 score. It was possible to conclude that forms filled in English and forms filled with print handwriting style had better accuracy than forms filled in Portuguese and forms filled with cursive handwriting style, respectively. The other experiment performed was user testing, in which two testers used official forms filled with fake data to test the application. The feedback was positive, with some improvements being suggested, such as the order of the fields in the form response details screen and the size of the numeric fields. A mean value of 7.9% incorrectly extracted fields was achieved during the user testing.

Keywords: Optical Character Recognition, Text extraction, Mass drug administration, Android, Backend

Resumo

A administração massiva de medicamentos é o processo de distribuição de medicamentos a pessoas que se encontram em áreas de risco para doenças tropicais negligenciadas. O monitoramento desse processo é apoiado pela aplicação de formulários nas comunidades, que são depois enviados para a sede, onde são inseridos manualmente nos sistemas de informação de saúde governamentais. Esta dissertação tem como objetivo encontrar uma solução para automatizar o processo, tornando-o mais eficiente e menos propenso a erros.

OCR é uma tecnologia que converte diferentes tipos de documentos em dados pesquisáveis. Ao rever a literatura, é possível concluir que a utilização de OCR é viável para uma extração de texto exata. A literatura também sugere que o AWS Textract é a ferramenta com maior exatidão na extração de texto manuscrito.

Foi desenhado e implementado um sistema composto por um Backend, uma aplicação Android e uma aplicação web Backoffice. Esta solução foi avaliada com dois ensaios. Foram realizados testes de extração de texto com a utilização de 8 formulários de teste preenchidos com dados falsos. Foram calculadas diferentes métricas de eficácia, resultando em valores médios de 0,8%, 2,5% e 3,7% para o CER, WER e FER, enquanto um valor médio de 96,2% foi alcançado para a Precisão, Recall e F1 Score. Foi possível concluir que os formulários preenchidos em inglês e os formulários preenchidos com estilo de caligrafia impressa tiveram melhores resultados do que os formulários preenchidos em português e os formulários preenchidos com estilo de caligrafia cursiva, respetivamente. O outro ensaio realizado foi o teste de usabilidade, no qual dois testadores utilizaram formulários oficiais preenchidos com dados falsos para testar a aplicação. O feedback foi positivo, com algumas melhorias sugeridas, tais como a ordem dos campos no ecrã de detalhes da resposta do formulário e o tamanho dos campos numéricos. Foi alcançado um valor médio de 7,9% de campos extraídos incorretamente durante o teste de usabilidade.

Acknowledgement

First of all, I would like to thank my parents and my sister for the unconditional support during all my academic, professional and personal life. To Margarida, thank you for all the patience and for being my backbone during this journey. To my close friends, thank you for all the leisure and breather moments that allowed me to relax a bit.

A special thanks to Professor Paulo Baltarejo Sousa for accepting being my advisor again, for all the feedback and for being attentive and rigorous in all the revisions. The inputs were crucial in shaping both the direction and quality of this dissertation.

I am grateful to Mindera for the opportunity to develop my dissertation with this project, and, especially, to Cláudio Teixeira and Pedro Vicente for all the availability and support in the project, as well as for all the help with both the external communication and the infrastructure. I am also grateful to Mindera School and Sara Cardoso for giving me the opportunity to mentor two wonderful students. Thank you, Ian and Raquel, for all your work and for being patient with me during your internships. I would like to extend my gratitude to the colleagues in my main project for bearing with me while I was working on the dissertation.

To The MENTOR Initiative, thank you for allowing me to be part of a project with such a philanthropic cause, and, especially, to Xavier for all the support with testing and for providing useful information and materials to aid the development of the solution.

Finally, I would like to thank all the colleagues and teachers that crossed my path during these 6 years at ISEP.

Contents

List of Figures

List of Tables

List of Source Code

List of Acronyms

AI	Artificial Intelligence.
ANN	Artificial Neural Networks.
API	Application Programming Interface.
APK	Android Package Kit.
AWS	Amazon Web Services.
CER	Character Error Rate.
CI/CD	Continuous Integration/Continuous Delivery.
CSR	Client-side Rendering.
DTO	Data Transfer Object.
FER	Field Error Rate.
FURPS	Functionality, Usability, Reliability, Performance, Supportability.
HTTP/S	Hypertext Transfer Protocol Secure.
I/E	inclusion and exclusion.
ISEP	Instituto Superior de Engenharia do Porto.
JSON	JavaScript Object Notation.
JWT	JSON Web Token.
LRRQ	Literature Review Research Questions.
ML	Machine Learning.
MVC	Model-View-Controller.
MVVM	Model-View-ViewModel.
NGO	Nongovernmental Organization.
OCR	Optical Character Recognition.
ORM	Object-Relational Mapping.
PICOCS	Population, Intervention, Comparison, Outcomes, Context, Study design.
PRISMA	Preferred Reporting Items for Systematic reviews and Meta-Analyses.
RBAC	Role-Based Access Control.

REST	Representational State Transfer.
SDK	Software Development Kit.
SPA	Single-page Application.
TCP/IP	Transmission Control Protocol/Internet Protocol.
UI	User Interface.
URL	Uniform Resource Locator.
WER	Word Error Rate.

Chapter 1

Introduction

This dissertation is part of the Master's degree in Informatics Engineering, specialization area of Software Engineering, at Instituto Superior de Engenharia do Porto (ISEP). It focuses on a project developed under the supervision of Mindera, in a partnership with the Nongovernmental Organization (NGO), The MENTOR Initiative.

In this chapter, the problem and its context are described, as well as the objectives to be reached with this dissertation. In addition, the ethical considerations will be presented along with the research methodology, the project planning and the document structure.

1.1 Context

The MENTOR Initiative is a Humanitarian NGO that works with the most vulnerable and hard to reach communities to reduce death and suffering from tropical diseases [1]. They cooperate closely with local communities, health workers, health authorities and other international organizations in insecure and high risk environments, such as conflict zones and other humanitarian emergencies [2]. Their work has had a significant impact, reaching millions of people and saving countless lives [3].

They act in multiple countries around the globe, with a higher focus on African countries, such as Angola, Mozambique or Nigeria. Their efforts include preventive and curative health work, as well as surveillance and long-term sustainable control of diseases.

Mass drug administration is one of the critical jobs that The MENTOR Initiative does to prevent the risk of neglected tropical diseases [4]. Monitoring the drug administration is an important part of the work, as it helps evaluate the coverage and identify communities in need.

The monitoring is supported by data gathering, through the application of forms about the amount of administered drug, side effects and supervision of the drug administration. The data is further analysed in order to effectively target vulnerable and hard to reach communities [4].

This process influences decision-making at The MENTOR Initiative and its partners, allowing them to adapt their strategies and allocate resources effectively.

1.2 Problem

The mass drug administration process is based in the communication between the people appointed to distribute the medicine in the villages and the country headquarters. For each

village, a paper form, that can have several pages, is filled with the amount of administered drug and health information. Collecting this data on paper makes the data compilation difficult, inefficient and error-prone.

The collected data is sent to the country headquarters, where a person is responsible for manually entering the data into a computer and submitting it to governmental systems. Besides the inefficiency, this process is both time-consuming and error-prone as well.

From the author's perspective, this is a perfect example of a process that could be improved, considering the current state of technology. The proposed solution of using Optical Character Recognition (OCR) is a good step towards that improvement, and it could be improved even further if the whole process was moved to the digital and the forms were filled digitally. However, the author understands that there can be some restrictions, such as poor network and device conditions [5] or bureaucratic constraints.

Solving this problem can impact positively the stakeholders of the project. The MENTOR Initiative and its partners will be able to improve the decision-making by accurately targeting the communities in need. Mindera will be able to offer this solution to other NGOs. The African communities, that suffer the most with tropical diseases, will get a more timely and accurate response for the neglected tropical diseases risk.

On the other hand, the author has concerns related with the infrastructure costs, since OCR services can get expensive in a large scale [6]. Considering that usually the NGOs run low on budget and depend on funding [7], this can be a limitation for the project.

The integration between multiple components, including mobile, and the possibility of impacting countless lives were the main motivators for the author to accept the project.

1.3 Objectives

The main goal of the project is the development of a tool that allows the digitization of drug administration forms, as well as the automation of data submission for different countries and, possibly, NGOs.

The specific objectives were defined as follows:

1. Identify the most suitable OCR tool for digitization of forms;
2. Develop an OCR Engine backend that processes the images and stores the data, so that it can be validated and submitted;
3. Develop a backoffice that connects to the backend and allows the form structure preparation, as well as user management;
4. Develop the frontend as an Android application to take pictures of the forms and review the data before submission;
5. Apply multi-tenancy to the project, in order to support multiple NGOs and countries;
6. Evaluate the effectiveness of the digitization process with real-world test trial.

1.4 Ethical Considerations

Addressing ethical considerations is crucial to ensure the integrity and social impact of the project. As a student of the Polytechnic of Porto, the Code of Good Practices and Conduct of Polytechnic of Porto [8] was completely followed, specially the 8th article by presenting the Declaration of Integrity at the beginning of the document.

Considering that this is an international software engineering project, the ACM/IEEE-CS Software Engineering Code of Ethics and Professional Practice [9] was also fulfilled, with special regard to the 2nd and 3rd principles, that focus on the relationship with the client/employer and the quality of the product, respectively.

Complying with data protection regulations is an important matter to be taken into consideration in a project like this. Since Mindera is a European-based company, the European Union General Data Protection Regulation [10] was considered. As The MENTOR Initiative works with African communities, their data protection regulations were followed, namely the Angolan Personal Data Protection Law [11].

Since the project involves a NGO, the Code of Ethics and Conduct for NGOs [12] was taken into account, even though it is more focused on the NGOs side.

Part of the dissertation was reused from the Dissertation Preparation course, namely the second and third chapters. No Artificial Intelligence (AI) tools were used for core activities of the dissertation, being used only for checking the grammar in the text.

1.5 Methodology

The research methodology will be described according to Oates, Griffiths and McLean's research model [13]. The diagram of the Figure 1.1 summarizes the research process that was followed in this dissertation.

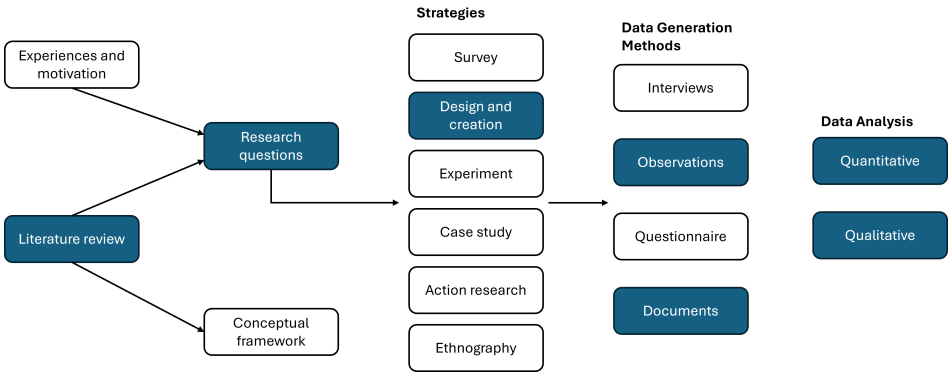

Figure 1.1: Research Process of the dissertation

As shown in the diagram, a **Literature review** was conducted to explore the existing knowledge and analyse previous research on the topic. The **Research questions** were defined to guide the selection of the research strategy, as well as the data generation and analysis methods.

The research area and research topic of the dissertation need to be specified in order to define the research questions. Taking that into consideration, the **research area** is "Text extraction from images", while the **research topic** is "Using OCR for handwritten characters' recognition".

The main **research question** that is answered with the dissertation is "How to create a tool that allows the digitization of handwritten forms in order to facilitate and improve the monitoring of the mass drug administration process?". This question can be divided in several research sub-questions that guide the dissertation:

- RQ1: How can OCR be used to extract text from images?
- RQ2: Is a solution that uses OCR to extract handwritten text feasible?
- RQ3: What is the OCR tool with better accuracy for handwritten characters?
- RQ4: What image enhancement techniques can be applied to improve OCR accuracy?
- RQ5: How to design a solution that supports the digitization of handwritten forms?
- RQ6: How can the designed solution be implemented?
- RQ7: How can the accuracy and quality of the developed tool be evaluated?

This project focuses on the development and evaluation of a digitization system that make the monitoring process easier. Therefore, the research strategy that was followed is **Design and Creation**. This research methodology involves multiple steps, such as defining how the system should be (Design), implementing it (Build), testing the solution (Evaluate) and interpret the results (Conclusions).

Regarding the data collection and generation, **Observation** methods were used to collect the results of the system application in real-world tests. In addition, **Documentation** provided by The MENTOR Initiative was another relevant source of data. It can include reports, form examples and information about the devices that are used in the field.

Based on the collected data, **Quantitative** data analysis methods were used to measure the accuracy and effectiveness of the developed tool. On the other hand, **Qualitative** methods were used to analyse the correctness and viability of the system.

As for the development, a Kanban methodology [14] was followed. Its continuous improvement and focus on flow were the main reasons for choosing this methodology, considering that the solution was developed only by one person outside working hours.

1.6 Project Planning

This section includes the work plan that was formulated for the dissertation. It is an important step, as it helps stay organized and manage the time effectively, by defining a clear roadmap with the tasks that need to be performed to achieve the deliverables. It also ensures that all the necessary aspects of the research and development are taken into account, in order to set realistic goals.

Figure 1.2 illustrates the plan that was obtained after defining the time estimation, order and resources of the tasks.

The project is planned to start at 13/01/2025 and considering the estimations, it should take approximately 167 days, ending at 02/09/2025.

The planning phase is estimated for 12 days and includes the skills management actions, as well as revisiting the work plan and state of the art that is being developed in this document.

WBS	Task Name	Duration	Start	Finish	Prede	Resource Names	% Complete
1	**Dissertation Project**	**167 days**	**Mon 13/01/25**	**Tue 02/09/25**			**92%**
1.1	**Planning**	**12 days**	**Mon 13/01/25**	**Tue 28/01/25**			**100%**
1.1.1	**Skills Management**	**7 days**	**Mon 13/01/25**	**Tue 21/01/25**			**100%**
1.1.1.1	Time Management Course	2 days	Mon 13/01/25	Tue 14/01/25		**Rafael Ribeiro**	100%
1.1.1.2	Define Deliverables and Work Plan	3 days	Fri 17/01/25	Tue 21/01/25	6	**Rafael Ribeiro**	100%
1.1.1.3	Oral Communication Course	2 days	Wed 15/01/25	Thu 16/01/25	4	**Rafael Ribeiro**	100%
1.1.2	Work Plan	3 days	Fri 17/01/25	Tue 21/01/25	6	**Rafael Ribeiro**	100%
1.1.3	State of the Art	5 days	Wed 22/01/25	Tue 28/01/25	7	**Rafael Ribeiro**	100%
1.2	**Design**	**8 days**	**Wed 29/01/25**	**Fri 07/02/25**	**2**		**75%**
1.2.1	Analysis	3 days	Wed 29/01/25	Fri 31/01/25		**Rafael Ribeiro;**	100%
1.2.2	Architecture	3 days	Mon 03/02/25	Wed 05/02/25	10	**Rafael Ribeiro;**	100%
1.2.3	Mockups	2 days	Thu 06/02/25	Fri 07/02/25	11	**Rafael Ribeiro;**	0%
1.3	**Build**	**77 days**	**Mon 10/02/25**	**Tue 27/05/25**	**9**		**86%**
1.3.1	Report v0.1	30 days	Mon 10/02/25	Fri 21/03/25		Rafael Ribeiro;	100%
1.3.2	**Backend**	**35 days**	**Mon 10/02/25**	**Fri 28/03/25**			**57%**
1.3.3	**Backoffice**	**20 days**	**Mon 31/03/25**	**Fri 25/04/25**	**15**		**100%**
1.3.4	**Mobile**	**22 days**	**Mon 28/04/25**	**Tue 27/05/25**	**18**		**100%**
1.4	**Evaluate**	**25 days**	**Wed 28/05/25**	**Tue 01/07/25**	**13**		**100%**
1.4.1	Report v0.2	20 days	Wed 28/05/25	Tue 24/06/25		**Rafael Ribeiro;**	100%
1.4.2	Test trial	25 days	Wed 28/05/25	Tue 01/07/25		Rafael Ribeiro;	100%
1.5	**Conclusions**	**45 days**	**Wed 02/07/25**	**Tue 02/09/25**	**24**		**100%**
1.5.1	Final Report	35 days	Wed 02/07/25	Tue 19/08/25		**Rafael Ribeiro;**	100%
1.5.2	Final Presentation	10 days	Wed 20/08/25	Tue 02/09/25	28	**Rafael Ribeiro;**	100%
1.6	**Milestones**	**160 days**	**Tue 21/01/25**	**Tue 02/09/25**			**0%**
1.6.1	Plan delivered	0 days	Tue 21/01/25	Tue 21/01/25	7		0%
1.6.2	Architecture designed	0 days	Wed 05/02/25	Wed 05/02/25	11		0%
1.6.3	Report 0.1 delivered	0 days	Fri 21/03/25	Fri 21/03/25	14		0%
1.6.4	Solution implemented	0 days	Tue 27/05/25	Tue 27/05/25	13		0%
1.6.5	Report 0.2 delivered	0 days	Tue 24/06/25	Tue 24/06/25	25		0%
1.6.6	Test trial performed	0 days	Tue 01/07/25	Tue 01/07/25	26		0%
1.6.7	Final report delivered	0 days	Tue 19/08/25	Tue 19/08/25	28		0%
1.6.8	Presentation read	0 days	Tue 02/09/25	Tue 02/09/25	29		0%

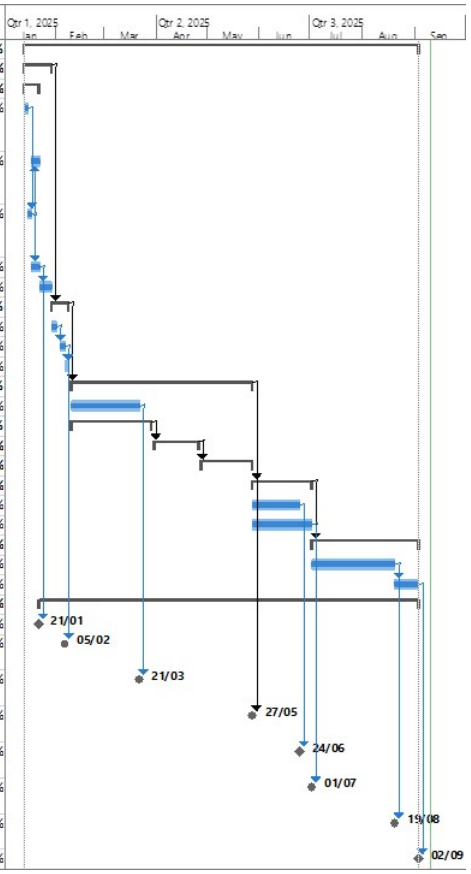

Figure 1.2: Dissertation project plan

The second phase includes the analysis of the problem, as well as the design of the solution and the mockups, taking up to 8 days.

At the beginning of the third phase, an initial version of the report is planned. Besides that, it includes the development of the Backend, Backoffice and Mobile app, estimated for 35, 20 and 22 days, respectively.

The Evaluate phase includes writing the second version of the report and performing a test trial, estimated for 25 days.

In the last phase of the project, the focus is the final version of the report and the presentation, until the end of the project.

The milestones of the dissertation project were also defined in the plan. Firstly, the revisited plan is delivered, followed by the design of the architecture and delivery of the initial version of the report. After two months, the solution is implemented and the second version of the report is delivered. After performing the test trial, the project ends with the delivery of the final report and presentation.

The progress of the project can be tracked through the "% Complete" column in the planning document.

1.7 Document Structure

This document is divided into seven chapters: Introduction, Background on OCR, State of the Art, Analysis & Design, Implementation, Experimentation & Evaluation and Conclusions.

The current chapter introduces the project by detailing the context, problem, objectives, ethical considerations, methodology and project planning.

The second chapter, Background on OCR, gives context to the reader about OCR and technologies, as well as optimization techniques.

The third chapter explores the existing knowledge about the best OCR services and their viability, as well as image enhancement techniques.

In the fourth chapter, the technical specification is detailed, including the analysis of the problem and design of the solution.

The fifth chapter describes the implementation of the solution, as well as the deployment and testing.

The sixth chapter outlines the quantitative and qualitative outcomes derived from the experimentation phase.

In the last chapter, the document is summarized. The achieved objectives are also assessed, as well as the challenges and future work.

Chapter 2

Background on OCR

This chapter will provide context about what OCR is, how it can be used and what its limitations are. This background information will serve as an answer to RQ1.

It includes sections that describe the historical development and fundamental concepts, as well as its applications, challenges and limitations. The chapter concludes with a summary, in which the answer to RQ1 is outlined.

2.1 Historical Development

The OCR concept started appearing during the 20th century. The Figure 2.1 summarizes some historical developments that will be described in this section.

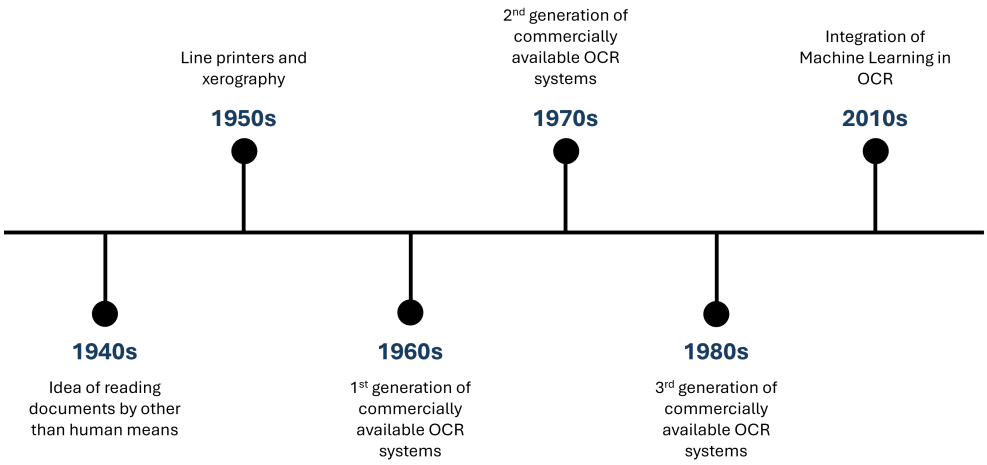

Figure 2.1: OCR timeline

The early developments of OCR started during the World War II, in which the idea of "reading documents by other than human means" appeared [15]. This idea was motivated by the necessity of automating the transcription of documents for faster processing compared to manual methods, such as perforating punched cards [15].

After some years, the concept evolved with the development of innovations like line printers and xerography, that addressed the speed discrepancies between document transcription and computer processing [15]. During the early ages of OCR, it was also used to support visually impaired individuals [16], as well as reading musical notations and small words, even though it had a lot of limitations [17].

The commercially available OCR systems originated in early 1960s, and can be divided in several generations [16]. The first generation only supported selected fonts and shapes of characters [16], [18]. Template matching methods were the most used in this first iteration [16], [19].

The second generation of OCR systems appeared during the mid 1960s and early 1970s. They demonstrated a higher capability, by recognizing both printed and handwritten characters, even though it was restricted to numeral characters [16], [20].

During the late 1970s and 1980s, the OCR technology evolved into supporting handwritten characters with a larger set, as well as printed characters with poor quality [16], [18].

In the latest generations, Machine Learning (ML) techniques, such as Artificial Neural Networks (ANN), were introduced and integrated into the OCR systems, leading to a wider supportability and higher accuracy [17], [20]. Because of that, currently, it is possible to extract text from complex documents that can include tables, mathematical symbols, unrestricted handwritten characters, noise and multiple languages [16].

2.2 Fundamental Concepts

OCR is a technology that extracts text from different types of documents, such as photos or scanned documents. The extracted data can then be indexed, stored and analysed. The process involves multiple phases that can vary depending on the implementation.

When dealing with physical documents or images, the first step of the process is converting to digital format [21]. This can be achieved by either scanning the document or taking a photo [17].

The Figure 2.2 summarizes the typical structure of an OCR-based system. The various phases will be detailed in this section.

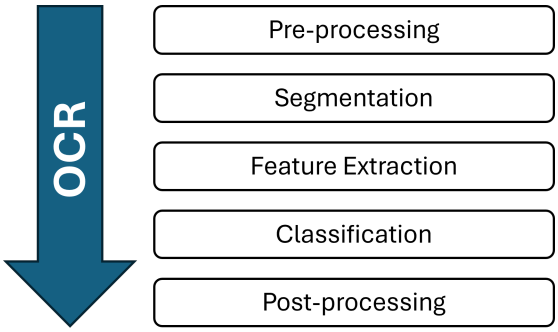

Figure 2.2: OCR phases

Most of the modern OCR systems include a **pre-processing** phase, in which the quality of the image is improved, in order to increase the possibility of better accuracy [22]. Different techniques can be used, such as noise removal, skew correction, filling and thinning [17], [21].

The **Segmentation** phase involves dividing the text into characters or words, which is crucial for an accurate recognition [21]. Segmentation can be categorized into two distinct types: external segmentation and internal segmentation. While the external segmentation focuses on isolating the writing units, such as paragraphs, sentences or words, the internal segmentation focuses on isolating the characters [22], [23]. Some examples of segmentation results are represented in the Figure 2.3.

Figure 2.3: Segmentation samples. Adapted from [23]

It is possible to notice the application of both external and internal segmentation in the boxes and lines, respectively.

There are some challenges that can emerge during the segmentation phase, such as separating cursive text or fragmented characters, distinguishing noise from text, and misinterpreting graphics as text [17].

In the **Feature Extraction** stage, relevant features of the text are identified to help in the recognition. A set of rules are compared against the shape of the characters, distinguishing their unique characteristics [23]. The feature set must have sufficient discriminating power to allow an accurate classification even among similar characters, and be efficient in computation time [22]. As a result of this stage, a feature vector is formed to be used in the next phase [17].

With the features extracted, the text segments can be identified during the **Classification** phase [23]. Different methods have been used to classify and extract text in OCR technologies.

One of the most used methods in modern OCR systems is ML, by using ANN [18], [20]. In ANN, the basic units are the nodes, that are also called neurons. These nodes are modelled by inputting data and then mapping it to predefined labels or classes [20]. With continuous stimulation and provision of inputs, the weights of each node are adjusted, leading to more accurate results [18].

Applying Template Matching techniques is another extraction method, and has been used for a long time [18]. It follows a sliding window approach, in which small portions of the image are matched against predefined templates [20]. Initially, this method was mainly used to recognize standard font characters, as it was easier to match against templates [18]. But there has been some research about supporting handwritten characters, through the usage of deformable template matching [18], [20].

The Structural Pattern Recognition method consists of analysing the pattern structures of the text and their relationships, to classify its characters [20]. This analysis includes extracting pattern primitives, such as edges, contours and connections [18]. It requires well-defined boundaries, which means that this method is difficult to use and is inaccurate for handwritten characters, as it has irregular edges [18], [20]. Structural Pattern Recognition was popular in early developments, but due to its limitations and popularization of other methods with higher accuracy, its usage and research has been declining [18], [20].

Some OCR systems also include a **post-processing** phase, that involves correcting errors in the recognized text, in order to improve the accuracy of the result [21]. This stage is specially useful for documents that have a lot of noise, such as historical or degraded documents [24]. There are several approaches for OCR post-processing, that can be divided into manual and automatic approaches [24]. While the manual approaches rely on people reviewing and fixing the resulting text line by line [24], the automatic (or semi-automatic) approaches can involve lexical validations or other contextual analysis, such as character frequencies, geometrical information, part of speech or other context information [17], [23], [24]. The automatic approaches usually involve the usage of statistical language models, as well as ML techniques, through neural network-based language models [24].

2.3 Applications of OCR

OCR has a wide range of applications across various fields. In this section, some of the key applications of OCR will be described.

Document Digitization and Archiving is one of the main applications of OCR. Digitizing printed and handwritten documents makes them searchable and editable, which allows the organization of vast archives with ease [25]. Converting physical documents supports an efficient indexing and access to valuable information [25].

The previous application can be divided into multiple uses. In **Banking**, it can be used to process check payments with minimal human interaction [26], reducing the time of the transaction [27]. In **Legal Industry**, digitizing legal papers facilitates searching for information of the legal processes [25]–[27]. Processing **historical records** allows the creation of large-scale archives that support the cataloguing and retrieval of valuable information [25]. In **Healthcare**, OCR can be used to extract information from medical forms and patient records [27]. Furthermore, by digitizing **invoices and receipts**, the businesses keep track of financial records and avoid accumulating overdue payments [26].

OCR is also useful for **Vehicle Registration Plate Recognition** [28], [29]. It can be used to control the vehicles passing through non-stop tolls, traffic management and law enforcement [26].

By combining multilingual OCR with **Language Translation** systems, it is possible to translate entire documents or images, reducing the language barriers and improving the cross-cultural communication [25].

Visually impaired individuals often rely on screen readers to be able to access digital content. OCR can be used to improve the **Accessibility** for these users, by allowing them access to printed and handwritten documents, or even text in images [25], [30].

2.4 Challenges and Limitations

Despite their significant advancements, OCR systems have some limitations and there are some challenges to their implementation.

One of the main challenges is supporting multiple languages and scripts. Some languages, such as Arabic and Indic, have complex scripts that are harder to recognize [27], [31]. This means that the accuracy of the OCR technology varies according to the language, due to their diverse character sets and unique challenges [18].

Recognizing handwritten text is another challenge for OCR systems, due to variations in individual handwriting styles [18], [20]. The development of machine learning and deep learning has enabled the improvement of handwritten recognition, and respective increase in accuracy [20].

Similarly, OCR technologies can also struggle with documents that use multiple fonts or forms that have complex layouts [30]. Deep learning is helpful in addressing this challenge and maintain higher accuracy levels [30].

Another technical challenge for OCR is recognizing text when the images have poor quality or resolution [32]. The environment conditions, such as distortion, noise, lighting and rotation, can also affect the performance of OCR systems [30], [32]. The introduction of post-processing techniques alleviated this problem, even though it still struggles when the original image is too deteriorated [24].

One of the main limitations of OCR is the potential resource-intensive consumption for larger datasets [30]. Cloud services can be an effective solution, that, in some cases, can even reduce the operational costs [33].

2.5 Summary

As an answer to RQ1, OCR is a technology that extracts text from different types of documents, such as printed and handwritten documents, or photos taken with a camera. The origins of OCR date back to the World War II, driven by the need to automate the transcription of documents. It can be composed of several phases, such as Pre-processing, Segmentation, Feature Extraction, Classification and Post-processing.

This technology is widely used in multiple areas, such as document digitization, archiving, vehicle plate recognition, language translation and accessibility. However, there are some challenges to its implementation, like supporting multiple languages, recognizing handwritten text and dealing with low quality input images.

Chapter 3

State of the Art

In this chapter, the literature will be reviewed in order to understand the existing knowledge about OCR viability, services, and enhancement techniques. These findings will enable RQ2, RQ3 and RQ4 to be answered, respectively.

A Systematic Mapping Review methodology [34] will be followed for the literature review, by performing a mapping study. It involves defining and framing the research questions, conducting the search for primary studies, and analysing them in order to discuss the findings.

The Literature Review Research Questions (LRRQ) will be mapped from the mentioned research questions of the dissertation, according to Table 3.1.

Table 3.1: Literature review research questions mapping

Dissertation RQ	Literature Review RQ	Research Question Specification
RQ2	LRRQ1	Are there case studies about solutions that use OCR to extract handwritten text with accuracy?
RQ3	LRRQ2	What studies are there about OCR tools with the best accuracy for handwritten characters?
RQ4	LRRQ3	What studies are there about image enhancement techniques that can improve OCR accuracy?

Each research question will be framed in accordance with the Population, Intervention, Comparison, Outcomes, Context, Study design (PICOCS) model [34] so that the search can be performed. The Preferred Reporting Items for Systematic reviews and Meta-Analyses (PRISMA) flow [35] will be used to describe the flow of information during the systematic review.

The search will be conducted in three data sources: ACM Digital Library, ScienceDirect and IEEE Xplore. Relevant papers from other sources will also be taken into account.

In the first three sections, the literature will be consulted in order to answer each LRRQ. In the last section, the findings of the systematic review will be summarized.

3.1 LRRQ1 - Feasibility of OCR for handwritten text extraction

In this section, the literature will be reviewed to answer the research question "Are there case studies about solutions that use OCR to extract handwritten text with accuracy?".

3.1.1 Search Process

The research question was framed according to the PICOCS model of the Table 3.2.

Table 3.2: LRRQ1 PICOCS model

PICOCS	RQ part	I/E	Sub string
P	Studies involving handwritten character recognition	I - studies that consider handwritten characters E - prior 2018	>=2018; Handwritten
I	Usage of OCR for handwritten text	I - studies about OCR E - studies that do not consider handwritten text	OCR; Optical Character Recognition
C	—		
O	Accuracy and limitations	E - studies that do not present results	Accuracy; Limitations
C	Practitioners or students	I - professionals I - academic projects	Professionals; Academic
S	Case studies	E - surveys E - literature reviews	Case study

The table includes the inclusion and exclusion (I/E) criteria related with each part of the research question, as well as the possible sub strings that can be used to conduct the search. Only case studies from 2018 onwards that contemplate OCR for handwritten character extraction were taken into consideration. Studies that do not present results, such as the accuracy or limitations, were excluded.

The PRISMA flow in Figure 3.1 describes the process of the search that was conducted.

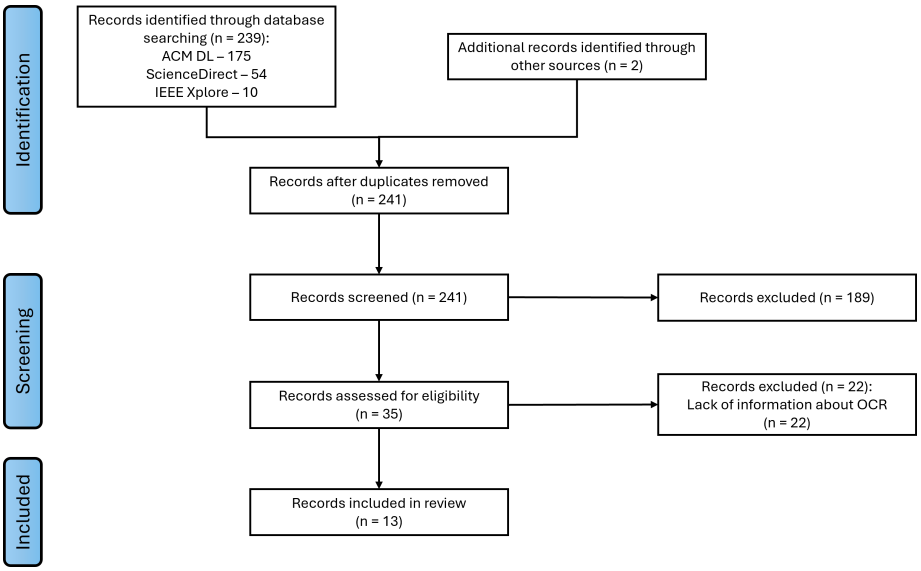

Figure 3.1: LRRQ1 PRISMA flow

Several records of the ACM Digital Library and ScienceDirect were excluded for being only references to entire journals or books. Some results were excluded for not being related with the topic of the search and not meeting the I/E criteria, such as the record not being a case

study. In the later stage of Screening, some studies were discarded for not focusing on OCR or not presenting useful results in that topic. **13** studies are going to be reviewed in the following subsection.

3.1.2 Discussion

Some of the studies that were found, explore the application and enhancement of OCR in specific types of handwritten characters.

In [36], the application of OCR for struck-out handwritten words was studied by training a ML model. It resulted in a high detection rate and quality measures, such as precision and recall [36]. However, the performance degraded in noisy or blurred environments [36].

The case study [37] explored the application of OCR for text written with Indic script. Even though the authors of the study mention that there is room for improvement, accuracies between 87% and 95% were achieved [37].

Similarly, in [38], the recognition of handwritten Georgian characters was explored with custom ML models, leading to over 94% accuracy.

In [39], it was concluded that it is possible to recognize text containing historical lexicon with a good precision. An open source OCR engine was used in the case study. However, a book-specific model training is required for a high quality recognition when there are several typographies [39].

When it comes to real-world case studies, there are several researches reported in the literature.

One of these studies is [40], which is related with the digitization of child protection cards. It involves a scenario similar to this dissertation, where completely moving to digital services, like digital forms, is not viable. Because of that, digitizing the forms was explored as an alternative [40]. Commercially available OCR services were not efficient on the field environment, so a custom solution was developed using ML. The solution was tested in the field and worked successfully, achieving accuracies between 93% and 98% [40].

Both [41] and [42] explored the application of OCR for historical documents with crowdsourced post-correction. The first study focus on ancient manuscripts, while the second focus on historical Dutch documents. Both researches achieved high accuracy and an effective process. In [41], maintaining a stable community of contributors was a great challenge. In [42], person names were a challenge, as they were harder to recognize.

In the case study [43], OCR was used to extract text from smartphone screenshots. An accuracy of 71% to 74% was achieved, which is lower than other studies. The application of pre-processing procedures slightly improved the results [43].

The case study [44] explores the analysis of images with malicious content in social networks, like toxic content or misinformation. The state of the art OCR-based systems were described as vulnerable to slightly manipulated images, but the accuracy could be improved by over 10% with a custom model [44]. By applying a post-correction algorithm, the authors of the study managed to improve the accuracy of the detector by over 20% [44].

A solution that helps match children with potential adoptive parents was developed in [45]. In this solution, a cloud service was used, which presented some difficulties dealing with images captured from a distance higher than 30 centimetres, with more than 30° deviation or with

noise in the background [45]. When these limitations are avoided, the solution is accurate at recognizing the text in the forms [45].

Similarly, an OCR cloud service was used in [46] to develop a social pension management system. The solution had a high accuracy, but struggled at differentiating uppercase and lowercase letters, as well as struck-out text [46].

In [47], a solution was developed for automating mark entry in educational institutions. It significantly accelerated data processing workflows by efficiently converting and processing the marks of students [47]. The main challenges were detecting decimal numbers, scalability and compatibility with different formats [47].

The case study [48] explores the recognition of handwritten code from Computer Science students. When combining an OCR cloud service with post-correction techniques, it proved to be accurate enough to be used in real learning environments [48]. One of the main challenges of the solution is dealing with struck-out text [48].

In general, the application of OCR for recognizing handwritten text has proven to be accurate in many case studies. However, in some cases, using only OCR technology was not accurate enough, and pre-processing or post-correction techniques were applied. In some studies, the commercially available OCR services were not sufficient, leading to training custom ML models. The main challenges found were struck-out text and noisy images.

3.2 LRRQ2 - OCR tools with high accuracy for handwritten text

In this section, the literature will be examined in order to answer the research question "What studies are there about OCR tools with the best accuracy for handwritten characters?".

3.2.1 Search Process

The research question was framed in accordance with the PICOCS strategy in the Table 3.3.

Table 3.3: LRRQ2 PICOCS model

PICOCS	RQ part	I/E	Sub string
P	Studies involving handwritten character recognition	I - studies that consider handwritten characters E - prior 2020	>=2020; Handwritten
I	OCR tools	I - studies about OCR	OCR; Optical Character Recognition
C	—		
O	Best tool	I - studies about OCR tools I - studies comparing OCR tools	Tool; Engine; Service; Comparison; Benchmark
C	Practitioners	I - professionals	Professionals
S	Empirical studies or surveys	I - Empirical study I - Surveys I - Case study	Empirical study; Case study; Survey

The review includes studies that consider handwritten character recognition and compare different tools. Only studies from 2020 onwards are going to be considered to reduce the probability of finding comparative studies with outdated or deprecated tools.

The flow of information of the conducted search is described in the PRISMA diagram in Figure 3.2.

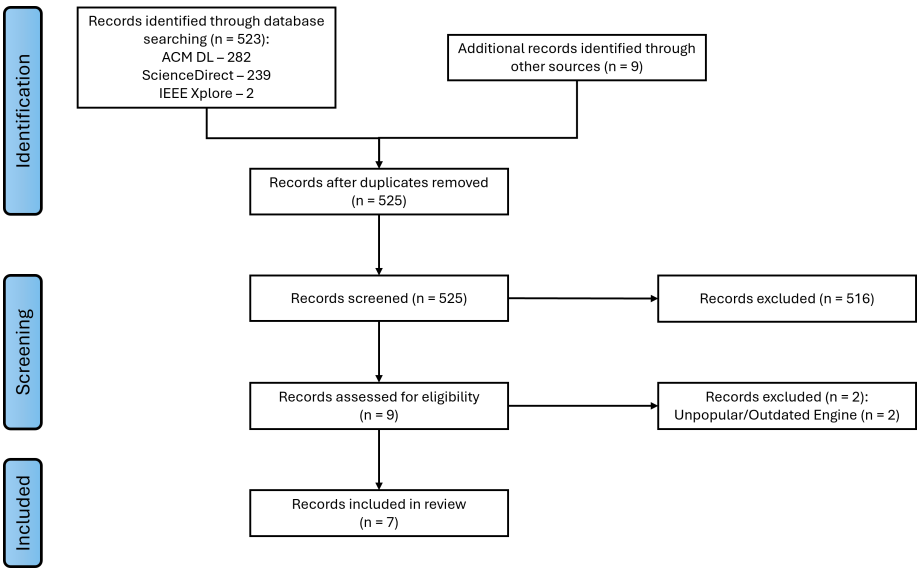

Figure 3.2: LRRQ2 PRISMA flow

Most of the records where excluded for not meeting the I/E criteria, specifically for not comparing OCR tools. In the last phase of Screening, some studies were excluded for comparing unpopular or outdated tools. In the discussion, **7** studies will be examined.

3.2.2 Discussion

There are many OCR tools that can be divided in three categories: proprietary software, open source engines and online services [49]. Some examples of proprietary software are ABBYY FineReader, Transym OCR or Readiris. As for the open source engines, Tesseract, OCRopus and Calamari are some of the most popular. Regarding online services, the cloud solutions provided by AWS (Textract), Google Cloud (Vision AI and Document AI) and Microsoft Azure (Document Intelligence) are some of the most used.

In the comparative study [50], the tools Tesseract, AWS Textract and Google Vision AI were compared using a dataset that includes printed and handwritten medical records. Both cloud services, Textract and Vision AI, performed significantly better than Tesseract. Between the services, AWS Textract achieved a slightly higher accuracy (88.89% against 87.63%) for handwritten text recognition.

Similarly, the study [6] compares Tesseract, AWS Textract and Google Document AI. The used dataset includes scanned documents in English and Arabic that were replicated multiple times, leading to over 18000 documents processed by each OCR engine. The results of this study also suggest that the cloud services, Textract and Document AI, perform significantly better than Tesseract, specially on noise documents. In this case, the Google cloud service, Document AI, achieved a slightly higher accuracy with lower error rates, compared to Textract.

The study [51] analyses all the types of OCR tool that were mentioned previously, by comparing Google Vision AI (online service), Tesseract (open source), ABBYY FineReader and Transym OCR (proprietary). The dataset included key-frames extracted from lecture videos, containing digital images, machine-written text and handwritten text, with variations in the text orientation, noise, skewness and blurriness. Google Vision AI was the tool that performed better, with an average accuracy of 96.7%.

On the open source side, the study [52] compared the engines Tesseract, Easy OCR and Keras OCR. A dataset with multiple categories, ranging from vehicle number plates, receipts, handwritten text and symbols, was used to compare the tools. Easy OCR was the tool that performed better on average. However, Keras OCR performed slightly better for the handwritten text category.

In [53], 180000 pages of a historial book with imperfections were used to compare the OCR services from the three leading cloud providers, that is AWS, Microsoft Azure and Google Cloud. In a larger scale, similar results were found for the services. AWS Textract achieved a lower median error rate, while the Azure Service achieved a lower processing time. On the other hand, Google Vision AI had a better combination of low error rate and duration. In terms of costs, they were similar for the analysed services.

The study [54] compares the tool Tesseract with Google Document AI by using a dataset with varying resolution, brightness, blurriness, skewness and noise. Higher accuracy values were achieved by Document AI both at the word and character levels. Nevertheless, the results for Tesseract were close.

In the study [55], the tools Tesseract, Google Document AI and AWS Textract. Three different dataset were used, containing distorted documents, photos of receipts and clean scanned multilingual forms, respectively. Textract performed significantly better with distorted documents, reaching over 95% accuracy. Document AI had a higher accuracy for photos of receipts, reaching over 93%. For the scanned multilingual forms dataset, Textract performed slightly better, with a higher combination of accuracy and precision.

In general, the results suggest that the OCR cloud services have better performance compared to the other types of tool. Furthermore, the services from AWS (Textract) and Google (Vision AI and Document AI) achieved higher accuracy results, with a slight advantage for AWS Textract in several studies.

3.3 LRRQ3 - Image enhancement techniques

This section includes the literature review for the research question "What studies are there about image enhancement techniques that can improve OCR accuracy?".

3.3.1 Search Process

The PICOCS model was used to frame the research question, as can be seen in the Table 3.4.

Studies from 2018 onwards that analyse image enhancement or other OCR pre-processing techniques will be considered in the review. Only studies that compare results will be taken into consideration, either comparing with the original image or different techniques. Both professional and academic studies will be considered.

Table 3.4: LRRQ3 PICOCS model

PICOCS	RQ part	I/E	Sub string
P	Studies involving OCR	I - studies that consider OCR E - prior 2018	>=2018; OCR
I	Image enhancement techniques	I - studies about image enhancement	Image enhancement; Pre-processing
C	—		
O	Best techniques	I - studies about image enhancement techniques I - studies comparing image enhancement techniques	Techniques; Comparison
C	Practitioners or students	I - professionals I - academic	Professionals; Academic
S	Empirical studies or surveys	I - Empirical study I - Surveys I - Case study	Empirical study; Case study; Survey

The PRISMA diagram in Figure 3.3 summarizes the flow of information during the conducted search.

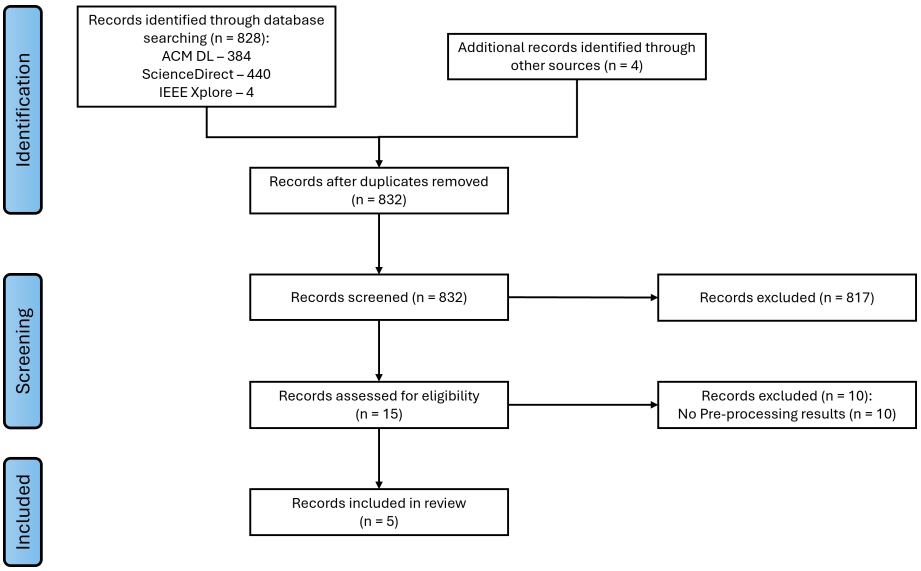

Figure 3.3: LRRQ3 PRISMA flow

Most records were discarded for not complying with the I/E criteria, particularly for not being OCR-related and for not using or describing the pre-processing techniques in use. Several papers were excluded for not presenting results related with the pre-processing techniques. In the next subsection, **5** studies will be reviewed to answer the research question.

3.3.2 Discussion

There are several pre-processing techniques described in the literature, ranging from geometric transformations to pixel transformations [56].

Skew correction is a geometric transformation that consists of identifying and fixing the skewness of the text [56]–[58]. The most used algorithm is Projection profile [57], but there are other algorithms that can be used, like Hough transform and Nearest neighbour approaches [56].

Another geometric transformation is Cropping, in which only the relevant part of the document image is considered [56]. This technique is useful for rectangular documents, and potentially improves OCR performance by removing unnecessary content from the image [56]. The process is similar to skew correction, which means that similar algorithms can be used, like Hough transform [56].

Regarding pixel transformations, Binarization is one of the most pre-processing popular techniques [56], [57], [59]. It consists of converting the image to black and white, in which usually the black colour represents the foreground layer and the white colour represents the background layer [56]. This can be achieved through the application of various algorithms, such as Otsu, Niblack, Savoula, Triangle or Adaptive thresholding [57], [59].

Noise removal is a pre-processing technique that involves eliminating unwanted imperfections and random variations from the image [56], [57], [60]. By removing the noise, the text in the image can be seen more clearly [57]. Methods like the Median filter or Gaussian filter can be used to implement this technique [56].

Another pixel transformation is Sharpening, which is a technique that involves reducing the blur of the image [56]. The sharpness of the image is improved by highlighting the edges and finer details of the components of the image [56].

In the study [59], the Binarization technique was tested with historical newspaper images. The results significantly improved by up to 18%, specially when using the Otsu thresholding method.

Similarly, the study [57] applied Binarization in a system that extracts text and converts to Braille. Even though the accuracy was already good, the application of image pre-processing improved the results.

The study [56] explored the application of Cropping, Binarization, Noise removal and Sharpening on a dataset of scanned receipts. Only a portion of the dataset showcased an improved OCR accuracy when applying the enhancement techniques individually. In some cases, the application of the techniques even impacted the results negatively. In the cases in which they impact positively, the accuracy improved slightly compared with the raw image. The results also suggest that combining multiple pre-processing techniques result in higher accuracy values.

In [60], the Noise Removal and Binarization techniques were applied. The Noise removal technique did not perform well when applied individually, and even degraded the quality of the image. A possible cause can be a poor implementation of the algorithm. Regarding Binarization, it slightly improved the accuracy in most cases. The authors of the study also state that when applying multiple techniques, the accuracy improved.

The study [58] explored the application of Skew correction in scanned documents with complex layouts. The results suggest a significant improvement of the OCR accuracy by up to 23%.

In general, the application of image enhancement and pre-processing techniques, such as Binarization, Skew correction and Sharpening, can improve the accuracy of text extraction. However, in some cases, these techniques can impact negatively the results and even degrade the quality of the original image, which is a concern that needs to be taken into account.

3.4 Summary

In this chapter, the literature was reviewed in order to answer the research questions about the feasibility of OCR for handwritten text, tools with high accuracy for the same type of text, and pre-processing techniques that can be applied to improve the accuracy of text recognition.

The answer to RQ2 is that a solution that uses OCR to extract handwritten text is feasible. Many case studies were found in which OCR was used to recognize handwritten text accurately. In some cases, pre-processing or post-correction techniques were needed to compensate shortcomings of the OCR engines, as well as training custom ML models. Struck-out text and noisy or blurry images were the main challenges during the application of OCR in the case studies.

Regarding the OCR tools, the comparative studies in the literature suggest that cloud services perform better compared to proprietary and open source software, with special regards for AWS Textract and Google Document AI. Answering RQ3, AWS Textract is the OCR tool with better accuracy, with Google Document AI coming close as a great alternative.

As for RQ4, methods such as Binarization, Skew correction, Noise removal and Sharpening can be used to improve OCR accuracy. Nonetheless, some awareness is needed as there is a possibility of these techniques degrading the quality of the original image.

Chapter 4

Analysis & Design

In this chapter, the technical description of the project will be detailed, including the analysis of the problem and the design of the proposed solution. The content of this chapter will allow answering RQ5.

The first section will detail the analysis of the problem by describing the domain and identifying the requirements. In the second section, the design of the solution will be presented, including the architecture and components of the system.

4.1 Analysis

In this section, the problem will be analysed, in order to provide a clear understanding of what needs to be solved and the expectations for the solution. The domain of the project will be described, as well as the functional and non-functional requirements.

4.1.1 Domain

The domain model in Figure 4.1 includes the main entities and relationships that are relevant to the project. These can be used to understand the problem and specify the requirements for the solution.

Each concept is annotated with a label that indicates the area of the domain it belongs to. The **Core** annotation identifies the entities that are related to data extraction and management. The **Validation** annotation indicates that the entity is related to the validation of filled data. Finally, the **Submission** annotation is used for entities that are related to the submission of the extracted data.

The base concept of the domain is the **Form** entity, which represents the structure of a form that can be filled by users. Besides the name and description, it includes a list of **Form Fields** to extract, with the respective type of value that will be extracted (**Form Value Type**). A **Form Field** can be identified as a key-value pair or a table cell, holding the specifications for the respective type.

Form Response represents the instance of a filled form that was uploaded. It contains **Form Response Fields**, that hold the values extracted from the image. Each **Form Response Field** is connected to a **Form Field** of the respective **Form**. The **Form Response Status** indicates the current state of the form response, depending on the extraction and submission result.

Regarding the **Validation** entity, it allows comparing **Form Field** values to validate that the forms were filled correctly. When the **Validation Type** is **CONSTANT**, the subject field is

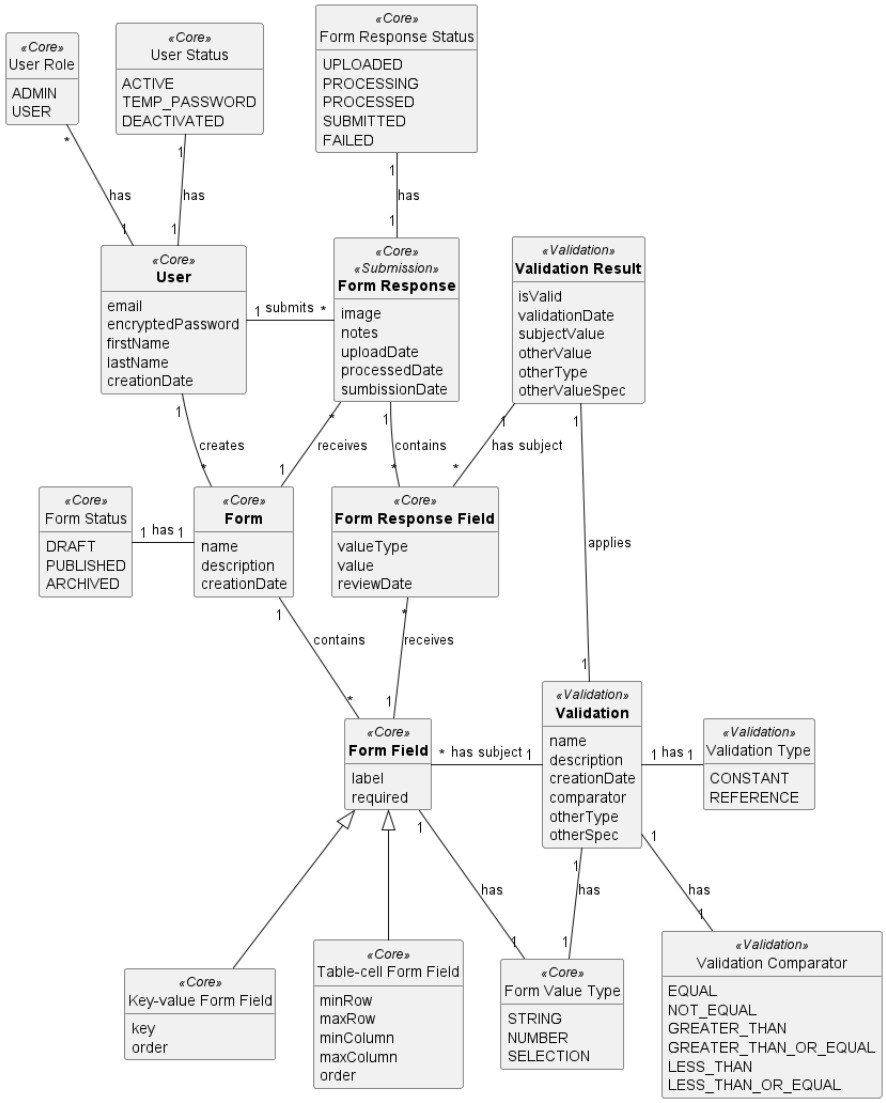

Figure 4.1: Domain model

compared with a constant value, and when it is **REFERENCE**, the field is compared with one or more other fields. Different **Validation Comparators** can be used when comparing the values, depending on the value type.

When a **Validation** is applied, a **Validation Result** is created, which indicates whether the validation was successful or not. It contains the **Form Response Fields** that were used in comparison, as well as their resulting values.

The **User** is a supporting entity that represents the user of the system. It can be either an **ADMIN** or a regular **USER**, leading to different levels of access. The **User Status** indicates the current state of the account. It can only be created by an **ADMIN** and stays in the **TEMP_PASSWORD** state until the user tries to sign in for the first time and changes the password.

4.1.2 Functional requirements

The activity diagram in Figure 4.2 illustrates the main journey of the user in the system, highlighting the main actions that are performed.

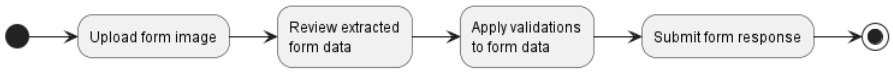

Figure 4.2: Activity diagram of the main user journey

The typical flow involves uploading a photo of the filled form, which is then precessed by the system to extract the values. The user can review the extracted data, in case there were mistakes, and apply validations to ensure that the form was filled correctly. After that, the user can submit the form response with the correct information.

Based on the domain model and the main user journey, the functional requirements were specified according to the use case diagram in Figure 4.3.

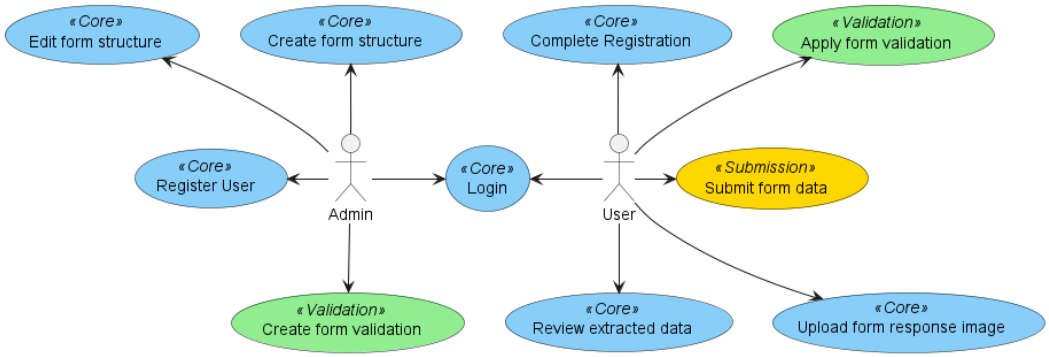

Figure 4.3: Use case diagram

The use cases of a regular **User** include the completion of its registration, uploading photos of filled forms, reviewing the data that was extracted from a form response, applying validations to the extracted data and submit form responses.

Besides the **User** use cases, an **Admin** is responsible for the back office operations, such as registering new users, creating and editing the structure of forms, and creating new form validation. The login use case is common to both actors, being required to access all the other functionalities of the system.

4.1.3 Non-functional requirements

The Functionality, Usability, Reliability, Performance, Supportability (FURPS) model [61] was used to specify the non-functional requirements of the system to be developed.

The **Functionality** requirements are architecturally significant and support the use cases identified as functional requirements in the previous subsection. The following requirements were specified:

- The user passwords must be stored securely, using a hashing algorithm, such as Bcrypt;
- Firebase Analytics must be used to track interactions and collect metrics about the usage of the application;

- Only authenticated users must be able to access the functionalities, except for the login;
- The system must support role-based access control;
- Only users authenticated with administrator role must be able to access back office operations.

In terms of **Usability**, the following requirements were identified:

- The user must be able to upload the photo by using the camera inside the application or by selecting a photo from the gallery;
- The mobile application must provide a consistent navigation pattern across all screens.

When it comes to **Reliability**, some requirements were specified to ensure that the system is robust and can handle errors gracefully:

- The system must be available 99.5% of the time;
- Unexpected application errors must be handled gracefully, without crashing the application.

Regarding **Performance**, the following requirements were identified:

- The backend must support rate limiting for the data extraction endpoint, allowing 3 requests per minute at most, to avoid abusing the OCR tool quotas;
- The mobile application must start up in less than 3 seconds to avoid frustration for the user;
- The dashboards in the backoffice web application must load within 5 seconds.

As for the **Supportability**, the following requirements were defined to ensure maintainability, testability and adaptability of the system:

- The codebase of the backend must have at least 80% line coverage with unit tests;
- The backend must be configurable through environment variables or configuration files;
- The mobile application must be accessible on Android devices with Software Development Kit (SDK) level 26 or higher (Android 8.0 version);
- The codebase of the mobile application must have at least 70% line coverage with unit tests;
- The mobile application must be implemented with regionalization in mind;
- The mobile application must use Firebase Crashlytics to report crashes and errors;
- The codebase of the backoffice web application must have at least 70% line coverage with unit tests;

4.2 Design

In this section, the design of the solution will be presented, including its architecture and the Continuous Integration/Continuous Delivery (CI/CD) that will support the development. The design will be based on the analysis, ensuring that the requirements are met.

4.2.1 Architecture

The architecture of the solution was designed according to a combination of the C4 [62] and 4+1 [63] models. The C4 model allows visualizing the system at different levels of granularity [62], while the 4+1 model describes the system from different perspectives [63]. Only the most relevant views will be presented, providing an overview of the global architecture.

The component diagram in Figure 4.4 represents the logical view of the solution at level 1.

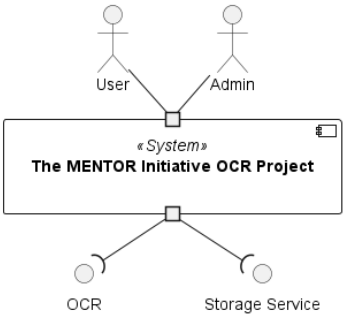

Figure 4.4: Level 1 Logical View

This view allows visualizing the interactions between the system and the external environment. It is possible to identify that the main actors are the **Admin** and the regular **User**. The system also interacts with third party services, such as **OCR**, for data extraction, and **Storage Service**, to store the uploaded photos.

At level 2, the containers of the system are identified, as shown in the logical view in Figure 4.5.

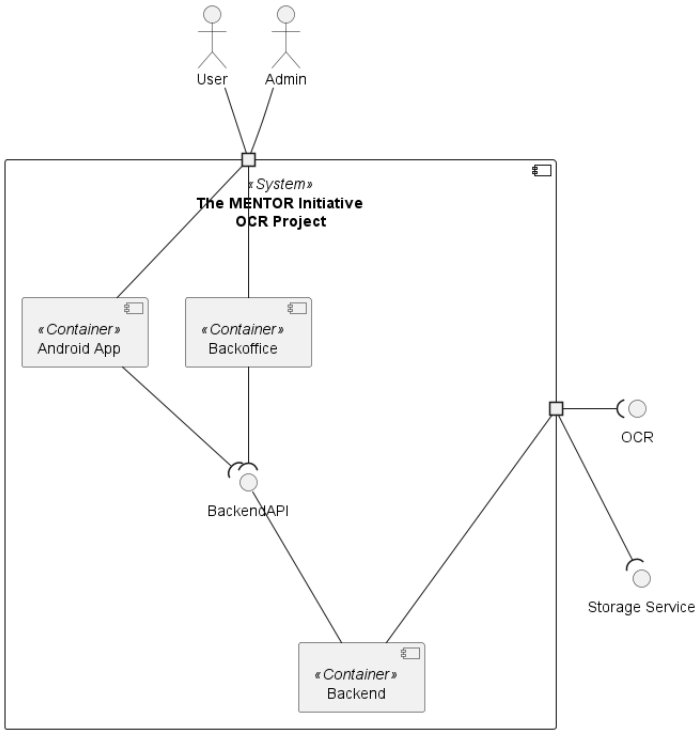

Figure 4.5: Level 2 Logical View

The **Backend** is responsible for the business logic and data management, serving as a foundation for the frontend containers. It interacts with the external services, for data extraction and image storage. The **Backend** exposes a Representational State Transfer (REST) Application Programming Interface (API) that acts as an interface and allows communicating with it.

Another container is the **Android App**, that allows the users to interact with the system and perform the main actions, such as uploading photos and reviewing the extracted data. As for the **Backoffice**, it is mostly accessible by the administrators and allows them to manage the system, such as creating new users and editing the structure of forms. Both frontend containers consume the API exposed by the **Backend**.

The physical view of the system was represented with the deployment diagram in Figure 4.6.

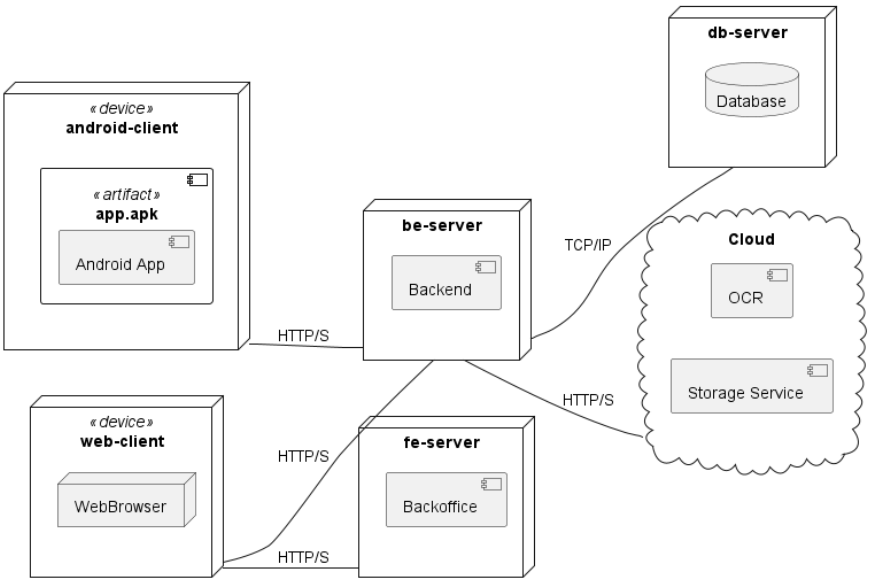

Figure 4.6: Level 2 Physical View

The physical view allows visualizing the planned infrastructure of the system. The **Backend** is deployed on its own server and connects to the **Cloud** services through Hypertext Transfer Protocol Secure (HTTP/S). It also connects to the **Database** server, where the data is stored, using Transmission Control Protocol/Internet Protocol (TCP/IP).

The **Backoffice** is a Single-page Application (SPA) that is hosted on a web server. As it will follow a Client-side Rendering (CSR) approach, the **WebBrowser** in the user device connects to the **Backoffice** server to download the page and its resources, and then communicates directly with the **Backend** through HTTP/S.

Regarding the **Android App**, it is packaged in a Android Package Kit (APK), which is the artifact that is distributed to the users, that can install it on their Android devices. The **Android App** connects to the **Backend** through HTTP/S to perform the main actions.

When it comes to the level 3, each container is separated into components that describe its architecture. Only the logical view will be described for each container.

The component diagram in Figure 4.7 represents the logical view of the Backend.

The architecture of the **Backend** is based on a combination of Model-View-Controller (MVC) and layered architecture. The **Controller** is responsible for intermediating the application

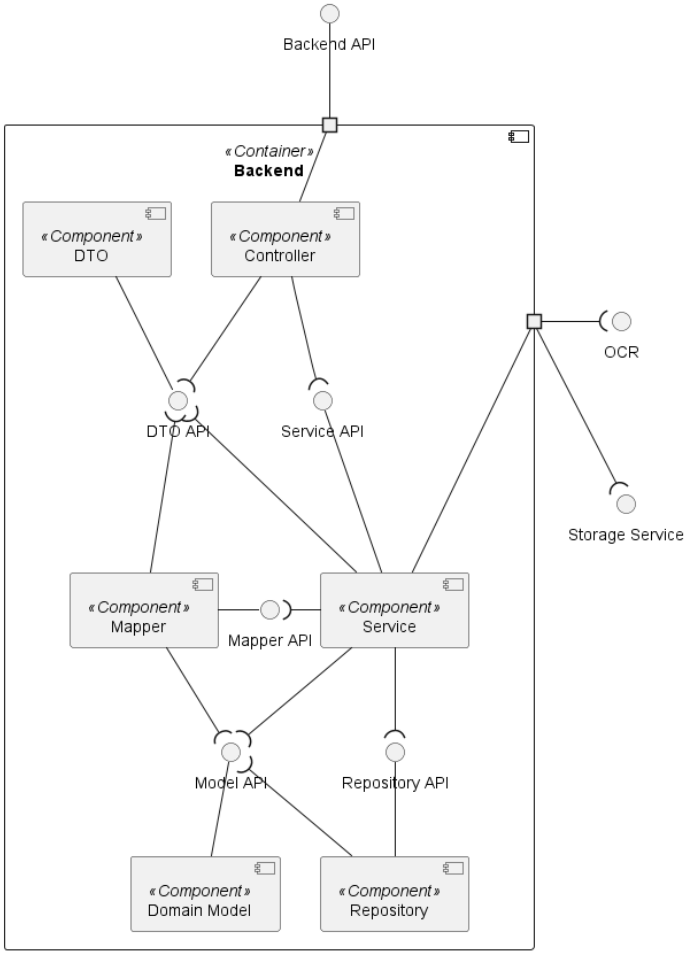

Figure 4.7: Level 3 Logical View of the Backend

data, in **Model**, and the **View**, which, in this case, is represented by the output returned by the API. The layers of the architecture are noticeable in the diagram, with the upper layers being closer to the user interaction, through the API, and the lower layer being closer to the persistence of the data.

The **DTO** component represents the data objects that are received and returned by the API. The **Controller** component is responsible for handling them in the requests and responses. In the **Service** component, the business logic is implemented, being responsible for processing the data and applying the necessary transformations. It makes use of the **Mapper** component to convert the data between the **Domain Model** and the **DTO**. The **Repository** is the component closer to the persistence, being responsible for accessing the database and performing the necessary operations.

Regarding the Android App, the logical view is represented by the component diagram in Figure 4.8.

The **Android App** follows a Model-View-ViewModel (MVVM) architecture, where the business logic and the presentation of the application are separated. The **View** is represented by the User Interface (UI) that is presented to the user, while the **Model** is represented by the data that comes from the API or the local persistence, as well as the business logic built on

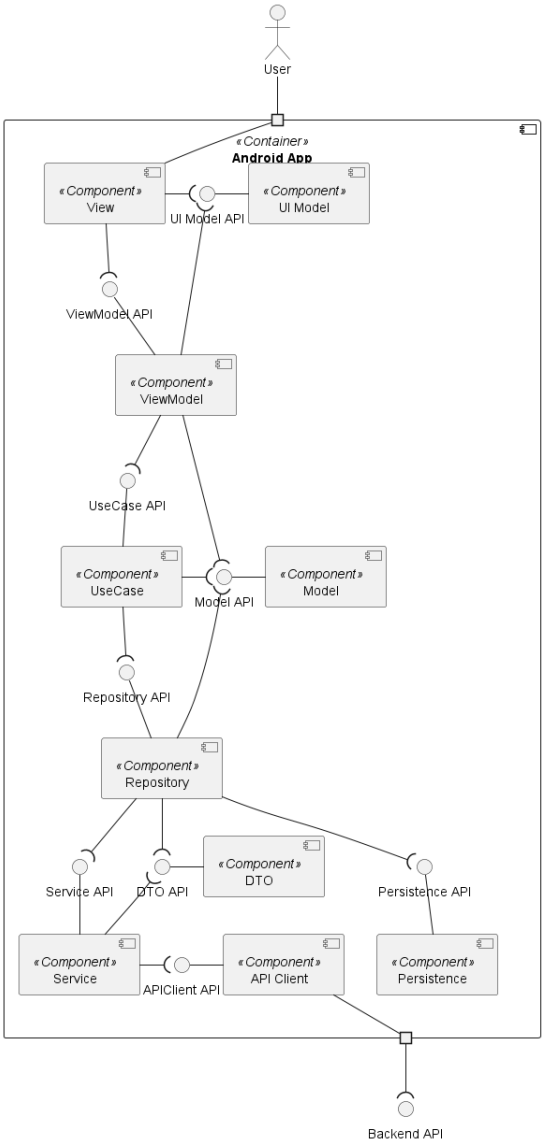

Figure 4.8: Level 3 Logical View of the Android App

the client-side. The **ViewModel** is responsible for preparing and updating the **Model** data that is then presented by the **View**.

When the data comes from the API, the **API Client** is used by the **Service** to perform the necessary requests and fetch the **DTOs**. The **Repository** is responsible for handling the data sources, including the local **Persistence** and the API. The **UseCase** serves as an abstraction for the data layer, as well as handling some extra business logic that is not directly related to the UI.

The **UI Model** component represents the data that is ready to be presented by the **View** component. The **ViewModel** is the component responsible for managing and updating that data, as well as notifying the **View** when it is changed. Besides rendering the UI, the **View** is also responsible for handling user interactions and notify them to the **ViewModel**, so that it can perform the proper action, such as calling an **UseCase** or directly updating the **UI Model**.

Finally, the logical view of the Backoffice is represented by the component diagram in Figure 4.9.

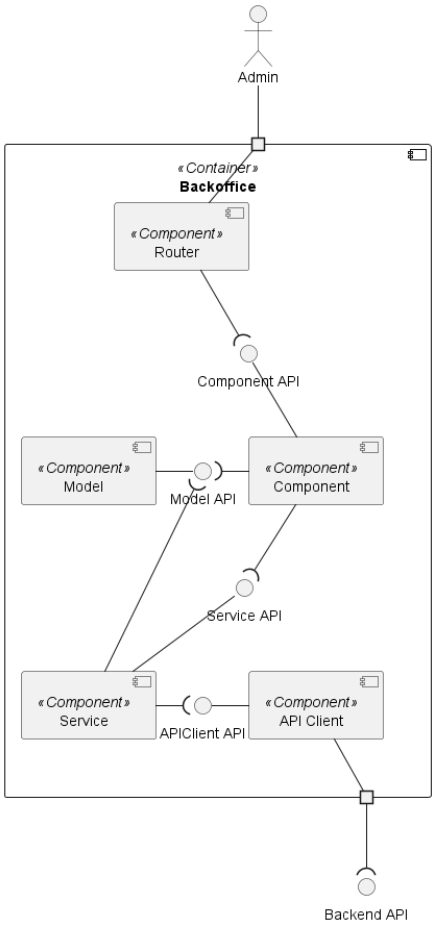

Figure 4.9: Level 3 Logical View of the Backoffice

The **Backoffice** follows a component-based architecture, where the UI is built using **Components** that can be reused across the pages of the application. Each page is also considered a **Component**, and its navigation is controlled by the **Router**, that renders the correct page according to the path in the Uniform Resource Locator (URL).

The **API Client** communicates with the **Backend API**, being then used by the **Service** component to perform the necessary requests and fetch the **Model** data. This data is then handled by local states in the **Component**.

4.2.2 CI/CD

In order to support the development of the solution, the CI/CD will be designed to automate the process and shorten the development lifecycle. Some pipelines will be created to automate the validation and deployment of the code.

Before designing the pipelines, the branching strategy needs to be defined to understand how they can be integrated into the workflow.

GitHub Flow [64] was chosen as the branching strategy, as it combines simplicity and flexibility. The diagram in Figure 4.10 illustrates how it can be used in a project.

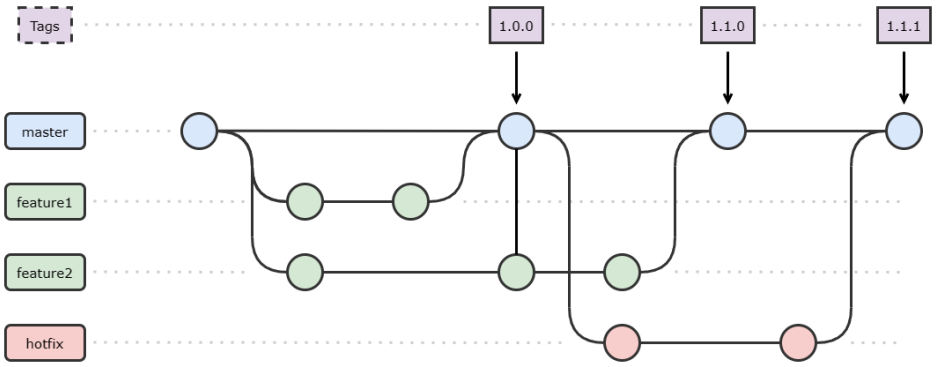

Figure 4.10: Branching model

The **master** branch is the main branch of the project, where the code is considered stable and ready for production. A **feature** branch is created for each new feature, task or bug fix, allowing the development to be isolated from the main branch. When other changes are merged into **master**, the **feature** branch can be updated to be in sync, resolving any conflicts that may arise.

A tag can be pushed to the **master** branch to indicate a new release and the cut-off point for the code that is going to be deployed in that release. If there is a critical bug in production that needs to be fixed, a **hotfix** branch can be created from the release tag, allowing the fix to be applied without affecting the ongoing development.

One of the pipelines that was planned is meant to apply a basic validation to the different parts of the project, following the flow in Figure 4.11.

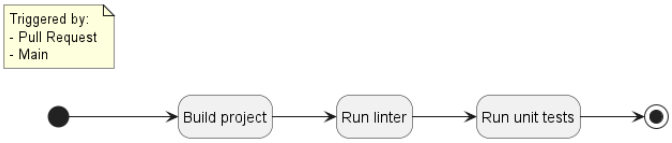

Figure 4.11: Basic validation pipeline

The pipeline is triggered when a pull request is opened or updated, or when changes are pushed to the main branch. Each container of the system has its own pipeline, which is only triggered when changes are made to the respective codebase.

The same steps are followed for all the containers, starting by building the code. After that, the linter is executed to ensure that the code complies with the coding standards. The final step involves running the unit tests, to guarantee that the code is working as expected. If any of the steps fail, the pipeline is marked as failed.

As for the deployment pipeline, the diagram in Figure 4.12 represents the flow that is followed to deploy the code to production.

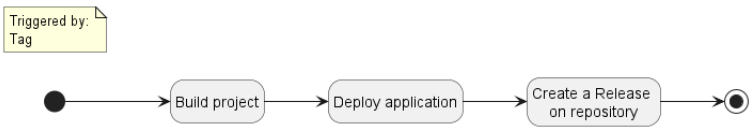

Figure 4.12: Deployment pipeline

This pipeline is triggered when a tag is pushed to the **master** branch. The name of the tag is composed of a prefix that indicates the container that is being deployed, followed by the version name.

In the first step, the project is built, ensuring that the code is ready for deployment. After that, the project is deployed to the respective location, which can be a server, for the Backend and Backoffice, or an application distributor, such as Firebase App Distribution, for the Android App. The final step is to document the release on the repository, providing the release notes and the source code that was deployed.

4.3 Summary

As an answer to RQ5, this chapter presented the analysis and design of the solution. The domain was modelled in order to raise the functional and non-functional requirements. These were taken into account when designing the architecture of the solution, which is composed of a Backend, an Android App and a Backoffice web application. The architecture was represented in different levels of granularity, through different perspectives. The CI/CD was also planned to support the development and deployment of the solution.

Chapter 5

Implementation

This chapter will focus on the details of the implementation of the project, including the different parts of the system. The functionalities, design patterns and testing will be described for each component of the system. The content of this chapter will allow answering RQ6.

In the first section, the implementation of the backend will be described, followed by a section for the Android Application, and, finally, the section of the Backoffice, which will only be described broadly.

5.1 Backend

The Backend project was built using NodeJS with the programming language TypeScript. The framework Express was used to expose a REST API, by handling the routing, requests and responses. As for the database, PostgreSQL was used with the Object-Relational Mapping (ORM) Prisma, which handles the database operations and migrations, making it easier to interact with the tables.

The API documentation is powered by Swagger, which exposes the endpoints and allows testing them in the browser, following the OpenAPI specification. This documentation will be used to present the functionalities of the Backend.

For the continuous integration, the project uses TypeScript Compiler for the Build phase, ESLint to perform lint checks, and Jest to run the unit tests. The pipeline was configured to run on GitHub Actions.

The deployment server is hosted on Amazon Web Services (AWS), using a virtual machine with Ubuntu (EC2 instance). The same instance is hosting both the Backend application and the PostgreSQL database. By running on AWS, the project benefits from an easier integration with other services, such as the storage service (S3) and OCR (Textract).

5.1.1 Functionalities

In this subsection, the implementation of the main functionalities of the Backend will be detailed, including Swagger documentation and code snippets.

User Management & Authentication

The user-related functionalities allow managing and requesting the access to the system. The available endpoints are listed in Figure 5.1.

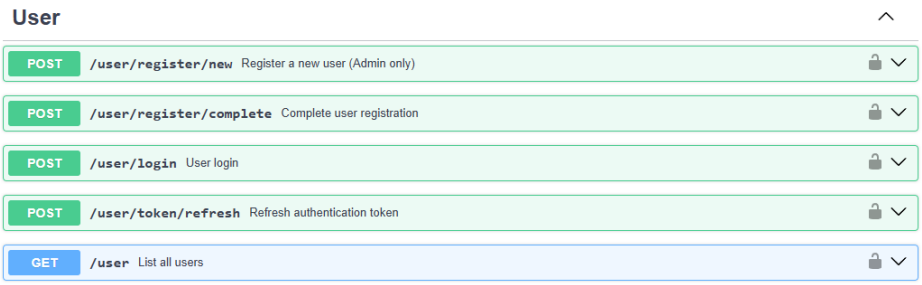

Figure 5.1: User Management & Authentication endpoints

The authentication system is based on a controlled registration process, where only the administrators can register new users. This means that Role-Based Access Control (RBAC) is being enforced, as the role of the user defines what endpoints can be accessed.

When an administrator registers a new user, the system generates a random password and sets the status of the user to flag that has temporary password. When the user first logs in, they are required to change the password, by calling the endpoint to complete the registration.

The password is stored securely after being hashed with Bcrypt algorithm. In the login endpoint, the password is verified against the stored hash, using the library of the algorithm, as can be found in the Code Snippet 5.1.

```
async hashPassword(password: string): Promise<string> {
    return await bcrypt.hash(password, 10);
}

async validatePassword(password: string, encryptedPassword: string): Promise<
    boolean> {
    return await bcrypt.compare(password, encryptedPassword);
}
```

Listing 5.1: Password hashing and validation

The functions from the **bcrypt** library are used to hash a new password before being stored, and to compare a provided password with the stored hash. A number of salt rounds was chosen in a way that the hashing process is secure, but not too slow, as it would impact the user experience.

When the user logs in, an access token and a refresh token are generated using JSON Web Token (JWT), as shown in Code Snippet 5.2.

```
generateToken(payload: object, expiresIn: number, secretKey: string): string {
    return jwt.sign(payload, secretKey, { expiresIn: expiresIn });
}

decodeToken<T>(token: string, secretKey: string): T | null {
    try {
        return jwt.verify(token, secretKey) as T;
    } catch (err) {
        return null;
    }
}
```

Listing 5.2: JWT Token generation and decoding

A payload is stored in the JWT token, including the identifier of the user in the system, the email, the roles and the status. The tokens are signed with a secret key that is fetched from the environment variables, differing for the access and refresh. The two types of token also have different expiration time, with the access token being short-lived and expiring in 1 hour, while the refresh token has an expiration time of 2 weeks.

The refresh token can be used to refresh the session of the user when the access token expires, by requesting the respective endpoint. That endpoint returns new access and refresh tokens that can be stored on the client side.

An endpoint that lists the users of the system was also implemented, being accessible only to the administrators.

Form Structure

The endpoints in Figure 5.2 allow specifying the structure of the forms from where the data is extracted.

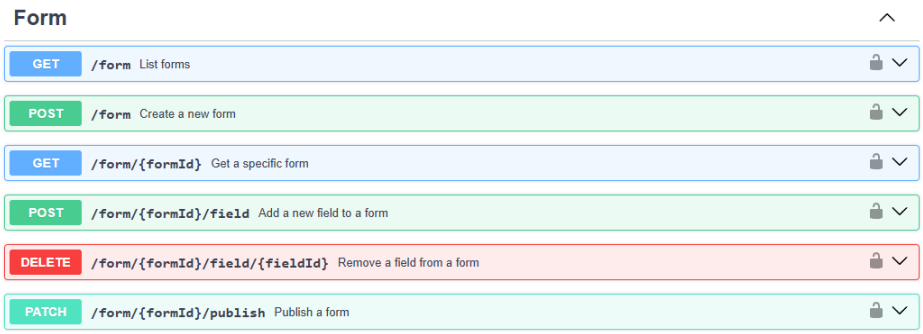

Figure 5.2: Form Structure endpoints

An administrator can start specifying a form by requesting the endpoint for the creation of a new form, passing the name and description. The form is created in draft status, meaning that it can not be used to extract data yet.

While in draft status, the fields of the form can be added or removed, specifying the type of data that is expected and the type of field, either key-value or a table cell. Since the specification of the field depends on the type of field, it is stored on the database in a column with JavaScript Object Notation (JSON) data type.

When the form structure is ready, the publishing endpoint can be requested, which updates the form status to published, enabling other operations on that form.

Only the **GET** endpoints are accessible to the regular users. There is an endpoint that can be requested to fetch a paginated list of forms and another that returns a specific form.

Form Response Extraction

The Form Response endpoints are accessible to any authenticated user and revolve around the extraction and review of the data from forms, as listed in Figure 5.3.

The first endpoint can be used to upload a new form response to the system, indicating which form it refers to. Unlike the other endpoints, which receive JSON bodies, this one receives the content as multipart/form-data, in order to accept image files.

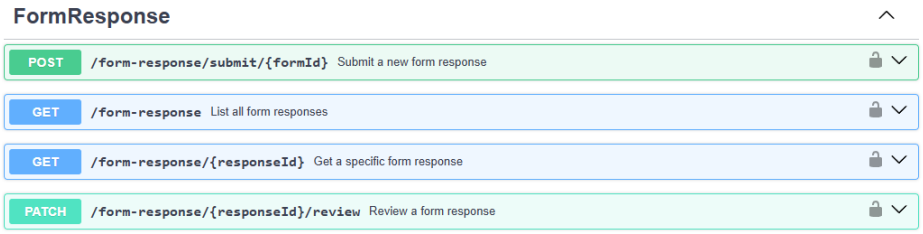

Figure 5.3: Form Response endpoints

After validating the size of the file and if the file is an image, it is uploaded to the storage service S3, with the public URL being stored on the database. A unique name is generated for the file, using the timestamp and a random number.

With the uploaded image, the data of the form is then extracted using the OCR tool AWS Textract. The Code Snippet 5.3 shows how the response of the tool is being orchestrated.

```
async extractFormResponseFields(imagePath: string, formFields: FormField[]):
    Promise<FormResponseField[] | null> {
    (...)
    const command = new AnalyzeDocumentCommand(params);
    const response = await this._textractClient.send(command);

    const blocks = response.Blocks;
    const keyBlocks = this.getKeyBlocks(blocks);
    const cellBlocks = this.getCellBlocks(blocks);
    const mergedCellBlocks = this.getMergedCellBlocks(blocks);
    return formFields.map(field => {
        if (field.type === FormFieldType.KeyValue) {
            return this.extractKeyValueFormField(blocks, field, keyBlocks);
        } else if (field.type === FormFieldType.TableCell) {
            return this.extractTableCellFormField(blocks, field, cellBlocks,
    mergedCellBlocks);
        } else {
            return null;
        }
    }).filter(field => field !== null);
}
```

Listing 5.3: AWS Textract usage

The response of the client includes a list of **Blocks** that were extracted from the image. Each **Block** has an identifier and a type, such as "KEY_VALUE_SET", "TABLE", "CELL", "MERGED_CELL" and "WORD", among others. It can also have relationships with other blocks, depending on the type, with "WORD" being the most elemental type of block, where the extracted text can be found. The other types of block usually have multiple child "WORD" blocks that can be joined to form a line or statement.

For each field of the intended form, the value is extracted according to its type. When it is a key-value field, the blocks with "KEY_VALUE_SET" are checked to find a block with the expected key. For table cell fields, the "CELL" and "MERGED_CELL" blocks are verified to find the cell with the expected column and row.

The expected type of value is also taken into account when extracting the data. For textual fields, multiple "WORD" blocks are joined with whitespace. In numeric fields, the textual representation is converted to number, if possible. When it is a selection field, the child "SELECTION_ELEMENT" block is checked.

In order to double-check the extracted data and check any missing fields, Generative Artificial Intelligence was used, specifically the model Gemini 2.5 Flash from Google, as presented in Code Snippet 5.4.

```
async extractFieldsFromImage(file: Buffer, contentType: string, formFields:
    FormField[]): Promise<FormResponseField[] | null> {
    const prompt = this.buildPrompt(formFields);
    const content = await this._genAIClient.models.generateContent({ ... });
    const response = JSON.parse(content.text);
    return formFields.map(field => {
        const fieldResponse = response.find(responseField => responseField.field
    === field.label);
        if (!fieldResponse || !fieldResponse.value) return null;
        const valueObject = this.getFormResponseFieldValue(field.valueType,
    fieldResponse.value);
        if (!valueObject) return null;
        return new FormResponseField("", field.valueType, valueObject, null, "",
    field.id);
    }).filter(field => field !== null);
}
```

Listing 5.4: Google Gemini usage

The **prompt** for the request is built using the information of the form fields that are meant to be extracted from the image. The request is also configured with structured output, in order to generate a JSON response with a custom schema, which is then parsed. The custom schema specifies that the output is an array of objects that includes a "field" property, containing the label of the extracted field, and a nullable "value" property that can contain the extracted value for that field.

Similar to what was done with AWS Textract, depending on the expected type of value, the text is converted. When it is a numeric field, the textual representation is converted to a number, if possible. If it is a selection field, the textual representation is either "true" or "false", being converted to the respective boolean value.

The **PATCH** endpoint allows reviewing the extracted data of a form response, as well as the additional notes. In the body, a list of objects is passed, in which the new values are associated with the respective field identifier. After validating the passed fields and updating the data in the database, the endpoint returns the updated object for the form response, as well as a list of requested fields that were not updated for being invalid. This can happen if no field was found with the requested identifier or if the value does not match with the expected type of value.

The form responses can be fetched using the listing endpoint or the endpoint that retrieves a specific form response. The listing route optionally allows filtering the form responses by form, as well as selecting the form responses that include specific form fields or form response fields. These filter options can be specified as query parameters.

Validation

The Validations allow checking that the extracted data is correct. The endpoints listed in Figure 5.4 allow managing, applying and fetching the validations.

The first endpoint is accessible only to the administrators and is used to create a new validation in the system. Besides the name and description, the subject field, comparator and type of validation is received in the body of the request. A comparator that checks the equality or

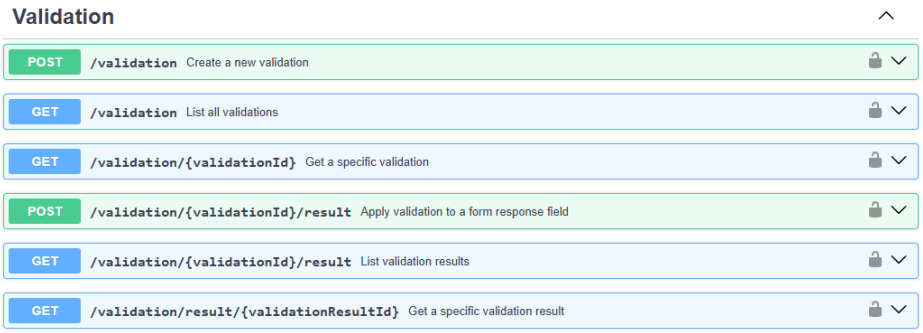

Figure 5.4: Validation endpoints

difference can be defined for any type of value, while the numeric fields can also be validated using comparators that check if the value is greater or less than the other.

When the validation is created with the type "CONSTANT", a constant value is specified, in order to be used against the field in the comparison. When the validation uses "REFERENCE", a list of fields that can be used in the validation are specified, as well as a flag that indicates if the validation can receive multiple form response fields or not.

The listing endpoint allows fetching the validations registered in the system, with the possibility to filter by form. There is also an endpoint that retrieves a specific validation by its identifier.

In order to apply a validation to a specific form response, the second **POST** endpoint can be requested. In the body of the request, the identifier of the form response field is specified, as well as the other field identifiers, in case it is a reference validation. The result of the validation is returned by the endpoint, containing a flag that indicates the result of the comparison, as well as the values that were used.

When it is a constant validation, the value of the form response field is directly compared with the constant value. In the reference validations, the other value depends on the specification. If allows only a single field, the subject form response field value is compared with the value of the referenced field. If multiple fields are allowed, the values of the referenced fields are joined, according to their type, before doing the comparison. For textual fields, the values are joined with whitespace, for numeric fields, the sum operation is applied, and for selection fields, the conjunction logical operation is applied.

Two endpoints are also provided to fetch validation results, either by listing the results of a specific validation or fetching a specific validation result.

5.1.2 Design patterns

In this subsection, some of the design patterns that were followed in the Backend project will be described.

As designed in the architecture, the **Repository** pattern was used to abstract the access to the data source and centralize the data operations. The ORM tool Prisma was used in the implementation of the repositories to facilitate the connection to the database and the handling of the data operations.

The **Service layer** pattern is another design pattern that was noticeable in the architecture and was used in the project. This layer is responsible for handling the business logic, connecting the controllers with the data layer.

The **Data Transfer Object (DTO)** pattern was also followed in the Backend. Through these flat objects, the contract of the inputs and outputs of the API is defined. They contain only the data and serialization logic, without any business logic. This allows the domain to be changed with less risk of breaking changes in the contracts of the API.

In the listing endpoints, the **Pagination** pattern was used to separate the complete list into smaller sets, leading to smaller payloads and, consequently, higher performance. An offset-based approach was followed in the project. In this type of pagination, an offset and a limit are provided to define the starting point and the size of the segment of data. This pattern is being implemented as shown in Code Snippet 5.5.

```
async fetchForms(offset: number, limit: number): Promise<{ forms: Form[], total:
    number }> {
    const [forms, total] = await Promise.all([
        this._prismaClient.form.findMany({
            skip: offset,
            take: limit
        }),
        this._prismaClient.form.count()
    ]);
    return {
        forms: forms.map(form => this.toFormDomain(form)),
        total: total
    };
}

toPaginationDTO(offset: number, limit: number, total: number): PaginationDTO {
    return {
        offset: offset,
        limit: limit,
        total: total,
        previousOffset: offset > 0 ? Math.max(0, offset - limit) : null,
        nextOffset: offset + limit < total ? offset + limit : null
    };
}
```

Listing 5.5: Pagination pattern in Backend

The first function is part of a repository class, in which the data is fetched from the database. Taking advantage of Prisma capabilities, two queries are performed concurrently. One of the queries fetch the segment of data, using the **offset** and **limit**, while the other query gets the total number of entries for that entity.

The second function is part of a shared mapper that is used by the service classes. It builds and returns and object with the pagination information, calculating the **previousOffset** and **nextOffset** when they are applicable.

5.1.3 Testing

Unit tests were implemented for multiple layers of the Backend, using Jest testing framework. Only some files with no logic were not considered in the unit tests, such as the route files and the configuration files for the database and third party tools.

The Code Snippet 5.6 includes an example of unit test for a service class.

```
describe('ValidationServiceImpl tests', () => {
    let validationRepository: MockProxy<ValidationRepository>;
    let formRepository: MockProxy<FormRepository>;
    let validationService: ValidationService;

    beforeEach(() => {
        validationRepository = mock<ValidationRepository>();
        formRepository = mock<FormRepository>();
        validationService = new ValidationServiceImpl(validationRepository);
    });

    describe('createValidation', () => {
        test('WHEN subject field does not exist THEN throw bad request error',
    async () => {
            formRepository.fetchFormFieldById.mockResolvedValue(undefined);

            const dto: CreateValidationReqDTO = { ... };
            await expect(validationService.createValidation(dto))
                .rejects.toThrow(new ClientError(ErrorType.BadRequest, ErrorMessage
    .Validation.SUBJECT_FIELD_DOES_NOT_EXIST));
        });
    }
}
```

Listing 5.6: Backend unit test example

All the test cases for the class are placed inside a **describe** block, with the mocks and subject class instance being configured at the beginning. The **beforeEach** block is used to reset the instances before each test. An inner **describe** block is created with the test cases for each public function.

In this case, the example test case is validating that when the subject field does not exist on the database, an error is thrown. Firstly, the mock of the repository is configured to return **undefined**, indicating that there is no form field with that identifier on the database. Then, the **dto** parameter is built and passed to the function. The execution of the function is validated to check if it throws an error with the **BadRequest** type and the correct message.

Using the same testing framework, the test coverage of the testable code was collected, resulting in the values in Figure 5.5.

Figure 5.5: Backend test coverage

A **90.1%** line code coverage was achieved in the Backend project, which complies with the non-functional requirement that was specified in Section 4.1. The branch coverage was lower than the line coverage, with a value of **79.2%**, which means that some paths in the code were not covered by tests. These uncovered paths are mostly concentrated in the logic of data extraction, which is more complex, with a lot of paths and edge cases.

5.2 Android Application

The Android Application was built natively using Gradle with the programming language Kotlin. The framework Jetpack Compose was used to build the UI directly in Kotlin code using a declarative style. Jetpack Navigation is responsible for handling the navigation within

the app. The API requests are handled by the library Retrofit with OkHttp, while the local persistent storage is handled by Jetpack DataStore with Protocol Buffers. Dependency injection is supported by the framework Hilt, which helps reducing coupling between the classes.

As part of continuous integration, the project can be built using a command from the Android plugin in Gradle. The linting is powered by the plugin Detekt, while the unit tests were built using the framework JUnit.

The application is configured to connect to the Backend server and is distributed using Firebase App Distribution. Firebase Analytics was used to track user interactions, using also Crashlytics capabilities to track unexpected errors in the application.

5.2.1 Functionalities

The implementation of the functionalities of the Android Application will be described in this subsection.

Authentication, Account & Home screens

The Figure 5.6 includes the authentication screens, as well as the home and account screens that can be accessed in the application.

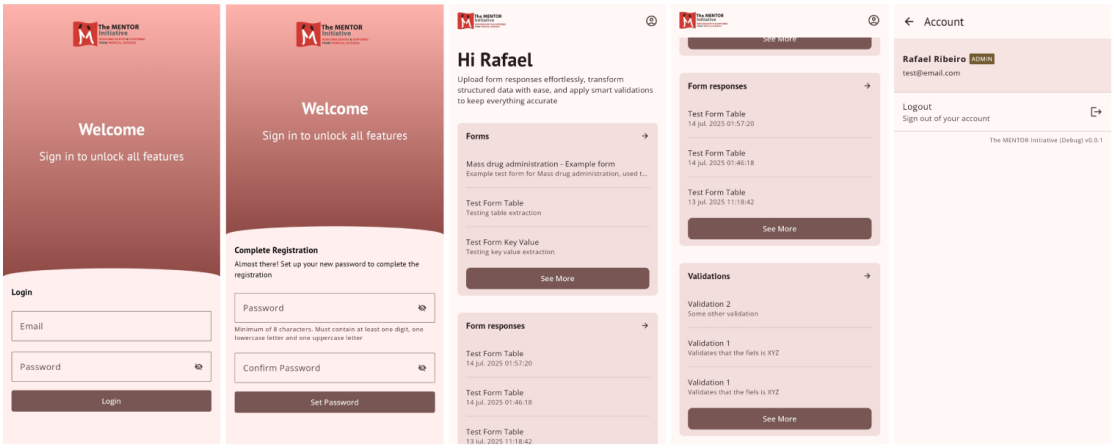

Figure 5.6: Authentication, Home and Account screens

In the **Login** screen, a welcome message is shown, as well as text inputs for the email and password. The password input allows showing and hiding the text. When the button is clicked, the login request is performed, with an error message being shown when it fails. If the user is using a temporary password, the **Complete Registration** screen is shown, where a new password needs to be defined. A password input is shown with the requirements, as well as an input to repeat the password. When the button is clicked, an API request is performed, showing an error message if it fails.

The **Home** screen is composed of an icon that allows opening the account, an informative message about the functionalities of the application and three cards with recent forms, form responses and validations. Each card has an arrow icon that allows opening the full list of items and shows at most three items. When there are more than three items in the full list, a "See More" button is shown.

In the **Account** screen, some information of the user is rendered, such as the name, email and the role, when it is an administrator account. A "Logout" button is also shown, which clears the stored session and redirects the user to the login screen.

The user needs to be authenticated to use the application, which means that the login screen is shown in the first time that the application is opened and after every logout.

The session of the user includes its information, as well as the access and refresh tokens. It is stored in a local persistent storage, that is encrypted with the algorithm AES256. The stored session is observed in order to render the correct content, which can be either the login screen or the normal application content, depending on whether the user is authenticated or not.

When the access token expires, it needs to be refreshed before doing more authenticated requests. The Code Snippet 5.7 shows how the authorization is being handled.

```
internal class TokenAuthenticator @Inject constructor(...) : Authenticator {
    override fun authenticate(route: Route?, response: Response): Request? {
        val newToken = runBlocking { tokenManager.refreshToken() }
        return if (newToken.isNotBlank()) {
            response.request.newBuilder()
                .header("Authorization", "Bearer $newToken")
                .build()
        } else {
            null
        }
    }
}
```

Listing 5.7: Authorization in mobile application

The **Authenticator** interface from OkHttp is used to intercept requests that return status code 401. In that case, the **authenticate** function is invoked, in which the refresh token request is performed to fetch a new access token. The new token is then used to retry the original request if the refresh operation succeeded.

List screens

The screens in Figure 5.7 show the full list of items for the forms, form responses and validations.

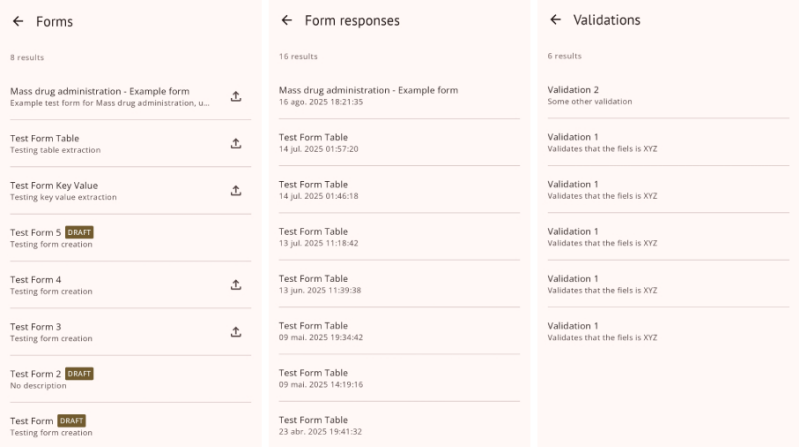

Figure 5.7: List screens

All the list screens follow a similar UI, with the same pagination logic. The total number of results is rendered at the top of the screen, with the items being rendered below, separated with horizontal dividers. The content of the item varies for each case. For the forms, the name and description are shown, as well as a tag when it is in draft status and an upload icon button when it is published. In the form response list, the respective form name and the upload date are rendered. For the validations, the name and description are used in the list item.

The size of the pagination pages is set to 30 in the list screens, with the next page being loaded when the user reaches the last third of the page. During the loading, a spinner is shown at the end of the list.

The library AndroidX Paging was used to facilitate the integration and handling of the pagination in the UI. A PagingSource was created for each list, with the logic to invoke the API request and define the previous and next keys, which are the offsets in this case. It returns an observer of PagingData that can be integrated in the lazy list UI components with the support of the library.

Details screens

When an item in the list screens is clicked, the respective details screen from Figure 5.8 is opened.

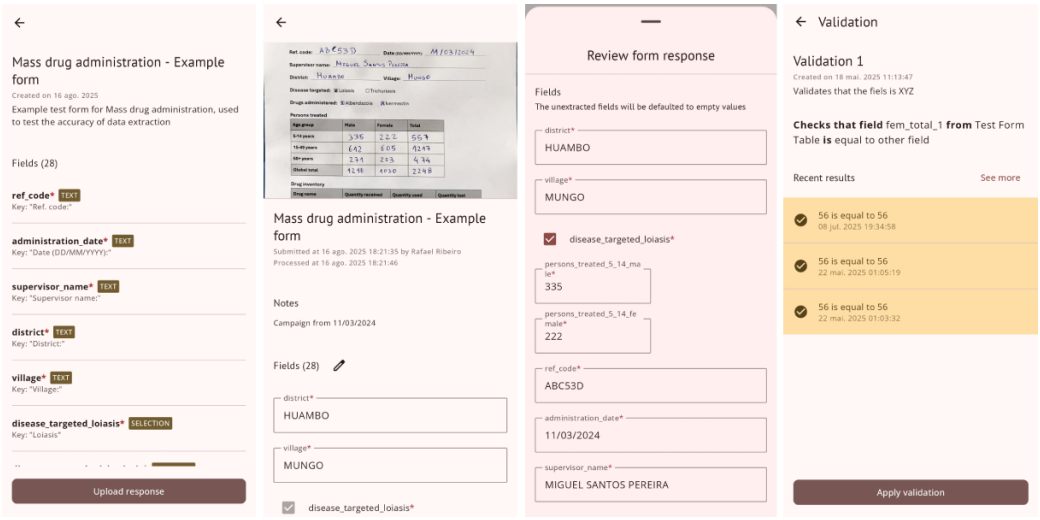

Figure 5.8: Details screens

The first screen shows the details of a form, including the name, description, creation date and the expected fields. When the form is in draft status, a tag is rendered below the name. When the form is archived, an informational message is shown at the top of the screen. For each field, an item is rendered, containing the label and type of value, as well as the specification, depending on the type of field. For key-value fields, it shows the **Key**, while for table cell fields, it shows the **Row(s)** and **Column(s)**. The screen is built with lazy list UI components to improve the performance.

In the second screen, the details of a form response are shown. At the top of the screen, the uploaded image is rendered, with a fullscreen view, that allows zooming, being shown when it is clicked. Below the image, the name of the form is shown, which can be clicked to open the respective details, as in the first screen. Besides that, the submission dates, notes and fields

are rendered lazily, similar to the previous screen. The fields are shown in read-only inputs or checkboxes, depending on the type of value, with the extracted value or the message "Could not extract value" when the extraction fails for that field.

When the pencil icon in the fields section is clicked, a bottom sheet is opened where it is possible to review the extracted data. The same inputs and checkboxes are rendered, but enabled for user interaction, allowing the values to be changed. When the "Save" button is clicked, the API is requested to review the form response, updating the details in the screen in case of success.

The last screen presents the details of a validation, including the name, description, creation date and specification. The specification is built based on the type of comparison, comparator and fields involved. Besides that, a list of the five latest results for that validation is also rendered, with the possibility of opening the full list. The "Apply validation" button at the bottom of the screen allows opening the screen for applying the validation.

Upload form response

The Figure 5.9 includes the flow for uploading a new form response, which can be accessed from the form list and form details screens.

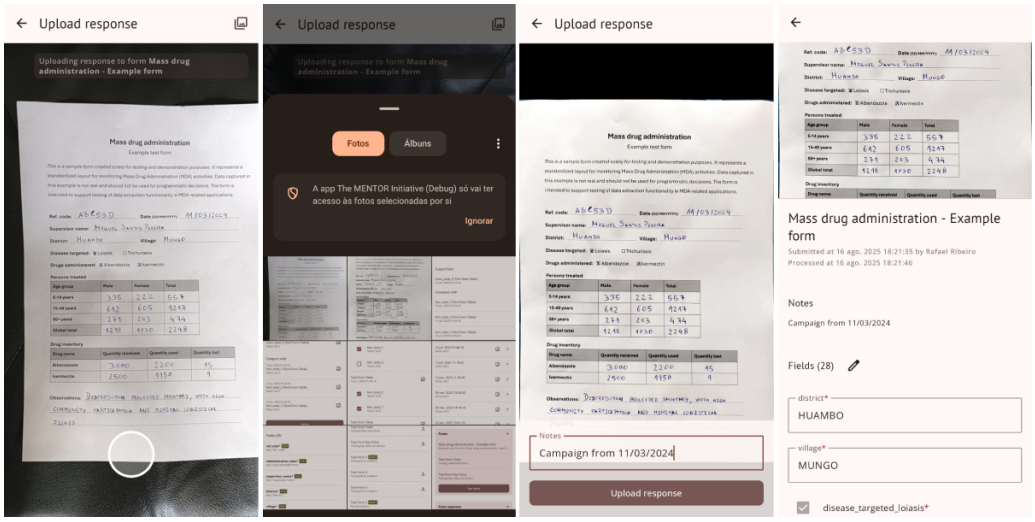

Figure 5.9: Upload Form Response screens

The **Upload response** screen contains a camera view that allows the user to take a picture inside the application. On top of the view, there is a label indicating the name of the form to which the response will be uploaded. At the bottom of the screen, there is a circle that can be clicked to take the picture. When the screen is opened for the first time, the user is asked to give permission to access the camera. If the user rejects the permission, the content is changed to show a button that redirects to the application settings, in order to allow the user to change the permission.

As an alternative, there is an icon in the title bar that can be clicked to choose a photo from the gallery. A bottom sheet from the system is shown, without any need for permissions. The option to pick an image from the gallery is available even if the user rejects the camera permission.

After taking a picture or choosing it from the gallery, a preview is shown so that the user can confirm that it is correct. At the bottom of the screen, the user can input notes about the form response, before clicking in the "Upload response" button. When the button is clicked, a request is performed to the API to upload the image and extract the data. If it succeeds, the details screen of the form response is shown. If it fails, an error banner is rendered above the notes input.

Apply validation

The flow to apply a validation includes multiple steps, as shown in Figure 5.10.

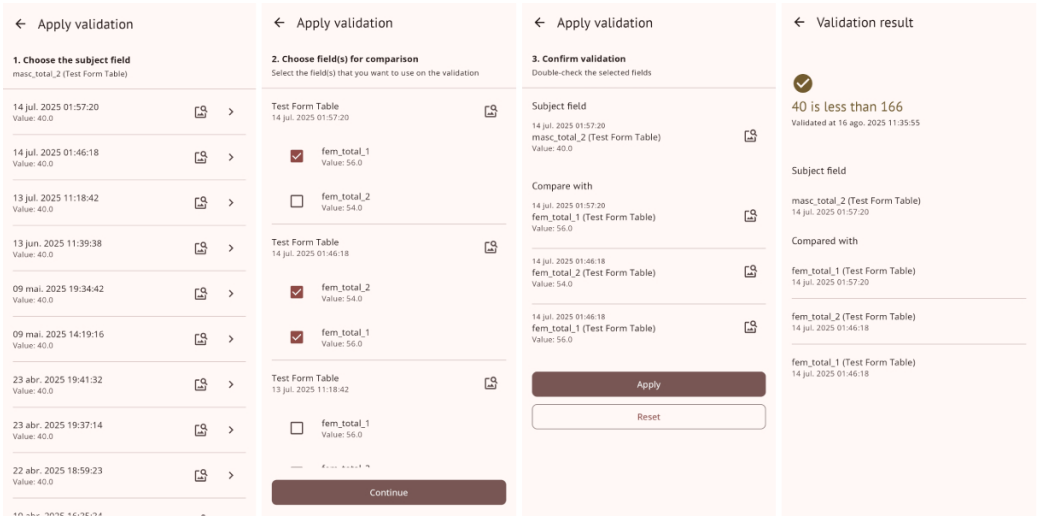

Figure 5.10: Apply Validation screens

In the first step, the form responses that include the subject field of the validation are listed, so that the user can choose one by clicking on the respective right icon. The form responses are identified by the submission date, with the extracted value of the field being presented as well. For each form response, there is an icon that allows previewing its image in a fullscreen view, in order to make sure that the correct field is being chosen.

When the validation is of reference type, a second step is shown with the form responses that have allowed fields for the comparison. For each form response, the name of the form and the submission date are presented, as well as the fields that are part of the allowed list, with the label of the field and the extracted value. The icon to preview the form response image is also shown in this step. If the validation is configured to accept multiple fields, a checkbox is rendered before each field to allow multiple selection. Otherwise, an icon pointing to the right is rendered for each field, allowing the user to choose that field.

In the last step of the flow, the information is presented to the user to confirm that it is correct. The information includes the chosen subject field, as well as the other fields that were chosen for the comparison, in case it is a reference validation. When the validation is configured to perform a comparison with a constant value, the confirmation step is shown right after choosing the subject field, without rendering any information about other fields.

The "Reset" button can be clicked to restart the process by going back to the first step. When the "Apply" button is clicked, the API request is performed, navigating to a screen with the validation result in case of success.

5.2.2 Design patterns

Similarly to the Backend, the **Repository** pattern was also applied on the Android Application to connect to the data sources. In this case, the repository classes coordinate the access to the Backend API and to the local persistent storage, where the session of the user is stored.

The **UseCase** pattern is being used in the domain layer to connect the data layer and the presentation layer. Each UseCase class has only one method and represents a single task, enforcing the **Single-responsibility Principle**. The business logic is applied on the UseCases, when applicable.

Instead of throwing and catching exceptions, the **Result** pattern was used to encapsulate the success or failure value. The **Result** class from Kotlin was used in the project, being returned in multiple layers, such as the API service, repository and UseCase.

Reactive programming, which extends from the **Observer** pattern, was used in the ViewModels to emit the data that the UI uses to render the content. This reactive apporach allows the UI to be constantly listening to changes in the data that has to be shown. In Code Snippet 5.8, the usage of the pattern is presented.

```
internal class FormDetailsViewModel @Inject constructor(...) : ViewModel() {
    private val _state = MutableStateFlow<FormDetailsUIState>(FormDetailsUIState.
    Loading)
    val state: StateFlow<FormDetailsUIState> = _state.asStateFlow()

    init {
        viewModelScope.launch {
            getFormUseCase(formId = args.id).fold(
                onSuccess = { form ->
                    val formDetailsUI = formDetailsUIFactory.buildFormDetailsUI(
    form)
                    _state.value = FormDetailsUIState.Data(formDetailsUI)
                },
                onFailure = {
                    _state.value = FormDetailsUIState.Error
                }
            )
        }
    }
}

@Composable
internal fun FormDetailsScreen() {
    val viewModel: FormDetailsViewModel = hiltViewModel()
    val state by viewModel.state.collectAsStateWithLifecycle()
    ...
}
```

Listing 5.8: Reactive programming in Android Application

In the first part of the snippet, part of the ViewModel of the form details screen is presented. At the beginning, the **StateFlow**, which is a built-in Kotlin structure, is initialized. On a **private** variable, the mutable version is initialized, with the **state** variable exposing a read-only version, so that it can only be modified inside the ViewModel. This is the data structure that holds the value of the state and emits it to the subscribers, which in this case is the UI. On the **init** block, the value of the **state** is being defined, depending on the result of the UseCase.

The second part of the snippet includes the code where the UI of the screen is built. The read-only **state** from the **viewModel** is subscribed and the collected value can be used to define the content. Jetpack Compose works in a way that when the collected value changes, the UI is re-rendered with the changes.

5.2.3 Testing

The unit tests were implemented using the testing framework JUnit, covering the data, domain and part of the presentation layer. Only UI-related code was not covered by unit tests, such as the screens built with Jetpack Compose and the navigation components.

The library MockK was used to mock objects in the tests, allowing the unit being tested to be isolated without using the real objects that it depends on. The library Turbine was used to facilitate the testing of Flows, which are reactive streams of data from Kotlin that are being used to emit the data that can be collected in the UI to render the content.

The Code Snippet 5.9 contains an example of unit test for a ViewModel class.

```
class AccountViewModelTest {
    private lateinit var getUserSessionUseCase: GetUserSessionUseCase
    private lateinit var accountInfoUIFactory: AccountInfoUIFactory
    private lateinit var viewModel: AccountViewModel

    @Before
    fun setup() {
        getUserSessionUseCase = mockk()
        accountInfoUIFactory = mockk()
    }

    @Test
    fun `failed user session load should update state with error`() = runTest {
        // Given
        coEvery { getUserSessionUseCase() } returns Result.failure(Exception())
        // When
        initViewModel()
        // Then
        viewModel.state.test {
            val state = awaitItem()
            assertEquals(AccountInfoUI.Error, state.accountInfo)
        }
    }
}
```

Listing 5.9: Android Application unit test example

The test classes are placed in the test directory with the same package as the original class. At the beginning of the test class, the subject object and its dependencies are defined with **lateinit**. Then, in the **setup** function, the mock dependencies and subject instance are initialized, except for the tests of ViewModels, which usually execute logic when they are initialized, so it needs to be initialized after configuring the mocks in the test cases. The tests are annotated with **@Test** and their name is defined with template string to allow a descriptive message.

In this example, the test is validating that when the user session fails to be fetched in the ViewModel of the Account screen, the **state** is defined to **Error**. Firstly, the mock of the **GetUserSessionUseCase** is configured to return a **failure**. The **viewModel** is then initialized before collecting the value of **state** with Turbine, and asserting that it is defined to **Error**.

The code coverage was collected for the testable code, using the plugin Kover. The results in Figure 5.11 were achieved.

app: Overall Coverage Summary

Package	Class, %	Method, %	Branch, %	Line, %	Instruction, %
all classes	88,1% (310/352)	71,8% (541/754)	52,2% (567/1087)	89,8% (2172/2419)	81,6% (13840/16970)

Figure 5.11: Android Application test coverage

A value of **89.8%** line code coverage was achieved in the Android Application, which complies with the non-functional requirements. On the other hand, the branch code coverage is significantly lower, with a value of **52.2%**. A possible reason for that value is the lack of tests for some paths in the ViewModel of the upload response screen, which involves handling images and files, that are hard to test.

5.3 Backoffice

The Backoffice project was built using React with the programming language TypeScript, in a NodeJS environment. The framework React Router was used to handle the routing of the web application and show the correct page. The library Axios was used to perform the requests to the Backend API.

For continuous integration, Vite was used to build the project in the Build phase. The lint check are performed with ESLint, while the tests are executed with Vitest.

The Backoffice was developed in a partnership with Mindera School, which is a learning programme that introduces young people to the world of technology and programming. As part of the programme, two students did their internships in this project, which consisted on the implementation of the Backoffice, with the mentorship of the author of the dissertation.

Even though the Backoffice was designed by the author of the dissertation, since it was implemented by the two students from Mindera School, the details of the implementation will not be described.

5.4 Summary

In this chapter, RQ6 was answered with a possible way of implementing the designed solution, by describing the implementation of the project.

The Backend was built with NodeJS and the framework Express, using TypeScript. The functionalities were detailed with the support of the generated Swagger documentation. Some design patterns were followed, such as Repository, Service layer and Pagination. Unit tests were implemented, achieving a value of 90.1% line code coverage.

As for the Android Application, it was built natively with Kotlin. The functionalities were detailed with screenshots of the application. Repository, UseCase and Result are some of the patterns that were used, with a value of 89.8% line code coverage being achieved.

The Backoffice was implemented by students of Mindera School, so the implementation was not detailed in this chapter.

Chapter 6

Experimentation & Evaluation

In this chapter, RQ7 will be answered by evaluating the accuracy of the text extraction in the developed solution, as well as the usability and efficiency of the mobile application. The accuracy will be measured using metrics and analytics data, while the usability will be evaluated through direct user feedback.

The first section will focus on measuring the accuracy of the text and field extraction, as well as testing the conditions that may affect it. The second section includes the analysis of the analytics data, as well as the direct feedback from the testers.

6.1 Text extraction accuracy

In this section, some metrics will be measured in order to analyse the accuracy of the text extraction in the developed solution.

6.1.1 Methodology

In this context, the accuracy represents the quality and classification performance of the text extraction, which will be evaluated with a set of appropriate metrics. The accuracy of text extraction was tested using an example test form that was filled by multiple people. The example form can be found in Appendix A and contains different types of field, such as key-value text fields, key-value selection fields and tables with numeric values. All the data captured in the responses is not real and was filled in for testing purposes only.

During the data gathering, the selection fields will be counted as one character and one word, while the numeric values will be considered as one word, with each digit being counted as a character.

The data extraction will be evaluated at different depths. At the character level, the metric Character Error Rate (CER) was used to quantify the percentage of characters that were incorrectly extracted from the form response [65]. It is calculated according to:

$$CER = \frac{LevenshteinDistance}{TotalCharacters}, \tag{6.1}$$

with a lower CER indicating better accuracy at finer granularity. The Levenshtein Distance algorithm is used to calculate the incorrect characters, by computing the minimum number of edits required to change the extracted text into the ground truth text [66], as follows:

$$LevenshteinDistance = Substitutions + Deletions + Insertions. \tag{6.2}$$

The number of $Substitutions$ is gathered by counting the characters that need to be replaced, while $Deletions$ and $Insertions$ are counted by the number of characters that need to be removed or added, respectively. The higher the $LevenshteinDistance$ value, the more errors there are in the extracted text.

At the word level, the metric Word Error Rate (WER) was used to measure the percentage of words that were incorrectly extracted, similarly to CER [65]. It is calculated using the equation:

$$WER = \frac{Substitutions + Deletions + Insertions}{TotalWords}, \tag{6.3}$$

where the incorrect words are calculated using a method similar to Levenshtein Distance, as in Equation 6.2, but applied to words instead of characters. A lower WER value indicates better accuracy at the word level.

For the field level, the metric Field Error Rate (FER) was used, based on CER and WER. It was calculated by comparing the number of fields that were incorrectly extracted to the total number of fields in the form response, as follows:

$$FER = \frac{Substitutions + Insertions}{TotalFields}. \tag{6.4}$$

In this case, the $Substitutions$ are represented by the number of fields where the extracted value does not match the ground truth value, while the $Insertions$ are the number of fields that were not extracted but are present in the ground truth, which means that they need to be added. Since the extracted fields are controlled by the application, there is no way to extract fields that are not present in the form response, so the $Deletions$ are not considered in this metric. Similarly to the previous metrics, a lower FER value represents a better accuracy at the field level.

As the domain revolves around form fields, other metrics were also collected at this level. The Precision [67] is measured to determine the proportion of correctly extracted fields out of the fields that were extracted, according to the following equation:

$$Precision = \frac{TP}{TP + FP}. \tag{6.5}$$

The True Positives, represented by TP, are the number of fields that were correctly extracted, while the False Positives, represented by FP, are the number of fields that were extracted but do not match the ground truth value.

The Recall [67] is also measured to determine the proportion of correctly extracted fields out of the number of fields that should have been extracted, as follows:

$$Recall = \frac{TP}{TP + FN}. \tag{6.6}$$

In this case, the False Negatives, represented by FN, are the number of fields that were not extracted but are present in the ground truth, which are the number of fields that the system failed to extract, as well as the fields that were extracted but do not match the ground truth value.

Finally, the F1 score is measured to provide a metric that combines both Precision and Recall, by calculating their harmonic mean [67], as shown in the equation:

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}. \tag{6.7}$$

The F1 score ranges between 0 and 1, similarly to the Precision and Recall. A higher value means that the system is both precise and thorough in extracting the fields from the form response. A lower value indicates that one or both of these aspects are lacking.

Following a non-probability convenience sampling technique, 8 forms were filled by different people that were asked to complete the form with fake data, in a way that resembled a real-world scenario. Some forms were filled in English, while others were filled in Portuguese, in order to test the multilingual capabilities of the solution. Different styles of handwriting were also used, such as cursive or print handwriting.

In order to compare the accuracy, the forms were captured in two different ways: by taking a picture with the camera inside the application or by scanning the form using an external scanner application and picking the scanned image from the gallery.

All the statistical analysis was performed using the programming language R 4.1.2 (2021-11-01) in RStudio 2022.02.0 (Build 443), with the packages dplyr (1.0.8), ggplot2 (3.3.5), BSDA (1.2.1), tidyr (1.2.0) and moments (0.14).

6.1.2 Results

After filling the forms and uploading them to the application, using both the in-app camera and scanned images, the metrics were calculated and gathered in a single dataset, which is presented in Appendix B.

Each reference code that was filled in the forms was mapped to a simpler identifier in the dataset. Information about the language used and the handwriting style was also associated with the entries in the dataset.

The data was summarized by calculating the mean, standard deviation, minimum and maximum values for each metric, as shown in Table 6.1.

Table 6.1: Dataset summary table

Metric	Mean	Standard deviation	Minimum	Maximum
CER	0.008	0.009	0.000	0.026
WER	0.025	0.026	0.000	0.063
FER	0.037	0.040	0.000	0.107
Precision	0.962	0.040	0.892	1.000
Recall	0.962	0.040	0.892	1.000
F1 score	0.962	0.040	0.892	1.000

A total of 16 images were uploaded to the application. The results indicate that the text extraction accuracy is generally high for the forms used, with a mean CER of **0.008**, which means that only **0.8%** of the characters were incorrectly extracted on average in a form. Since WER and FER have lower granularity, an incorrect character leads to higher error rates, even

if CER is low. The mean WER was **0.025** (2.5%), while the mean FER was **0.037** (3.7%), reaching a maximum of **0.063** (6.3%) and **0.107** (10.7%) for certain forms, respectively.

The Precision, Recall and F1 score presented a similar standard deviation to the FER (**0.040**), which could be expected since they were all measured at the field level. The lack of fields where the system failed to extract can explain the parity of the field-level metrics, as it leads to the same value of False Positives and False Negatives, and, consequently, the same values of Precision, Recall and F1 score, as per their formula. A minimum value of **0.892** (89.2%) was achieved for certain forms.

To further analyse the results, the data was grouped by the method of capturing the form, language and handwriting style, with the mean and standard deviation being presented in Table 6.2.

Table 6.2: Mean & Standard deviation by upload type, language and handwriting style

	Upload type		Language		Handwriting style	
Metric	**Camera**	**Scanned**	**English**	**Portuguese**	**Print**	**Cursive**
CER	0.011 ± 0.011	0.006 ± 0.007	0.001 ± 0.002	0.015 ± 0.009	0.003 ± 0.008	0.011 ± 0.010
WER	0.030 ± 0.028	0.021 ± 0.027	0.006 ± 0.011	0.046 ± 0.022	0.009 ± 0.021	0.036 ± 0.025
FER	0.045 ± 0.042	0.031 ± 0.040	0.009 ± 0.017	0.067 ± 0.035	0.012 ± 0.029	0.054 ± 0.039
Precision	0.955 ± 0.042	0.969 ± 0.040	0.991 ± 0.017	0.933 ± 0.035	0.988 ± 0.029	0.946 ± 0.039
Recall	0.955 ± 0.042	0.969 ± 0.040	0.991 ± 0.017	0.933 ± 0.035	0.988 ± 0.029	0.946 ± 0.039
F1 score	0.955 ± 0.042	0.969 ± 0.040	0.991 ± 0.017	0.933 ± 0.035	0.988 ± 0.029	0.946 ± 0.039

For the method of capturing the form, the scanned images presented slightly better mean values for all the metrics compared to the pictures taken inside the application, with lower mean error rates (0.6% CER, 2.1% WER and 3.1% FER) and higher field-level performance metrics (96.9% Precision, Recall and F1 score) compared to the values achieved when using the camera (1.1% CER, 3% WER, 4.5% FER and 95.5% Precision, Recall and F1 score). However, the standard deviations indicate that there can be some overlap in the values of the two upload types.

Regarding the language used to fill the forms, the difference between the mean values was higher, with the forms filled in English achieving better results (0.1% CER, 0.6% WER, 0.9% FER and 99.1% Precision, Recall and F1 score) than the forms filled in Portuguese (1.5% CER, 4.6% WER, 6.7% FER and 93.3% Precision, Recall and F1 score). In this case, the standard deviations indicate that there is little to no overlap in the values of the two languages.

For the handwriting style, the difference between the mean values was smaller than for the language, but the standard deviations indicate that there is small overlap in the values of the two styles. The forms filled with print handwriting style achieved better results (0.3% CER, 0.9% WER, 1.2% FER and 98.8% Precision, Recall and F1 score) compared to the forms filled with cursive handwriting style (1.1% CER, 3.6% WER, 5.4% FER and 94.6% Precision, Recall and F1 score).

To better visualize the distribution of the metrics per upload type, the box plot in Figure 6.1 was created.

Looking at the box plot, it is possible to notice that the values partially overlaps for both upload types, with a slight tendency for the scanned images to achieve better results, which is in line with the previous analysis based on the mean and standard deviation.

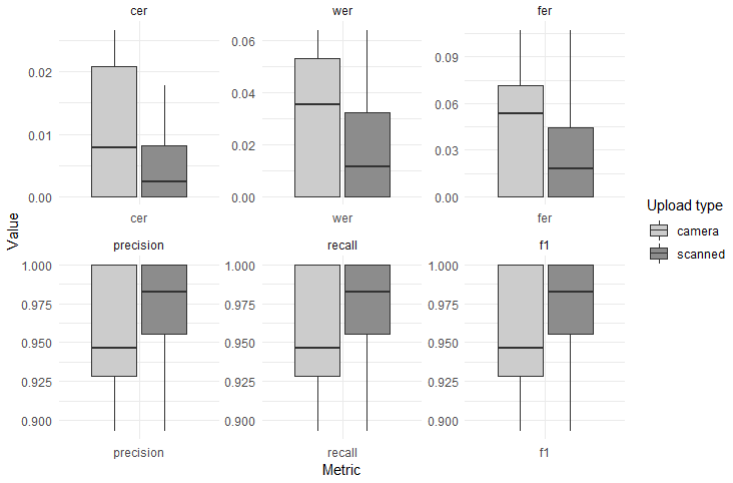

Figure 6.1: Box plot of the metrics grouped by upload type

For the error rates, the same first quartile was achieved for both upload types, while the same third quartile was achieved for the Precision, Recall and F1 score, indicating that part of the values were similar for both distributions. However, the opposite quartiles and median values were slightly better for the scanned images.

The minimum and maximum values were similar for all the metrics, except for the maximum CER, which was lower for the scanned images than the pictures taken with the camera in the application.

In order to determine if the differences in the metrics for the different groups are statistically significant, statistical tests were performed to compare the distributions for each group. Given the small sample size and the non-normal distribution of the data, non-parametric tests were used. The non-normality of the data can be verified by the tendency of the data to be closer to the minimum value for the error rates or the maximum for the Precision, Recall and F1 score.

For the upload type, the null hypothesis states that there is no difference in the metrics for the scanned images and the pictures taken with the camera. The alternative hypothesis states that the error rates are lower for the scanned images, while the field-level performance metrics are higher compared to the camera images. Since the two distributions are paired, the skewness coefficient for their difference was calculated to determine the appropriate statistical test to use.

The Table 6.3 includes the skewness coefficient and the results of the statistical tests for the differences of each metric for the two upload types.

Table 6.3: Statistical tests for differences in the metrics for Scanned vs Camera upload types

	CER	**WER**	**FER**	**Precision**	**Recall**	**F1 score**
Skewness	-1.132	-1.627	-1.564	1.564	1.564	1.564
p-value	0.0625*	0.25*	0.25*	0.25*	0.25*	0.25*

*Sign test

Since the skewness coefficient for all the metrics indicates that the differences are not symmetric, the Sign test was used to compare the distributions. Considering a significance level of 0.05, the null hypothesis cannot be rejected for any of the metrics, which means that there is not enough evidence to conclude that the scanned images allow for significantly better results than taking the picture with the camera inside the application.

CER was the metric that came closest to rejecting the null hypothesis, with a p-value of **0.0625**, but it was not enough to be considered statistically significant.

For the language, the null hypothesis states that there is no difference in the metrics for the forms filled in English and Portuguese. The alternative hypothesis states that the error rates are lower and the field-level performance metrics are higher for the forms filled in English. Since the two distributions are independent, the Mann-Whitney U test was used to compare them. The results of the tests can be found in Table 6.4.

Table 6.4: Statistical tests for differences in the metrics for English vs Portuguese filled forms

	CER	**WER**	**FER**	**Precision**	**Recall**	**F1 score**
p-value	0.0018*	0.0017*	0.0023*	0.0023*	0.0023*	0.0023*

*Mann-Whitney U test

Considering a significance level of 0.05, the null hypothesis can be rejected for all the metrics, which means that there is enough evidence to conclude that the forms filled in English had significantly better results than the forms filled in Portuguese.

Regarding the handwriting style, the null hypothesis states that there is parity in the metrics for the forms filled with print and cursive handwriting styles. As for the alternative hypothesis, it states that the error rates are lower for the forms filled with print handwriting style, while the Precision, Recall and F1 score are higher compared to the forms filled with cursive handwriting style. As the distributions are independent, the Mann-Whitney U test was also used in this case. The Table 6.5 includes the results of the statistical tests.

Table 6.5: Statistical tests for differences in the metrics for Print vs Cursive handwriting styles

	CER	**WER**	**FER**	**Precision**	**Recall**	**F1 score**
p-value	0.0307*	0.0178*	0.0195*	0.0195*	0.0195*	0.0195*

*Mann-Whitney U test

As the p-value is lower than the significance level of 0.05 for all the metrics, the null hypothesis can be rejected, making it possible to conclude that the forms filled with print handwriting style had significantly better results than the forms filled with cursive handwriting style.

Besides the grouped analysis, the metrics were also analysed in the context of the whole dataset in order to understand their relationship.

Firstly, the correlation matrix between the metrics was calculated using Kendall rank correlation coefficient, as the dataset is small and does not follow a normal distribution. The heatmap in Figure 6.2 was created to visualize the correlation matrix. Since the matrix is symmetric, only the lower triangle is shown, to avoid redundancy.

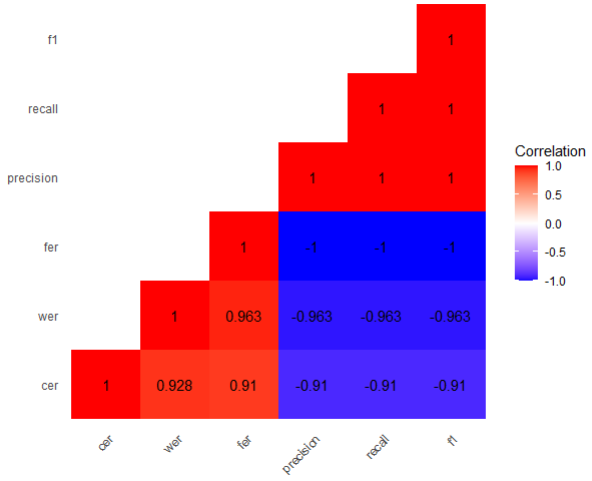

Figure 6.2: Heatmap of the correlation matrix for the metrics

From the heatmap, it is possible to notice that there is a strong correlation between all the metrics, with the absolute values of the correlation coefficients being higher than 0.9 in all the cases.

As expected, the error rates are negatively correlated with the Precision, Recall and F1 score, since a higher number of errors leads to lower accuracy in the field extraction. The correlation is particularly strong between FER and the Precision, Recall and F1 score, with a coefficient of **-1**, due to the fact that they are all measured at the field level. For the same reason, their absolute correlation coefficients with CER and WER are the same, with values of **0.91** and **0.963**, respectively.

The Precision, Recall and F1 score have a perfect positive correlation with each other, which can be explained by the lack of fields where the system failed to extract, leading to the same values of False Positives and False Negatives, and, consequently, the same values for these three metrics.

Besides the correlation analysis, the error rates were compared to each other to check if they have significant differences. The null hypothesis states that there is no difference between the error rates, while the alternative hypothesis states that there is a difference between at least two of them. Given the non-normal distribution of the data and the fact that the distributions are paired, the Friedman test was used. A p-value of **0.0001** was achieved, which is lower than the significance level of 0.05. This means that the null hypothesis can be rejected, indicating that there is enough evidence to conclude that there is a difference between at least two of the error rates.

In order to determine which error rates are different from each other, a *post hoc* analysis was performed. For each pair of error rates, the null hypothesis states that there is parity between them, while the alternative hypothesis states that there is a difference between them. The skewness coefficient for their difference was calculated to determine the appropriate non-parametric test to use. The Table 6.6 includes the results of the *post hoc* analysis.

For the pairs CER & WER and CER & FER, the skewness coefficient indicates that the differences are moderately symmetic, so the Wilcoxon signed-rank test was used to compare the distributions. For the pair WER & FER, the skewness coefficient indicates that the differences are not symmetric, so the Sign test was used instead.

Table 6.6: *Post hoc* analysis for differences between error rates

	CER & WER	CER & FER	WER & FER
Skewness	-0.241	-0.488	-1.093
p-value	0.0091*	0.0091*	0.0039**

*Wilcoxon signed-rank test; **Sign test

Considering a significance level of 0.05, the null hypothesis can be rejected for all the pairs of error rates, which means that there is enough evidence to conclude that there is a difference between all the error rates.

Even though some factors were found to have a significant impact on the accuracy of the text extraction, such as the language and handwriting style, the overall results indicate that the developed solution is capable of accurately extracting text from the filled forms, with low error rates and high field-level performance metrics. However, there are other confounders that could affect the accuracy, such as the camera quality, photo focus problems, handwriting quality, complexity of the form or lighting conditions.

6.2 User testing

In this section, the methodology and results of the user testing session will be presented to evaluate the usability and efficiency of the application in real-world scenarios.

6.2.1 Methodology

The user testing session was conducted in a partnership with The MENTOR Initiative, with the objective of gathering feedback from the target users and evaluate the extraction of fields. Two persons from the NGO were involved in the testing session by accessing the application distributed through Firebase App Distribution.

Before starting the testing, a demonstration meeting was held to showcase the application and its features, as well as to provide instructions on how to use it. The access to the application and the test accounts was provided after the meeting.

The tests were performed during the first week of August 2025, with the testers using the application in their own time and environment. After the testing period, a meeting was held to gather direct feedback from the testers regarding their experience with the application, as well as any issues or suggestions they had.

Official forms from one of the partners of The MENTOR Initiative were used during the testing session, which will not be disclosed due to confidentiality reasons. There was no authorization from the partner to use real filled forms, so the testers filled the forms with fake data for the testing session.

To monitor the usage of the application and track the results of some actions, Firebase Analytics was integrated into the application. Three custom events were created to track successful form response upload and extraction, form response review and validation result creation.

One of the events is triggered when a form response is successfully uploaded and the text extraction is completed, according to Code Snippet 6.1.

```
data class FormExtractionAnalyticsEvent(...) : AnalyticsEvent {
    override fun eventName(): String = "form_extraction"
    override fun eventParams(): Bundle = Bundle().apply {
        putString("form_response_id", responseId)
        putString("form_id", formId)
        putString("form_name", formName)
        putInt("fields_total", totalFields)
        putInt("fields_extracted", extractedFields)
        putInt("fields_non_extracted", nonExtractedFields)
        putDouble("extraction_rate", extractionRate)
    }
}
```

Listing 6.1: Form extraction analytics event

A **form_extraction** event is tracked, with parameters that identify the form response in the system, as well as data about the field extraction. This includes the total number of fields in the form, the number of fields that were extracted, the number of fields that could not be extracted and a calculated rate of fields extracted. This rate does not represent a real accuracy metric, as the field can be extracted incorrectly or the field can be left empty, leading to the system not being able to extract it.

When the user successfully reviews the data of a form response, another event is triggered, according to Code Snippet 6.2.

```
data class ResponseReviewAnalyticsEvent(...) : AnalyticsEvent {
    override fun eventName(): String = "response_review"
    override fun eventParams(): Bundle = Bundle().apply {
        putString("form_response_id", responseId)
        putString("form_id", formId)
        putString("form_name", formName)
        putInt("fields_total", totalFields)
        putInt("fields_reviewed", reviewedFields)
        putInt("fields_non_reviewed", nonReviewedFields)
        putDouble("review_rate", reviewRate)
    }
}
```

Listing 6.2: Form response review analytics event

An event with the name **response_review** is tracked when the review request succeeds. Similarly to the previous event, the metadata of the form response is sent, as well as data about the review. In this case, the rate is calculated for the number of fields that were reviewed, compared to the total number of fields. During the review, the fields that were not extracted are automatically included, being defaulted to an empty value if the user does not fill them.

Lastly, an event is also triggered when a validation is applied, according to the specification in Code Snippet 6.3.

```
data class ValidationResultAnalyticsEvent(...) : AnalyticsEvent {
    override fun eventName(): String = "validation_result"
    override fun eventParams(): Bundle = Bundle().apply {
        putString("validation_id", validationId)
        putString("validation_name", validationName)
        putString("valid", isValid.toString())
    }
}
```

Listing 6.3: Validation result analytics event

The **validation_result** event is tracked after a validation is successfully applied, with parameters that identify the respective validation and the resulting state.

The analytics data will be combined with data collected through observation in the application and system, in order to analyse the results of the user testing session.

Even though the extraction and review rates are not accuracy metrics, it is possible to infer an indicator from them. Considering that the review from the tester is done correctly and it includes the fields that were not extracted, the rate of incorrectly extracted fields can be calculated as follows:

$$IncorrectExtractionRate = ReviewRate - (1 - ExtractionRate). \tag{6.8}$$

Since the review rate includes the rate of unextracted fields, by calculating their difference, it is possible to estimate the rate of fields that were extracted incorrectly. The rate of unextracted fields can be achieved by calculating the complement of the extraction rate.

The incorrect extraction rate can be considered an indicator of the accuracy of the field extraction, as it represents the proportion of fields that were extracted but were not considered correct during the review. However, it is not a precise metric, as it does not account for the fields that were not extracted and were filled in the original form, which would also be considered an error in the extraction.

Besides the analytics data, Firebase Crashlytics was also integrated into the application to monitor any unexpected errors or crashes that may occur during the testing phase.

6.2.2 Results

After performing the tests on the application, a meeting was held with the testers. The feedback gathered from them during the meeting was generally positive, with some annotations being shared about their experience using the application. Even though the solution is missing the capability to submit the form response directly to the health information system of the partner, the testers saw the potential of the application to improve their workflow and congratulated the work done in the current state of the solution.

One of the main points for improvement mentioned was the ordering of the fields in the details screen of a form response. They are not following the same order as in the form, which confused the testers and made it harder to review the extracted data.

Another improvement suggested was to allow the user to delete an uploaded form response, in case the wrong form was uploaded. During the testing, there was a form that was uploaded twice by mistake, which led to the suggestion.

The testers also mentioned that the numeric fields in the form response details screen are small, making it harder to read bigger field names. They suggested increasing the size of the numeric fields to have the same size as the text fields.

Regarding the text extraction, the testers mentioned that it was generally accurate, except for some cases where the extraction was not perfect. In some situations, the system was extracting fields that were not filled, with the values being the same as other fields that were filled nearby. In another situation, there was a selection field that was considered filled, even though it was not selected. Besides that, most of the filled data was extracted correctly.

Another point raised was about some fields that were being marked as mandatory in the application even though they were not. In this case, it is not an issue with the system itself, but rather with the configuration of the structure of the form, which can be updated using the Backoffice or the respective Backend endpoints.

As for the validations, the testers did not understand well how to apply them, as they were expecting them to be applied automatically when the form response was uploaded. This can be improved by adding an extra step to apply the validation during the flow of uploading a form response instead of having to go through the list of validations. The testers think that it would be more intuitive and would provide a better user experience.

During the tests, Firebase Analytics were tracking the custom events, which were complemented with direct observation of the application and system to obtain the dataset in Appendix C.

The extraction and review events were tracked, with their respective rates being calculated. These variables were summarized by calculating the mean, standard deviation, minimum, maximum and count values, as shown in Table 6.7.

Table 6.7: Summary table of extraction and review rates during user testing

	Mean	**Standard deviation**	**Minimum**	**Maximum**	**Count**
Extraction rate	0.828	0.242	0.308	1.000	10
Review rate	0.250	0.263	0.045	0.782	10
Incorrect extraction rate	0.079	0.064	0.000	0.227	10

A total of 10 form responses were uploaded and had their fields extracted during the testing session. The mean extraction rate was **0.828** (82.8%), which means that, on average, 17.2% of the fields could not be extracted from the uploaded forms. In some cases, all the fields were extracted, according to the maximum value of **100%**, while in other cases, only **30.8%** of the fields were extracted, according to the minimum value.

These values do not represent the accuracy of the system, as they do not account for the correctness of the extracted fields or the empty fields in the original form. After observing the uploaded forms, it was possible to verify that some of them had fields left empty, which led to lower extraction rates, specially in one of the forms that had a higher quantity of fields.

When a form response is reviewed, the unextracted fields are always considered, even if they are defaulted to an empty value. This means that the review rate takes into account the fields that were not extracted, as well as the fields that were extracted incorrectly. The mean value of **0.250** indicates that, on average, 25% of the fields were reviewed. In some cases, the review rate was higher, reaching a maximum of **78.8%**, which can partially be explained by the empty fields and, consequently, lower extraction rates in those cases.

The calculated incorrect extraction rate had a mean value of **0.079** (7.9%), with at least one form having no incorrectly extracted fields, according to the minimum value of **0%**, while a maximum of **22.7%** was achieved. The standard deviation of **0.064** reflects the variability of the values, which could have been caused by the complexity of the forms or the empty fields in the uploaded forms.

For the validation events, the results were summarized by counting the occurrences in which a form response field was marked as valid or invalid, resulting in the pie chart in Figure 6.3.

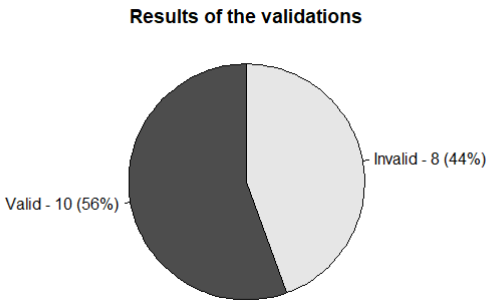

Figure 6.3: Pie chart of the validation results during user testing

A total of 18 validations were applied during the testing session, with **10 (56%)** of them returning a valid result and **8 (44%)** returning an invalid result. Some forms were purposely filled with incorrect data to test the extraction and the validations in error scenarios.

During the testing session, no crashes or unexpected errors were detected by Firebase Crashlytics, indicating that the application was stable and was gracefully handling any errors that may have occurred.

6.3 Summary

As an answer to RQ7, text extraction accuracy tests and user testing were conducted to evaluate the accuracy and quality of the developed tool.

In the text extraction tests, mean values of 0.8%, 2.5% and 3.7% were obtained for the CER, WER and FER, respectively, while a mean value of 96.2% was achieved for the Precision, Recall and F1 score. On the one hand, it was not possible to conclude that scanned images had better accuracy than images taken with the camera inside the application. On the other hand, it was possible to conclude that forms filled in English had better accuracy than forms filled in Portuguese, as well as the forms filled with print handwriting style, which achieved better accuracy compared to the forms filled with cursive handwriting style. All the metrics are strongly correlated, even if the error rates have significant differences between them.

As part of the user testing, two testers from the NGO experimented with the application using official forms filled with fake data. The feedback was positive, with some improvements being suggested, such as the order of the fields in the form response details screen and the size of the numeric fields. Mean values of 82.8% and 25% were obtained for the extraction and review rates, respectively, which were influenced by the number of empty fields in the uploaded forms. No crashes or unexpected errors were detected during the testing session.

Chapter 7

Conclusions

In this chapter, the dissertation will be summarized, considering the objectives that were defined. The challenges and future work will also be identified, finishing with some final considerations.

7.1 Document summary

This dissertation aimed at finding a solution for the paper-based mass drug administration process, in which the forms filled in the communities are sent to the headquarters and manually entered in the health information systems. This makes the process inefficient and error-prone.

To understand how the problem of the extraction of data could be solved, the OCR technology was explored. It can be used to extract text from different types of document, such as printed and handwritten documents, and usually includes several phases, such as Pre-processing, Segmentation, Feature Extraction, Classification and Post-processing. It can be used in multiple areas, such as document digitization, archiving and vehicle plate recognition.

By reviewing the literature, it was possible to find multiple case studies that used OCR to recognize handwritten text accurately, concluding that its usage is feasible. However, in some cases, custom ML models or pre-processing and post-correction techniques were used to compensate shortcomings of the OCR engines. Regarding the OCR tools, the comparative studies suggest that cloud services perform better compared to proprietary and open source software, with AWS Textract and Google Document AI being the better cloud services in the market. Some studies in the literature also suggest that using methods such as Binarization, Skew correction, Noise Removal and Sharpening can improve the OCR accuracy. However, there are some cases in which these techniques can degrade the quality of the original image.

The problem was analysed and mapped to a domain model, in order to raise the functional and non-functional requirements. Based on them, the architecture of the solution was designed in different levels of granularity, resulting in a system composed of a Backend, an Android Application and a Backoffice web application. The development process was also defined, including the branching model and CI/CD.

The Backend was built with NodeJS, using the framework Express with the programming language TypeScript. The functionalities were documented using Swagger, following the OpenAPI specification. Regarding the Android Application, it was built natively using the programming language Kotlin, with the UI framework Jetpack Compose. As for the Backoffice, it was implemented by students of Mindera School, using React with the programming language TypeScript.

To evaluate the solution, two experiments were conducted using the application. A text extraction test session was performed using a fake example test form with the different types of fields. 8 forms were filled with fake data and uploaded to the system using both the camera and scanned images of the forms. Different metrics where calculated for each response, resulting in mean values of 0.8%, 2.5% and 3.7% for the CER, WER and FER, while a mean value of 96.2% was achieved for the Precision, Recall and F1 score. After performing statistical tests, it was possible to conclude that forms filled in English had better accuracy than forms filled in Portuguese, and forms filled using print handwriting style had better results than forms filled with cursive handwriting style. On the other hand, it was not possible to conclude that using scanned images resulted in better results than using the camera inside the application. All the metrics were strongly correlated, and the error rates had significant differences between them.

The other experiment performed was a user testing session, in a partnership with the NGO. Official forms were filled by two testers using fake data, after a demonstration meeting. The feedback from the testers was positive, with some improvements being suggested and some points being raised, such as the order of the fields in the form response details screen, the size of the numeric fields and a capability to delete uploaded form responses. Firebase Analytics was integrated in the application to track some metrics, such as the extraction rate, review rate and validation results. For the extraction and review rate, mean values of 82.8% and 25% were obtained, respectively. The quantity of empty fields in the filled forms might have impacted the results. From the previous rates, it was possible to measure the rate of incorrectly extracted fields, achieving a mean value of 7.9%. During the test session, 56% of the validations returned success, catching some cases were the forms were filled with wrong values purposely. Firebase Crashlytics was also integrated in the application, with no crashes or unexpected errors being detected during the tests.

7.2 Achieved objectives

The main goal of the project was to develop a tool that supports the digitization of mass drug administration forms and automation of the data submission for different countries and NGOs. To achieve that, several specific objectives were specified and followed during the project. The Table 7.1 presents the objectives and their respective status at the end of the dissertation.

Table 7.1: Achieved objectives

Id.	Objective	Status
1	Identify the most suitable OCR tool for digitization of forms	Done
2	Develop an OCR Engine backend that processes the images and stores the data, so that it can be validated and submitted	Partially done
3	Develop a backoffice that connects to the backend and allows the form structure preparation, as well as user management	Delegated
4	Develop the frontend as an Android application to take pictures of the forms and review the data before submission	Partially done
5	Apply multi-tenancy to the project, in order to support multiple NGOs and countries	Done
6	Evaluate the effectiveness of the digitization process with real-world test trial	Done

The first objective was completed by reviewing the literature in the state of the art of Chapter 3. The objectives of developing the Backend and the Android Application were mostly completed, with their design and implementation being documented in Chapter 4 and Chapter 5, respectively. However, the functionality of submitting the form response to the external health information systems was not implemented due to the lack of time. Besides that, even though the Backoffice was designed by the author of the dissertation in Chapter 4, the implementation was delegated to the students of Mindera School.

Regarding the fifth objective, it was completed by designing a generic solution that allows structuring the forms with different types of fields, instead of supporting only the forms that the NGO needed. The Android Application can be adapted to different NGOs by refactoring the metadata and changing the icon and theme, even though it could be improved to be easier to apply the multi-tenancy.

The last objective was achieved by conducting a user testing session and text extraction tests, as specified in Chapter 6.

By achieving the objectives, it was also possible to answer the research questions of the dissertation. In Chapter 2, the RQ1 was answered with a description of the historical developments and fundamental concepts of OCR, as well as its applications, challenges and limitations. RQ2, RQ3 and RQ4 were answered in Chapter 3 with a literature review that allowed concluding that a solution that uses OCR to extract handwritten text is feasible, AWS Textract is the OCR tool with the better accuracy, and multiple methods can be used to improve OCR accuracy, respectively. In Chapter 4, RQ5 was answered with the design of the architecture of the solution. The description of the implementation of the solution, in Chapter 5, allowed answering RQ6. Finally, RQ7 was answered in Chapter 6 with two experiments being conducted to evaluate the accuracy and quality of the developed solution.

7.3 Challenges and future work

The project turned out to be ambitious for the timeframe of the dissertation, leading to the functionality of submitting the form response not being implemented. Even though it was not a direct consequence, the delegation of the Backoffice implementation helped in easing the work to be implemented, even if it was a part of the project that had less impact and importance, as the functionalities could be accessed directly on the Backend. The lack of time associated with the complexity of the project, was revealed to be the biggest challenge in the dissertation.

Another challenge was the lack of authorization from the governmental partners of The MENTOR Initiative to use real data in the user testing. Because of that, fake data was used, even though the official forms were used in the tests.

The integration of the developed solution with the external health information systems for submitting the form responses is the main piece of future work that is needed on the project. With that functionality, the solution can be used by the NGO to improve the efficiency of their mass drug administration processes and data handling.

The points suggested by the testers can also be considered for future developments, such as the order of the fields in the form response details screen, the capability to delete form responses, the size of the numeric fields in the form response details screen and the capability to apply the validations automatically after uploading a filled form.

Besides that, other minor functionalities can also be implemented, such as search capabilities with filtering, listing recently viewed forms and offline mode for uploading a form response when the connection is reestablished. The registration flow can also be improved to automatically send an email with the credentials instead of relying on the administrator.

7.4 Final appreciation

In conclusion, even though the objectives were not completely achieved, the development of the project went well and the proposed solution showed the potential that the system has for improving the efficiency of mass drug administration monitoring processes in the NGOs and their partners. The research questions were also answered through the document.

This dissertation represents a significant milestone for the author, allowing both personal and professional growth, with the development of a solution in different areas, as well as the opportunity to mentor and supervise Mindera School students. The impact that this project can have on several lives was also a great motivator to embark on this journey.

Bibliography

[1] "The mentor initiative," Accessed: Nov. 6, 2024. [Online]. Available: https://mentor-initiative.org/.

[2] "Our vision - the mentor initiative," Accessed: Nov. 9, 2024. [Online]. Available: https://mentor-initiative.org/our-vision/.

[3] "The mentor initiative's 22nd anniversary! - the mentor initiative," Accessed: Dec. 23, 2024. [Online]. Available: https://mentor-initiative.org/the-mentor-initiatives-22nd-anniversary/.

[4] "Mass drug administration - the mentor initiative," Accessed: Feb. 14, 2025. [Online]. Available: https://mentor-initiative.org/activity/neglected-tropical-diseases/mass-drug-administration/.

[5] J. James and M. Versteeg, "Mobile phones in africa: How much do we really know?" *Social Indicators Research*, vol. 84, pp. 117–126, 1 Oct. 2007, issn: 03038300. doi: 10.1007/S11205-006-9079-X/TABLES/1.

[6] T. Hegghammer, "Ocr with tesseract, amazon textract, and google document ai: A benchmarking experiment," *Journal of Computational Social Science*, vol. 5, pp. 861–882, 1 May 2022, issn: 24322725. doi: 10.1007/S42001-021-00149-1/FIGURES/10.

[7] S. Bawa, "Autonomy and policy independence in africa: A review of ngo development challenges," *Development in Practice*, vol. 23, pp. 526–536, 4 2013, issn: 09614524. doi: 10.1080/09614524.2013.790935.

[8] "Regulamento do código de boas práticas e de conduta do instituto politécnico do porto," Accessed: Nov. 17, 2024. [Online]. Available: https://diariodarepublica.pt/dr/detalhe/despacho/11171-2020-148327696.

[9] D. Gotterbarn, K. Miller, and S. Rogerson, "Software engineering code of ethics," *Communications of the ACM*, vol. 40, 11 1997, issn: 00010782. doi: 10.1145/265684.265699.

[10] "General data protection regulation (gdpr) – legal text," Accessed: Feb. 15, 2025. [Online]. Available: https://gdpr-info.eu/.

[11] "Lei 22/11 angola - lei de protecção de dados pessoais," Accessed: Feb. 15, 2025. [Online]. Available: https://apd.ao/fotos/frontend_1/editor2/110617_lei_22-11_de_17_junho-proteccao_dados_pessoais.pdf.

[12] "World association of non-governmental organizations - code of ethics and conduct for ngos," Accessed: Nov. 17, 2024. [Online]. Available: https://www.wango.org/codeofethics.aspx.

[13] B. J. Oates, M. Griffiths, and R. McLean, *Researching information systems and computing*, 2nd ed. Sage, 2022, isbn: 978-1-5297-3268-9.

[14] R. B. Wakode, L. P. Raut, and P. Talmale, "Overview on kanban methodology and its implementation," *IJSRD-International Journal for Scientific Research & Development*, vol. 3, 2015.

[15] A. Robertson, "Optical character recognition: A study of success and failure in innovation," *Management Decision*, vol. 9, pp. 213–223, 1971. doi: 10.1108/EB000971.

[16] D. Berchmans and S. S. Kumar, "Optical character recognition: An overview and an insight," *2014 International Conference on Control, Instrumentation, Communication and Computational Technologies, ICCICCT 2014*, pp. 1361–1365, Dec. 2014. doi: 10.1109/ICCICCT.2014.6993174.

[17] M. R. Dubey, "Machine learning in the field of optical character recognition (ocr)," *International Journal of Trend in Scientific Research and Development (IJTSRD)*, vol. 4, 2020.

[18] J. Wang, "A study of the ocr development history and directions of development," *Highlights in Science, Engineering and Technology*, vol. 72, pp. 409–415, Dec. 2023, issn: 2791-0210. doi: 10.54097/BM665J77.

[19] S. Mori, C. Y. Suen, and K. Yamamoto, "Historical review of ocr research and development," *Proceedings of the IEEE*, vol. 80, pp. 1029–1058, 7 1992, issn: 15582256. doi: 10.1109/5.156468.

[20] J. Memon, M. Sami, and R. A. Khan, "Handwritten optical character recognition (ocr): A comprehensive systematic literature review (slr)," *IEEE Access*, vol. 8, pp. 142 642–142 668, 2020. doi: 10.1109/ACCESS.2020.3012542.

[21] J. D. Thanki, D. Davda, and P. Swaminarayan, "A review on ocr technology," vol. 8, 2021, issn: 2349-5162.

[22] K. Karthick, K. B. Ravindrakumar, R. Francis, and S. Ilankannan, "Steps involved in text recognition and recent research in ocr; a study," *International Journal of Recent Technology and Engineering (IJRTE)*, vol. 8, pp. 2277–3878, 1 2019, issn: 2277-3878.

[23] A. Kaur, S. Baghla, and S. Kumar, "Study of various character segmentation techniques for handwritten off-line cursive words: A review," *International Journal of Advances in Science Engineering and Technology*, vol. 3, pp. 2321–9009, 3 2015, issn: 2321-9009.

[24] T.-T.-H. Nguyen, A. Jatowt, M. Coustaty, and A. Doucet, "Survey of post-ocr processing approaches," *ACM Computing Surveys (CSUR)*, vol. 54, pp. 1–37, 2021. doi: 10.1145/3453476.

[25] Gowrishankar, "Optical character recognition (ocr): A comprehensive review," *International Research Journal of Modernization in Engineering Technology and Science*, 2023. doi: 10.56726/irjmets43530.

[26] A. Singh, K. Bacchuwar, and A. Bhasin, "A survey of ocr applications," *International Journal of Machine Learning and Computing*, pp. 314–318, 2012. doi: 10.7763/IJMLC.2012.V2.137.

[27] S. Faizullah, M. S. Ayub, S. Hussain, and M. A. Khan, "A survey of ocr in arabic language: Applications, techniques, and challenges," *Applied Sciences*, 2023. doi: 10.3390/app13074584.

[28] J. Majumdar and R. Gupta, "An accuracy examination of ocr tools," *International Journal of Innovative Technology and Exploring Engineering*, 2019. doi: 10.35940/ijitee.i1102.0789s419.

[29] Z. Akhtar, J. W. Lee, M. A. Khan, M. Sharif, S. Khan, and N. Riaz, "Optical character recognition (ocr) using partial least square (pls) based feature reduction: An application to artificial intelligence for biometric identification," *J. Enterp. Inf. Manag.*, vol. 36, pp. 767–789, 2020. doi: 10.1108/jeim-02-2020-0076.

[30] P. Sharma, "Advancements in ocr: A deep learning algorithm for enhanced text recognition," *International Journal of Inventive Engineering and Sciences*, 2023. doi: 10.35940/ijies.f4263.0810823.

[31] E. al. Shaik Moinuddin Ahmed, "Handwritten ocr for indic scripts: A comprehensive overview of machine learning and deep learning techniques," *International Journal on*

Recent and Innovation Trends in Computing and Communication, 2023. doi: 10.17762/ijritcc.v11i9.9230.

[32] C.-W. Park, V. Palakonda, S. Yun, I.-M. Kim, and J.-M. Kang, "Ocr-diff: A two-stage deep learning framework for optical character recognition using diffusion model in industrial internet of things," *IEEE Internet of Things Journal*, vol. 11, pp. 25 997–26 000, 2024. doi: 10.1109/JIOT.2024.3390700.

[33] O. Krasynskyi and O. Markovets, "Possibilities of using ocr technologies from google for recognition and digitalization of archive documents," *Visnyk of Kharkiv State Academy of Culture*, 2024. doi: 10.31516/2410-5333.065.16.

[34] A. Sampaio, "Improving systematic mapping reviews," *ACM SIGSOFT Software Engineering Notes*, vol. 40, pp. 1–8, 6 Nov. 2015, issn: 0163-5948. doi: 10.1145/2830719.2830732.

[35] D. Moher *et al.*, "Preferred reporting items for systematic reviews and meta-analyses: The prisma statement," *Annals of Internal Medicine*, vol. 151, pp. 264–269, 4 Jan. 2009, issn: 15393704. doi: 10.7326/0003-4819-151-4-200908180-00135/SUPPL_FILE/PRISMA_FIGURE-S1.DOC.

[36] D. Zhong, S. Palaiahnakote, U. Pal, and Y. Lu, "Struck-out handwritten word detection and restoration for automatic descriptive answer evaluation," *Signal Processing: Image Communication*, vol. 130, p. 117 214, Jan. 2025, issn: 0923-5965. doi: 10.1016/J.IMAGE.2024.117214.

[37] P. K. Singh, R. Sarkar, A. Abraham, and M. Nasipuri, "A case study on handwritten indic script classification: Benchmarking of the results at page, block, text-line, and word levels," *ACM Transactions on Asian and Low-Resource Language Information Processing*, vol. 21, 2 Mar. 2022, issn: 23754702. doi: 10.1145/3476102/SUPPL_FILE/SINGH.ZIP.

[38] D. Soselia, M. Tsintsadze, L. Shugliashvili, I. Koberidze, S. Amashukeli, and S. Jijavadze, "On georgian handwritten character recognition," *IFAC-PapersOnLine*, vol. 51, pp. 161–165, 30 Jan. 2018, issn: 2405-8963. doi: 10.1016/J.IFACOL.2018.11.279.

[39] C. Reul, S. Göttel, U. Springmann, C. Wick, K. M. Würzner, and F. Puppe, "Automatic semantic text tagging on historical lexica by combining ocr and typography classification a case study on daniel sander's wörterbuch der deutschen sprache," *ACM International Conference Proceeding Series*, pp. 33–38, May 2019. doi: 10.1145/3322905.3322910.

[40] PantDevesh *et al.*, "Robust ocr pipeline for automated digitization of mother and child protection cards in india," *ACM Journal on Computing and Sustainable Societies*, vol. 1, pp. 1–24, 1 Sep. 2023. doi: 10.1145/3608114.

[41] A. J. Wecker *et al.*, "Tikkoun sofrim: Making ancient manuscripts digitally accessible: The case of midrash tanhuma," *ACM Journal on Computing and Cultural Heritage (JOCCH)*, vol. 15, 2 Apr. 2022, issn: 15564711. doi: 10.1145/3476776.

[42] M. Koolen *et al.*, "The value of preexisting structures for digital access: Modelling the resolutions of the dutch states general," *Journal on Computing and Cultural Heritage*, vol. 16, pp. 2019–2024, 1 Jun. 2023, issn: 15564711. doi: 10.1145/3575864/ASSET/D262D520-E752-4D06-8012-329CCFB74EEF/ASSETS/GRAPHIC/JOCCH-21-0143-F08.JPG.

[43] A. Chiatti *et al.*, "Text extraction and retrieval from smartphone screenshots: Building a repository for life in media," *Proceedings of the ACM Symposium on Applied Computing*, vol. 8, pp. 948–955, Apr. 2018. doi: 10.1145/3167132.3167236.

[44] N. H. Imam, V. G. Vassilakis, and D. Kolovos, "Ocr post-correction for detecting adversarial text images," *Journal of Information Security and Applications*, vol. 66, p. 103 170, May 2022, issn: 2214-2126. doi: 10.1016/J.JISA.2022.103170.

[45] C. M. Rosca, A. Stancu, and A. V. Ariciu, "Algorithm for child adoption process using artificial intelligence and monitoring system for children," *Internet of Things*, vol. 26, p. 101 170, Jul. 2024, issn: 2542-6605. doi: 10.1016/J.IOT.2024.101170.

[46] M. J. C. Samonte, A. M. L. Bejar, H. C. L. Bien, and A. M. D. Cruz, "Senior citizen social pension management system using optical character recognition," *ICTC 2019 - 10th International Conference on ICT Convergence: ICT Convergence Leading the Autonomous Future*, pp. 456–460, Oct. 2019. doi: 10.1109/ICTC46691.2019.8940013.

[47] A. T. Shaju, J. T. Jo, E. S. Abraham, K. G. Vishnuprasad, V. Deepa, and J. Mathew, "Document digitization using optical character recognition - a case study of mark entry automation in educational institutions," *2024 1st International Conference on Trends in Engineering Systems and Technologies, ICTEST 2024*, 2024. doi: 10.1109/ICTEST60614.2024.10576116.

[48] M. S. Islam, M. K. B. Doumbouya, C. D. Manning, and C. Piech, "Handwritten code recognition for pen-and-paper cs education," *L@S 2024 - Proceedings of the 11th ACM Conference on Learning @ Scale*, pp. 200–210, Jul. 2024. doi: 10.1145/3657604.3662027.

[49] P. Jain, K. Taneja, and H. Taneja, "Which ocr toolset is good and why : A comparative study," *Kuwait Journal of Science*, vol. 48, pp. 1–12, 2 Apr. 2021, issn: 2307-4116. doi: 10.48129/KJS.V48I2.9589.

[50] A. Kumar, P. Singh, and K. Lata, "Comparative study of different optical character recognition models on handwritten and printed medical reports," *International Conference on Innovative Data Communication Technologies and Application, ICIDCA 2023 - Proceedings*, pp. 581–586, 2023. doi: 10.1109/ICIDCA56705.2023.10100213.

[51] G. S. Hukkeri, R. H. Goudar, P. Janagond, and P. S. Patil, "Machine learning in ocr technology: Performance analysis of different ocr methods for slide-to-text conversion in lecture videos," *International Journal of Advanced Computer Science and Applications*, vol. 13, 8 2022, issn: 21565570. doi: 10.14569/IJACSA.2022.0130839.

[52] U. Poudel, A. M. Regmi, Z. Stamenkovic, and S. P. Raja, "Applicability of ocr engines for text recognition in vehicle number plates, receipts and handwriting," *Journal of Circuits, Systems and Computers*, vol. 32, 18 Dec. 2023, issn: 0218-1266. doi: 10.1142/S0218126623503218.

[53] W. Ughetta and B. W. Kernighan, "The old bailey and ocr: Benchmarking aws, azure, and gcp with 180,000 page images," *Proceedings of the ACM Symposium on Document Engineering, DocEng 2020*, Sep. 2020. doi: 10.1145/3395027.3419595.

[54] A. Kumar and G. S. Lehal, "Performance comparison of tesseract and google document ai in punjabi newspapers digitization," *Library Progress International*, vol. 44, pp. 19 191–1931, 3 Sep. 2024, issn: 2320-317X.

[55] K. Olejniczak and M. Šulc, "Text detection forgot about document ocr," *CEUR Workshop Proceedings*, vol. 3349, Oct. 2022, issn: 16130073.

[56] R. Kleinhans, "Towards a framework for intelligent document image enhancement in pursuit of improved ocr performance," 2023.

[57] A. Ignasius, J. C. Chandra, R. Oscadinata, and D. Suhartono, "Image pre-processing effect on ocr's performance for image conversion to braille unicode," *Procedia Computer Science*, vol. 227, pp. 922–931, Jan. 2023, issn: 1877-0509. doi: 10.1016/J.PROCS.2023.10.599.

[58] V. A. Pham, D. T. K. Nguyen, M. D. Tran, and V. D. Pham, "Improved ocr quality for smart scanned document management system," *Journal of Science and Technique - Section on Information and Communication Technology*, vol. 9, 01 May 2020, issn: 1859-0209. doi: 10.56651/LQDTU.JST.V9.N01.60.ICT.

[59] I. Cetintas and A. A. Mungen, "Using pre-processing methods to improve ocr performances of digital historical documents," *Proceedings - 2021 Innovations in Intelligent Systems and Applications Conference, ASYU 2021*, 2021. doi: 10.1109/ASYU52992.2021.9598972.

[60] A. Tamboli, A. Kadam, A. Talegaonkar, A. Tekam, and B. Wakchaure, "Optimizing ocr performance: An investigation into image preprocessing techniques," *INTERANTIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING AND MANAGEMENT*, vol. 08, pp. 1–5, 04 Apr. 2024, issn: 25823930. doi: 10.55041/IJSREM31769.

[61] D. J. Barth, "Software quality in the design of reliable medical instrumentation," *Symposium Record - Policy Issues in Information and Communication Technologies in Medical Applications*, pp. 133–136, 1988. doi: 10.1109/ICTMA.1988.669600.

[62] A. Vazquez-Ingelmo, A. Garcia-Holgado, and F. J. Garcia-Penalvo, "C4 model in a software engineering subject to ease the comprehension of uml and the software," *IEEE Global Engineering Education Conference, EDUCON*, vol. 2020-April, pp. 919–924, Apr. 2020, issn: 21659567. doi: 10.1109/EDUCON45650.2020.9125335.

[63] P. B. Kruchten, "The 4+1 view model of architecture," *IEEE Software*, vol. 12, pp. 42–50, 6 1995, issn: 07407459. doi: 10.1109/52.469759.

[64] "Github flow," Accessed: Aug. 4, 2025. [Online]. Available: https://docs.github.com/en/get-started/using-github/github-flow.

[65] "Benchmarking performance analysis of optical character recognition techniques," *Proceedings of the IEEE International Multi Topic Conference, INMIC*, 2024 2024, issn: 28358864. doi: 10.1109/INMIC64792.2024.11004392.

[66] L. Yujian and L. Bo, "A normalized levenshtein distance metric," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, pp. 1091–1095, 6 Jun. 2007, issn: 01628828. doi: 10.1109/TPAMI.2007.1078.

[67] C. Goutte and E. Gaussier, "A probabilistic interpretation of precision, recall and f-score, with implication for evaluation," *Lecture Notes in Computer Science*, vol. 3408, pp. 345–359, 2005, issn: 1611-3349. doi: 10.1007/978-3-540-31865-1_25.

Appendix A

Example test form

The form in Figure A.1 was used to evaluate the text extraction accuracy.

Mass drug administration

Example test form

This is a sample form created solely for testing and demonstration purposes. It represents a standardized layout for monitoring Mass Drug Administration (MDA) activities. Data captured in this example is not real and should not be used for programmatic decisions. The form is intended to support testing of data extraction functionality in MDA-related applications.

Ref. code: _____ **Date** (DD/MM/YYYY): _____

Supervisor name: _____

District: _____ **Village:** _____

Disease targeted: ☐ Loiasis ☐ Trichuriasis

Drugs administered: ☐ Albendazole ☐ Ivermectin

Persons treated

Age group	Male	Female	Total
5-14 years			
15-49 years			
50+ years			
Global total			

Drug inventory

Drug name	Quantity received	Quantity used	Quantity lost
Albendazole			
Ivermectin			

Observations: _____

Figure A.1: Example test form

Appendix B

Text extraction tests dataset

The Table B.1 represents the dataset that was used to perform the statistical analysis of the text extraction accuracy.

Table B.1: Text extraction tests dataset

id	ref_code	upload_type	language	handwriting_style	cer	wer	fer	precision	recall	f1
1	ABC53D	scanned	English	Print	0.000	0.000	0.000	1.000	1.000	1.000
2	123AZ	scanned	Portuguese	Cursive	0.016	0.057	0.071	0.929	0.929	0.929
3	8A6HR5	scanned	English	Print	0.000	0.000	0.000	1.000	1.000	1.000
4	XEYNO-A	scanned	Portuguese	Cursive	0.005	0.024	0.036	0.964	0.964	0.964
5	ALG1-C	scanned	Portuguese	Print	0.000	0.000	0.000	1.000	1.000	1.000
6	A35BI11	scanned	English	Cursive	0.005	0.023	0.036	0.964	0.964	0.964
7	A4732YIJ	scanned	Portuguese	Cursive	0.018	0.064	0.107	0.893	0.893	0.893
8	5Q3AT3	scanned	English	Cursive	0.000	0.000	0.000	1.000	1.000	1.000
1	ABC53D	camera	English	Print	0.000	0.000	0.000	1.000	1.000	1.000
2	123AZ	camera	Portuguese	Cursive	0.024	0.057	0.071	0.929	0.929	0.929
3	8A6HR5	camera	English	Print	0.000	0.000	0.000	1.000	1.000	1.000
4	XEYNO-A	camera	Portuguese	Cursive	0.011	0.048	0.071	0.929	0.929	0.929
5	ALG1-C	camera	Portuguese	Print	0.020	0.051	0.071	0.929	0.929	0.929
6	A35BI11	camera	English	Cursive	0.005	0.023	0.036	0.964	0.964	0.964
7	A4732YIJ	camera	Portuguese	Cursive	0.027	0.064	0.107	0.893	0.893	0.893
8	5Q3AT3	camera	English	Cursive	0.000	0.000	0.000	1.000	1.000	1.000

Appendix C

User testing analytics dataset

The Table C.1 represents the dataset that was gathered for the extraction and response review analytics events. Each response review was associated to the extraction of the same form response, according to its identifier, which has been partially omitted. The rate of incorrectly extracted fields was calculated as well.

Table C.1: Extraction and review dataset from user testing

response_id	form_name	fields_total	fields_extracted	extraction_rate	fields_reviewed	review_rate	inc_extr_rate
40a125dd-...	Ficha No1 A	22	21	0.955	2	0.091	0.045
d8567aac-...	Ficha No1 A	22	22	1.000	5	0.227	0.227
89e74a35-...	Ficha No1 A	22	21	0.955	2	0.091	0.045
1f6168f4-...	Ficha No1 A	22	21	0.955	2	0.091	0.045
21e46f45-...	Ficha No1 A	22	21	0.955	1	0.045	0.000
fdee2b54-...	Ficha No2 B	78	35	0.449	54	0.692	0.141
cdbfaadf-...	Ficha No1 A	22	20	0.909	4	0.182	0.091
15c72d43-...	Ficha No1 A	22	20	0.909	3	0.136	0.045
5b3a8501-...	Ficha No1 B	18	16	0.889	3	0.167	0.056
02116c3e-...	Ficha No2 B	78	24	0.308	61	0.782	0.090

The validation results analytics events are summarized in Table C.2, which shows the total number of valid and invalid results, as well as their respective percentages.

Table C.2: Validation results summary from user testing

Validation result	Count	Percentage
Valid	10	55.6%
Invalid	8	44.4%