Form Management in Public Administration

Miguel Póvoas Figueira Verdelho

Thesis to obtain the Master of Science Degree in
Electrical and Computer Engineering

Supervisor(s): Prof. João Nuno de Oliveira e Silva
Lic. André Ricardo Gonçalves Valentim

Examination Committee
Chairperson: Prof. Teresa Maria Sá Ferreira Vazão Vasques
Supervisor: Prof. João Nuno de Oliveira e Silva
Member of the Committee: Prof. André Ferreira Ferrão Couto e Vasconcelos

September 2020
DECLARATION

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.
Acknowledgments

First, I want to thank my Supervisor, Professor João Silva for his acceptance of this topic and resilience when accepting the limitations of developing this work with the company.

Second, I would like to thank everyone at Mind for all the availability and generosity sharing their knowledge and experience, specially André Valentim for the help and guidance throughout all my work with the company but also Diogo Reis, David Gomes and Luís Ferreira as part of the development team where I felt extremely welcome during the execution of this work.
Abstract

Some Portuguese Municipalities currently provide online most forms and documentation their citizens might need to perform any type of requests. Managing all the documentation and its provisioning is executed manually, without any type of supporting tool or validation. The way this management is performed is problematic when updating and distributing hundreds of forms.

The system presented in this report aims to ease the delivery of sets of versioned documents through automation of tasks, a clear interface and error detection features. Its major benefits are the validation of human actions, automated control of versioned documents, data coherency when fetching up-to-date documentation and mostly the automation of document/packages delivery. This application provides some document management functionalities having the extra possibility of scheduling deliveries and gathering documents in sets to be automatically delivered.

The goal was to develop a Web tool with new workflows that could ease the process of managing versioned forms as well as their availability for the population. In order to evaluate the gains obtained from the use of this application, tests were performed with a diversified universe of users who executed the same actions with and without it, giving a clear feedback on the improvements brought from its use.

Keywords

Process Dematerialisation, Document Management, PDF Forms, Web Application, Urban Management
Resumo

Algumas Câmaras Municipais Portuguesas disponibilizam através da Internet os formulários necessários para que os seus cidadãos possam fazer qualquer tipo de pedido. A tarefa de gerir e disponibilizar toda a documentação é, atualmente, executada manualmente sem nenhum tipo de apoio ou validação digital, o que é problemático quando se tem de gerir e atualizar centenas de formulários.

O sistema apresentado neste relatório pretende facilitar a distribuição de conjuntos de documentos versionáveis através de automatização de fluxos, uma interface clara e detecção e prevenção de erros. Os seus maiores benefícios são a validação de ações humanas, o controlo automático de documentos versionáveis, coerência de informação aquando da utilização de documentos atualizados e acima de tudo a automatização da criação de pacotes de documentos para distribuição. Esta aplicação possui outras funcionalidades, nomeadamente a possibilidade de agendar a disponibilização de documentos e o agrupamento de documentos em pacotes de distribuição.

O objetivo foi desenvolver uma ferramenta Web com novos fluxos de trabalho que facilite o processo de gestão de documentos versionáveis e a sua distribuição à população. Para avaliar os ganhos obtidos pelo uso da ferramenta desenvolvida foram feitos testes com um universo diversificado de utilizadores que executaram o mesmo conjunto de ações dando uma ideia clara das melhorias que foram trazidas pelo uso da nova aplicação.

Palavras Chave

Desmaterialização de Processos, Gestão Documental, Formulários PDF, Gestão Urbanística
## Contents

<table>
<thead>
<tr>
<th>List of Figures</th>
<th>x</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Tables</td>
<td>xi</td>
</tr>
<tr>
<td>List of Acronyms</td>
<td>1</td>
</tr>
</tbody>
</table>

### 1 Introduction

1.1 Context .................................................. 3
1.2 Problem Description .......................................... 4
1.3 Objectives ................................................ 5
1.4 Proposed Solution ........................................ 5
1.5 Document Structure ........................................... 6

### 2 Related Work

2.1 Process Dematerialisation in Public Services .................. 8
2.2 Electronic Document Management Systems ...................... 9

### 3 ePaper - System Description

3.1 Basic Concepts of The ePaper System .......................... 15
3.2 Users and Use Cases ........................................ 16
3.3 Implementation Technologies ................................... 18
3.4 System’s Structure .......................................... 18
3.5 Relevant Functionalities’ Implementation .................. 19
    3.5.1 Form configuration ..................................... 20
    3.5.2 Form Update ........................................... 22
    3.5.3 Zip Package Providing ................................ 22
3.6 ePaper’s Limitations - Problem Analysis .................. 23
    3.6.1 Document Versioning ................................... 23
    3.6.2 Package Creation and Update ........................... 24
# eFormManager

## 4.1 Objectives

## 4.2 Solution Requirements

- **4.2.1 User Classes and Use Conditions**
- **4.2.2 Functional Requirements**
- **4.2.3 Non-Functional Requirements**

## 4.3 Use Cases

## 4.4 Architecture

## 4.5 Data Model

## 4.6 Implementation

- **4.6.1 Architectural Models**
- **4.6.2 Used Technologies**
- **4.6.3 Development Process**
- **4.6.4 Code Base**

## 4.7 User Interface

## 4.8 Document Configuration Workflow Details

## 4.9 REST Endpoints

- **4.9.1 Documents**
- **4.9.2 Request**
- **4.9.3 Request Groups**
- **4.9.4 Generic**

## 5 System Evaluation

## 5.1 Functional Requirements

- **5.1.1 System Administrators’ functionalities**
- **5.1.2 Citizens’ functionalities**

## 5.2 Performance Evaluation (N.F.R.1)

- **5.2.1 Performance Tests**
- **5.2.2 Performance Metrics**

## 5.3 Accessibility Evaluation (N.F.R.2)

## 5.4 Usability Evaluation (N.F.R.3)

- **5.4.1 Testing Set Up**
- **5.4.2 Results and Analysis**

## 5.5 Openness Evaluation (N.F.R.4)
### 6 Conclusions and Future Work

#### 6.1 Conclusions

#### 6.2 Future Work

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bibliography</td>
<td>64</td>
</tr>
<tr>
<td>A System Evaluation Guide</td>
<td>67</td>
</tr>
<tr>
<td>B Evaluation Survey</td>
<td>80</td>
</tr>
<tr>
<td>C Evaluation Results</td>
<td>87</td>
</tr>
</tbody>
</table>
List of Figures

3.1 High Level Architecture of ePaper .................................................. 14
3.2 ePaper’s Use Cases ........................................................................... 17
3.3 ePaper’s Architecture ......................................................................... 19
3.4 Form Configuration workflow diagram ................................................. 20
3.5 Form Configuration Interface on the Administration Menu ...................... 21
3.6 Request Configuration Menu Interface in the Administration Menu ....... 21
4.1 Solution’s Use Cases ........................................................................... 31
4.2 eFormManager Architecture ................................................................. 32
4.3 eFormManager’s Data Model ................................................................. 33
4.4 Web Page Sections .............................................................................. 38
4.5 Web Page Input Menu .......................................................................... 39
4.6 Setting up a document in eFormManager - Workflow Diagram ............. 40
5.1 Application’s Main Page Screenshot ...................................................... 44
5.2 FR1 - Document insertion ..................................................................... 45
5.3 FR2 - Request Creation ......................................................................... 46
5.4 FR2 - Editing Documents Associated to a Request .............................. 46
5.5 FR2 - Editing Documents Associated to a Request .............................. 47
5.6 Version Comparison Interface ............................................................... 48
5.7 Request Log History ........................................................................... 49
5.8 FR8 - Submission of an outdated document ......................................... 50
5.9 Download Requests’ documents and packages ..................................... 50
5.10 ASQ: Tasks’ score .............................................................................. 55
5.11 SUS scores ......................................................................................... 56
5.12 Time Taken for each Task ................................................................... 57
5.13 Types of errors occurred by application ............................................. 58
List of Tables

5.1 Google Lighthouse Metrics for eFormManager ........................................ 52
5.2 Number of errors in each task per system .................................................. 59
C.1 After Scenario Questionnaire for ePaper .................................................. 87
C.2 After Scenario Questionnaire for eFormManager ........................................ 88
C.3 System Usability Scale for ePaper .............................................................. 88
C.4 System Usability Scale for eFormManager .................................................. 88
# Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBMS</td>
<td>Database Management Systems</td>
</tr>
<tr>
<td>EDMS</td>
<td>Electronic Document Management Systems</td>
</tr>
<tr>
<td>ICT</td>
<td>Information and Communication Technology</td>
</tr>
<tr>
<td>IDE</td>
<td>Integrated Development Environment</td>
</tr>
<tr>
<td>IST</td>
<td>Instituto Superior Técnico</td>
</tr>
<tr>
<td>PDF</td>
<td>Adobe’s Portable Document Format</td>
</tr>
<tr>
<td>REST</td>
<td>Representational state transfer</td>
</tr>
<tr>
<td>WCAG</td>
<td>Web Content Accessibility Guidelines</td>
</tr>
<tr>
<td>ZIP</td>
<td>Archive file format for data compression</td>
</tr>
</tbody>
</table>
1

Introduction
Process's dematerialisation began revolutionising all types of businesses and industries in the past few years. Digital documents are now prevailing over physical files and big archive rooms, not only because of environmental concerns but mostly for all the advantages they bring to organisations: cheaper storage costs and automation of document management processes like archiving and searching, and document tracking.

This change in the paradigm opened a wide variety of possibilities as documents can be accessed by multiple people in several different locations simultaneously. People no longer need to wait in lines to fill paper forms and submit them in some physical help desk in a governmental institution, a bank or an insurance company who will keep documents in an archive next to thousands of other documents never to be looked at again.

Following a European Union's tendency, the Portuguese Government (2017) [1] issued a resolution in order to reduce public expense by cutting down paper usage costs and paper-related processes, investing in a modernisation of the public sector with a more sustainable and paper-free administration. Thus, software solutions with document management features keep increasing in popularity.

With these technologies, maintaining a system with vast quantities of digital documents becomes less complex. In spite of that, new functionalities such as multiple users accessing the same files simultaneously and document versioning/edition rise a different set of problems related with data coherency.

1.1 Context

Portuguese municipalities are in charge of their own urban management which means that whenever a citizen plans on having a construction, has any kind of tax related issue or needs licensing, an application is made to the Municipality. Nowadays these applications are done by the submission of a form. Usually, these requests are associated with different kinds of documents like forms, responsibility statements or certificates that need to be submitted for a successful application.

To submit these type of requests, citizens would have to go and pick up a paper document at the physical help desk to come back to hand it over filled and signed by diverse entities. This whole process used to take days or weeks but paper digitalisation allowed for a process dematerialization. Now-a-days, through the use of PDF Forms, supported by ubiquitous internet connection the process of fetching and submitting a document is now able to be executed from any where at the distance of a few clicks.

The ePaper solution [2], by Mind, provides municipalities with the possibility of performing all these tasks online. This system is currently used by some of the most populated Portuguese Municipalities and makes process dematerialisation possible by providing digital documentation needed to perform any type of request. It also provides a Web platform that allows the submission of those requests. ePaper's core functionalities ease and automate the Back Office processing that each request goes through, with
simplified and automated workflows: from the moment it is submitted until the point where an answer is
given back to the applicant after going through the process of being analysed by different entities (such
as architects and/or lawyers).

Some of those Municipalities provide more than a hundred request possibilities to citizens and re-
ceive hundreds of those requests per day. Each request needs to comply with a considerable amount of
legislation and is subject to adjustments any time the latter is modified and is composed by a multitude
of PDF documents and forms. On the other hand, the population demands and the services addressed
by Municipalities change throughout time contributing to more documentation alterations. Therefore, a
frequent change in requirements leads to multiple versions/iterations of each set of documents. The
focus of this Master Thesis will be on studying the management of all those versions’ coexistence in the
ePaper application and creating a solution to improve it.

1.2 Problem Description

The current version of ePaper has a public platform for citizens to download and submit forms that allows
limited configuration/maintenance from the municipality. The application is still not prepared to deal with
this document’s versioning and the following deployment. Specific problems arise:

1. Providing a new online form demands a complex sequence of tasks, including pasting files in
   specific folders mapped by the application, with an exact name that ePaper needs to recognise.

2. Municipality workers need to perform complex replication and copy of files between the different
   systems.

3. The moment of adding a new version and making it available to the public is a liable procedure
   that includes replacing the version with a the new file that must have the same name generating
   many human errors. This means that there is no mechanism to store the history and versions of
   the forms.

Even though there is a way to go around the fact that the application does not support versioned
documents (by executing manually part of the tasks needed), it is a very liable procedure that is asso-
ciated with a large amount of human errors. There are strict rules in what concerns file management
such as specific file paths and file names that users must follow on the replication process and it does
not exist any kind of control or verification to assure the actions taken are valid. Some clients have
multiple servers in their internal networks and it may happen that the public platform and the Back Office
application run in different servers (since Back Office may not be exposed to the Internet) and do not
share the same file storage. In practise, this means that the File System management procedures have
to be replicated in both servers manually and, because of human errors, it frequently leads to inconsistencies between servers. This process of managing all the files, their versioning and availability for download will be the problem addressed in this work. Tasks like manual creation of ZIP packages with dozens of documents, the update of those documents (i.e. its substitution for different files) alongside the scheduling of those updates have to be re-engineered.

1.3 Objectives

Given the contextualisation presented in 1.1 and the problem emphasised, this report aims to describe in full detail the study performed on this topic as well as all the decisions taken that led to the solution found. The work presented in this report aims to achieve the following objectives:

1. A problem analysis giving an insight on why the current implementation is problematic.
2. A clear specification of what the project requirements are, within the scope of ePaper, where it is meant to be integrated.
3. Construction of the web application that meets all the requirements.
4. Evaluation of the tool built and the gains obtained from its use.

Overall, the main goal of the work presented is to improve the management of versioned forms easing and simplifying the ePaper mechanism.

1.4 Proposed Solution

This report presents a possible solution to face the problem described in the ePaper platform. It consists on a tool that automates the majority of the document update related tasks as well as centralises the management and the storage of the forms that ePaper makes available to users (opposing to the management of multiple servers and storage performed before).

To achieve this centralisation, the tool is meant to be ran in the Back Office server (where it should have more logical connections) and be called through web services when the public platform receives a download request.

The new tool should give, as much as possible, some type of feedback and validation towards actions that users perform in order to help identifying mistakes right before they occur. This should be a major concern once it is known that the previous file management is associated with many human errors due to the complexity of its workflow. Besides, it should have workflows and interfaces that minimise the errors identified as frequent on ePaper.
The results presented in this report are extremely positive. Tests performed with real users using ePaper and the developed tool showed a significant improvement concerning the time users take to execute specific task and also verified a decreasing occurrence of human errors related with the execution of the tasks. Usability scores given by users are detailed in Chapter 5 and are also satisfying.

1.5 Document Structure

This report is structured in the following way: in Chapter 2 are presented concepts and studies done regarding the topics where this work fits, namely, Process Dematerialisation and Electronic Document Management Systems; in Chapter 3 the problem is described in detail, considering the application on top of which the whole work is developed and its problematic functionalities. There are also defined the requirements the solution needs to match in order to be considered successful; Chapter 4 covers in detail several aspects of the solution’s development from its architecture to its user interface. It also has a description of its functionalities relating them to the functional requirements they aim to match; taking in consideration all the requirements set to the project, Chapter 5 aims to verify the compliance of the solution with them. It presents the evaluation methods and the results obtained; Chapter 6 contains a general conclusion regarding all the work developed and also includes some guidelines for future work to be done and continue the product created so far.
Related Work
Within the context of this work, it is relevant to consider some concepts which are closely related with the topic at hands such as process dematerialisation and document management.

2.1 Process Dematerialisation in Public Services

Paper has been in use for information exchange for many centuries now. In the last two/three decades Information and Communication Technology allowed corporations and consumers to increasingly use technologies for many of the paper demanding tasks and digital files are overshadowing paper documents faster every day.

Public organisations are no exception in this evolution towards documents digitalisation in the services they provide. The public encounter, defined by Goodsell [3] as "the interaction of citizen and official as they communicate to conduct business" is suffering a revolution in some of its major aspects. The means and settings in which citizens communicate with municipalities change once it has became an online process. The actors involved change because requests can be automatically handled without a receiving actor. Besides, the initiation, duration and scope of these interactions also changed given the new paradigm of digital services [4].

Process dematerialisation, within this context, can be understood as the replacement of paper documentation for digital files as information carriers in order to eliminate the circulation of paper and its related tasks.

The term digital public services is used to refer to the public services provided using internet based technologies where citizens may interact with organisations entirely through an IT-system. One of its advantages is the implementation of this self-service concept where citizens should be able to demand data and services from their homes or somewhere else.

Public administrations in many countries are taking the step of moving towards paper digitisation and process dematerialisation. The Portuguese Government resolution [1] encouraged public administration institutions to reduce public expense by moving towards process dematerialisation and consequently to speed up administrative processes.

Casalino et.al [5, 6], performed studies over the Italian public administration system and what should be the gains and losses of paper digitisation and process dematerialisation. Concerns and drawbacks of this transition are pointed, namely the constant re-investments in new technologies and in training people on how to use them, and all the actions needed to encourage the population for the use of newer systems. Despite this, it has been concluded that all those costs are small when compared to the advantages brought by innovative and more efficient systems. They argue that benefits would be clear from an economical point of view (where the use of new technologies will lead to relevant savings) to an increase of HR productivity and a less intense environmental impact. With System Dynamics Models[7].
their simulations show how, within a scope of just a few years, break even points can be achieved and administrations can profit from all the expected advantages.

These type of works led to an increasing desire of governments and administrations to start the modernisation process and, currently, we find ourselves in the middle of the transition process in which there are already a lot of solutions supporting this type of business but they still have considerable room for improvement.

Within the context of public services, technologies such as PDF may offer process dematerialisation in two major ways:

- **Information Distribution**: static PDFs generated by administrative software such as ePaper automate the diffusion of information as a consequence of a deadline expiration or any other notification. Citizens may fetch all kinds of information provided by the municipality in a form of a PDF and ease all the information gathering.

- **Automated Requesting and Processing**: Technologies such as PDF Forms help eliminating paper usage by providing citizens with the possibility of remotely download, fill, sign and submit forms online. Not only this type of process becomes more comfortable to citizens as well as it may become much simplified for institutions that receive these requests: templatised documents can be read automatically and data can be processed, filtered and organised with less (or none) human intervention.

Besides these technologies, there are software solutions that provide process dematerialisation in this context of public administration and urban management.

Diverse Electronic Document Management Systems are commonly used by public administrations in order to manage all the documentation they have to maintain, their versioning and lifetime among other things. Despite not being the focus of this thesis, Document Management systems, their properties and functionalities are studied ahead.

On the other hand, there are Workflow management systems extremely useful inside organisations for the automation and process dematerialisation that they bring. All requests that any citizen submits in a digital format can be automatically processed by this type of applications with adequate workflows for each business bringing an end to all the paper circulation that was attached to this type of processing.

### 2.2 Electronic Document Management Systems

Electronic Document Management Systems are tools that can be utilised by governments and organisations to manage valuable information and documents. Through the use Information and Communication Technologies to process documents, they improve organisations’ performance and productivity.
Multiple authors have defined what a document is, in this context, and while some define it as "a unit of recorded information structured for human consumption" [8], others describe it as a "record and storage of information which may be in the form of clay tablet inscriptions, recorded conversation or speech into transcripts" [9]. Either way, the fact that technology is able to handle multiple types and formats of documents transforms greatly their roles in organisations once they can now be electronically processed, generated, stored, disseminated, consulted, etc.. From this perspective, an EDMS can be conceptually presented as a tool that eases and automates the processes that each document can be subject to (like storing, consulting or dissemination) as well as manages and relates all the data extracted from documents inside an organisation.

Huge technological advances in EDMS led to a pivoting in every type of business towards paper-free business models for a diverse amount of reasons. Electronic Document Management Systems are gaining relevance in the market due to their several advantages [10]:

- Physical storage reduced, as big warehouses can be replaced by small cheap hard-drives.
- Lower expenses, not only for the cost of the actual paper (that is not negligible over long periods of time) but also for the cost of using paper as an information carrier inside organisations.
- Time saving, once search and distribution of documents becomes instantaneous whereas it can take hours to find a document in a stacked pile of paper.
- Less error liability, with an automated filing system there is less room for errors on physical files’ management.
- Competitive edge over other businesses, for the efficiency improvement that an EDMS brings to a company can be crucial in tight competition markets.

For these and some other reasons, since EDMS were adopted by organisations, they became crucial tools for the maintenance and development of their businesses. Despite this, such evolution also presents some adversities [11] [12]. The high rate of technology changes and the desire and need of implementing those in organisations in order to make them more efficient and productive is a challenge that Information System(IS) managers face frequently.

Bigger organisations frequently face the challenge of fragmented data where different departments handle data in different ways. EDMS will have the job of managing data across departments becoming a more useful knowledge source for managers and will eventually lead to performance improvements and easier organisational management. The challenge for Information Systems’ executives is to integrate all the existing technologies and linking all the separated data into one unified system. This issue might grow in complexity as technology evolves unevenly in different but related areas [11] of action inside
an organisation. If new and emerging technologies are implemented individually it can increase divergences between different departments and consequentially the difficulty of merging their information and documentation all together.

EDMS are already used by organisations and can provide them more efficiency by automating processes as well as improving dissemination of documents and information inside and outside organisations. Besides, they should reduce data redundancy by providing a centralised data management. Despite all the advantages presented, its implementation might face several challenges at an organisational and technological level.

Nowadays there are countless EDMS’s in the market and they are, usually, associated with a core set of features that may vary slightly from tool to tool and their target business. Some of these features were studied during the development of this thesis:

- **Cloud Access**: which is a fundamental characteristic given that most business take place online. Documents should be available from anywhere and at anytime.

- **Document input**: that in most cases has to be a combination of inserting digital files as well as paper documents.

- **Document indexing and searching**: by giving documents tags and associations for them to be more easily found in the system. Also, documents should be easily found through metadata searches such as dates, authors, etc..

- **Document processing**: the reading and conversion of paper and hand-written typified documents to digital formats.

- **Document version control**: allowing to update documents, to revert to previous iterations of documents, and also to multiple users’ editions on the same document.

- **Workflow automation**: this feature varies significantly with the type of business that the tool is used for. It should automatically route and forward documents according to business logic needed each specific organisation.

- **User permissions**: not all users inside an organisation should have the same type of permissions and responsibilities. This should also be reflected in the Document Manager by giving each user a set of permissions to access only the documents they need to work on/with.

This type of system contributes directly to the dematerialisation presented in the previous section: automated workflows allow automation on request processing and all the document management features ease the distribution of information in the form of pdf documents.
There are registered patents of Electronic Document Management Systems (EDMS) since the early 90’s. Although their core functionalities are relatively similar and close to the ones listed above, each solution develops some features regarding their target business specifications. Given that, in the context of this work, a brief analysis on the existing tools for municipalities and urban management existing in Portugal will be presented.

The biggest Portuguese organisations in this field of action were contacted within the context of this Master Thesis but unfortunately no answer was obtained in any case. Given this, an analysis of their software solutions has to be made through the products’ specifications provided in their respective web sites.

*MyDoc* [13] developed by AIRC and *SigmaAtendimento* [14] developed by Medidata aside with Mind’s *ePaper* [2], are the main Document Management Systems used in Portuguese municipalities. Each one of these solutions provides functionalities of document and process management concerning their internal workflows, helping and automating the tasks performed inside municipalities following citizens’ requests. Despite this, no one (besides *ePaper*) claims to have functionalities related with the online distribution of PDF documentation to the population and none of the systems provides management and/or versioning of the forms available to the public.
3

ePaper - System Description
The ePaper system is a solution for digital files and processes management, document management and process dematerialisation. Several Portuguese Municipalities use this product since they began the administrative system’s modernisation. It is now used not only as a platform for citizens to download and submit their requests but also as a back office tool to analyse and process received applications. ePaper is used in more than 50 Municipalities in Portugal, covering over one third of the Portuguese population.

As Figure 3.1 illustrates, the application an be divided in two major components: the Front Office and the Back Office. Front Office is the platform available on the Internet that will serve as an help desk where citizens may communicate with the municipality, download the files or submit their requests. Back Office, on the other hand, is typically only accessible within the Municipality network and is the application where requests are processed and answered by the Municipality’s workers. In the Back Office, system administrators can manage a big amount of the Front Office’s available information, as it will be detailed later on. As these components may be running on different servers, the system’s complexity increases because file storage is not shared and needs to manually be maintained coherent.

Besides allowing an automatic management of all the documents submitted, not only as a repository but also as a processor of their contents, the Back Office component also associates all information submitted by an applicant (in the front office) and generates virtual processes where all information is gathered and handled, eliminating all these tasks that were previously executed manually by the Municipality. Thus, Municipality employees, have their role changed. Before, they used to be behind an help-desk, receiving paper forms from citizens and archiving them in some way it can be fetched later on and then mailed to some other department and now, when a document is submitted online, they have to verify its compliance with some rules, for example, and forward it to be reviewed by architects or other technicians to whom it can become instantly available.

ePaper’s main objectives are: providing a modernisation of the whole administrative system, by
process dematerialisation and procedures simplification; receiving and processing a set of standardised files (PDFs and DWFs, for example); security and integrity of documents and how they are managed and finally contributing to a more environment friendly solution.

Clearly, ePaper increases urban management productivity while providing a more comfortable service for the population. With this kind of system, it is much simpler for administrators to keep track of processes and the stages they are in, to know who is responsible for each task in a process and to have access to the history of actions taken inside each process.

Despite this, there are still some processes in the Back Office work environment that can be re-engineered in order to be simplified, more efficient and less vulnerable to errors. These will be described and analysed in the following sections.

3.1 Basic Concepts of The ePaper System

This section contains a general description of ePaper software and an explanation of the way some functionalities work (using the existing interface) in order to give some context for the remaining work and its implementation.

Being ePaper such a complex system, it wouldn’t fit in the context of this work to describe all its functionalities and possible configurations and, therefore, only a subset of its features will be detailed in this section.

Since the work presented is built on top of ePaper and in order to ease the understanding of the topics stated, it is relevant to clarify the meaning of some definitions used on the existing work that will be adopted in the current report. These concepts are related to the direct application of this project, namely the Municipality management of citizen’s needs in the topic of urbanism.

- **Process**: gather all the information regarding an applicant and all his requests. Generally it is created after the first delivery of a form and it usually follows a certain set of phases until its conclusion.

- **Request**: is a type of application that citizens can ask the municipality. Each request has, typically, one or more forms associated to it and this can be consulted and modified by users. These Requests are listed on the Front Office’s public interface for citizens to download ZIP packages with all the needed documentation.

- **Document**: in the current context, a document is a digital file (usually forms in PDF format, but may be instruction documents or legislation) that has meta data associated to it such as the fields it can have. There might be data that is part of the document even though it is not contained in it like its creation date and its creator, for example.
Some forms can originate new requests and others are used as complementary information for existing ones.

Documents are created by the Municipality and provided online in the Front Office’s list of requests interface. After being filled by a citizen or some external entity (such as architecture companies), documents are uploaded to the Front Office platform and they start being read and processed by all the Back Office users until the process is closed.

These documents play an important role because they might need to be updated (adding or removing fields, for example) and the management of these new versions, inside the whole system, is the beginning of this problem.

- **Package**: typically is a ZIP file available on the public interface (one for each request), and it contains all the documents a citizen needs to start a process.

  Each package is necessarily associated to a request having the currently active version of each document the request contains. Packages are currently created manually by gathering a set of documents into a Zipped folder.

### 3.2 Users and Use Cases

Figure 3.2 is a representation of some use cases of this application. Being *ePaper* a very complex system, there are only represented the relevant use cases for this work. The use case “Requests Processing...” represents the generic set of actions that take place to the processes a citizen’s request.

There are also a set of possible users involved in the whole application process:

- **Municipality Workers**: users who are responsible for handling processes. They review/approve submitted documents and in case of necessity notify applicants of any missing data.

- **Municipality Administrators**: who can manage all system configurations, including documents and request associations as well as package creation and availability.

- **Citizens**: entities who perform requests to the municipality. They download packages on the web platform and submit the filled forms.

Differences between Municipality workers and Administrators are shallow and differ from Municipality to Municipality. In practise, administrators have more privileges inside the application due to their job description and, therefore, are able to access a wider set of configurations but a single person might have both roles in a smaller Municipality with less employees.
From the use cases presented in Figure 3.2, there are some worth emphasising: "Add Document" and "Manage Download Availability". They represent the core of the problematic workflow once they are the tasks needed to perform precariously the non-existing task of updating a document.

For each use case illustrated in Figure 3.2 a brief description is given:

- **Download Forms** - Citizens access ePaper's Front Office public interface and download all the documents they need to start a Request. There it can be downloaded only the initial form (which is going to generate the process) or the ZIP package with more documentation.

- **Submit filled documents** - After filling the forms, citizens start an application to the municipality by uploading or delivering the filled initial form.

- **Managing Forms Download availability** - System Administrators are responsible for managing available files in the Front Office public interface. This means they create (manually) the ZIP files to be downloaded and they update documents (replacing old outdated files for their most recent version)

- **Create/Set Up Requests** - Documents can only be submitted to a set of typified requests. Those requests types have to be created _a priori_ by the municipality.
• **Process Submitted Requests** - this use case would include an enormous set of possible tasks depending on the type of request made. In any case, for the purpose of this report, it is only relevant to know this whole processing exists and not its details.

A typical scenario of *ePaper*’s use starts with a Municipality Administrator configuring a Request and associating a Document to it. Afterwards, the Administrator can make the document available on the online public platform where citizens may download it. After filling the document, the citizens upload it and through that action, origin a request. Later on, this request will be processed by Municipality Workers and will follow a determined set of stages until an answer is given back to the citizen with the outcome of the Request.

### 3.3 Implementation Technologies

*ePaper* is a web application built in the [ASP.NET Web Forms Framework](#) [15]. The DBMS is [SQL Server](#) [16] and [Entity Framework](#) [17] and [LINQ](#) [18] are used to manage interactions between the application and the database. Due to its complexity, the system uses third party libraries, from which the one worth mentioning (in this work’s scope) is [iTextSharp](#) [19], a PDF management library used to manage PDF forms.

### 3.4 System’s Structure

The application is divided in two main components: the Front Office which is the public interface citizens should access when they desire to make a request, and another which is the Back Office that works as a platform for users (municipality employees) to manage documents, requests and their availability on the Front Office web page among many other functionalities not necessary for the understanding of this work.

As Figure 3.3 illustrates, the Front Office module is independent from the Back Office module despite the fact they both access the same database server. They might be installed in different servers which increases the system’s complexity once they do not share the same local storage/paths.

Back Office is used only by municipality workers and administrators and helps managing all internal processes and files. Some Back Office workflows will be covered, once they are a source of errors and data inconsistencies in the system and need to be re-designed.

Front Office, on the other hand, is the platform where citizens can access and download packages, submit forms or check the status of their processes. In the Front Office, a registered citizen can find all the files needed for a determined request and submit those afterwards.
Files available on the Front Office need to be configured by the system’s Administrators on the Back Office and to perform this configuration, administrators have to use PDF files stored in the Back Office local storage. After this step is concluded, the Front Office will look for the file names configured previously in its own local storage, which demands a replication of the storage of both server’s storage. This is a problematic situation, to be studied in this report.

3.5 Relevant Functionalities’ Implementation

Within the scope of this work, ePaper’s set of functionalities can be split in two parts being the first all the management and configuration needed previously to forms’ submission and the second being the processing to which submitted forms are subject to. Even though this is an uneven division (since the application’s focus is handling received documents and consequential processes’ management), works as a pretty efficient filter on what functionalities are relevant for this study and therefore should be analysed in detail moving forward - the pre-submission functionalities.

The functionalities presented in this section are part of what is a major task inside the application - the insertion/update of forms to be available for download.
3.5.1 Form configuration

In order to explain the steps needed to insert and configure a form inside ePaper, Figure 3.4 illustrates the workflow of actions that a municipality administrator needs to perform to successfully configure a form in ePaper. The multiple steps will be explained below with screenshots of the interfaces of the application.

![Diagram](image)

Figure 3.4: Form Configuration workflow diagram

The actions enumerated in Figure 3.4 are detailed below:

1. Inserting the PDF file that will correspond to the form in the storage of the Back Office server, in a previously configured folder.

2. Inserting the same file with the same name in another previously configured folder in the front office server.

3. In the menu shown in Figure 3.5, the administrator needs to choose the previously inserted file in the first drop down box (that shows the names of all the PDF files in the folder).

   Afterwards, a few more settings have to be configured such as the type of form, or the code that will be written in the form to validate its submission. Most of these configurations are only relevant to the municipality who creates them and will not be covered now.

4. The application keeps all the data inserted, but it does not keep the file. Instead, only the file name is kept in the database and once a form is created the application will always look for the file by its name in the directory where all the files should be, meaning they cannot be moved or renamed.
5. Once a form is inserted in the system through this interface, it can be associated with a request in the menu showed in Figure 3.6. It is the interface where system administrators set up what is available for download on the Front Office. It allows the creation of a tree (for organisational purposes) in which each childless node corresponds to a request and is supposed to be associated with a preconfigured form as it can be seen in Figure 3.6. Administrators can create new tree nodes as well as reorder existing nodes. The “Formulário” drop down box lists all forms in the system created in the menu shown in Figure 3.5.

6. In order to associate a package with the Form created, the Municipality Administrator has to create (manually) a ZIP file containing all the files that are supposed to be available for the citizen when
applying for the specific request. This ZIP package needs to have the same name of the PDF form (only being different the file extension) and needs to be placed in the same folder as the form.

7. This ZIP package has to be copied manually to the Front Office local storage exactly like the form.

8. Only after all these tasks it is possible to download a PDF form and a ZIP package from the public interface.

3.5.2 Form Update

As it was previously explained, ePaper does not have an implementation for form versioning, which means that editing a form and putting it to use leads to a whole set of tasks before the up-to-date form starts being used instead of the outdated one.

If, at any time, a form needs to be replaced by a newer version, there are two options for users to follow:

1. Replacing the out-dated file directly in the file system by the new one, maintaining the exact same name. This needs to be replicated for both servers and naturally, as these storage folders contain hundreds of documents, human errors are extremely frequent in this procedure.

2. The second option is to follow the process described in the Section 3.5.1 and edit the form configuration to change the file selected and associate it with a Request.

3.5.3 Zip Package Providing

The forms mentioned so far are PDF files that can originate new processes after their submission. There are many other forms or documents that, not being able to initiate a process, are still crucial for its development. When a citizen wants to make a request and start a process, he should be able to have access to all those documents at once, and that is why packages are useful. Packages encapsulate several other forms and documents that are known to be needed in advance, facilitating the citizen’s job.

As it is shown in Figure 3.4 - actions 6 and 7 - for a package to be available in the Front Office, a ZIP file has to be manually created in the file system and it must have the exact same name as the document it should be associated to. The application will look for a file with the same name as the PDF file but with the “.zip” extension whenever a package download request arrives and will return nothing if the ZIP file is not found.
3.6 **ePaper’s Limitations - Problem Analysis**

In theory, all human tasks have an associated failure probability and that should be a main concern when designing software and workflows. Not only the way workflows are designed but also the number of validation tasks they contain can have a severe impact on the number of human errors that actually affect a system.

Reijers et al. [20], associates dysfunctional workflows with some typical characteristics:

- **Data redundancy**, that happens in ePaper, considering that every PDF and ZIP files have to be replicated between Front Office and Back Office Servers.

- **Conflicts between different servers**. The system presents inconsistencies if the files in one server are not exactly the same as the files on the other.

- **Long throughput times**. On tasks such as documents and packages’ provisioning for download, given the number of steps it takes to conclude these actions.

All these points can justify the redesign of this workflow, but to confirm this and to get a better understanding on how to approach the problem, the specifications of the problematic tasks should be analysed in detail.

Currently, ePaper is not prepared to support some tasks related with document management and package deployment. Despite this, there are a few ways to go around the existing implementation and execute operations that the system is not designed to cover. There is nothing in the system that allows forms to be updated. Although, as forms are only stored in the file system, if a user replaces a form for another totally different with the same name in the same path, ePaper will not distinguish between files and will not know the file was changed, however, in practise the form corresponds to a different file. The lack of rules and procedure validations brings to the system a huge vulnerability to some of the errors described later on this section.

Evidences from customer complaints and users observation led Mind to the belief that the significant majority of the human errors associated with this kind of workflows is directly related with the execution of tasks inside Windows File System.

This section contains an overview of the tasks that are not implemented in the application and yet are needed in the Municipalities. It also emphasises the problems and consequences related with the absence of these functionalities.

### 3.6.1 Document Versioning

As previously explained, for a different amount of reasons, documents suffer changes throughout time. Forms might need new fields or to have obsolete fields removed, as legislation documents might be to
be fully replaced, or even instructions documents might need to be updated whenever there is some change on internal processes.

The current implementation of the ePaper system does not expect to deal with this document management complexity and so form’s update becomes a vulnerable task. In practise, there are two ways of “updating” a form in the system: a) over write the file we are updating with a file with the same - losing the previous version and also losing track of whether it was or not updated, at least without opening the file; b) in the same folder paste the new version with a different name (usually adding the suffix “.v2” to the name of the first version), afterwards going to the form menu (Figure 3.5), instantiating the form in the system and finally going to Request Configuration Menu (Figure 3.6) and changing the association of the requests to which the first version was mapped.

This is not an actual implementation of document versioning, but is a work around possibility that allows to solve some of the versioning related problems.

Obviously, every change in the Back Office has to be reproduced in the Front Office (tasks 1 and 2 displayed in Figure 3.4), and on top of that, if a form is updated, all packages where its first version was present have to be remade and this is very likely to have human errors associated.

### 3.6.2 Package Creation and Update

For each request available in the public interface, the Front Office will be expecting a ZIP file that must have exactly the same name of the PDF form associated (for instance, if the form mapped in the Back Office is called "exampleRequestForm.pdf" there has to be an "exampleRequestForm.zip" in a predefined folder of Front Office in order for downloads to work correctly). There is not any support in ePaper to create or verify these files which means they are created manually by human users that gather all the files the package should have and name them accordingly and finally save them in the specific folder.

If, for any reason, a package needs to be updated, i.e, it needs to have more documents, or an already present form needs to be changed, this task is performed manually without any control or validation. There are real case scenarios in which packages are composed by dozens of documents and so it’s clear the complexity and vulnerability of this procedure to human errors.

### 3.6.3 Liability to Human Errors

There are numerous studies and researches on what type of human errors can occur more frequently and why. Bagnara et al. [21], identify several types of human errors eligible to happen (for example, in IT systems), and some of them are described as likely to happen when in routine situations characterised by automatic and over-practised behaviours. It’s a fair assumption that, for everyone working behind a computer on a daily basis, managing files in the File System is quite a common task and one can easily
mishandle files and folders without noticing the error. These so called slips are situations in which users have a correct intention and yet the action does not happen accordingly (slips include miss clicks, miss typing or even miss reading on file system).

Another error pointed in this research [21] is called lapse: usually related to memory failures and actions that are not completely or correctly executed because of those. For a common user of ePaper, a likely mistake would be to forget one form when building a package or only updating a form in the Back Office, forgetting about the Front Office server.

ePaper’s customer support has records of complaints from customers about errors related with lack of data coherency between the two servers (usually caused for human errors) and this is a problem Mind aims see to solved. These human related type of errors are meant to be minimised by the end of this work. To achieve that, a tool has to be developed in order to automate as many processes as possible as well as providing an interface that eases the execution of sensible tasks while validating and controlling actions taken towards the system.

After studying the problem at hands, the goal was to set the requirements for the solution to be built. This eases agreement between all the stakeholders and can be an objective criteria of evaluation on if the product is in accordance with what was discussed.

Furthermore, in this section will be described the initial plan for the solution in what concerns technologies to be used and generic implementation strategies.

To guide these decisions, ePaper characteristics will be taken in consideration in topics like interfaces, data models and functionalities implementation. Despite this, the aim of this work is to build a tool that can work separately from ePaper and be as versatile as possible in what concerns integration with bigger systems.
4
eFormManager
The solution presented in this thesis is called eFormManager and it answers the limitations of ePaper described in the previous chapter. This chapter contains a detailed presentation of eFormManager: its requirements, architecture and development process.

4.1 Objectives

In order to clarify this thesis requirements, was followed Wiegers' [22] approach where several types of requirements are to be defined in the beginning of the work and will be used as guidelines as the project goes forward. A generic scope of the project should be defined as well as what should be achieved with the project's outcome. The vision statement, following a Wiegers' template helps clarifying all the generic aspects of the project:

Most Portuguese Municipalities currently provide online all the forms and documentation their citizens might need to perform any type of requests. Managing all the documentation and its provisioning is executed manually at present, without any type of support tool or validation, which is problematic when updating and distributing hundreds of forms. The product presented in this report is a document and request manager that aims to ease the delivery of sets of versioned documents through automation of tasks, a clear interface and error detection features. Its major benefits are the validation of human actions, automated control of versioned documents, data coherency when fetching up-to-date documentation and mostly the automation of document/packages delivery. Unlike typical document managers whose focus is on organising and presenting different versions of documents, this application provides some of their functionalities having the extra possibility of scheduling deliveries and gathering documents in sets to be automatically delivered.

Given this, the overall objective for eFormManager becomes more clear: to build a tool that eases and automates the task of managing forms and deliverable packages increasing data coherency and implementing validation and records of the actions performed. To conclude, the goal is to create a better solution that can replace the one currently in use in ePaper.

4.2 Solution Requirements

After an agreement on the project's generic objectives, users and use cases were studied in order to define functional and non-functional requirements.

4.2.1 User Classes and Use Conditions

To refine requirements elicitation, stakeholders must understand and be clear about what are the use cases of the tool to develop, who is going to use it, with what frequency and under what circumstances.
There will be two different types of users:

- **Municipality system’s Administrators** who perform all the management of the application. Usually there are very few administrators per municipality so it is very unlikely that the tool is being used by two users simultaneously.

- **Citizens** that can access the public platform and download packages - the number of downloads can be considerably big (may occasionally reach hundreds per day)

Besides this, an analysis was made on the use of the currently installed versions of the *ePaper* application, and how often these various types of actions are performed.

Usually administrators access Request Configuration Menu on the first configuration of the application to create all the requests and assign them the forms needed to apply and after that, documents and packages are changed sporadically (when legislation or internal processes change, for example) meaning its capacity to scale in terms of simultaneous user accesses is negligible. On the other hand, *ePaper*’s Back Office application is not accessible from the outside of Municipalities once it runs on their internal networks and, therefore, capacity to answer diverse requests is not a concern of this tool but instead of the servers that run the Front Office public platform.

### 4.2.2 Functional Requirements

Functional requirements are a crucial part of projects like the one at hands. In order for them to work as evaluation criteria for the work developed, they should follow the SMART requirements [23] designation being Specific, Measurable, Attainable, Realisable and Traceable. With functional requirements well defined, it is possible to evaluate objectively the final product in what concerns its features. They mainly expose all the functionalities that the product should present.

A list of functional requirements divided by who will use it is presented here.

**Functional requirements for System Administrators:**

- **F.R.1** - Document insertion - the system should accept files (typically PDFs) and store them for download and management.

- **F.R.2** - Request creation - the system should allow the configuration of requests citizens may apply for.

- **F.R.3** - Association between Requests and Documents - the system should have sets of documents associated with each request that may be editable and that need to be delivered in order to perform the respective request.

- **F.R.4** - Document update - the system should allow the administrator to add files as documents’ new versions.
• **F.R.5** - Document version schedule - the system should allow the assignment of a date to every document version from which it becomes active and is used instead of previous ones.

• **F.R.6** - Version comparison - the system should allow the comparison of different versions of a PDF form, concerning its input fields.

• **F.R.7** - Input testing/validation - the system should detect and warn against “potentially” incorrect inputs (files, dates, text inputs, etc). For example, if two files are totally different and one is inserted to replace the other as a new version, probably either the selected file was wrong or the document being updated is not the one desired and the user should be warned (avoiding as much as possible the human errors).

• **F.R.8** - Automatic package creation - the system should automatically provide packages for every Request that have the active versions of the all associated documents.

• **F.R.9** - Activities’ logging - keep logs of all the changes made to each Request and each document.

Functional requirements for the citizens’ platform:

• **F.R.10** - Submitted files’ validation - the system should verify whether each file that citizens submit matches a valid version of the document (in order to avoid submitting outdated files).

• **F.R.11** - Download of documents and packages - the system should allow the download of documents and packages for every request providing automatically the up-to-date version of every document.

For evaluation and validation purposes, the implementation of all these functionalities should be demonstrated in the current report.

### 4.2.3 Non-Functional Requirements

Nonfunctional requirements are described as characteristics the system must exhibit or properties it must have. These requirements are highly dependant on the type of system being built, as well as its use circumstances.

The concrete goal of *eFormManager* is to run inside the *ePaper* application which runs in Municipalities internal networks. Given that, all its security requirements (such as users, logins or permissions) will be inherited from *ePaper* after its integration.

Besides this, as explained before, the new tool is not going to have more than half a dozen users in total and therefore, server capacity to deal with many accesses or even simultaneous requests is not one of its main priorities either.

The non functional requirements set as the most relevant for this project are listed bellow:
• **N.F.R.1** - Performance - users should be able to complete tasks like creating packages or updating documents in less time and with a fewer number of errors than they were before.

• **N.F.R.2** - Accessibility - for this application to be used in public administration, it has to some accessibility rules issued by the Portuguese Government in [24]. A verification tool [25], also provided by the government will verify that the system's interface complies with all the rules.

• **N.F.R.3** - Usability - can be expressed by the complexity of the actions needed to perform a task or the number of clicks needed to conclude the desired task. The usability requirement in this context is to have a tool that is easier and more intuitive to use than the previous system and it should be reflected in user satisfaction.

• **N.F.R.4** - Openness - even though it is built to be integrated with ePaper, this tool is supposed to be able to be integrated with other systems. Therefore, its architecture has to be ready for integration with other applications.

All the requirements referred in the present section will be covered again in the evaluation chapter of this report (chapter [5]).

### 4.3 Use Cases

In order to introduce the tool developed, its use cases (illustrated in figure 4.1) are described briefly in the following list divided user. For a better understanding, concepts from section 3.1 should be remembered.

- **Create Requests**: the creation of a request entity, that can be followed by an association with existing Documents.

- **Insertion of new Documents**: inserting in the system a new PDF file that will be read and can afterwards be associated with any existing Requests.

- **Update Documents**: consists on inserting a new PDF file on the system that will be considered the most recent version of the current Document. When a new version is inserted, the user has to choose a date from which the version becomes active (downloadable).

- **Compare Versions**: a user can at any time select two versions of a document and have a visual comparison of the PDF forms selected where the different fields of the forms are highlighted. This comparison is also made whenever a new version is to be inserted so that the user can verify the correctness of his action.

- **Consult Log History**: Admins can consult the actions taken into each request and the history of versions of each document.
• **Package Download:** The application has an interface (and an API) for package download. When an applicant downloads a request, the system verifies the package is up-to-date and creates a new one in the case the current package is out-dated.

![Solution's Use Cases](image)

**Figure 4.1: Solution's Use Cases**

### 4.4 Architecture

The *eFormManager* Application’s structure is illustrated by the diagram of figure 4.2. Contrarily to *ePaper* (figure 3.3), the *eFormManager* is designed to run in a single server, having all the business logic and file system in the same place which increases data coherency.

One of the components of the system is a C# library where all the logic is encapsulated and divided in multiple modules as shown in the diagram above. These modules correspond to classes created in order to separate the code of different functionalities. Apart from this, there is a controller where web requests are received and forwarded to be processed in the logic modules.

There is also a component of web pages and scripts that make the User interface module and a data layer that provides an interface to access the database.

### 4.5 Data Model

The Data Model of *eFormManager* that helps understanding the application’s behaviour, is illustrated in figure 4.3. It suffered changes throughout all the projects execution due to scope changes and new demanded functionalities.
There are four major related classes around which the whole application works:

- **Document** is a conceptual entity since it does not represent a physical element but instead it is used to gather a set of files that should be editions/versions of the same original file. To instantiate a document, it is logically required to have a version attached to it because a document does not make sense without its first version. Versioning is what allows documents to be updated having a controlled management system.

- **Version** represents one iteration of a document and has necessarily a physical path attached to it. For record purposes has its creation date and the ID of a user who created it. Besides this, there are two validation properties: the hash and the validationID that is inserted hidden in the form. Finally one of the most relevant properties for the application’s logic: the Activation Date - is the date from which the system considers that one specific version is the one to deliver to the public.

- **Request** represents one of the many available applications that a citizen or other entity can make to the municipality. In practice, those applications are just a set of files that the applicant has to submit therefore a request is an organised set of documents (not versions). A request is also a conceptual entity and its specific usage happens through packages. All actions related to a Request are stored as logs, keeping the history of each request in the database.

- **Package** is an automatically generated entity. It represents a ZIP file that contains the up-to-date version of each document associated with the request it belongs to. The ZIP files are also
automatically created by the application whenever a there is a download request and the the respective package is outdated.

Important to clarify the relation between Package and Version. Not only a version can be included in multiple packages but also a Package can contain multiple versions (of different documents, necessarily).

This structure allows having an application where entities have a strong degree of independence and stability, which eases the whole process of maintaining consistency among the system.

Versions of the same document are completely independent entities as intuition would suggest. After all, they are completely separate files even though they can/might be similar. This property simplifies the needed processes of synchronisation in the system because once a version is committed, there is no editing that could generate inconsistencies.

This associativity between entities allows great flexibility: a document is associated with a request or more and, if there is the need to change it (by adding new versions), their relationship can remain

![Figure 4.3: eFormManager's Data Model](image-url)
untouched since that change will be considered for every request associated to the document.

Besides these major entities, there are a few more classes that are used for extra functionalities. Both the following classes entities are consequences of requirements that were added to the project during its execution:

- **VersionPage** is used for visual representation of documents. To display the files inside the application, images were generated for each page of the file and its data is kept on this entity.
- **RequestGroup** that is used for organisational purposes as it is used to group requests in a hierarchical tree.

### 4.6 Implementation

This section contains a description of the architectural models used in the implementation of *eForm-Manager* in a lower level than what is presented in section 4.4. There is a brief description on how and why specific architectural models were implemented within the context of this solution. Besides, this section contains a list of the technologies used in the development process as well as a description of the development process itself. Finally there is an overview of the code base of this project.

#### 4.6.1 Architectural Models

*eFormManager* is to be integrated with other applications and, given that, it needs to be easily integrated with other systems. It is true that interfaces and styles have to be frequently remade and redesigned, but the same does not apply with the logic behind the program and building a pure REST application provides total independence between front-end and back-end code. This architectural style implies that the back-end has a well defined set of endpoints to be called for different tasks/services and these endpoints are the ones who may facilitate an integration with any system.

REST was introduced by Roy Fielding [26] and has a specific set of characteristics, some of which were extremely relevant for its choice:

- **Client-Server architecture:** If for any reason, this application is to be included in an existing interface, the API can be seen as an encapsulated set of services independent from the interface that can be imported and integrated with any system.

- **Implementational freedom:** Both client and server’s implementation can use different technologies. From a perspective of integration this is a valuable characteristic. If for some reason the integration needs a specific technology on either client or server side, the other may remain untouched and will not be affected by any change on first.
• URI representation of resources: the REST API uses a URI (Uniform Resource Identifier) to represent resources and to make them available for clients. Remembering that this application is to be included in ePaper’s Back Office but it still needs to provide resources - packages for applicants to download - to the Front Office a URI format to make packages and other resources available is an ideal characteristic to have on the system.

• Stateless applications: Even though it was already explained that user concurrency is not a major problem on the system, this characteristic of REST can be a very useful tool on maintaining data coherence (even more considering a centralised system and database).

Besides a REST Client and Server, the Web Server is itself divided in smaller components with different concerns.

There is a Request Handler that has defined all the REST endpoints, where requests are processed and, consequentially, services are called. Request Handler validates all the requests and returns an answer to the Client after all the processing is done.

Database Handler is a component that works as proxy between the application and its database. All the direct contacts with database are done in this module.

The Service Handler is where all the business logic is implemented. It calls Database Handler for database queries and is called by Request Handler whenever a REST request is received. All the processing done here happens only a consequence of a request.

An external Database running on SQL Server, with a Data Model detailed on 4.5.

In section 3.5 one can understand how the Request Configuration Menu (Figure 3.6) and all its features are presented in the same screen/Web page. For that reason, and considering this tool aims also to substitute Request Configuration Menu, it is reasonable to have the same approach of having a tool that is a SPA (single page application). Besides, SPA’s and REST API’s are a good combination to have: information fetched from the server in JSON format can be processed by the client and displayed asynchronously without reloading the page.

No framework was used to develop the client-side application. Instead, some JavaScript design patterns [27] were analysed and an MVC (model, view, controller) design pattern was chosen.

Client’s architecture is divided in three components:

• **Model**: is responsible for every communication with the server; is where all requests are sent and received. It should contain all data needed to display the current page. When JSON data is received from the server, Model processes it and triggers an event to be handled by the view. Model does not have any knowledge on the page structure and does not contact directly with HTML elements.
• **View**: handles all the events triggered by the model. Is responsible for displaying data stored in the Model as well as managing all the visible contents in the page.

• **Controller**: listens to every user action in the page and reacts accordingly. If the user entered data that needs to be sent or fetched from the server, Controller will call the Model and if user action only demands client side processing the Controller will call the View.

### 4.6.2 Used Technologies

As already mentioned, when choosing technologies to be used, *ePaper*’s technologies were taken as a reference and used whenever it was considered adequate within the context of this Thesis. It was decided that building this tool separately would make it more versatile. Despite this, all the technologies chosen were already used in *ePaper*. Following, a list of the used technologies is provided.

In the back-end:

• **.NET Framework**[^1] developed by Microsoft that provides language interoperability and can be used to build Windows Applications and Web Services,

• **Entity Framework**[^2] and **LINQ**[^3] which are Object-Relational Mapping and were introduced in the version 3.5 of the .NET Framework,

• **Microsoft SQL Server**[^4] which is a Relational Database Management System

Being front end composed only of JavaScript libraries, this list contains the most relevant:

• **jQuery**: used to simplify HTML manipulation,

• **Ajax**: a tool from jQuery used to perform web communications between server and client,

• **DataTables**: a plugin for jQuery that allows the use of customised tables for display and management of information in tables.

### 4.6.3 Development Process

In order to develop this tool, Mind’s development strategies were adopted, specifically Agile Development[^30], that allowed clear, well defined and compartmented and validated development tasks. Besides, this development method allowed frequent validation of the work developed from the different stakeholders.

[^1]: https://dotnet.microsoft.com/
[^2]: https://docs.microsoft.com/en-us/ef/
As a code repository was used Microsoft's Azure DevOps Services[^5] as it is in use for Mind's applications and has a simple integration with Visual Studio[^6], the IDE used.

Testing is a crucial part of development and creating a set of automated tests was also a stage of this project's execution. Different types of tests were written for the client and server sides with different technologies, according to Mind's testing policies:

- To test the user interface and all the JavaScript code present in the client, Jasmine Framework[^7] was used. Client side tests were focused on interface behaviour and unit tests were implemented to assert every function's answer to different inputs. Besides, there were integration tests where the client makes REST requests (caught by the framework) and its reaction to different answers is asserted.

- This application's logic is majorly composed by queries to its database and some processing of those queries results. As such, and following the company's policy, there were not as many unit tests as typically can be expected. Integration testing was the core testing made in the Server side code where query functions were tested using a mock database to simulate different possible outputs for every query.

Despite this, some exceptional unit tests were implemented on the few algorithms built inside the system, namely the package creation algorithm and the version comparison algorithm.

Finally, with Jasmine some end-to-end tests were implemented in order to verify the system's core functionalities.

### 4.6.4 Code Base

The solution built on Visual Studio contains three projects that can be compiled separately:

- Front-End Project: this project contains all the HTML pages as well as javascript files and also the REST endpoints. This project contains a total amount of 5685 lines of code, divided in a total of 18 files, from which 2194 lines are javascript scripts while the 2150 are c# files, 579 are HTML files and the remaining 762 are css (style) files. These numbers do not include external javascript and css plugins used.

- The back-end project: a code library which is composed by a total amount of 22 c# files that contain in total 3,366 of source code.

- The Testing project: contains only c# code as well, 9 files with a total of 1566 lines.

[^6]: https://visualstudio.microsoft.com/
[^7]: https://jasmine.github.io/
4.7 User Interface

Being this system a work tool, its interface should be representative of that, meaning the aim is not necessarily to have a colourful and very dynamic web page but a functional, intuitive and straightforward one providing a better user experience.

Obviously, interface design is a subjective topic and therefore one of the most complex to evaluate since a good interface is an interface that satisfies users and what is a simple/intuitive interface to someone can seem complex to another user and with that come all the difficulties usually faced when designing interfaces.

On the one hand, interface design is not the main focus of this project, but on the other, being its target to create a tool that eases human tasks and eliminates errors, interfaces are a major factor on its success.

Given all this, followed some guidelines and principles from the Portuguese Government usability portal [28] for web pages design to decide on the base structure and organisation of the page should be. Besides, ePaper’s styles and structure was also taken in consideration on some design decisions.

The first and major concern for a page like this should be the clarity on content display. The way they are organised should take the most advantage of all the available space in the page in order to display as much useful and relevant information while keeping readability and awareness.

The application has only one page where the user can navigate and its structure is detailed in this section. A page map (divided in sections) is in figure 4.4: the breadcrumb section, that is important to give the user information its location on the web page and grant the awareness mentioned. The section right bellow the breadcrumb is the main navigation section, that selects the content to be displayed in the secondary navigation section.
The choice was to have all the navigation tools on a sidebar and all the main content displayed in a centered section of the screen. A clear distinction between all these sections is imperative for an easier understanding of the page’s structure and that can be achieved through different background colours and different font styles, for example. The colour scheme and generic styles choice was based on ePaper’s colours.

Apart from this page map, there is a menu that appears as an overlay container (as shown in figure 4.5) used for major inputs in the system, such as new documents or requests, or even inserting/rescheduling versions. This implementation decision was taken considering that a significant part of human errors in the interface happened by miss selection of items in drop-down lists and similar menus. It was considered that giving an isolation such as fig 4.5 illustrates would highlight all the inputs and help error detection by the user.

![Web Page Input Menu](image)

Figure 4.5: Web Page Input Menu

Screenshots of the application are displayed in the following section where all functionalities are demonstrated.

### 4.8 Document Configuration Workflow Details

As in section 3.5.1 is presented a Workflow diagram that illustrates the task of setting up a document until it is ready to be downloaded.

The tasks described in Figure 4.6 are the following:

1. In the browser, the user uploads a file, inserts the document metadata and associates the document to existing requests (This interface will be shown in Chapter 5).

2. The eFormManager will store the file automatically in its storage.

3. The inserted metadata is stored in the database,
4. The document is ready to be downloaded in the public interface.

Figure 4.6: Setting up a document in eFormManager - Workflow Diagram

The major difference consists on the number of tasks needed to be executed by the municipality administrator. The user only performs the actions of uploading the document and setting up its metadata, all in the same interface, which will certainly lead to less human errors.

Comparing Figure 4.6 with Figure 3.4, the new workflows only demand one task from the Municipality Admin. Tasks 1, 2, 6 and 7 from Figure 3.4 (that consisted in managing files in servers’ storage) no longer exist which is an improvement moving towards less complex workflows and, therefore, less human errors.

4.9 REST Endpoints

The REST endpoints used in this implementation are listed and grouped by its context in this section. There are get and post requests and each endpoint has a brief description of its purpose.

4.9.1 Documents

The endpoints used to get data from documents and versions as well as edit their information are the following:

• *document/[id]/edit*: Post request to edit a determined document's details such as its name or description.

• *document/[id]/versions*: Get request that returns a specific document's version.

• *document/[id]/requests*: Get request that returns all the requests associated with a document.

• *document/[id]/addReqs*: Post request to associate requests to a document.

• *document/[id]/testVersion*: Post request that receives a file and returns its comparison with the latest version of a document.

• *document/[id]/addVersion*: Post request to add a new version to a document (always after its analysis).

• *document/[id]/versionDate/[vID]*: Post request to reschedule the activation of one version.

• *document/version/[vid1]/compare/[vid2]*: Get request that returns the comparison result of two versions in a JSON format.

• *document/version/[id]/delete* Post request that deletes a version.

### 4.9.2 Request

This subsection contains a list of the endpoints used to create and edit Requests, as well as manage their associations:

• *request/[id]/reqDocuments*: Get request that returns all the documents associated to a request.

• *request/add*: Post request with data attached in a JSON format that creates a new request.

• *request/[id]/addDocs*: Post request to associate a set of Documents to a Request

• *request/[id]/edit*: Post request to update the details of a request, such as his name, description, and identifier code.

• *request/[id]/logs*: Get request that returns all the system logs associated to a request.

• *request/[id]/packages*: Get request that returns the zip file corresponding to a request package.

### 4.9.3 Request Groups

The following list contains a set of endpoints used to perform basic operations over the Request Groups such as read, edit and delete:
• *requestgroup/add* : Post request that creates a new request group.

• *requestgroup/{id}/edit* : Post request to edit a request group details such as names and descriptions.

• *requestgroup/sort* : Post request to edit the request group in the organisational purposes.

### 4.9.4 Generic

There are some endpoints used out of the context of any of the previous entities specifically or commons for everyone:

• *tree* : Get request that returns the full tree structure.

• *details/{type}/{id}* : Get request that returns the details/metadata of documents, requests, versions, packages and request groups.

• *delete/{type}/{id}* : Post request used to delete documents, requests, versions, packages and request groups.
5 System Evaluation
This section presents the evaluation of the proposed platform. The functional validation follows the requirements presented in section 4.2.2 and describes through screenshots how those requirements were achieved and how the functionalities were implemented. Non functional requirements are also presented accordingly to what was defined in section 4.2.3.

5.1 Functional Requirements

This section presents how the functionalities were implemented and, as such, this section presents screenshots available to the users that demonstrate how functional requirements are met.

Each of the following subsections refers to a functional requirement defined in section 4.2.2. Their title has the code of the specific functional requirement that is matched with each functionality.

Before describing in detail each functionality, a screenshot of eFormManager’s main page is presented (relating with the map shown in figure 4.4). A generic screen of the application can be observed on figure 5.1.

![Figure 5.1: Application's Main Page Screenshot](image)

The main navigation section has the tree menu selected. This option means the secondary navigation section has a tree view where all Requests and Request Groups appear with their hierarchical organisation. In this tree, a node can be selected in order to display content on the center part of the screen. The first line with node controls allows (in this case, where a request node is selected) to move the request up and down on the tree, as well as delete it or to download the package automatically built for that request. The Details section is editable and allows the user to change some metadata of the
selected node such as its name, description or parent request group. In the main content section, two tables appear: one showing all the documents associated with the request and the other showing the history of the request. All the controls and content tables change with the type of node selected.

Functionalities are divided by their logical use/execution in two sections: one that contains the functionalities used by the municipality’s administrators and another one for the citizen’s functionalities, such as form download and submission.

5.1.1 System Administrators’ functionalities

The functionalities to be used by system administrators are listed in this section.

F.R.1 - Document Insertion

For users to insert documents in the system, it is only needed to upload a PDF file and choose a code to be associated with this document. Usually this code is used to distinguish between forms, certificates, instructive documents, etc..

To access the document creation menu, users have to access the request menu and click in the button “Novo Formulário” as figure 5.2(a) highlights. As mentioned above, this input is presented in an overlay highlighted container as shown in figure 5.2(b). A message will appear on this menu informing about the success or failure of this operation.

F.R.2 - Request Creation

The interface for request creation is very similar to the Document insertion as figure 5.3 demonstrates. The user goes to the Request menu and has to click on the button that says “Novo Pedido”, as shown in figure 5.3(a). The input necessary for creating the request is a name for the request, a description
(optional) that gives the applicant some detailed information on the selected request and finally, the drop down list that appears in figure [5.3(b)] shows all the request groups to which the request will belong. A feedback message is also provided in this menu giving the user information about the success of the task executed.

![Open New Request Form](image1)

(a) Open New Request Form

![Request Creation Form](image2)

(b) Request Creation Form

Figure 5.3: FR2 - Request Creation

**F.R.3 - Association between Requests and Documents**

This can be considered one of the most crucial functionalities of this application. To associate a Document with Requests or vice versa, one selects the node that is to be associated with others. The left content table will have an edition icon as it can be observed on figure [5.4(a)]. A click on this button will open the overlay menu shown bellow on figure [5.4(b)] - a situation where a user can pick between all the documents which ones should be associated to the selected request.

![Request Selection](image3)

(a) Request Selection

![Editing Documents associated to the selected request](image4)

(b) Editing Documents associated to the selected request

Figure 5.4: FR2 - Editing Documents Associated to a Request

When changes are saved, a feedback message is shown and changes should be instantly visible on the node main menu.
Adding a new version is quite similar to adding new documents. In the version’s menu, available whenever a document is selected on the sidebar (figure 5.5(a)), there is an icon that opens the overlay menu present on figure 5.5. This form demands a file input as well as an activation date from which this version should be included in the packages of all the requests associated with this document.

![Document Selection and Updating Document with a new version](image)

(a) Document Selection  
(b) Updating Document with a new version

Figure 5.5: FR2 - Editing Documents Associated to a Request

When a file is selected, it is analysed before its submission and a small report is displayed (as figure 5.5(b) shows) in order to avoid potentially wrong inputs. A submission is considered potentially wrong when a file presents many differences when compared to the last version added to a document (one should remember that document updates usually are just small changes in a small group of fields). Whenever the system considers there might be an error it prompts the user with a warning.

There is always an option of having a detailed comparison with different fields visually represented - this feature is similar to the Version comparison tool, described below.

**F.R.6 - Version Comparison**

The application provides a graphical interface for version comparison. It is still only prepared to compare PDF Forms and its input fields, since those are the core target of the system and it consists on highlighting the fields that are different between the two files shown.

To access this feature, users should click on the Comparison button, in the versions menu, shown in figure 5.5(a). If there are two or more versions selected for comparison, the screen in figure 5.6 appears.

In the example presented on 5.6, we have a simple but representative example of an update. The newer version (on the right) had a field removed because the municipality thought it was no longer useful.
to ask applicants for a landline number. This field is marked on the version where it is present with a tool-tip that contains the field name as well as its type (a text field, in this case).

**F.R.8 - Package Creation**

Automated package creation can be considered the most valuable and innovative feature of this application. It has not any kind of interface, once its purpose is to manage data and files without user intervention but it automatically provides up-to-date ZIP packages for every request in the system that has documents associated. When a user clicks on the download button, in a request, the system performs a quick verification to assert if there is a package in the database (corresponding to a file in the file system) with the active versions of all documents associated. If the last package generated is out of date, a new one is built and send to the applicant.

Typical Document Management tools are not equipped with this functionality given how specific it is. Despite this, the automation of package creation is one of the most important functionalities regarding the elimination of human errors from the workflows.

**F.R.9 - Logging**

When a system like the one discussed in this report is accessed by multiple users, logging the actions executed important not only to keep track of the changes executed but also to hold users responsible for their actions.
The application was built with a generic layer over the database that registers all major actions taken towards data. This layer creates a record in the table "Log" with a small predefined description of the action taken, the user who executed the action and the time when the action was performed.

These actions are displayed in the context of each request, as it is visible in figure 5.4(a), in the right side content table name "Histórico". Its content is displayed in detail.

![Request Log History](image)

Figure 5.7: Request Log History

These logs track whenever a document or request are created, deleted, updated, or their associations with other documents/requests change.

5.1.2 Citizens’ functionalities

A prototype of ePaper's citizen interface was built in order to demonstrate these functionalities.

F.R.10 - Submitted files’ validation

Given that this application manages all the downloadable forms, it should also be responsible for granting that documents out dated are not submitted by applicants. For this validation, the system inserts in each version of each file an hidden identifier code and whenever a document is submitted in the platform, that code is verified confirming whether the code corresponds to the most recent version of the document.
If a delivered file corresponds to an old version of a document in the system, the applicant is notified to download the currently active version of the form.

Even though this tool is not yet integrated with all the post submission processing present in ePaper, the prototype of the public interface has a validation area where documents can be uploaded and are validated by the system.

To access this submission form, users should click on "Novo pedido", highlighted in figure 5.8(a). It opens the form present in figure 5.8(b) (where an invalid/outdated document was already submitted and therefore an error message is prompted).

Figure 5.8: FR8 - Submission of an outdated document

F.R.11 - Download of Documents and Packages

In the mock of the public's Interface, there is a link (highlighted in figure 5.8(a)) with the text "Formulários". This link leads to the interface shown in figure 5.9 where it is replicated the organisational tree build in the main page of eFormManager - figure 5.1.

With this interface users can navigate through requests and request groups and download files and packages associated with the each request as displayed in figure 5.9.

Figure 5.9: Download Requests’ documents and packages
5.2 Performance Evaluation (N.F.R.1)

As stated in section 4.2.3, performance in this context is mostly about usability, i.e., if users can execute the same tasks they were able to execute before, in a smaller time, with less errors. All this detailed analysis is described in section 5.4.2. Despite this, regarding usability, some aspects of computational performance of this application analysed and described in this section.

5.2.1 Performance Tests

Performance tests consist on using software tools to simulate certain circumstances and see how the system behaves. Even though there are several types of performance tests, some of them are not as relevant once they evaluate situations that are not the use cases of eFormManager.

Typical performance tests that evaluate web applications are related with its CPU usage, memory consumption and scalability. Despite this, this tool is to be integrated with another application and installed over each municipality infrastructure. This implies that all the three performance metrics mentioned are totally dependant on the configuration/resources of the production system instead of the performance of the code. For this reason, these metrics were not considered in this work.

Instead, tests were performed in a local installation of the application regarding the response times of the application in the browser, which is the most relevant metric regarding user satisfaction. The computer where tests were ran is a laptop with a great computational capacity (8GB of RAM and an Intel i5 processor) so it would be expectable to have better results when running on a real server with more capacity.

On the other hand, performance tests should consider a data volume that corresponds to the worst real case scenario and to achieve that a database from the most populated municipality (using ePaper) was used. This dataset is contains:

- more than 10000 processes,
- more than 500 requests,
- more than 230 000 submitted documents.

There is a small part of eFormManager that is available outside of the municipalities intranet - the package download feature. Although, it would not make sense to test its capacity to deal with huge loads of requests once all those requests are made to the Municipality server (which is out of the scope of this project) and it will use the url provided by the application for a request. Given this, the application only executes work when a document or request is updated (which does not happen often enough to be relevant) and, therefore, the workload is on the municipality server and not on eFormManager.
5.2.2 Performance Metrics

To obtain some performance metrics, Google Lighthouse\footnote{https://developers.google.com/web/tools/lighthouse/} was used and a report was generated with the results presented in table 5.1 (the table has the average results of ten audits performed in order to grant consistency).

In order to perform tests, Lighthouse is provided with a url and provides a series of tests before generating a detailed report with an objective and concrete evaluation of the website.

According to Lighthouse scoring measures, tests had a score of 100/100. These tests were executed with eFormManager running on the local-host which will not be much different from the reality once it will always be ran in internal networks but with more powerful and resourceful servers. As mentioned before, the data volume used to execute the tests was a database collected from the most populated municipality that uses ePaper, so it can be considered a real case and still be working with the large data volume mentioned in the introduction of this section.

Table 5.1 shows the results of the tests ran by Lighthouse as well as a brief description of what each test measures. All the times measured are extremely satisfying in the context of user experience.

<table>
<thead>
<tr>
<th>Test</th>
<th>Summary</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>First Contentful Paint</td>
<td>First Contentful Paint marks the time at which the first text or image is painted.</td>
<td>0.45 s</td>
</tr>
<tr>
<td>First Meaningful Paint</td>
<td>First Meaningful Paint measures when the primary content of a page is visible.</td>
<td>0.45 s</td>
</tr>
<tr>
<td>Speed Index</td>
<td>Speed index shows how quickly the contents of a page are visibly populated.</td>
<td>0.75 s</td>
</tr>
<tr>
<td>Time to Interactive</td>
<td>Time to interactive is the amount of time it takes for the page to become fully interactive.</td>
<td>0.46 s</td>
</tr>
<tr>
<td>First CPU Idle</td>
<td>First CPU Idle marks the time at which the page’s main thread is quiet enough to handle input.</td>
<td>0.46 s</td>
</tr>
<tr>
<td>Max Potential First Input Delay</td>
<td>The maximum potential First Input Delay that your users could experience is the duration, in milliseconds, of the longest task.</td>
<td>40 ms</td>
</tr>
</tbody>
</table>

Table 5.1: Google Lighthouse Metrics for eFormManager

5.3 Accessibility Evaluation (N.F.R.2)

Concerning the human-computer interaction, accessibility refers to the use of software by people with some kind of disability. Systems should be built with some special characteristics in order to be understood and read by assistive technology software used by disabled users such as screen readers that read HTML pages, in the case of blind people. Naturally, these software tools have some limitations and the pages they read must be compliant with the Web Content Accessibility Guidelines (WCAG) 2.0 \cite{wcag20}.

Following an European Union’s directive\cite{eu директивы}, on the 19th of October 2018, the Portuguese Government emitted a Law\cite{portuguese law} with a set of requirements that every web site of a public institution must comply with. To help verifying this, a tool\cite{accessibility tool} was provided online for developers to insert their files or URI’s and a score is given in a scale of 1 to 10 regarding all the rules set in the legislation.
This tool was used to validate the web page developed and a score of 10/10 was obtained in the end, having zero fails and complying with WCAG.

Besides this institutional validator, other development helping tools evaluated the application such as Lighthouse Audit tool used by Google Chrome. With Lighthouse, a perfect validation of all accessibility rules was obtained.

5.4 Usability Evaluation (N.F.R.3)

Usability is a crucial parameter for measuring the success of this work, since it aims to improve user experience in a set of workflows as well as to decrease the number of errors associated to them. In this section the goal to quantify the improvement that this system brings in comparison to the previous workflows.

Usability was measured through user experiences and feedback as a result of tests performed with real users.

5.4.1 Testing Set Up

There are a few standardised software usability tests that can be used to quantify the quality of users’ experience. In the context of this work, two different tests were chosen for two different reasons.

- **System Usability Scale (SUS)** [32], introduced by Brooke in 1996, was chosen for being a broadly recognised usability measure that could classify globally, not only the system built, but also the previous system and provide a comparison based on the same criteria. For the SUS test, results can be calculated as described by Brooke in order to obtain a score between 0 and 100. These scores are illustrated in figure 5.11. In order to create a relative grading, Sauro and Lewis(2016) [34] used data from a large dataset of usability studies and created a scoring system with a range of A+ to F corresponding to a percentile range. This can give the reader a better understanding of the results obtained.

- **After-Scenario Questionnaire (ASQ)** [33], consists on having a small set of three questions after the execution of a guided task. This method was considered extremely useful for the purpose of comparing workflows and specific scenarios. If a tester executes the same task with both tools and answers the same set of questions rating the tools within the same scale, that would be an explicit measure of comparison.

While SUS asserts the usability of the system as a global unit, with ASQ there will be specific information about each task and how the users experienced their execution. For these tests, the simplicity of workflows is rewarded as well as the ease users feel through the execution.
In order to simplify data analysis, results for each task were merged according to Lewis [33], who states that all the three questions can be "reasonably condensed" to a single scale through summation in order to allow an easier interpretation of the results.

For the execution of both tests, an ideal test would be to have users experimenting all the features of both systems having a full experience and giving feedback afterwards. Having this in consideration, three tasks were chosen so that the tester would go through the vast majority of the implemented features. These tasks are related with the management and deployment of forms and packages and are representative of a real world scenario:

1. The creation of a ZIP package correspondent to a Request with a form inside and its deployment for download;

2. Adding two auxiliary forms to the previously created package and its deploy for download;

3. The change/update of the main form and the deployment of an up-to-date package (with the form’s most recent version).

The guide that was built (and can be found in appendix A) includes tasks, instructions and screenshots that testers should follow. After each task, users answer the three ASQ questions and, in the end of the three tasks, they answer the ten questions of SUS (the questionnaire is in appendix B). For each task, was measured the time the user took for its completion as well as were registered all the errors occurred during its execution.

Ideally, the target population would be composed by real users of the software and potential future users, i.e, municipalities administrators. Unfortunately, that was not a possibility during the execution time of this project.

On the one hand, it is relevant to have testers with good knowledge of ePaper’s workflows once they are the ones that can actually measure the improvement brought by new system. This subset of the population could only be found inside the company and so, 10 members of the Support, Development and Testing teams were asked to participate in the evaluation.

On the other hand, for the purpose of this work, a non-biased comparison between workflows is a good qualifier of the work done and users that have never used ePaper should give a relevant input for this study as well as they can represent potential new users’ behaviour. In order to limit the universe of testers and consequently have data as close to reality as possible, was studied the typical set of skills of ePaper’s administrators (who would ultimately be the users of these tools). In general, users come from diverse backgrounds, being the major common ground that administrative software such as ePaper and file management is a core part of their jobs. Age or gender does not follow any patterns and so was not taken into consideration (even though variety on both aspects was a concern) and the only
background checks for the testers were their job (which implies the how much the person is familiarised with this administrative software) and the frequency of use of Windows File System. The final goal was to find testers with the same type of intuition as real users. A set of 10 testers that did not know ePaper was collected making a total of 20 testers.

In order to ensure that the order in which each tool was used was not a decisive factor on the final results, half the population that did not know ePaper executed the tasks with ePaper in the first place and the other half used the new tool initially. For users that knew exactly the existing workflows the order of execution was considered irrelevant.

### 5.4.2 Results and Analysis

Given the variety of tests executed and results obtained, different analysis had to be made. Despite this, it can be concluded for each and every test that the results were very satisfying.

The results can be majorly divided in two subgroups: the testers that had prior knowledge of ePaper and the ones having their first contact with both systems. Obviously, being this a comparison between two systems, it is extremely relevant to know whether testers knew ePaper in advance or not. As expected, results are quite distinct for these two groups and it is worth to analyse them separately.

#### After-Scenario Questionnaire

Every question of each task is answered with a score from 1-5 regarding how easy it is to complete the task, the time spent, and the support information were given. And average of these results is presented in figure 5.10.

![Task 1 scores](image)

(a) Task 1 scores

![Task 2 scores](image)

(b) Task 2 Scores

![Task 3 Scores](image)

(c) Task 3 Scores

Figure 5.10: ASQ: Tasks’ score

Task 1 results are illustrated in figure 5.10(a) and it is the task where the results for the eFormManager are lower (4.07). This could be expected because, in fact, this is the most difficult task to execute - the extra complexity present on this task allows for decreasing all the complexity on the rest of the
workflows. Despite this, there is a substantial improvement when comparing ePaper (3, 1.87) with eFormManager (4.67, 4.07) for both populations of testers. It is interesting to highlight that users who know the complexity of existing functionalities gave a significantly higher score to the new system.

Task 2 (figure 5.10(b)) is the task where there is less discrepancy between both systems. This does not mean that the results are negative, once scores for the eFormManager are above 4.5 out of 5 and it can be explained by the following reason: the execution of this task in the ePaper application consisted only in the creation of a ZIP file in the File System and its renaming which is in fact quite simple. For this task, major improvements should be observed when analysing the number of errors occurred in its execution.

Task 3 has very satisfying results for eFormManager (4.73, 4.53) (figure 5.10(c)) and it is where there can be observed the biggest improvement in terms of classification. When compared to ePaper, the new system has some significant differences, namely in the automation of package delivery and it justifies the user satisfaction.

In all the tasks, eFormManager had better scores than ePaper which is a very satisfactory result.

System Usability Scale

For both groups of testers, results show a better score for the eFormManager which is the most relevant fact. As predicted, the SUS score for ePaper is not satisfying and, even though its absolute value almost doubles for users that are familiar with it, both scores correspond to an F on the benchmark used. For the eFormManager, scores are quite similar for both groups and they float between an A+ and an A grade.

Scores of 83 and 86.5 are extremely satisfying scores for the usability of eFormManager, mostly
when compared with the scores of ePaper which are considered poor in this task.

It is understandable that users that knew ePaper in advance are more aware of its workflows and therefore gave it a better score than users who did not know ePaper. But this only increases the relevance of the improvement noticed when comparing scores from ePaper and eFormManager.

Regarding the usability of the eFormManager, there is a clear improvement considering the previous implementation of such functionalities and, therefore, it can be said the objective was achieved.

**Task Completion Times**

The time users spent completing the tasks can be considered a relevant measure for the users facility to complete tasks. For each task, that time was measured and the average of every user is presented in figure 5.12. Obviously, there are many factors that may influence the time taken by a user to complete a task and those will be mentioned in this section.

![Figure 5.12: Time Taken for each Task](image)

At first sight, can be observed that for each and every task the average execution time was smaller for eFormManager which is a positive result. Curiously though, ePaper users spent more time using eFormManager than new users. During test execution, ePaper users had more comments and questions towards the new system, its different workflows and some conceptual changes and that can justify the extra time they spent. As new users were not biased into any workflow and the discrepancy between the times they spent in each system is remarkable, this is an extremely satisfying result.

**Errors occurred**

During test execution, an observer noted down the errors users made when following the guide. On purpose, users were only notified of their errors when they led to a point in which the user could not understand why their actions did not give the expected output.

The following image and table display the number of errors that occurred during the execution of the tests for both systems. In order to simplify its analysis, the errors were aggregated in 4 categories:
• Wrong ZIP creation: this error category includes all the errors associated with the creation of a ZIP file, namely, the forms that it contains and the folder where it was placed.

• Wrong ZIP naming: even though, the name of the ZIP is directly related with its creation, this was a very common error and so it is displayed in a different category.

• Wrong File Selection: when through the interface users have to pick a file (in a dialog box or drop down list).

• Wrong Form Input: when users inserted wrong information in the system through miss clicking or writing mistakes.

Figure 5.13 shows the number of errors by type that happened during the tests. The eFormManager presents a significant improvement regarding error occurrence.

The last line of table 5.2 has the number of errors the system detected and consequentially prompted the user.

The fields in the table marked with “-“ correspond to non applicable situations, for example, eFormManager not having any contact with ZIP files. These were errors that were removed because the tasks started being executed by the application and not by the users.
Table 5.2: Number of errors in each task per system

<table>
<thead>
<tr>
<th>Error Type</th>
<th>ePaper</th>
<th>eFormManager</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>T1</td>
<td>T2</td>
</tr>
<tr>
<td>Wrong ZIP Creation</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>Wrong ZIP Naming</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>Wrong File Selection</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>Wrong Form Input</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>Total</td>
<td>68</td>
<td>5</td>
</tr>
<tr>
<td>Warnings Prompted</td>
<td>12</td>
<td>5</td>
</tr>
</tbody>
</table>

As table 5.2 shows, ePaper usage is associated with many human errors (which in a way is a validation of the problem that led to this work) while, in that matter, the solution developed presents way more satisfying results.

Another relevant data from this results is the number of errors that each system detected. Not only eFormManager had fewer errors but also all the errors were detected by the system. The version submission validation feature was extremely helpful in this topic since the most common error was users choosing the wrong file as new version and the verification performed alerted the users for a potentially wrong insertion allowing a clear confirmation. The majority of improvements was cause by the fact that users stopped executing the tasks but there were many improvements made by the user interface, namely regarding the Wrong File Selection and Wrong Form Input errors.

On the other hand, ePaper has an enormous percentage of undetected errors once it is not ready to verify the operations executed exclusively inside Windows File System (where the majority of errors occurred).

The number of errors and their detection is an extremely positive measure of success for eFormManager considering that one of the initial goals was to increase control and change error related workflows making them less vulnerable.

### 5.5 Openness Evaluation (N.F.R.4)

A qualitative evaluation can be performed regarding how easy it would be to integrate eFormManager with other systems. As described in section 4.6.1 REST architecture provides some characteristics to the system built that were considered to be crucial in order to ease the integration of eFormManager in other systems.

The key point about this architecture is that it provides a clear interface of endpoints that can be used not only by JavaScript Web Browser clients but also by any other entity with the right privileges that has access to those endpoints.

This means that another system with its own interface can make HTTP requests to this system and perform actions in its database as well as fetch data or files from it. Considering this, REST endpoints
can be used as an API that provides undoubtedly a huge degree of openness to this system.

If we consider the specific example of ePaper and its two major modules, eFormManager should be fully integrated in the Back Office module but it should easily be called from the Front Office module through HTTP Requests in order to fetch the organisational tree with the requests and also the packages for download.
6

Conclusions and Future Work
This chapter contains a description of the conclusions taken from the development of this work and from its results by presenting an analysis on what was done and what is the value of the work in the given context. This chapter also presents an overview is given on how this work can be followed in the future and which steps should be taken in order to increase its relevance and value.

6.1 Conclusions

Public administrations are constantly increasing their investment in tools that automate document management related jobs, not only tasks directly related with files themselves, but also their distribution for the population and the following reception. Online and automated help desks are becoming common and the features they present are growing in complexity and variety.

The application developed in this work - eFormManager - aims to increase the automation of Portuguese Municipalities’ documents life cycle since they are created until they are deployed for citizens to submit them online. This process became more efficient by reducing the human intervention needed and supervising the remaining human action through algorithms that validate tasks executed by users prompting them if potential errors are detected.

The work was developed on top of concepts and paradigms of the ePaper system by Mind on which the application built is to be integrated. ePaper is used for urban management by Portuguese Municipalities having tools for PDF form management and their distribution to the population who can also use ePaper’s public platform to submit their filled requests.

Even though the final goal of this work is to integrate eFormManager into Mind’s application ePaper, an independent web tool was built, tested and evaluated. Since this tool is to be integrated with another system, it was built with an architecture that provides isolation between the front-end, the back-end and its database so that each one of these components could be re-used independently.

Despite being built within the context of the ePaper application and having certain requirements of features and specifications to follow, the tool presents some extra features such as document comparison and changes’ tracking. These features are described throughout this report and increase its value beyond objectives initially defined.

In order to evaluate this tool (its features and interfaces), an evaluation survey was performed where 20 participants compared the use of the new tool with the previous workflows (in ePaper). Statistical results were extremely satisfying since usability and user workflows results are indicative of significant improvements. The following results should be highlighted:

- User satisfaction results were clearly better for eFormManager than ePaper,

- The time spent on the execution of the same tasks was smaller when using eFormManager,
• The number of errors occurred during the execution of the tests was extremely smaller for eForm-Manager when compared with ePaper.

Besides, the evaluation survey was very relevant because of the inputs received from the participants and led to some corrections and improvements of the application.

6.2 Future Work

The final result of this master thesis consisted on a web application tool that presented some functionalities of document versioning, management and deployment. Although this tool’s value is limited on its own, its integration with a bigger system (which is the obvious next step to follow) will increase greatly value of the work done. When integrated with the ePaper application, it will allow Municipalities to automate the management of the forms they provide to the population.

At Mind, the goal is to make the integration of the tool where all its business logic can be re-used and its interfaces will only need to be adapted.

There are still several features that can be implemented in this tool to increase its value related with the graphical interface for the PDF Form Fields. Mapping Form Fields with the entities in the data model where field’s data will be stored will allow reading the automatic read of forms in the moment of their input. This is beyond the context of this work since its focus was the management of documentation during their construction and deployment phases and not the processing of citizens’ requests.

Despite what was previously mentioned, the integration of the work developed with ePaper has started and, after proper testing and documentation, this should be ready to be released.
Bibliography


A

System Evaluation Guide
Guia de Avaliação e Comparação de Aplicações

Miguel Verdelho

Dezembro 2019

1 Introdução

Serve este guia para a avaliação e comparação de duas ferramentas distintas na execução de tarefas semelhantes. Como tal, possui um conjunto de instruções que ajudam o utilizador a executar as mesmas 3 tarefas com as duas aplicações diferentes seguindo-se a execução de questionários sobre as tarefas executadas.

Desde já muito obrigado pela sua participação neste estudo.

2 Contexto

O contexto específico do questionário é o seguinte: imagine-se que o utilizador gere uma câmara municipal e tem de providenciar aos seus cidadãos um conjunto de formulários que eles precisarão de preencher no caso de precisarem de fazer um pedido à câmara.

O que se pretende é disponibilizar online é o formulário referente a um pedido - Pedido de Início de Obra, assim como adicionar documentos auxiliares necessários para este pedido. Mais tarde há uma decisão da câmara de retirar um dos campos do formulário (nº de telefone fixo) por achar obsoleto e torna-se necessário substituir o documento e colocar o mais recente disponível online.
3  Aplicação 1 - ePaper

3.1 Tarefa 1: Disponibilizar um ficheiro ZIP com o formulário para o pedido

1. No ambiente de trabalho, copiar o ficheiro Formulario_Test.pdf para a pasta Atendimento PDFs.

2. No ambiente de trabalho, copiar o ficheiro Formulario_Test.pdf para a pasta ePaper PDFs.

3. Ir ao Menu Mapeamentos e clicar em Criar Formulário

4. Para preencher este Formulário, usam-se os valores abaixo:

Ficheiro: Formulario_Test.pdf
Descrição: Início de Obra
Valor Identificador: 1
Tipo de processo: [1]OB – Processo obra
Tipo de Requerimento: [1]Alteração_Dados_Pessoais
Tipo de Documento: [DOCREQ] Requerimento
Click em Criar
5. Ir ao Menu de assistente de Downloads

6. Expandir a árvore e selecionar Início depois clicar em **Criar resposta** (para adicionar um novo tipo de pedido)

Descrição: **Pedido de Início de Obra**

Formulário: **Formulario_Teste.pdf**

Click **Guardar alterações**
7. Voltar à pasta **Atendimento PDFs** um ficheiro do tipo Zip com o ficheiro **Formulario_Test.pdf** e com o nome **Formulario_Test.zip**

8. Ir ao separador **Atendimento** atualizar a página e garantir que o download do Zip funciona clicando no pedido criado.

Inquérito!
3.2 Tarefa 2: Associar mais documentos ao pedido e criar novo package
Pretende-se que os cidadãos possam fazer download também dos ficheiros auxiliares.

1. Ir ao ambiente de trabalho e copiar os dois ficheiros auxiliares `auxiliar1.pdf` e `auxiliar2.pdf` e copiá-los para a pasta *Atendimento PDFs*.

2. Criar um ficheiro Zip que contenha os dois ficheiros auxiliares e *Formulario_Test.pdf*. A pasta zip tem de ter o nome *Formulario_Test.zip*.

3. Verificar o Download na pasta do atendimento e o conteúdo (3 ficheros pdf)

Inquérito!

3.3 Tarefa 3: Atualizar o Formulário para a nova versão

1. Copiar o ficheiro *Formulario_Test_versao2.pdf* (que se encontra no ambiente de Trabalho) para a pasta *Atendimento PDFs*.

2. Ir ao Menu *Mapeamentos* e clicar em *Criar Formulário*
3. Para preencher este Formulário, usam se os valores abaixo:

**Ficheiro:** Formulario_Testo_versa2.pdf

**Descrição:** Inicio de Obra

Valor Identificador: 1

Tipo de processo: [1] OB – Processo obra

Tipo de Requerimento: [1] Alteração_Dados_Pessoais

Tipo de Documento: [DOCOPDF] Outros pdf Click em Criar

4. Ir ao Menu de assistente de Downloads
5. Seleccionar o *Pedido de Inicio de Obra* e alterar o Formulário para *Formulario_Teste_versao2.pdf*

6. Voltar à pasta *Atendimento PDFs* um ficheiro do tipo Zip com o ficheiro *Formulario_Teste_versao2.pdf* e com o nome *Formulario_Teste_versao2.zip*

7. Ir ao separador *Atendimento* e garantir que o download do Zip funciona clicando no pedido criado.
4 Aplicação 2 - DocManager

Abaixo estão as instruções que guião a execução das mesmas tarefas através de outra ferramenta.

4.1 Tarefa 1: Disponibilizar um ficheiro ZIP com o formulário para o pedido

1. Ir a Formulários e adicionar Novo Formulário

2. Preencher os campos com as instruções abaixo

Ficheiro: Formulario_Test.pdf
Código: 1
Descrição: Alteração de Dados
Tipo de Processo: [1] OB - Processo obra
Tipo de Requerimento: [1] Alteração_Dados_Pessoais
Tipo de Documento: [447] Planta com informação de cadastro - Informação emitir
Click em Adicionar Formulário
3. Voltar ao menu _Árvore_, click em **Novo Pedido**

Nome: **Pedido de Início de Obra**
Inserir no Grupo de Pedidos **Início**

4. Expandir árvore até ao pedido **Pedido de Alteração de Dados** e no menu de Formulários Associados escolher o botão “Editar”

5. Ao editar, escolher o formulário inserido **Formulario_Test.pdf** e **Guardar Alterações**
6. Ir ao separador *Atendimento* e garantir que o download do Zip funciona clicando no pedido criado.

![Image](image)

Inquérito!

4.2 **Tarefa 2: Associar mais documentos ao pedido e criar novo package**

1. Ir a Formulários e adicionar Novo Formulário

![Image](image)

2. Preencher os campos com as instruções abaixo

   **Ficheiro:** auxiliar1.pdf  
   **Código:** 1  
   **Adicionar Formulário**

   **Ficheiro:** auxiliar2.pdf  
   **Código:** 1  
   **Adicionar Formulário**
3. Expandir árvore até ao pedido **Pedido de Início de Obra** e no menu de Formulários Associados escolher o botão "Editar".

4. Selecionar os Formulários auxiliares e Guardar Alterações.

5. Ir ao separador **Atendimento** e garantir que o download do Zip funciona clicando no pedido criado.

**Inquérito!**

### 4.3 Tarefa 3: Atualizar o Formulário para a nova versão

1. No Menu **Formulários**, selecionar o formulário que queremos atualizar: **Formulario_teste.pdf**
2. Na tabela Versões, clicar no "+" para adicionar uma nova versão

3. Preencher os campos como mostrado abaixo

![Image of Nova Versão](image)

- **Ficheiro:** Formulario_Testes_versao2.pdf
- **Data de Ativação:** Data atual (dia de hoje)

4. Ir ao separador *Atendimento* e garantir que o download do Zip funciona clicando no pedido criado e verificando que se faz download da versão 2 do formulário.
Evaluation Survey
Inquérito de Avaliação e Comparação de Sistemas

* Required

1. Com que frequência usa o gestor de ficheiros do Windows?
   * Mark only one oval.
   - Diariamente
   - Com frequência
   - Esporadicamente
   - Nunca

2. Qual a sua profissão?

---

ePaper - Tarefa 1: Disponibilizar um ficheiro ZIP com o formulário para o pedido

3. Estou satisfeito com a facilidade de completar esta tarefa *
   * Mark only one oval.

   1 2 3 4 5

   Discordo totalmente  ○  ○  ○  ○  ○  Conordo totalmente

4. Estou satisfeito com o tempo que demorou a executar esta tarefa *
   * Mark only one oval.

   1 2 3 4 5

   Discordo totalmente  ○  ○  ○  ○  ○  Conordo totalmente

5. Estou satisfeito com a informação de suporte (mensagens, documentação) durante a execução da tarefa *
   * Mark only one oval.

   1 2 3 4 5

   Discordo totalmente  ○  ○  ○  ○  ○  Conordo totalmente

---

ePaper - Tarefa 2: Associar mais documentos ao pedido e criar novo package

6. Estou satisfeito com a facilidade de completar esta tarefa *
   * Mark only one oval.

   1 2 3 4 5

   Discordo totalmente  ○  ○  ○  ○  ○  Conordo totalmente
7. Estou satisfeito com o tempo que demorou a executar esta tarefa *
   *Mark only one oval.*
   1 2 3 4 5
   Discordo totalmente ○ ○ ○ ○ ○ Convido totalmente

8. Estou satisfeito com a informação de suporte (mensagens, documentação) durante a execução da tarefa *
   *Mark only one oval.*
   1 2 3 4 5
   Discordo totalmente ○ ○ ○ ○ ○ Convido totalmente

**ePaper - Tarefa 3: Atualizar o Formulário para a nova versão**

9. Estou satisfeito com a facilidade de completar esta tarefa *
   *Mark only one oval.*
   1 2 3 4 5
   Discordo totalmente ○ ○ ○ ○ ○ Convido totalmente

10. Estou satisfeito com o tempo que demorou a executar esta tarefa *
    *Mark only one oval.*
    1 2 3 4 5
    Discordo totalmente ○ ○ ○ ○ ○ Convido totalmente

11. Estou satisfeito com a informação de suporte (mensagens, documentação) durante a execução da tarefa *
    *Mark only one oval.*
    1 2 3 4 5
    Discordo totalmente ○ ○ ○ ○ ○ Convido totalmente

**ePaper - Usabilidade**

12. 1. Gostaria de usar este sistema mais frequentemente *
    *Mark only one oval.*
    1 2 3 4 5
    Discordo totalmente ○ ○ ○ ○ ○ Convido totalmente

13. 2. Achei o sistema demasiado complexo *
    *Mark only one oval.*
    1 2 3 4 5
    Discordo totalmente ○ ○ ○ ○ ○ Convido totalmente
14. 3. Achei o sistema fácil de usar *
*Mark only one oval.*

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Conordo totalmente</td>
</tr>
</tbody>
</table>

15. 4. Acho que ia precisar de apoio técnico de uma pessoa para usar este sistema *
*Mark only one oval.*

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Conordo totalmente</td>
</tr>
</tbody>
</table>

16. 5. Achei que as várias funções do sistema estavam bem integradas *
*Mark only one oval.*

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Conordo totalmente</td>
</tr>
</tbody>
</table>

17. 6. Achei que havia demasiada inconsistência no sistema *
*Mark only one oval.*

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Conordo totalmente</td>
</tr>
</tbody>
</table>

18. 7. Acho que as pessoas iriam aprender rapidamente a usar este sistema *
*Mark only one oval.*

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Conordo totalmente</td>
</tr>
</tbody>
</table>

19. 8. Achei o sistema complicado de usar *
*Mark only one oval.*

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Conordo totalmente</td>
</tr>
</tbody>
</table>

20. 9. Senti-me confiante a utilizar este sistema *
*Mark only one oval.*

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Conordo totalmente</td>
</tr>
</tbody>
</table>

21. 10. Precisava de aprender muitas coisas antes de usar este sistema com facilidade *
*Mark only one oval.*

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Conordo totalmente</td>
</tr>
</tbody>
</table>
DocManager- Tarefa 1: Disponibilizar um ficheiro ZIP com o formulário para o pedido

22. Estou satisfeito com a facilidade de completar esta tarefa *
   *Mark only one oval.*

   1 2 3 4 5

   Discordo totalmente ☐ ☐ ☐ ☐ ☐ Convido totalmente

23. Estou satisfeito com o tempo que demorou a executar esta tarefa *
   *Mark only one oval.*

   1 2 3 4 5

   Discordo totalmente ☐ ☐ ☐ ☐ ☐ Convido totalmente

24. Estou satisfeito com a informação de suporte (mensagens, documentação) durante a execução da tarefa *
   *Mark only one oval.*

   1 2 3 4 5

   Discordo totalmente ☐ ☐ ☐ ☐ ☐ Convido totalmente

DocManager- Tarefa 2: Associar mais documentos ao pedido e criar novo package

25. Estou satisfeito com a facilidade de completar esta tarefa *
   *Mark only one oval.*

   1 2 3 4 5

   Discordo totalmente ☐ ☐ ☐ ☐ ☐ Convido totalmente

26. Estou satisfeito com o tempo que demorou a executar esta tarefa *
   *Mark only one oval.*

   1 2 3 4 5

   Discordo totalmente ☐ ☐ ☐ ☐ ☐ Convido totalmente

27. Estou satisfeito com a informação de suporte (mensagens, documentação) durante a execução da tarefa *
   *Mark only one oval.*

   1 2 3 4 5

   Discordo totalmente ☐ ☐ ☐ ☐ ☐ Convido totalmente

DocManager- Tarefa 3: Atualizar o Formulário para a nova versão
28. **Estou satisfeito com a facilidade de completar esta tarefa**
   *Mark only one oval.*

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Discordo totalmente</td>
<td></td>
<td></td>
<td></td>
<td>Conordo totalmente</td>
</tr>
</tbody>
</table>

29. **Estou satisfeito com o tempo que demorou a executar esta tarefa**
   *Mark only one oval.*

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Discordo totalmente</td>
<td></td>
<td></td>
<td></td>
<td>Conordo totalmente</td>
</tr>
</tbody>
</table>

30. **Estou satisfeito com a informação de suporte (mensagens, documentação) durante a execução da tarefa**
   *Mark only one oval.*

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Discordo totalmente</td>
<td></td>
<td></td>
<td></td>
<td>Conordo totalmente</td>
</tr>
</tbody>
</table>

**DocManager - Usabilidade**

31. **1. Gostaria de usar este sistema mais frequentemente**
   *Mark only one oval.*

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Discordo totalmente</td>
<td></td>
<td></td>
<td></td>
<td>Conordo totalmente</td>
</tr>
</tbody>
</table>

32. **2. Achei o sistema demasiado complexo**
   *Mark only one oval.*

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Discordo totalmente</td>
<td></td>
<td></td>
<td></td>
<td>Conordo totalmente</td>
</tr>
</tbody>
</table>

33. **3. Achei o sistema fácil de usar**
   *Mark only one oval.*

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Discordo totalmente</td>
<td></td>
<td></td>
<td></td>
<td>Conordo totalmente</td>
</tr>
</tbody>
</table>

34. **4. Acho que ia precisar de apoio técnico de uma pessoa para usar este sistema**
   *Mark only one oval.*

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Discordo totalmente</td>
<td></td>
<td></td>
<td></td>
<td>Conordo totalmente</td>
</tr>
</tbody>
</table>
35. **Achei que as várias funções do sistema estavam bem integradas** *Mark only one oval.*

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Discordo totalmente</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Concordo totalmente</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

36. **Achei que havia demasiada inconsistência no sistema** *Mark only one oval.*

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Discordo totalmente</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Concordo totalmente</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

37. **Acho que as pessoas iriam aprender rapidamente a usar este sistema** *Mark only one oval.*

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Discordo totalmente</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Concordo totalmente</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

38. **Achei o sistema complicado de usar** *Mark only one oval.*

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Discordo totalmente</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Concordo totalmente</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

39. **Senti-me confiante a utilizar este sistema** *Mark only one oval.*

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Discordo totalmente</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Concordo totalmente</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

40. **Precisava de aprender muitas coisas antes de usar este sistema com facilidade** *Mark only one oval.*

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Discordo totalmente</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Concordo totalmente</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Evaluation Results

This section contains a set of tables with the results of the survey in detail. Tables C.1, C.2 have the results for the ASQ while tables C.3, C.4 show the results for the SUS.

<table>
<thead>
<tr>
<th>Score</th>
<th>ePaper</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Score</td>
<td>Ease</td>
<td>Time</td>
<td>Supp</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>6</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>11</td>
<td>12</td>
<td>11</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>6</td>
</tr>
</tbody>
</table>

Table C.1: After Scenario Questionnaire for ePaper
### Table C.2: After Scenario Questionnaire for eFormManager

<table>
<thead>
<tr>
<th>Score</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ease</td>
<td>Time</td>
<td>Supp</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>13</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>7</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Score</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ease</td>
<td>Time</td>
<td>Supp</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>13</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>7</td>
<td>3</td>
</tr>
</tbody>
</table>

### Table C.3: System Usability Scale for ePaper

<table>
<thead>
<tr>
<th>Score</th>
<th>1.</th>
<th>2.</th>
<th>3.</th>
<th>4.</th>
<th>5.</th>
<th>6.</th>
<th>7.</th>
<th>8.</th>
<th>9.</th>
<th>10.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>9</td>
<td>0</td>
<td>9</td>
<td>1</td>
<td>6</td>
<td>0</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>0</td>
<td>9</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td>10</td>
<td>0</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>9</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>7</td>
<td>1</td>
<td>13</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>3</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>12</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>14</td>
<td>1</td>
<td>6</td>
</tr>
</tbody>
</table>

### Table C.4: System Usability Scale for eFormManager

<table>
<thead>
<tr>
<th>Score</th>
<th>1.</th>
<th>2.</th>
<th>3.</th>
<th>4.</th>
<th>5.</th>
<th>6.</th>
<th>7.</th>
<th>8.</th>
<th>9.</th>
<th>10.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>12</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>16</td>
<td>0</td>
<td>11</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>12</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>9</td>
<td>0</td>
<td>15</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>13</td>
<td>0</td>
<td>8</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>15</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>0</td>
<td>7</td>
<td>0</td>
<td>12</td>
<td>0</td>
<td>14</td>
<td>0</td>
<td>5</td>
<td>0</td>
</tr>
</tbody>
</table>