UNIVERSITY "POLITEHNICA" OF BUCHAREST
FACULTY OF ENGINEERING IN FOREIGN LANGUAGES
COMPUTERS AND INFORMATION TECHNOLOGY - INFORMATION ENGINEERING

DIPLOMA PROJECT

Project coordinator:

Conf. dr. ing. Andrei VASILĂŢEANU

Student:

Nándor-Mátyás IARCA

Bucharest
2021

UNIVERSITY "POLITEHNICA" OF BUCHAREST
FACULTY OF ENGINEERING
IN FOREIGN LANGUAGES
COMPUTERS AND INFORMATION TECHNOLOGY

Document Management System

Project coordinator
Conf. dr. ing. Andrei VASILĂŢEANU

Student:
Nándor-Mátyás IARCA

Bucharest
2021

UNIVERSITY "POLITEHNICA" OF BUCHAREST
FACULTY OF ENGINEERING
IN FOREIGN LANGUAGES
COMPUTERS AND INFORMATION TECHNOLOGY

Sistem pentru gestionarea documentelor

Project coordinator
Conf. dr. ing. Andrei VASILĂȚEANU

Student:
Nándor-Mátyás IARCA

Bucharest
2021

"POLITEHNICA" UNIVERSITY OF BUCHAREST
FACULTY OF ENGINEERING IN FOREIGN LANGUAGES
COMPUTERS AND INFORMATION TECHNOLOGY

Approved
Director of department:
Prof. dr. ing. Georg DRAGOI

DIPLOMA PROJECT THEME FOR:
Nándor-Mátyás IARCA

1. Theme title:
   Document management system
   Sistem pentru gestionarea documentelor

2. Initial design data:
   The project consists of implementing an algorithm which identifies templates of documents and extracts data from them, based on a self-learning process. This will be integrated in a web application which helps the user to upload and manage documents.

3. Student contribution:
   Bibliographical research
   Project analysis
   Algorithm implementation
   Web application development

4. Compulsory graphical material:
   Block scheme, functioning diagram, graphs

5. The paper is based on the knowledge obtained at the following study courses:
   Programming Languages, Databases, Object Oriented Programming, Software Development Methods, Web Applications Development, System engineering

6. Paper development environment:
   Microsoft Visual Studio, Microsoft SQL Server Management Studio, Microsoft Office

7. The paper serves as:
   Didactic purposes

8. Paper preparation date:
   June 2021

Project coordinator

Conf. dr. ing. Andrei VASILĂȚEANU

Student:

Nándor-Mátyás IARCA

Iarca

Academic Honesty Statement

I, Nándor-Mátyás IARCA, hereby declare that the work with the title “Document Management System”, to be openly defended in front of the diploma theses examination commission at the Faculty of Engineering in Foreign Languages, University "Politehnica" of Bucharest, as partial requirement for obtaining the title of Engineer is the result of my own work, based on my work.

The thesis, simulations, experiments and measurements that are presented are made entirely by me under the guidance of the scientific adviser, without the implication of persons that are not cited by name and contribution in the Acknowledgements part.

The thesis has never been presented to a higher education institution or research board in the country or abroad.

All the information used, including the Internet, is obtained from sources that were cited and indicated in the notes and in the bibliography, according to ethical standards. I understand that plagiarism is an offense and is punishable under law.

The results from the simulations, experiments and measurements are genuine. I understand that the falsification of data and results constitutes fraud and is punished according to regulations.

Nándor-Mátyás IARCA 23/06/2021

Table of Contents

1 Introduction *6*

1.1 Documents everywhere **6**

1.2 Motives for using a Document Management System **7**

1.3 Advantages of automated data extraction **8**

1.4 Structure of the paper **9**

2 State of the art *10*

2.1 Scientific state of the art **10**

2.1.1 Optical Character Recognition 10

2.1.2 Document classification 11

2.2 Commercial state of the art **12**

2.2.1 SENSEtask 14

2.2.2 DxInvoice 15

2.2.3 Nanonets Enterprise Automation 16

2.2.4 Azure Form Recognizer 18

2.3 Summary and comparison of existing approaches **21**

3 Requirements analysis *24*

3.1 Functional requirements **24**

3.1.1 Actors and agents 24

3.1.2 Software use case diagram 24

3.1.3 Use case functional requirements 25

3.1.4 Use case descriptions and system sequence diagrams 28

3.1.5 Activity diagrams 38

3.1.6 Operation contracts 40

3.2 Non-functional requirements **41**

4 Design *42*

4.1 Architecture **42**

4.1.1 Monolithic architecture 42

4.1.2 Package diagrams 43

4.1.3 Deployment 43

4.2 Detailed design **45**

4.2.1 Design Class Diagrams and Design Sequence Diagrams 45

4.2.2 Database structure 55

5 Implementation *58*

5.1 Technologies **58**

5.1.1 Development 58

5.1.2 Microsoft Azure Cognitive Services 59

5.2 Plug-in **59**

5.2.1 Text Extraction Phase 61

5.2.2 Properties Mapping Phase 62

5.2.3 Training Phase 64

5.2.4 Prediction Phase 66

5.3 Web application **68**
5.3.1 Walkthrough for employees 69
5.3.2 Walkthrough for administrators 73
6 Validation ***76***
Conclusions ***77***
References ***78***
Glossary ***81***

Table of Figures

Figure 1: Structured vs Unstructured Data **7**

Figure 2: Manual vs Automatic Data Extraction **9**

Figure 3: Existing document classification methods **11**

Figure 4: SENSEtask workflow **15**

Figure 5: DocProcess BEA platform components **16**

Figure 6: Nanonets new model creation steps **17**

Figure 7: Nanonets OCR service example **18**

Figure 8: Form Recognizer Labeling Tool **21**

Figure 9: Software Use Case Diagram **24**

Figure 10: Log In System Sequence Diagram **28**

Figure 11: Change Password System Sequence Diagram **29**

Figure 12: Upload Documents System Sequence Diagram **30**

Figure 13: View Documents System Sequence Diagram **31**

Figure 14: Filter Data System Sequence Diagram **31**

Figure 15: Export Data System Sequence Diagram **32**

Figure 16: Validate Document System Sequence Diagram **33**

Figure 17: Train Template System Sequence Diagram **34**

Figure 18: Delete Template System Sequence Diagram **35**

Figure 19: Manage Document Models System Sequence Diagram **36**

Figure 20: Manage Users System Sequence Diagram **37**

Figure 21: Validate Document Activity Diagram **38**

Figure 22: Manage Document Models Activity Diagram **39**

Figure 23: DocsMetaExtractor Architecture **42**

Figure 24: DocsMetaExtractor Package Diagram **43**

Figure 25: DocsMetaExtractor.Web Package Diagram **43**

Figure 26: DocsMetaExtractor.Web Publish Configuration in Visual Studio **44**

Figure 27: DocsMetaExtractor.DataModel Publish 44

Figure 28: Log In Design Class Diagram 45

Figure 29: Log In Design Sequence Diagram 45

Figure 30: Change Password Design Class Diagram 46

Figure 31: Change Password Design Sequence Diagram 46

Figure 32: Upload Documents Design Class Diagram 47

Figure 33: Upload Documents Design Sequence Diagram 47

Figure 34: View Documents Design Class Diagram 48

Figure 35: View Documents Design Sequence Diagram 48

Figure 36: Filter Data Design Class Diagram 49

Figure 37: Filter Data Design Sequence Diagram 49

Figure 38: Validate Document Design Class Diagram 50

Figure 39: Validate Document Design Sequence Diagram 50

Figure 40: Train Template Design Class Diagram 51

Figure 41: Train Template Design Sequence Diagram 51

Figure 42: Delete Template Design Class Diagram 51

Figure 43: Delete Template Design Sequence Diagram 52

Figure 44: Manage Document Models Design Class Diagram 52

Figure 45: Add and Modify Document Models Design Sequence Diagram 53

Figure 46: Delete Document Model Design Sequence Diagram 53

Figure 47: Manage Users Design Class Diagram 54

Figure 48: Add and Edit Users Design Sequence Diagram 54

Figure 49: Delete User Design Sequence Diagram 55

Figure 50: Plug-in Database ERD 55

Figure 51: Web App Database ERD at Deployment 56

Figure 52: Purchase Orders Custom Properties 56

Figure 53: Web App Database ERD at Runtime 57

Figure 54: Technologies Used 58

Figure 55: App Settings Structure 61

Figure 56: Text Extraction Phase - Straightened Invoice Example 63

Figure 57: Example of Fixed Properties 64

Figure 58: Example of Dynamic Property 65

Figure 59: JSON Example of a Trained Model 66

Figure 60: DocsMetaExtractor – Login Popup 69

Figure 61: DocsMetaExtractor – Homepage 69

Figure 62: DocsMetaExtractor – Change Password Popup 70

Figure 63: DocsMetaExtractor – Documents List View 70

Figure 64: DocsMetaExtractor – Documents List View Filtering Example 71

Figure 65: DocsMetaExtractor – Refresh & Export List 71

Figure 66: DocsMetaExtractor – Excel Export 71

Figure 67: DocsMetaExtractor – Processing Document Notification 72

Figure 68: DocsMetaExtractor –Document Properties Mapping 72

Figure 69: DocsMetaExtractor –Confirmed Document View 73

Figure 70: DocsMetaExtractor –Document Model Administration 73

Figure 71: DocsMetaExtractor –Adding New Document Model 74

Figure 72: DocsMetaExtractor –New Document Model List 74

Figure 73: DocsMetaExtractor – Save/Edit User Popup 75

Figure 74: Handwritten English Text Recognition 75

Figure 75: Arggo Consulting's Invoice Processing System Statistics 76

1 Introduction

1.1 Documents everywhere

In these days, as the world is modernizing, digitalization gets more and more attention from the business domain. Companies are very trying to digitalize their workflows, tasks and processes to increase their productivity. In this way they could save a lot of processing time and resources needed to finalize jobs, that are currently done manually.

Each business, independently of their working area, processes documents. Every company, no matter of its size, needs to keep official records of its activities. This is the reason why documents have such an important role, they serve as legal evidence. For example, contracts, invoices, statistics, orders or payments. These documents must be at hand for any further investigation.

As business are growing, the number of documents processed are exponentially growing with them as well. Businesses are not only receiving documents, but they are also issuing them. This means a huge number of files that need processing and maintenance from a lot of sources. Companies try to manage these documents in an efficient way in order to have full control over them. Therefore, each business has its system to store, classify, track and manage documents. These systems can work in two different domains based on their environment, they can be physical systems, for example filing cabinets, or they can be electronic systems. Nowadays, most of the companies try to use an electronic system because it has many advantages. It saves a lot of resources as no more paper is used and searching for documents is easier and way faster. These systems are called Document Management Systems (DMS).

Another vital aspect for businesses is data extraction, everything is moving around data, a company cannot exist without it. Data-driven decision making takes an enormous role in a business. Good strategic choices are a necessity for businesses to evolve, and this can be done only by constantly actualizing, analyzing and reporting data. Data comes in a huge variety of formats, one way would be through documents. Businesses usually transform the unstructured data from documents to structured data, in order to analyze them faster and more accurately. This is because structured data has a well-defined type, it can be easily stored in a huge amount and can be filtered as it is needed. Statistics can be easily derived from it and analyzed in order to take good business strategic decisions.

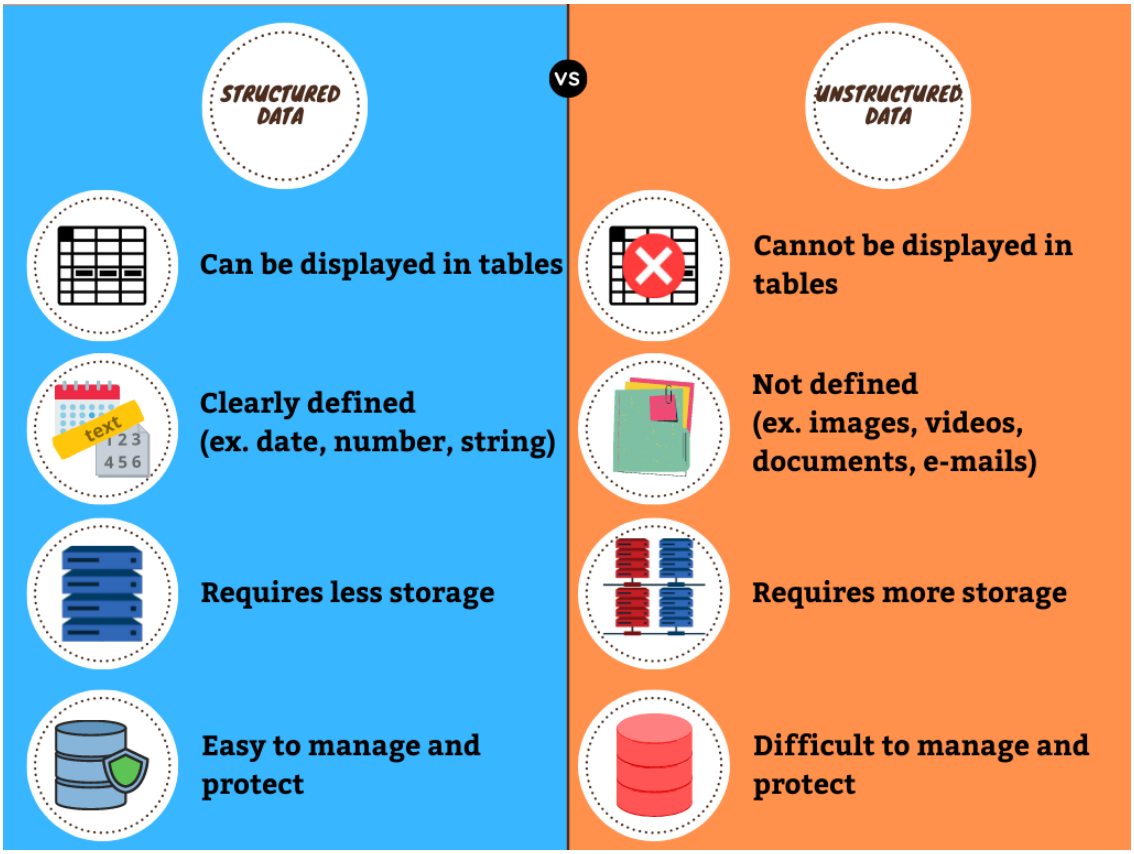


Figure 1: Structured vs Unstructured Data

1.2 Motives for using a Document Management System

As it is clear businesses need to keep records of the documents processed by them. Each document must be analyzed and retrieved as fast as possible and the number of these files can reach a very large number. This is why a DMS is the best way to keep everything in order.

A DMS is responsible for tracking, sorting, managing and securely storing documents. It also replaces repetitive unsatisfactory tasks, that have been done by employees, with a higher performance and accuracy rate. In this way allowing companies to focus their workers on other, more important jobs. Therefore, the productivity is increasing, the costs decrease and the business can become more profitable.

It is evident now that businesses profit if they have a properly implemented DMS. On the other hand, they have to process each received and emitted document. Data extraction is a key to growth. As it is mentioned before, unstructured data from files are transformed to structured data. Automatizing this procedure brings a lot of advantages for companies. A DMS which provides automatic data extraction eases a lot the life of a business.

Using such a system, provides an efficient way to analyze and work with documents. It helps a business to grow significantly faster and to assign tasks to employees that really matter.

1.3 Advantages of automated data extraction

Automatic data extraction brings a lot to a company, the most upfront advantages is that it is a fast, reliable and cost-efficient solution in comparison to manual extraction. Currently the majority of businesses are using manual data extraction, this has to change.

The first huge advantage is cost related. It is costly to manually extract data and as the business grows more and more employees are needed to process documents. Automatizing this process saves money by replacing long and repetitive tasks, also it is easy to scale as the needs change.

Another essential aspect is time, for businesses it is important to not waste time. Automatic extraction is faster, with it unstructured data from different kind of files are converted directly to structured data saving a lot of processing time. Moreover, employees can make mistakes and take extra time to correct them if they even notice the error.

Continuing the idea that people make mistakes, machines do not get tired or distracted, they are more reliable for this kind of tasks. Working daily with a huge amount of information leads to a high change of human errors. Most common mistakes are incomplete or missing records, duplicates or simply incorrect data. Automatization reduces the errors and improves the accuracy overall.

Besides the fact that it significantly reduces errors, it also increases the employees productivity. Manual data extraction does not require high level skills, it is a repetitive and tiring task and it can demotivate workers. Replacing this process gives the possibility to employees to work on other, more meaningful tasks which provide more satisfaction and increases motivation.

Lastly, when data is gathered fast and is at hand, decisions can be concluded faster. Furthermore, by having more data available in less time, with better accuracy leads to better decisions.

It can be observed that automatic data extraction would lead to an overall increase in productivity. These kinds of demotivating, low level manual tasks are a waste of time, which is also a waste of money and this is an issue for businesses. Automatization is profitable in both, short and long terms, it has many advantages.

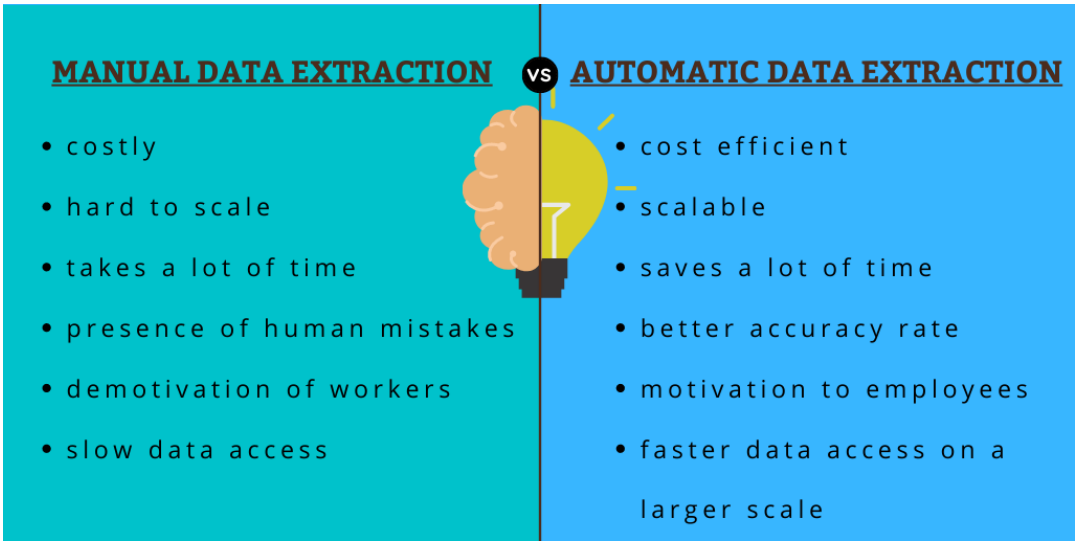


Figure 2: Manual vs Automatic Data Extraction

1.4 Structure of the paper

The purpose of this paper is to present an approach of DMS with automatic data extraction by proposing a web-based application, called DocsMetaExtractor, that integrates a plug-in to data extraction.

To describe the project this document is structured into seven main chapters as follows:

- Introduction
- State of the Art
- Requirements Analysis
- Design
- Implementation
- Validation
- Conclusion

Where it is needed additional UML diagrams, tables and figures are present to help in understanding the features and functionalities of the project. At the end of the documentation the references are mentioned and a glossary that describes every abbreviation used.

The figures are created by using Canva (https://www.canva.com/), Draw.io (https://app.diagrams.net/) and Visual Paradigm (https://www.visual-paradigm.com/).

2 State of the art

2.1 Scientific state of the art

The main challenge in this project is the document processing. How to get data from documents, how to manipulate them and how to implement the process so it would be easily used or integrated in other applications.

The majority of companies get their documents in Portable Document Format (PDF), therefore in this project these kinds of documents are analyzed. There are two types of PDF documents, real ones and scanned ones. A real PDF file is a native, digitally created document, they are also called “true” PDFs. These contain metadata, the text or images present in the document and are searchable. Scanned PDFs are “image-only” documents, they are not searchable. These files, where existent physically and were scanned by using a printer in order to digitalize them. There is no metadata from where the content of the document can be extracted. Therefore, Optical Character Recognition (OCR) is used to get information about the content of the file.

2.1.1 Optical Character Recognition

OCR, also known as Text Recognition, is a methodology to transform visual data to computer-editable text. OCR programs can differentiate handwritten text from machine printed text. There are also some special cases, for example historical languages, hieroglyphs, where OCR extracts and matches these characters. To accurately recognize characters, high end technologies, like Artificial Intelligence (AI), are used. The first step of an OCR program is to identify the format of text in the input image. Next, segmentation is applied, the input data is divided into pages, pages are divided into separate blocks. These blocks may or may not contain text, they can contain images or empty spaces. After that, the blocks with text are distributed into words and finally to characters. Characters are represented as vectors and a classifying happens to recognize the character. OCR pipelines usually also include image enhancement methods to reduce noise and provide a quality picture for better accuracy. Nowadays, OCR services also detect the orientation of the input and the angle of the text.

In this project, to extract text from real PDFs the metadata is used and for scanned documents an OCR service. To train and to predict data a generic algorithm will be implemented.

2.1.2 Document classification

Document classification became vital due to the rapid growth of the documents. In the article *Document image classification: Progress over two decades*[1] the current document image classification methods are listed and presented that are used by Document Image Processing (DIP) systems. They identified four main categories: textual-based, structural-based, visual-based and hybrid methods. These can be broken down to other categories as showed in *Figure 3* bellow.

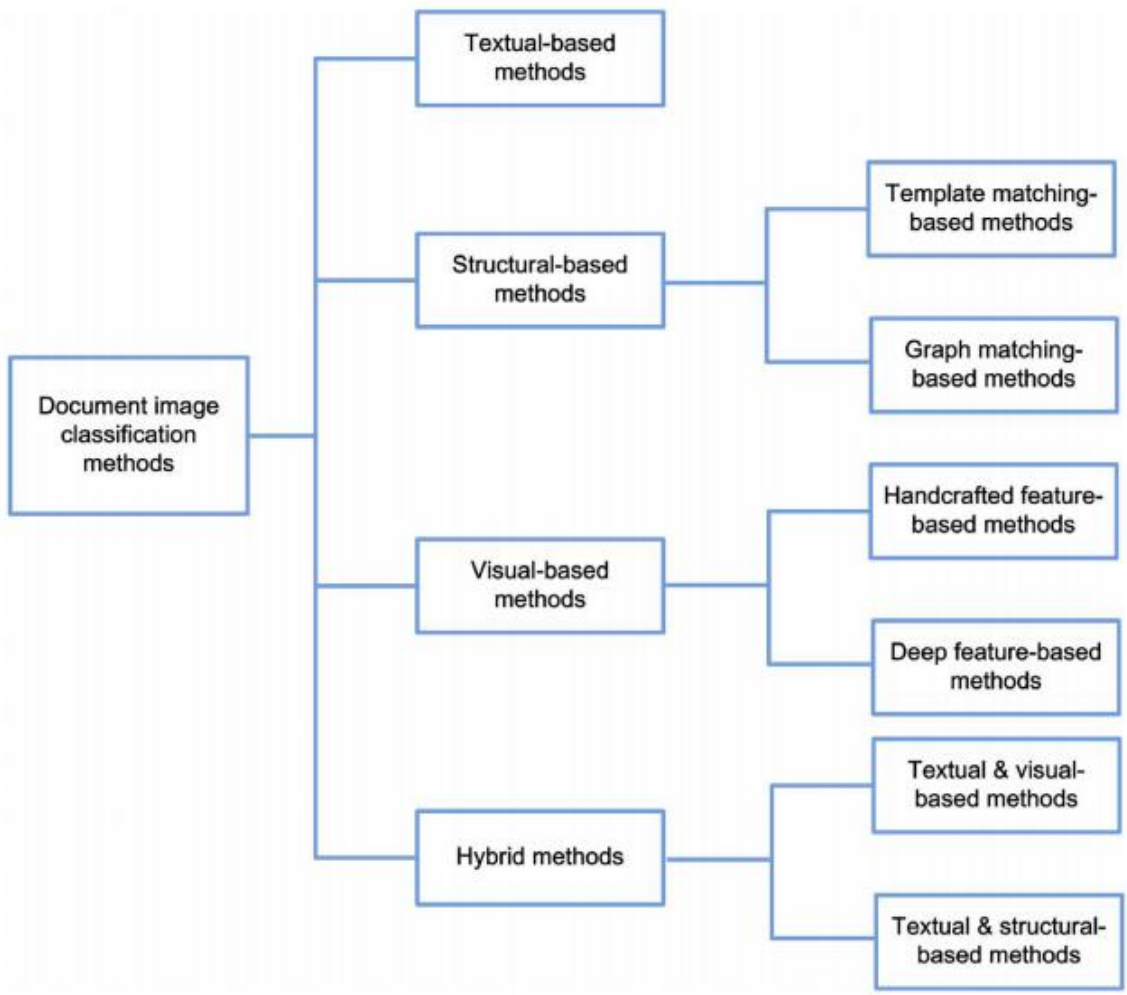


Figure 3: Existing document classification methods[1]

Textual-based methods are using the content of documents. In most cases an OCR service returns the text, which is then analyzed to categorize the document into predefined categories. These methods are completely depending on the textual content of a document, if the text is unusual or contains mistakes it is harder to identify a template. Moreover, it is important the language, the predefined models must include data for the language of the document else no matching can be done.

[1] *Li Liu, Zhiyu Wang, Taorong Qiu, Qiu Chen, Yue Lu, Ching Y. Suen. (2021). Document image classification: Progress over two decades. Neurocomputing, 453, 223-240.*
https://www.sciencedirect.com/science/article/abs/pii/S0925231221006925?via%3Dihub

Structural-based methods are analyzing the documents structure only. There are two subtypes, template and graph matching methods. Template matching methods are using manually predesigned templates. For matching, a metric is measuring the similarities between the templates and the document. These methods are mostly recommended for documents with fixed layouts because they depend on the orientation, angle and quality of the image. Graph-matching methods are dividing the document into parts, like title, body, footer. These are represented as a graph, where the nodes are the components in which the document is divided and the edges representing the spatial relationship between them. To identify a template the maximum likelihood is taken into consideration between the trained graphs and the new document's graph. This solution is not affected by the quality of the image, but it comes with high computational costs.

Visual-based methods are depending on the appearance of the document. Handcrafted feature-based methods rely on identifying features and classifying them. Such a method is the Bag of Visual Words (BOVW) model. BOVW has two steps, vocabulary learning and image representation generation. Basically, the model counts the visual appearances of a word, based on the trained vocabulary set by comparing them visually as vectors. This can be extended to also store the location of the words to a better matching. Another visual-based method is the deep feature-based one. This uses convolutional neural networks, which have two parts, a feature extractor that extracts so called deep features and a classifier that uses those features. It is relevant to choose a suitable classifier as it majorly affects the performance.

Hybrid methods mix textual-based ones with the others. For example, textual and visual-based methods work on two levels, a textual and a visual. The text is extracted from the document and a classifier is run for the text and for its visual representation. There is also another possibility to embed the text into its corresponding image, in this way there is no need for two different classifying. Textual methods can also be mixed with structural ones, which uses the layout with its content to match documents.

2.2 Commercial state of the art

Nowadays, businesses try to automatize their work, to focus employees on jobs that only a human can do. To understand the advantages of automatic document processing a case study[2] published in 2019 is presented by Carrefour, a French multinational retail corporation, the biggest in Europe and leader in the Romanian market. Worldwide Carrefour is the eight largest retailer by its revenue.

[2] *DocProcess. (2019, August 30). Cum a reușit un mare retailer din România să reducă rata de eroare a facturilor sub 0.1%. Softlead. https://softlead.ro/noutati-it-c/cum-a-reusit-un-mare-retailer-din-romania-sa-reduca-rata-de-eroare-a-facturilor-sub-0-1.html*

As it can be deduced, they are working with a vast number of suppliers. During the years they realized that they are wasting a lot of time with slow and costly processes done manually. In one year, they processed 1,200,000 invoices manually, from these 45% contained at least one mistake. To correct these required a higher volume of workers and additional time. While waiting for corrections payments where in hold, this leaded to delayes, which affected the relationship with suppliers. To overcome these difficulties, Carrefour started to automatize their invoice processing to increase the speed and accuracy. They converted their physical invoices into electrical ones and managed them in a software system provided by DocProcess.

During four years of work, they got the following results:

- 2.6 million of electronically processed invoices.
- An error rate less than 0.1%.
- The time to process invoices was reduced with more than 50%.
- An integration rate more than 95% with suppliers.
- A better relation maintained with suppliers.

As it can be observed by analyzing the results, digitalization improved a lot the function of the company. Other statistics provided by SourceTask[3] showed the followings about common offices:

- Over 20% of daily loss is attributed to document issues.
- An average of 18 minutes is needed to find a document.
- 80-90% of data is unstructured and it is doubling every 3 years.
- A document gets photocopied 19 times in its lifetime.
- It takes on tree to produce 10,000 A4 pages, that is how much an employee consumes in one year average. That means one tree per employee per year.

These statistics present that manual document processing is a problem, it holds back business, it is time consuming and cost inefficient. Moreover, automatization leads to a greener office by reducing the number of physical papers, which would be a major impact on the environment.

[3] *SenseTask. (n.d). Real benefits for your business. Retrieved June 16, 2021, from https://sensetask.com/benefits*

Going on, other applications, from the same domain, will be presented in order to study different types of approaches to the problem. Four applications will be analyzed as follows:

- SENSEtask
- DxInvoice
- Nanonets Enterprise Automation
- Azure Form Recognizer

For each software, the technologies used, the usage and the results will be presented. In this way a general idea is introduced about the current state of document processing.

2.2.1 SENSEtask

SENSEtask is a Romanian startup project launched in 2019 by SourceTask. It is an AI powered platform that extracts data from four types of documents, invoices, receipts, contracts and forms. This solution is not a DMS, but it is a good example of automated data extraction. It provides a solution of data extraction that can be used by other applications.

First step in SENSEtask's workflow is to upload a document. The document can be a scanned document or an image. Clients can upload files from several resources. They can send them directly from scanners, from a mobile application, from a cloud storage, from email or from local storage. Once the document is received, an AI-powered classification happens to sort the document in one of the four pre-trained types. Next, based on the identified category, pre-trained key values are extracted, each value having a confidentiality level. These are extracted by using Machine Learning (ML) with a defined set of key data, like invoice total, invoice date, company name or vendor name. The labels are represented as a Knowledge Graph that creates relationships between document types and key data. Unfortunately, these labels currently are only implemented in Romanian language, so it will not work for other languages. Finally, when the results are returned and reviewed an approval process is available for the user. The last thing is to export the data, for this there are available multiple formats or customizable templates.

SENSEtask can be accessed by using a web application, where clients can upload documents and review and validate them. This site has the role to present the application and how it is working, is a demo of the application. The components of the system can be reached by REST API calls in order to integrate it in an existing software program.

With this solution, SENSEtask managed to reduce processing costs by up to 80% and to reduce office paper usage up to 95%, by eliminating wasteful and redundant copies.

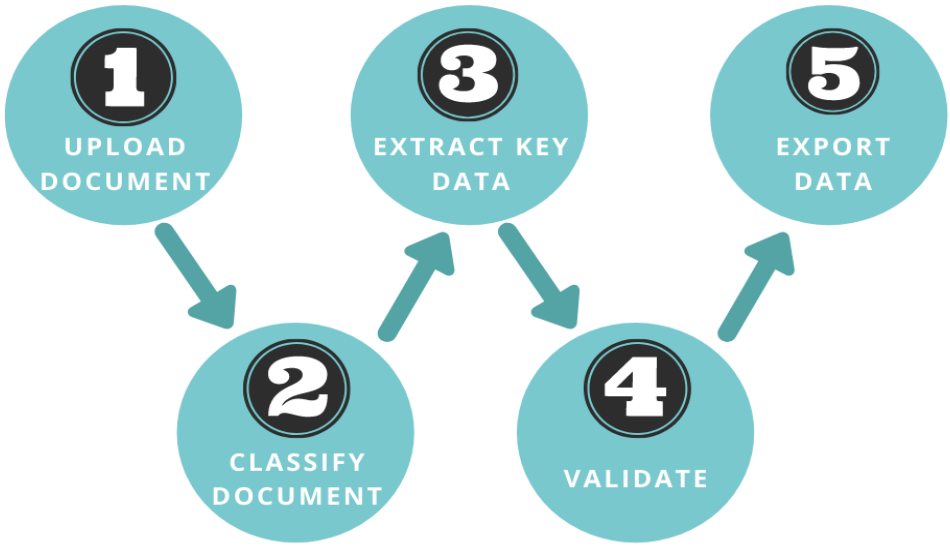


Figure 4: SENSEtask workflow

2.2.2 DxInvoice

DxInvoice is an e-invoice validation system provided by DocProcess, a company with the headquarter in Romania, which offers business process automations. Their definition of an e-invoice is a document with a structured electronic form, so scanned documents are not in this category[4].

DxInvoice is an end-to-end e-invoice automated cloud solution. It is an out-of-the-box integration with over 200 Enterprise Resource Planning (ERP) tools. This means that it can run by itself separately, or it can be integrated in an already existing ERP. DxInvoice is one of the products included in the beforehand mentioned case study with Carrefour. The suppliers, the invoice issuers, are allowed to create invoices directly in the system or to integrate their own system with it. In this way suppliers are also added to the environment and can track in real-time the status of their documents. The invoices must be generated with their associated XML format or in PDF with embedded XML, this provides a great visibility over the data. As the invoices are created by using the system there is no need for data extraction, because everything is known from each invoice. Therefore, a 100% automated solution can be achieved, without any human interaction. The whole process is controlled from the creation of an invoice to the time it is paid. When created the invoice is in pending, from pending it can be rejected or paid, this is the whole lifecycle of an invoice.

[4] *DocProcess. (2021, May 17). Mandatory E-invoicing in Saudi Arabia – Are you ready?.* https://doc-process.com/resources/e-invoicing-in-saudi-arabia

DocProcess achieved an 80% saving on invoice managements cost, 40 times more processed invoices per FTE/month and 97% less paper usage. This solution also requires the suppliers to board and to generate invoices based on a structured format, no more scanned or photographed documents are allowed.

Finally, in addition to DxInvoice, DocProcess has a Business Ecosystem Automation (BEA) solution, which includes other services, like DxContract, DxCatalog, DxOrder, DxLogistics, DxFin, DxArchive, to provide an advanced end-to-end automatization. The BEA system is useful for companies that can move their entire activities into an electronic ecosystem. This takes time and for many industries is hard to implement, the solution fits perfectly for retail businesses. DxContract is used to follow contract terms, DxOrder and DxLogistics ensures tracking orders from creation to delivery and invoicing. With DxCatalog the consistency of a product data through the whole process is verified and finally the data is archived by using DxArchive.

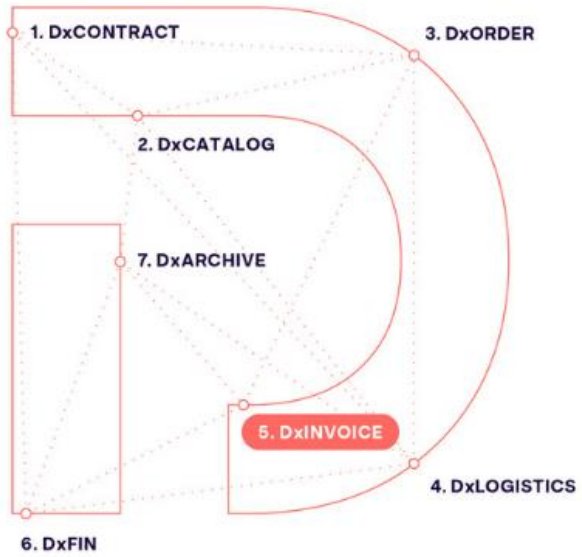


Figure 5: DocProcess BEA platform components[5]

2.2.3 Nanonets Enterprise Automation

Nanonets Enterprise Automation is a solution that provides document digitalization with deep learning. It transforms unstructured data into validated structured data, it is an example for data extraction and not for a DMS.

Nanonets supports any kind of documents, there are already some predefined templates like invoices, purchase orders, ID Cards or Mortgage forms. To define a new model four steps are taken in the following order, create, upload, annotate and review.

[5] *DocProcess. (2021, June 10). Achieve 100% touchless processing with end-to-end e-invoicing. https://doc-process.com/products_services/dxinvoice*

When creating a new model, the categories of text must be added, what the user wants to extract. Next, examples must be uploaded for each text category, a minimum of 50 images is required per category. After uploading the images, the user must annotate them, by drawing bounding boxes around the text and writing the value in it for each category. After this step the training itself happens, it might take 1-2 hours, the user will be notified when the data is ready to be reviewed.

Figure 6: Nanonets new model creation steps

These steps can be done by using a platform provided by Nanonets or an API. Also, Nanonets has language bindings in Shell, Ruby, Golang, Java, C# and Python. Nanonets API can be easily integrated with an existing software, it is compatible with the most ERP, CRM and RPA system. Moreover, it is available on-premises if customers want to run it in their own environment. The customers can directly import documents from their own system and also can directly export data back to it.

When a model is created and trained, the next thing to do is to use the API to upload an image or document. According to the defined model the data is extracted and can be exported in several formats, like CSV, JSON or DOC. A validation step can be also added before exporting the data to improve the prediction accuracy. With the Nanonets AI, the client extracts only the data defined by him. As it uses a custom model it is easy to re-train the model and to add new fields. Also, models can be trained for any languages and multiple languages too at the same time. With each document uploaded the Nanonets AI keeps learning and improves the model.

Nanonets OCR, also provides image quality improvements built in. It can recognize handwritten text and low resolution, blurry, noisy or shadowy images. These are extra features to handle any kind of input data. Bellow an example is presented of how the OCR service interprets a dirty document with handwriting and corrections.

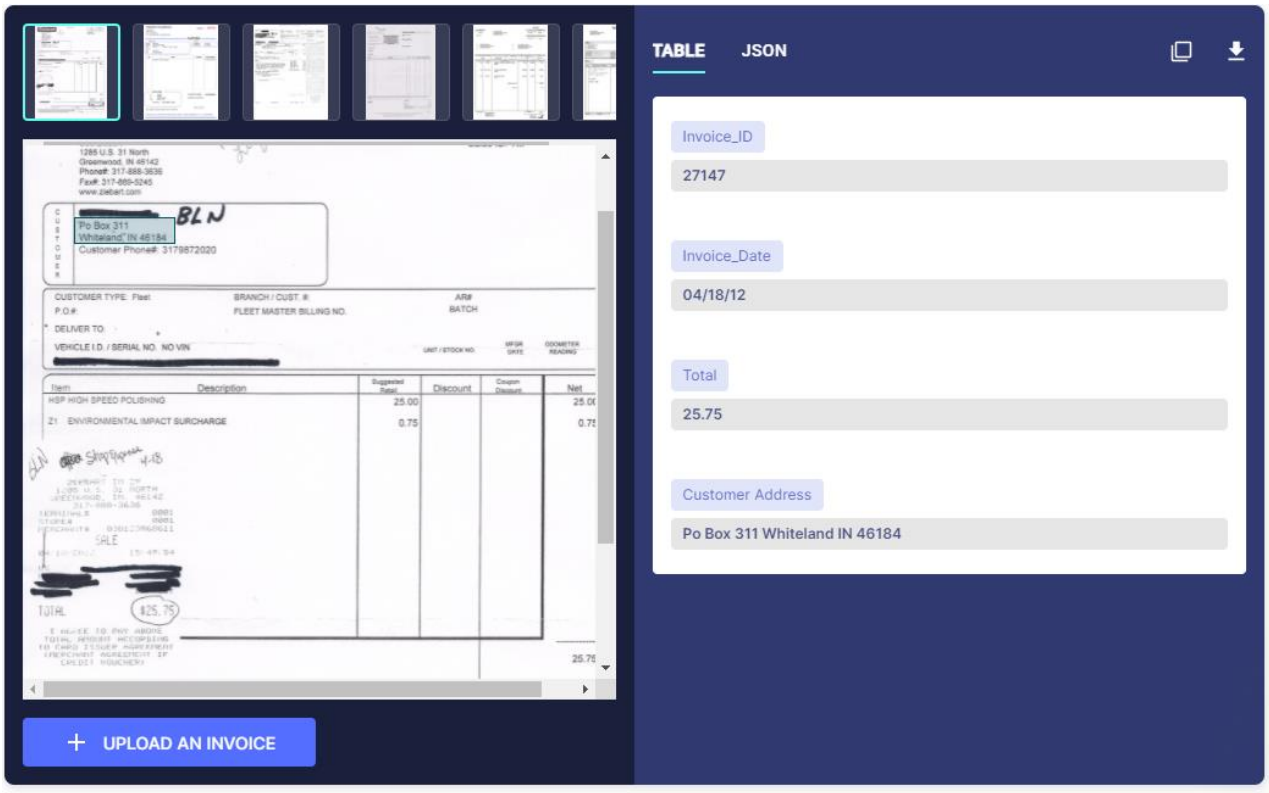


Figure 7: Nanonets OCR service example[6]

Statistics calculated by Nanonets showed that the time taken for invoice processing can be reduced by 90%, the inspections costs with 50% and moreover, the time needed to process the documents can be reduced with 90% as well.

The features offered by Nanonets PRO version costs in total is 411.55 EUR per model per month for each trained model up to 10000 pages. This includes 5 re-trains, any additional train costs 41.24 EUR and any additional field adds 0.0082 EUR to the whole.

2.2.4 Azure Form Recognizer

Azure Form Recognizer is a cognitive service offered by Microsoft Azure. It is a part of Azure Applied AI Services, a package which helps to build automated data processing systems, by using ML. Form Recognizer consists of three different services, the Layout API, Custom Models and Prebuilt Models. These can be accessed by REST API or by using client library SDKs. Therefore, it can be easily integrated in existing or new applications. SDKs are available for .NET, Python, Java and JavaScript.

To access the features provided by Azure, first a subscription must be made. Next on the Azure Portal platform a Form Recognizer resource is created, this is where the client can find its key and endpoint to connect with the REST API.

[6] *Nanonets. (n.d). Capture Data from invoices. Retrieved June 16, 2021, from https://nanonets.com/invoice-ocr*

The services have the following input requirements:

- Supported formats are PDF, JPG, PNG and TIFF.
- Size of the file must be less than 50 MB.
- Image dimensions mut be at least 50 x 50 pixels and at most 10000 x 10000 pixels.
- PDF dimensions must be at most 17 x 17 inches (A3 paper).
- For PDF and TIFF the first 200 pages are processed.
- Training data set must be at most 500 pages.

To use the API for each request an authentication header must be sent which contains the API key to verify the identity. The data is encrypted during the transit using HTTPS URLs. When retrieving the results, the same API key must be sent in the same manner. If the status of the operation is complete the extracted text is returned in a JSON format, else just the status is returned. The results are temporarily stores in Azure Storage for 48 hours.

The Layout API extracts everything from a document based on well-defined layouts. This API does not predict anything it extracts the text, tables, section marks and returns them as a structured JSON. It provides the bounding box and a confidentiality level for each extracted text. This feature is available in 73 languages, these do not include Romanian language. For English, handwritten text is acceptable as well.

Form Recognizer contains four prebuilt models, receipts, business cards, invoices and identification documents. These are currently available only in English, and the identification documents model is only available for US driver licenses and passports. These models extract fields based on the trained models and in the same way as for the Layout API, the bounding box and confidentiality is returned as well. For example, the receipts model looks for the following information presented in *Table 1*. The fields for prebuilt models are fixed, there is no possibility of adding new fields by the client. However, usually these satisfy the needs for each model as the most important properties are present.

	Property Name	Property Type
1	ReceiptType	string
2	MerchantName	string
3	MerchantPhoneNumber	phoneNumber
4	MerchantAddress	string
5	TransactionDate	date
6	TransactionTime	time
7	Total	number
8	Subtotal	number
9	Tax	number
10	Tip	number
11	Items	array of objects
12	Name	string
13	Quantity	number
14	Price	number
15	TotalPrice	number

Table 1: Form Recognizer Prebuilt Receipt Model Fields

The custom model feature provides the possibility to train custom documents. There are two ways to train a model, with or without labels. Training with labels is realized by supervised learning, where the user labels the data needed and specifies its location. First the content is extracted by using the Layout API, next, based on the labeling, key/value pairs are extracted. When training without labels an unsupervised learning is trying to identify and understand the layout of the document, once the document is processed the values associated to important keys are extracted. For this the keys must appear above or to the left of the values. This service is available in the same languages as the Layout API, because it uses it to get the document content.

The steps to configure a custom model are the following: create a training dataset, upload the dataset, train the model, test the model and manage it. The training dataset should contain examples with different values in each field. A minimum number of 5 examples are recommended, but if the images have low quality a larger set, like10-15 examples is needed. The training set must be uploaded to an Azure blob storage container. If the labeled training is selected, two files for each document might be added, the *.labels.json* and the *.ocr.json*.

To generate these, a labeling tool is provided, where the user can map and tag the properties. Once the model is trained, documents can be uploaded to test it. In the same manner as for the Layout API and Prebuilt Models the bounding box and a confidentiality level is returned for each extracted text. At any time, any custom model can be accessed, modified or deleted.

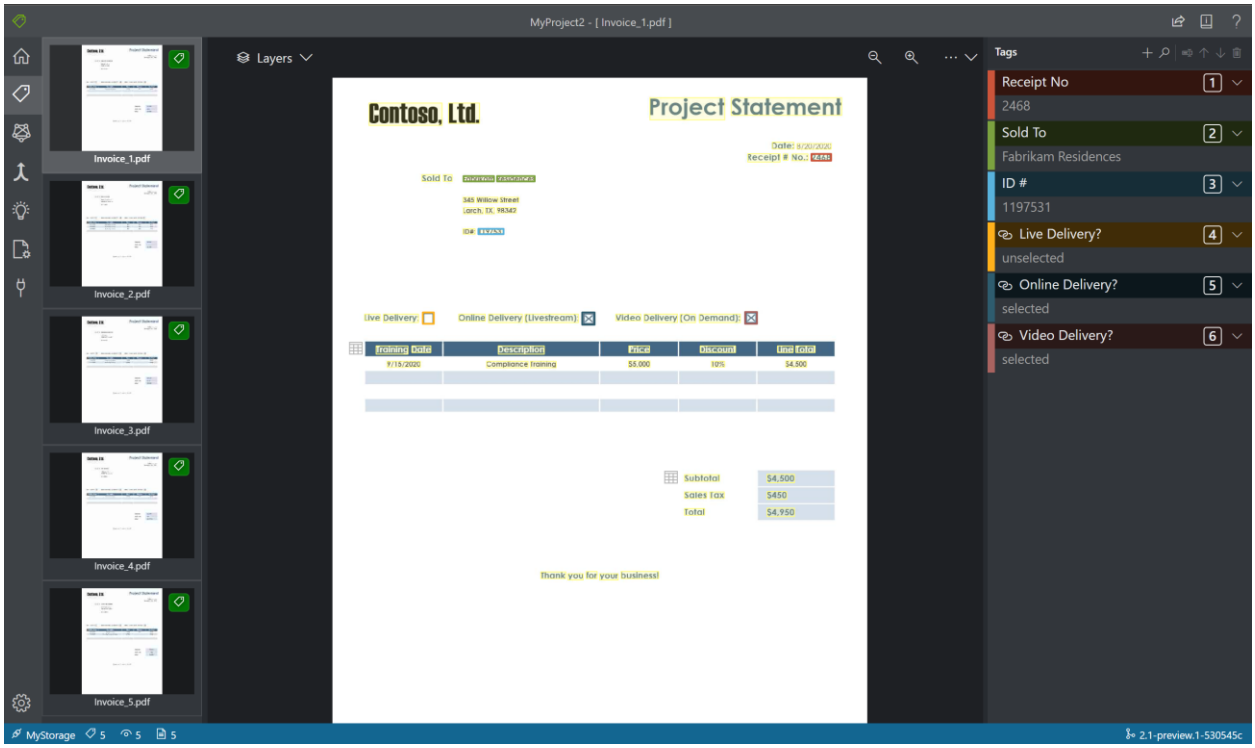


Figure 8: Form Recognizer Labeling Tool[7]

The services provided by Form Recognizer are free up to 500 pages per month. The custom model and the prebuilt invoice model cost 42.165 EUR per 1000 pages, the Layout API, and the rest of the prebuilt models are 8.433 EUR per 1000 pages.

2.3 Summary and comparison of existing approaches

All in one, it is clear now that automated data extraction significantly helps businesses. Integrating it in a DMS would increase the advantages, as for everything would be on-premises.

In DocsMetaExtractor, in contrary with DxInvoice, any kind of document models can be defined, not only invoices. Moreover, it is not bound to e-documents, it works with real and scanned PDFs as well. Statistics provided by ReadySoft[8] show that 90% of the invoices processed by companies are either paper, PDFs or scanned images and only 10% is in electronic formats, like EDI or XML.

[7] *La Laujan, Rohit Mungi, JP Park, Michael Bullwinkle. (May 11, 2021). Train a custom model using the sample labeling tool. Microsoft Docs. https://docs.microsoft.com/en-us/azure/cognitive-services/form-recognizer/label-tool?tabs=v2-1*

[8] *ReadySoft. (n.d). Invoice Processing Overview. Retrieved June 16, 2021, from https://readysoft.ro/invoice-processing-overview?lang=en*

The challenging part was the creation of the plug-in. In comparison with SENSEtask, where everything is extracted and there is no way of defining the fields wanted, like in the solutions provided by Nanonets and Azure, in DocsMetaExtractor when creating a document model, the properties wanted can be added with the type for each property. There are three types to structure the data, text, date and number. These can be modified and deleted any time and new properties can be added continuously.

The training requires minimum 2 examples. The model does not learn every time a new document is added, like in Nanonets and Form Recognizer. It retains only when it is needed in order to not use extra resources unwisely. For the user, a document preview page is available where the properties can be selected with a similar role as for the other applications. Moreover, a validation step is added in order to get a feedback about the results and to update the status of a document.

Another very important point is the language support. The plug-in is able to process documents in any languages as it is not working based on keys, like in the other data extraction solutions presented, which are bound to language limitations.

The main goals of DocsMetaExtractor are to be generic, customizable, easily integrated and to provide full control for the client. Like in Form Recognizer, the user can at any time to modify, delete and add new models. It can also change, delete and add at any time fields.

Exporting and filtering data is also possible with advanced filtering. As these are basic functionalities required for these kinds of applications.

In the table on the next page a comparison is presented between DocsMetaExtractor, SENSEtask, DxInvoice, Nanonets Enterprise Automation and Form Recognizer’s custom model service.

	SENSEtask	DxInvoice	Nanonets	Form Recognizer	DocsMetaExtractor
DMS	NO	YES	NO	NO	YES
Automatic data extraction	YES	YES	YES	YES	YES
Data extraction tool interface	REST API	X	REST API	REST API, SDKs	Plug-in
File formats	PDF, Images	E-Invoices	PDF, Images	PDF, TIFF, Images	PDF
Trainable	NO	NO	YES	YES	YES
Min. training dataset	X	X	50 images / property	5 pages (10 -15 for low quality images)	2 documents
Supported languages	Romanian	Language independent	Language independent	73 languages (not including Romanian)	Language independent
Data export	JSON	X	JSON, CSV, DOC	JSON	EXCEL, CSV
Data filtering	X	YES	X	X	YES

Table 2: Comparison with existing applications

3 Requirements analysis

3.1 Functional requirements

3.1.1 Actors and agents

The actors of the system are the employees of a company who are responsible for document processing. In most cases this would be the AP team, as they work with the majority of incoming documents. The employees by authentication can access the system, where they can upload, manage and validate the documents. They also are allowed to change their passwords, as only administrators can add new users with a password that must be changed by the employee.

Administrators are the agents, they have the power to use the same functionalities as employees and also an administration section is available for them. There, they have full control over users, document models and the already trained templates. They are responsible for updating and changing the application content.

3.1.2 Software use case diagram

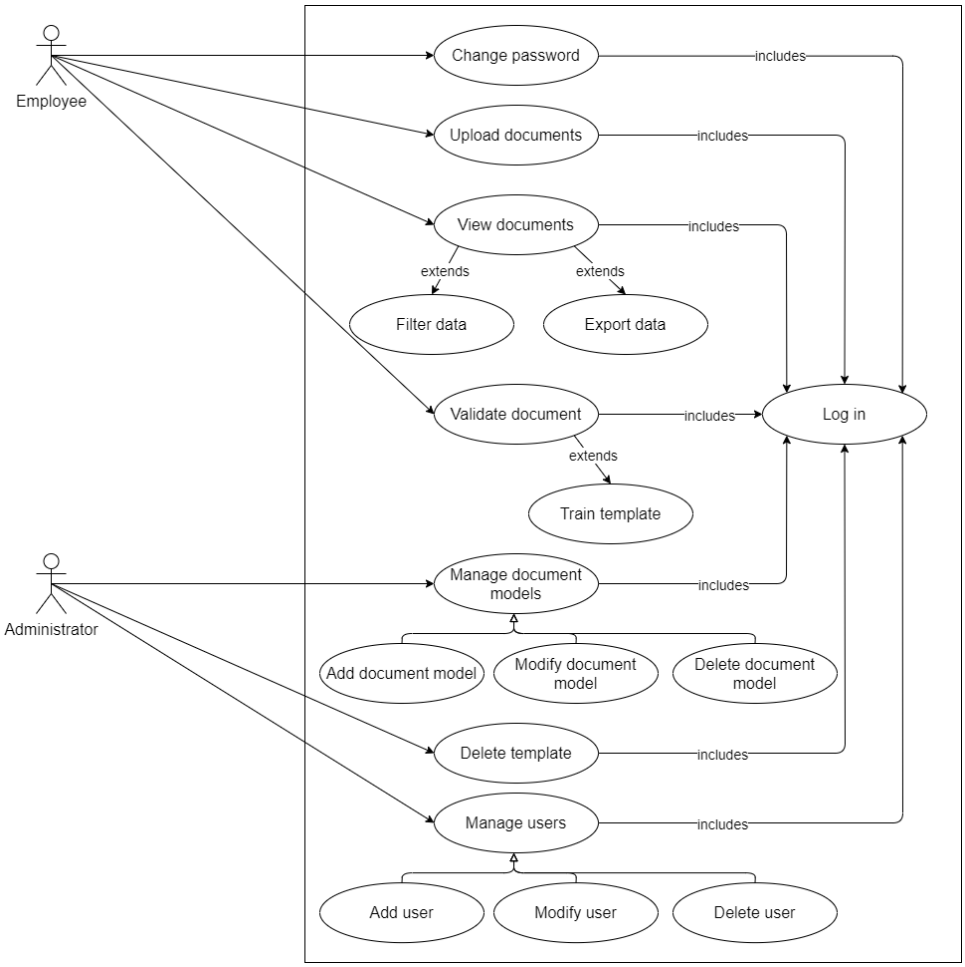


Figure 9: Software Use Case Diagram

3.1.3 Use case functional requirements

3.1.3.1 Log in – UC1

a) System shows login form -10

b) System validates form -10

c) System generates authentication token - 10

d) System displays error message – 6

3.1.3.2 Change password – UC2

a) System shows change password form - 10

b) System requires old password – 10

c) System validates old password - 10

d) System validates new password - 10

e) System updates user password - 10

f) System displays error messages - 6

3.1.3.3 Upload documents – UC3

a) Systems shows upload area -10

b) System checks files formats - 10

c) System stores the documents – 10

d) System starts text extraction – 10

e) System displays error message – 6

3.1.3.4 View documents – UC4

a) System gets document model data – 10

b) System displays documents list -10

c) System provides filter inputs - 9

d) System refreshes documents list – 6

3.1.3.5 Filter data – UC5

a) System provides filter inputs – 10

b) System validates filter values – 10

c) System shows error messages – 6

d) System filters the data - 10

3.1.3.6 Export data – UC6

a) System shows exporting options – 10

b) System generates export file – 10

c) System downloads file - 10

3.1.3.7 Validate document – UC7

a) System displays document -8

b) System displays extracted text blocks – 10

c) System displays document properties form – 10

d) System selects text block – 10

e) System add value to property – 10

f) System validates selected text blocks – 9

g) System updates document status – 10

h) System validates the view – 9

i) System displays errors – 8

3.1.3.8 Train template – UC8

a) System checks if new template was added – 10

b) System checks if properties were modified – 10

c) System trains template – 10

d) System updates training model – 10

e) System notifies user when train is finished – 6

3.1.3.9 Delete template – UC9

a) System displays templates – 10

b) System deletes template - 10

c) System notifies user - 6

3.1.3.10 Manage document models – UC10

a) System displays document models management view – 10

b) System adds new model -10

c) System modifies model – 10

d) System deletes model – 10

e) System notifies user on success – 6

f) System displays error messages – 6

g) System saves changes – 10

3.1.3.11 Manage users – UC11

a) System displays users management view – 10

b) System adds new user – 10

c) System modifies user – 10

d) System deletes user – 10

e) System encrypts sensitive data - 10

f) System saves changes – 10

g) System notifies user on success – 6

h) System displays error messages – 6

3.1.4 Use case descriptions and system sequence diagrams

3.1.4.1 UC1

<table>
<tr><td>Use Case Name:</td><td colspan="2">Log in</td></tr>
<tr><td>Scope:</td><td colspan="2">System</td></tr>
<tr><td>Actor:</td><td colspan="2">User (employee/administrator)</td></tr>
<tr><td>Preconditions:</td><td colspan="2">The system must be up and running.</td></tr>
<tr><td>Post-conditions:</td><td colspan="2">The user is logged in and the home page is displayed.</td></tr>
<tr><td>Main success scenario:</td><td colspan="2"><table>
<tr><th>User</th><th>System</th></tr>
<tr><td>1. access website</td><td></td></tr>
<tr><td></td><td>2. show log in form</td></tr>
<tr><td>3. enter credentials</td><td></td></tr>
<tr><td></td><td>4. validate credentials – A1</td></tr>
<tr><td></td><td>5. generate authentication token</td></tr>
<tr><td></td><td>6. display homepage</td></tr>
</table></td></tr>
<tr><td>Extension:</td><td colspan="2">A1. Incorrect username or password.
Return to step 2 and display error message.</td></tr>
</table>

Table 3: Log In Description

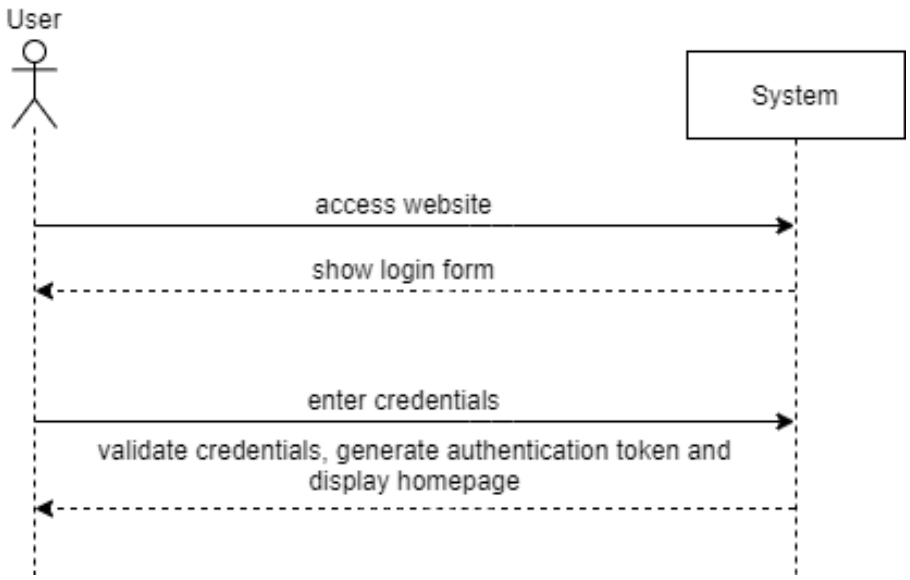


Figure 10: Log In System Sequence Diagram

3.1.4.2 UC2

Use Case Name:	Change password
Scope:	System
Actor:	User (employee/administrator)
Preconditions:	The user must be logged in.
Post-conditions:	The user's password is changed.

Main success scenario:

User	System
1. request password change	
	2. open password change form
3. fill in form	
	4. validate new password – A1
	5. validate old password – A2
	6. encrypt new password
	7. save encrypted password
	8. close form

Extension:

A1. Confirmation password and new password do not match.
Return to step 3 and display error message.
A2. Incorrect old password
Return to step 3 and display error message.

Table 4: Change Password Description

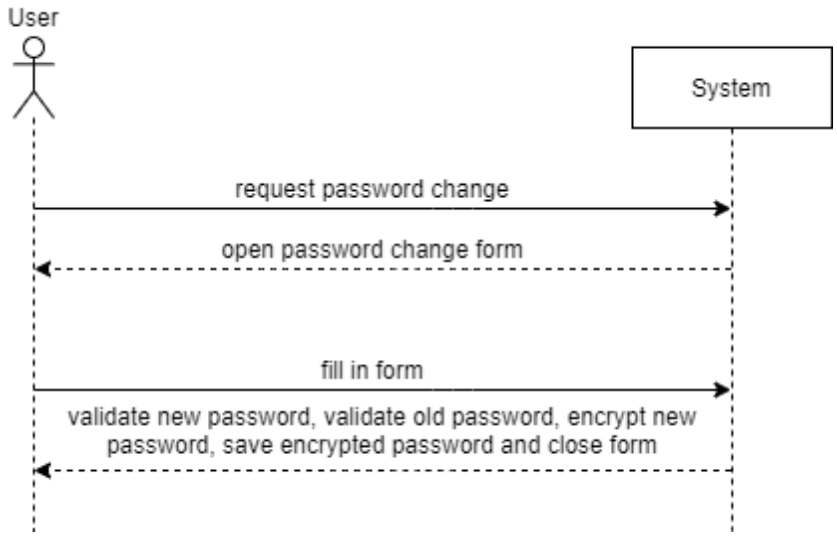


Figure 11: Change Password System Sequence Diagram

3.1.4.3 UC3

Use Case Name:	Upload documents
Scope:	System
Actor:	Employee
Preconditions:	The employee must be logged in.
Post-conditions:	New documents are added and the text extraction is started for each.

Main success scenario:

Employee	System
1. click upload button	
	2. show upload form
3. select documents	
	4. save documents – A1
	5. start text extraction
	6. close upload form

Extension:	A1. Incorrect document format. Return to step 2 and display error message.

Table 5: Upload Documents Description

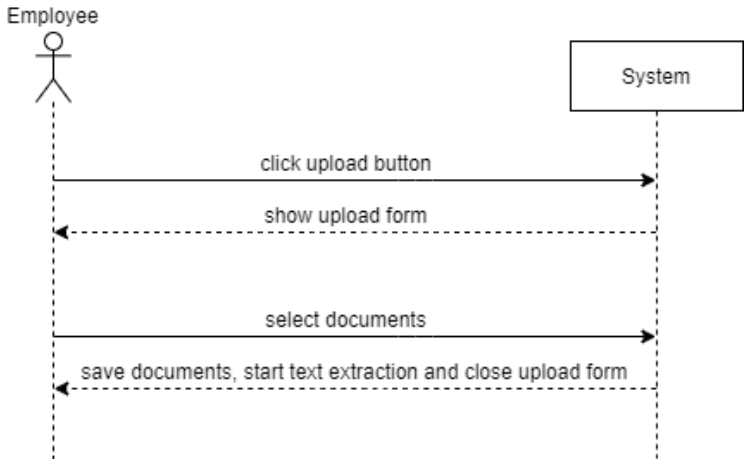


Figure 12: Upload Documents System Sequence Diagram

3.1.4.4 UC4

Use Case Name:	View documents
Scope:	System
Actor:	Employee
Preconditions:	The employee must be logged in.
Post-conditions:	The documents list is displayed.

Main success scenario:

Employee	System
1. navigate to documents list view	
	2. display documents

Table 6: View Documents Description

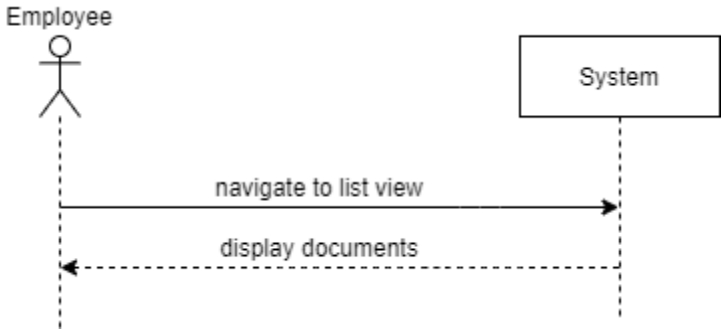


Figure 13: View Documents System Sequence Diagram

3.1.4.5 UC5

Use Case Name:	Filter data
Scope:	System
Actor:	Employee
Preconditions:	The current page must be the documents list.
Post-conditions:	Documents are filtered.
Main success scenario:	

Employee	System
1. type in filter values	
	2. filter columns – A1
	3. display filtered data

Extension:	A1. Incorrect filter values. Stop filtering, return to step 1 and display error message.

Table 7: Filter Data Description

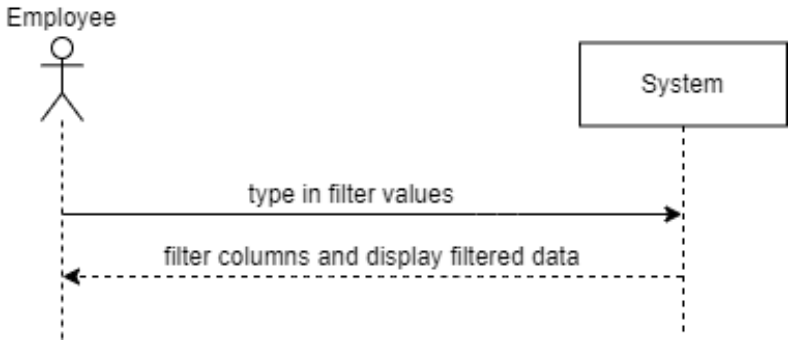


Figure 14: Filter Data System Sequence Diagram

3.1.4.6 UC6

Use Case Name:	Export data
Scope:	System
Actor:	Employee
Preconditions:	The current page must be the documents list.
Post-conditions:	Filtered data is exported to the local device.
Main success scenario:	

Employee	System
1. click on export dropdown button	
	2. show export formats
3. select a format	
	4. generate export file
	5. download file

Table 8: Export Data Description

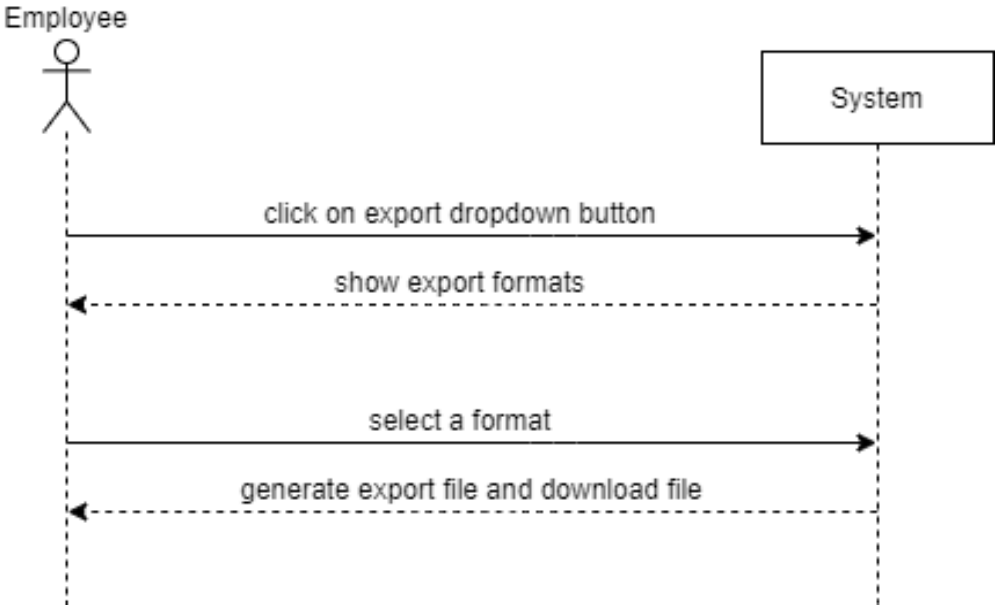


Figure 15: Export Data System Sequence Diagram

3.1.4.7 UC7

<table>
<tr><td>Use Case Name:</td><td colspan="2">Validate document</td></tr>
<tr><td>Scope:</td><td colspan="2">System</td></tr>
<tr><td>Actor:</td><td colspan="2">Employee</td></tr>
<tr><td>Preconditions:</td><td colspan="2">The current page must be the documents list.</td></tr>
<tr><td>Post-conditions:</td><td colspan="2">The document status is updated to confirmed.</td></tr>
<tr><td>Main success scenario:</td><td colspan="2"><table><tr><th>Employee</th><th>System</th></tr><tr><td>1. click on edit button – A1</td><td></td></tr><tr><td></td><td>2. show document preview</td></tr><tr><td></td><td>3. display extracted text blocks layer</td></tr><tr><td></td><td>4. predict properties</td></tr><tr><td>5. confirm results – A2</td><td></td></tr><tr><td></td><td>6. update document status</td></tr><tr><td></td><td>7. return to list view</td></tr></table></td></tr>
<tr><td>Extension:</td><td colspan="2">A1. The document is still processing.
Notify user of the status of document.
A2. Prediction was incorrect or new template is added.
Map the correct properties and return to step 5.</td></tr>
</table>

Table 9: Validate Document Description

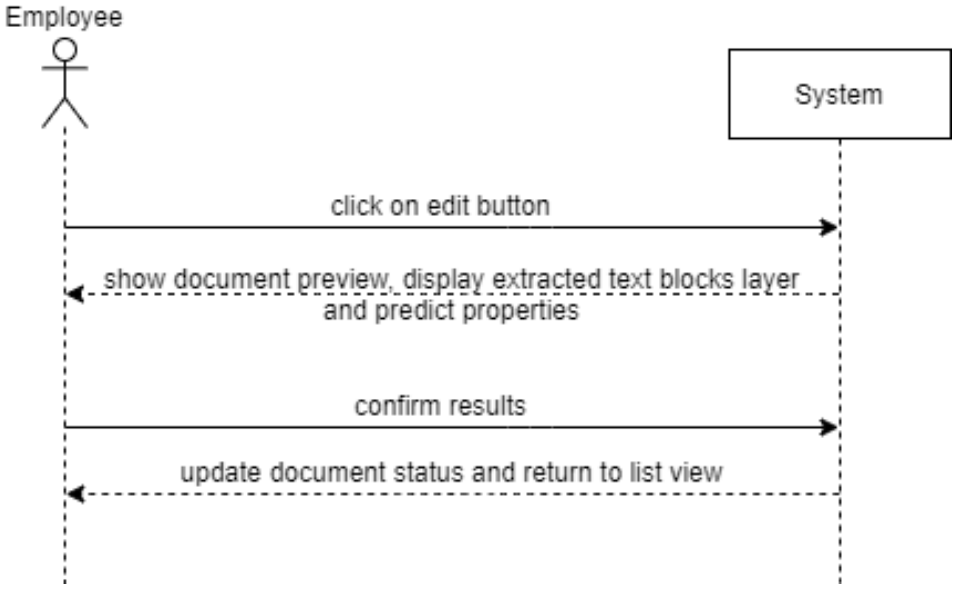


Figure 16: Validate Document System Sequence Diagram

3.1.4.8 UC8

Use Case Name:	Train template
Scope:	System
Actor:	Employee
Preconditions:	The document is validated.
Post-conditions:	The template is trained.
Main success scenario:	

Employee	System
1. validate document	
	2. add document to training set
	3. train the template – A1
	4. return to list view

Extension:	A1. Incorrect example. Delete document from training set and move to step 4.

Table 10: Train Template Description

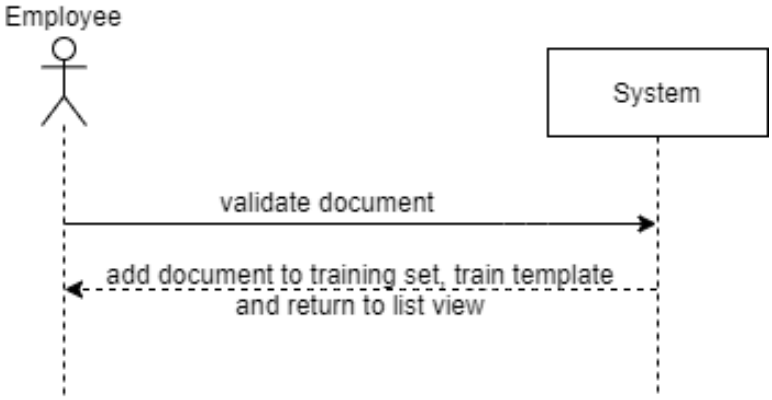


Figure 17: Train Template System Sequence Diagram

3.1.4.9 UC9

Use Case Name:	Delete template
Scope:	System
Actor:	Administrator
Preconditions:	The administrator is logged in.
Post-conditions:	Template is deleted.
Main success scenario:	

Administrator	System
1. navigate to templates administration view	
	2. display administration view
3. select template to delete	
	4. delete template
	5. refresh templates list

Table 11: Delete Template Description

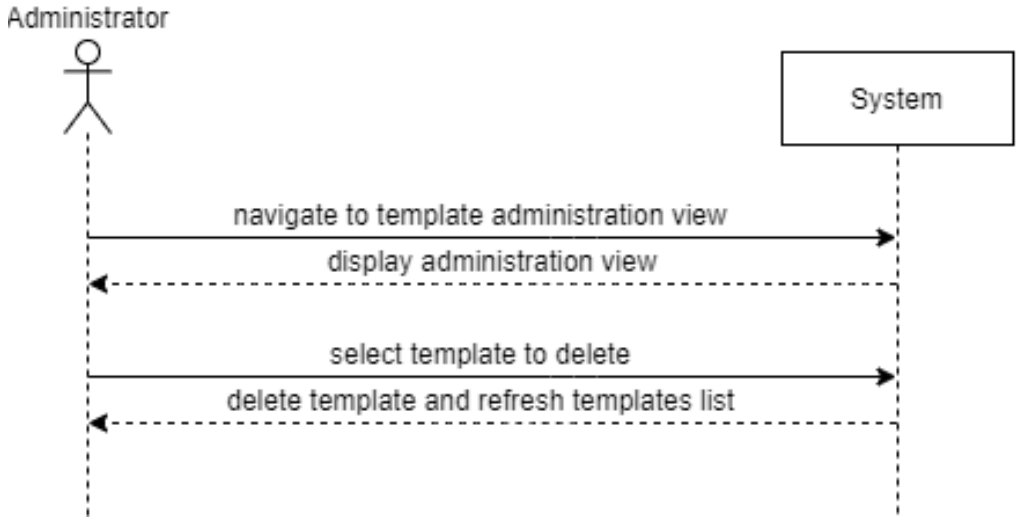


Figure 18: Delete Template System Sequence Diagram

3.1.4.10 UC10

Use Case Name:	Manage document models
Scope:	System
Actor:	Administrator
Preconditions:	The administrator is logged in.
Post-conditions:	Document model is added, modified or deleted.

Main success scenario:

Administrator	System
1. navigate to document model administration view	
	2. display administration view
3. select operation (add, edit, delete)	
	4. open operation form
5. fill in form	
	6. validate form – A1
7. click on submit button	
	8. save changes
	9. close operation form

Extension:	A1. Form contains invalid or missing data. Show error message and return to step 5.

Table 12: Manage Document Models Description

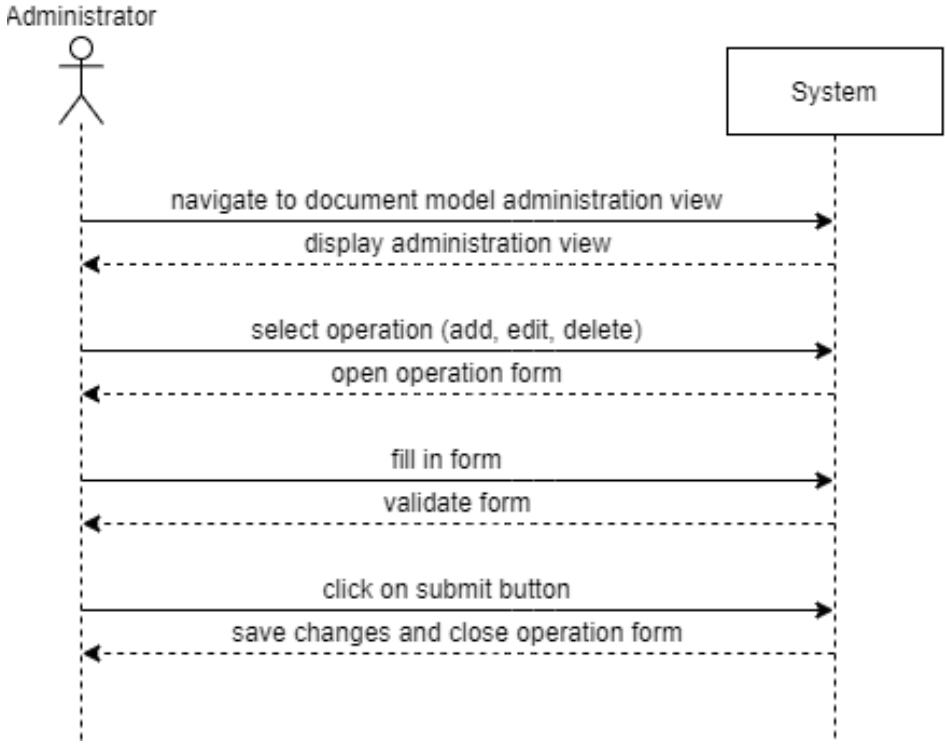


Figure 19: Manage Document Models System Sequence Diagram

3.1.4.11 UC11

Use Case Name:	Manage users
Scope:	System
Actor:	Administrator
Preconditions:	The administrator is logged in.
Post-conditions:	User is added, modified or deleted.

Main success scenario:

Administrator	System
1. navigate to user administration view	
	2. display user administration view
3. select operation (add, edit, delete)	
	4. open operation form
5. fill in form	
	6. validate form – A1
7. click on submit button	
	8. save changes
	9. close operation form
	10. refresh users

Extension:	A1. Form contains invalid or missing data. Show error message and return to step 5.

Table 13: Manage Users Description

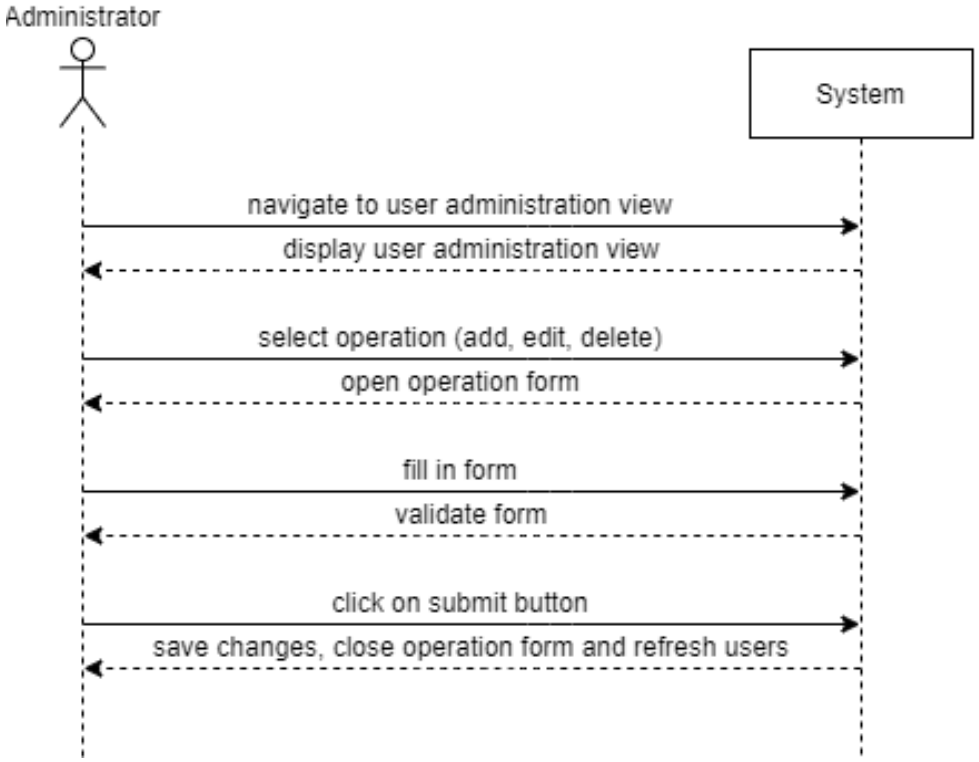


Figure 20: Manage Users System Sequence Diagram

3.1.5 Activity diagrams

3.1.5.1 UC7

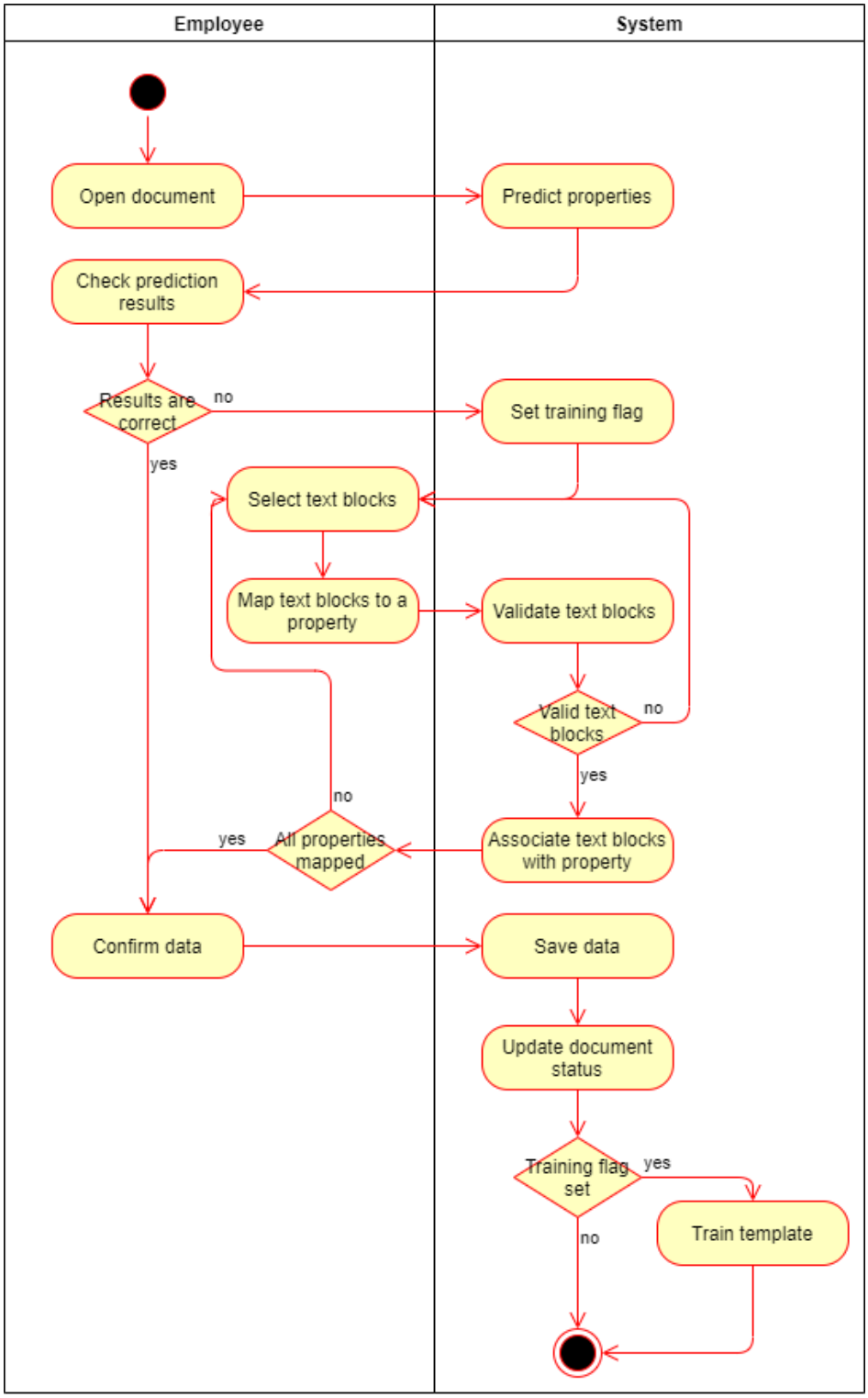


Figure 21: Validate Document Activity Diagram

3.1.5.2 UC10

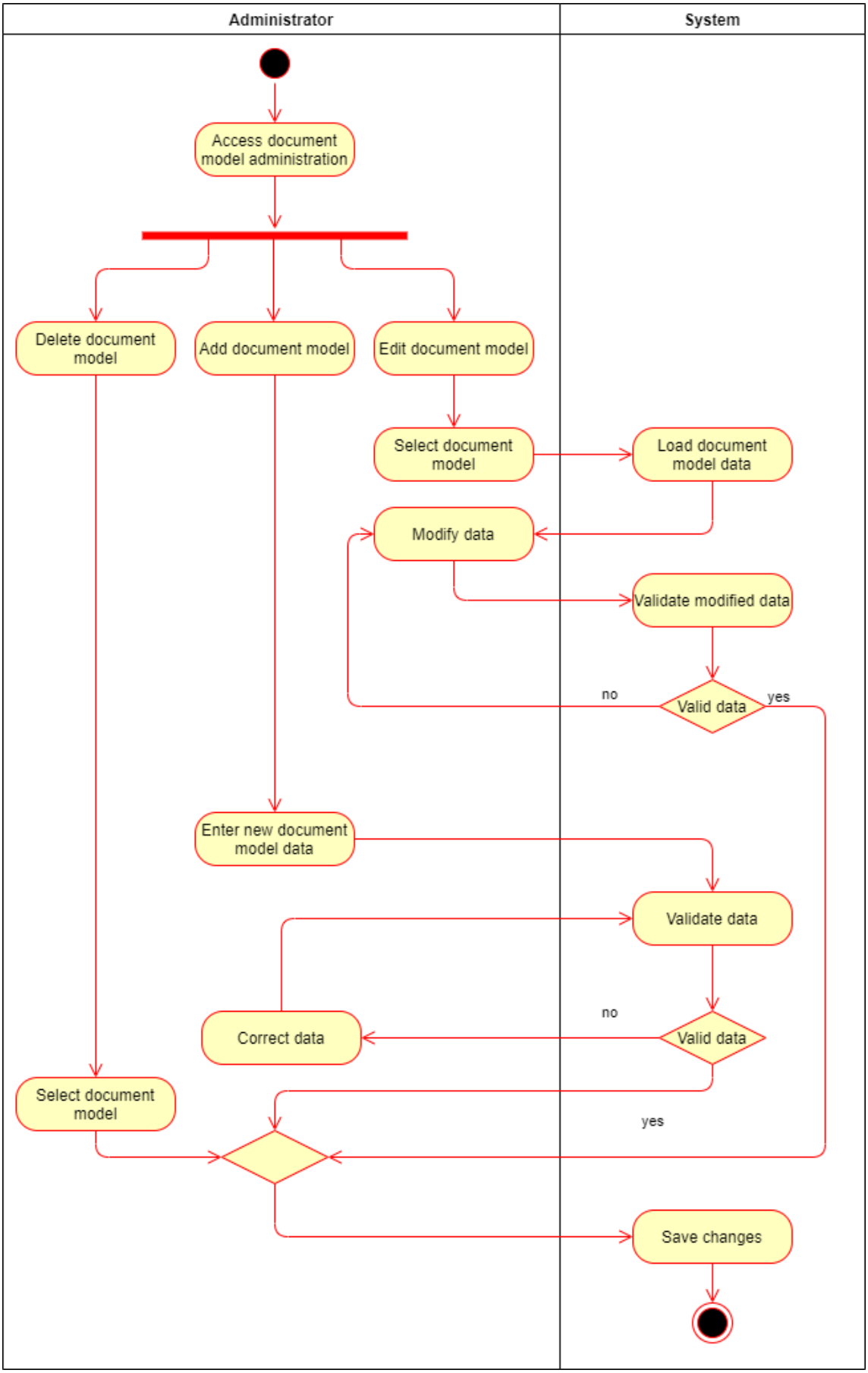


Figure 22: Manage Document Models Activity Diagram

3.1.6 Operation contracts

Operation Name:	StartPdfTextExtract(*pdfData*)
Use Case Name:	Upload documents – UC3
Preconditions:	New documents were saved in database.
Post-conditions:	Real PDFs the text blocks are extracted and saved. Scanned ones are sent to the OCR service. Each document is analyzed and their type is checked. For a real PDF the text is extracted by using the document's metadata. The result is synchronous. Scanned documents are sent asynchronously to an OCR service, where they are being processed and the results must be later queried.

Table 14: StartPdfTextExtract Operation Contract

Operation Name:	StartPdfTextExtractService(*docsMetaExtractorService*)
Use Case Name:	Upload document – UC3
Preconditions:	Scanned documents are uploaded.
Post-conditions:	Scanned documents OCR result are queried and response is saved. On a separate thread the scanned documents OCR operation ids is stored. Each operation id is popped from the list and the results are checked. If there is a result then it is saved, else the operation id is added to the list again to check it later. When all the scanned documents have results, the thread goes to sleep. When a new scanned document is uploaded if the thread is in sleeping mode, then it is started.

Table 15: StartPdfTextExtractService Operation Contract

Operation Name:	MatchTemplate(*modelId, pages, cultureCode*)
Use Case Name:	Validate document – UC7
Preconditions:	The document is opened.
Post-conditions:	Properties are predicted and returned. For the document model each template is iterated until the current document matches one. The documents text blocks are scaled and translated based on the current template model in the iteration. To match the template the identifier fields are checked. If the template is found the properties are predicted based on the trained model, converted to their type and returned. If the template is not found the pages are sent back.

Table 16: MatchTemplate Operation Contract

Operation Name:	TrainTemplate(*docModelId, templateId, newExampleId*)
Use Case Name:	Train template – UC8
Preconditions:	Document is validated and training is required.
Post-conditions:	The given template is trained and the model is updated. If new template is added the document is added to training set. If an existing template needs training the new example is scaled and translated to the first one, by determining two key labels. The key labels are fixed fields on the first page, by starting the search from the start and end of the page. If key labels could not be found the new example is deleted from the training set and no training happens. If the key labels are found the template is trained, there are two field types fixed and dynamic ones. For fixed fields, the coordinates are checked, and for dynamic ones their label is found. If the training fails the incorrect new document is deleted from the training set. If the first train fails, the two documents are deleted since cannot decide which one was wrongly mapped. If the train is successful the template model is updated.

Table 17: TrainTemplate Operation Contract

3.2 Non-functional requirements

- **Availability**: is available on working days, between working hours
- **Usability**: can be used by mid-educated people after less than an hour training
- **Reliability**: mean time between failures (MTBF) of 5 years
- **Portability**: system works on all platforms with internet access
- **Robustness**: the inputs are validated and the user is notified
- **Capacity**: 500,000 documents per document model
- **Scalability**: horizontally scalable, by adding new server, 500,000 documents => 1,000,000 entries per document model
- **Privacy**: pages are filtered and can be accessed based on the user's role
- **Security**: sensitive data is encoded
- **Reusability:** the training, predicting and text extraction services are published as a plugin, that can be integrated by systems

4 Design

4.1 Architecture

4.1.1 Monolithic architecture

The software architecture followed for this system is a monolithic one. A monolithic application is self-contained, and it is deployed as a single unit. These kinds of applications are also called all-in-one applications because the entire application is in a single project. This is the simplest way of creating and maintaining small and medium sized projects. Everything is in one place and can be accessed immediately.

A disadvantage of all-in-one applications comes when the size of the project grows, this means more and more files in addition and leads to a complex structure which is hardly understood.

DocsMetaExtractor web application is a medium sized project, it is a perfect choice to implement it monolithically. The complex parts are the training, prediction and text extraction services. These might be more and more complex during the development process, therefore these functions are integrated as a plug-in, so they can be modified separately without modifying the web application. Moreover, the structure of the web application projects remains as simple as possible. This overcomes the disadvantages of the monolithic structure and avoids so called spaghetti codes. The web application interacts with these services and runs the core, the presentation, business and data behaviors of the system.

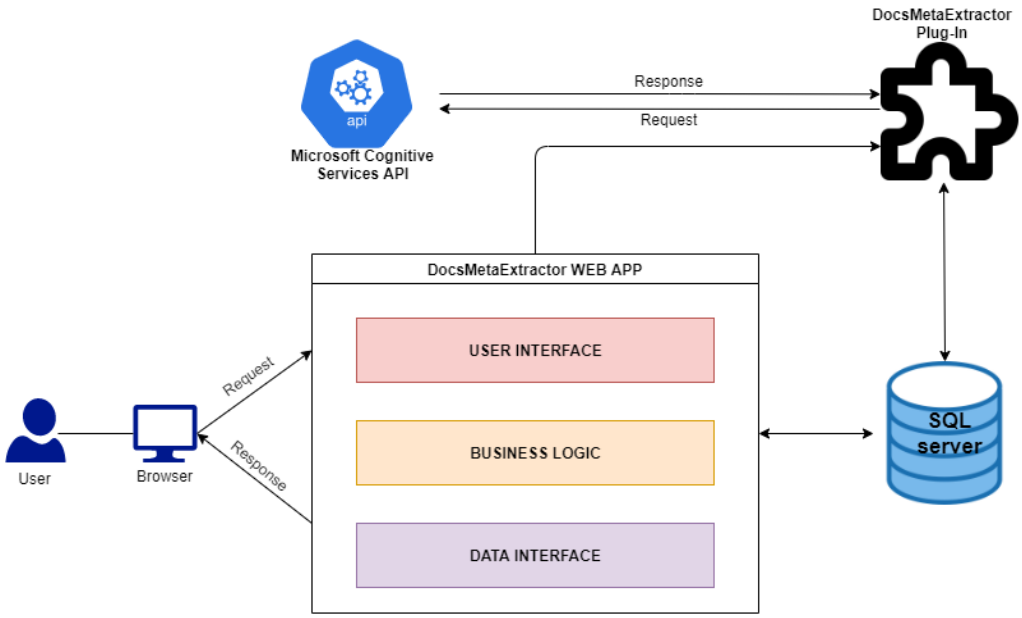


Figure 23: DocsMetaExtractor Architecture

4.1.2 Package diagrams

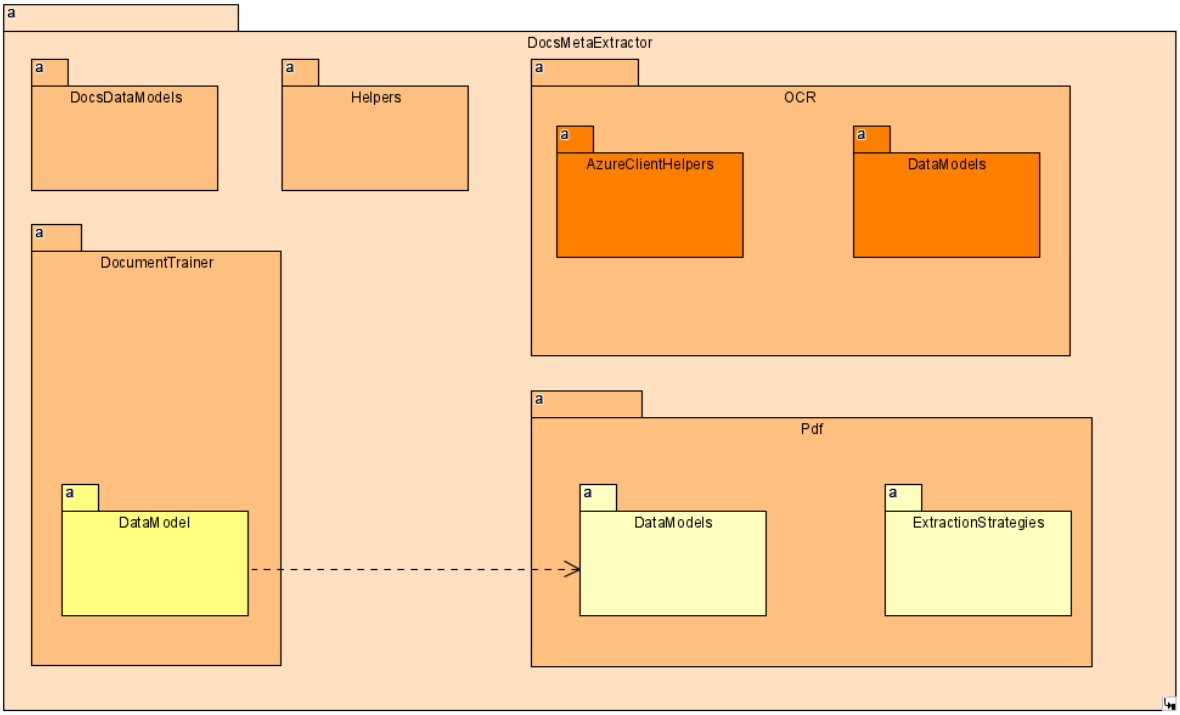


Figure 24: DocsMetaExtractor Package Diagram

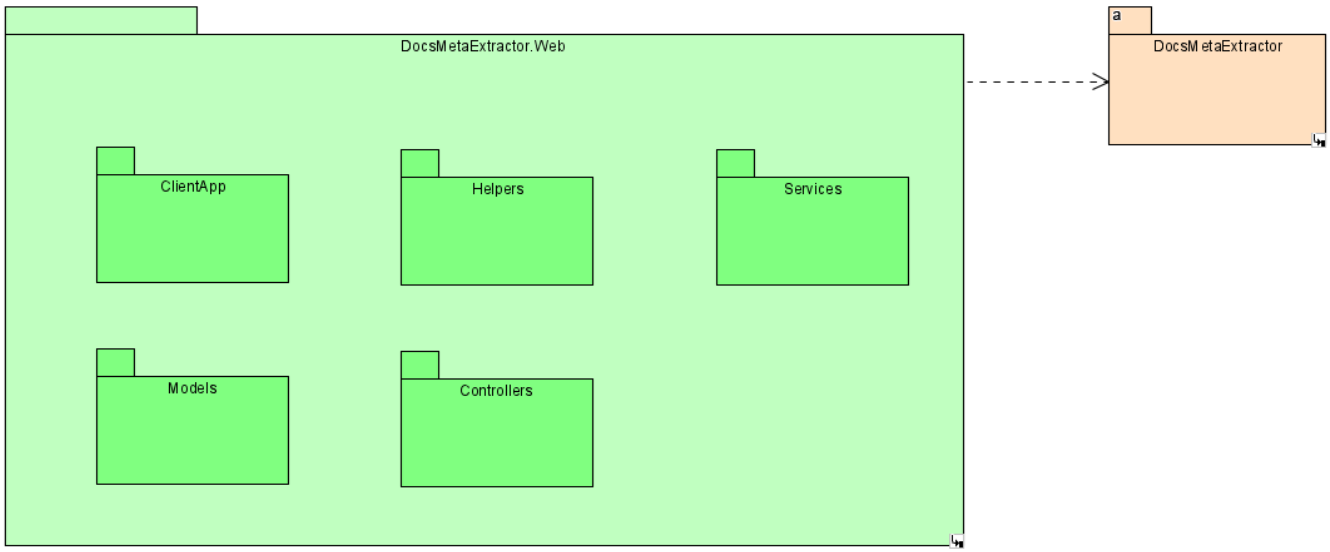


Figure 25: DocsMetaExtractor.Web Package Diagram

4.1.3 Deployment

Monolithic applications are deployed as a single unit. This is the easiest and simplest deployment model.

In this project IIS (https://www.iis.net/), by Microsoft, is used to host the application. The web application project is published as a File System to the IIS root folder. Next the folder is

converted to an application, after what it can be accessed. For this to work the *.NET Core Hosting Bundle* must be downloaded for the IIS to be able to host .NET5 applications.

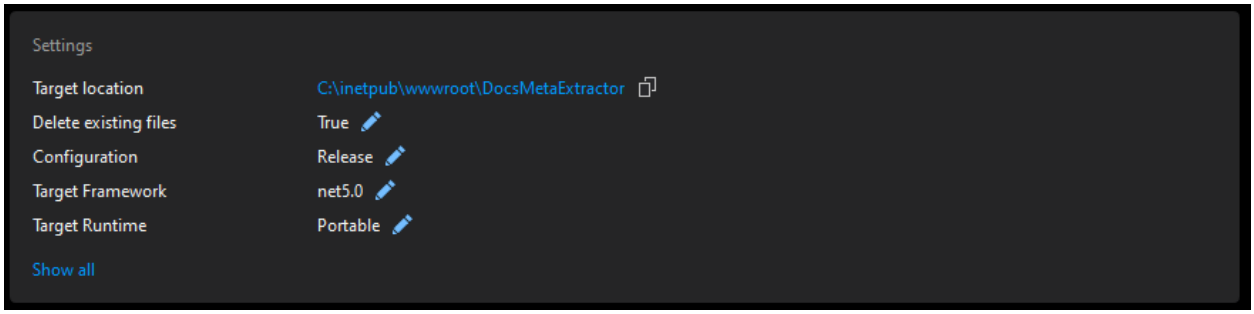


Figure 26: DocsMetaExtractor.Web Publish Configuration in Visual Studio

To deploy or integrate the application easier on other servers, or to move the database I created an SQL project which contains all the entities and members needed by the plugin. From this project the target database can be created or updated automatically. Moreover, also a script can be generated which can be run on the target database if we do not want to change its structure completely. This script can be modified and adapted in order to provide as much flexibility as possible.

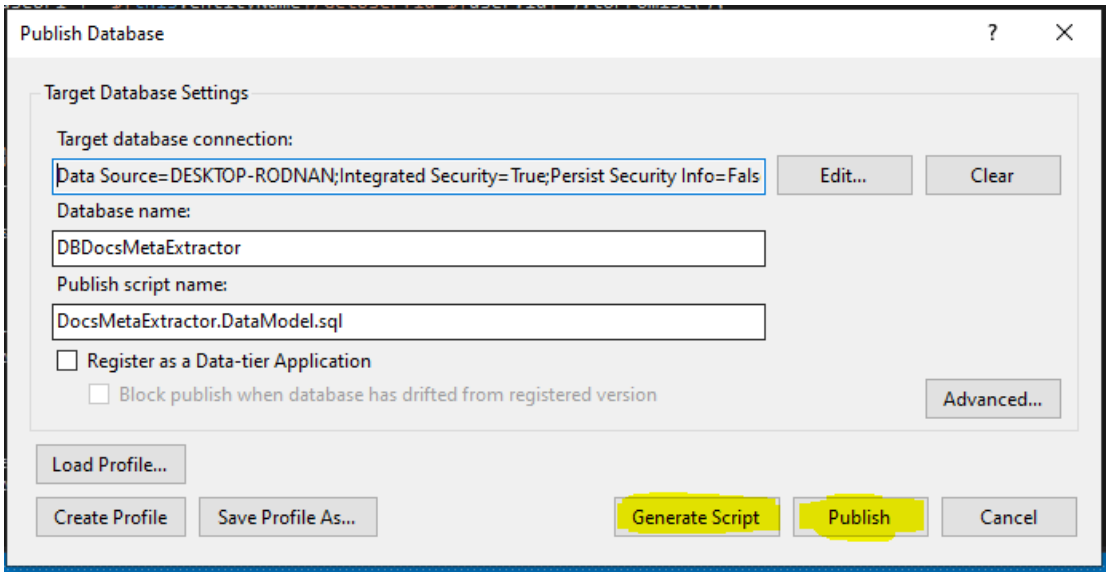


Figure 27: DocsMetaExtractor.DataModel Publish

4.2 Detailed design

4.2.1 Design Class Diagrams and Design Sequence Diagrams

4.2.1.1 UC1

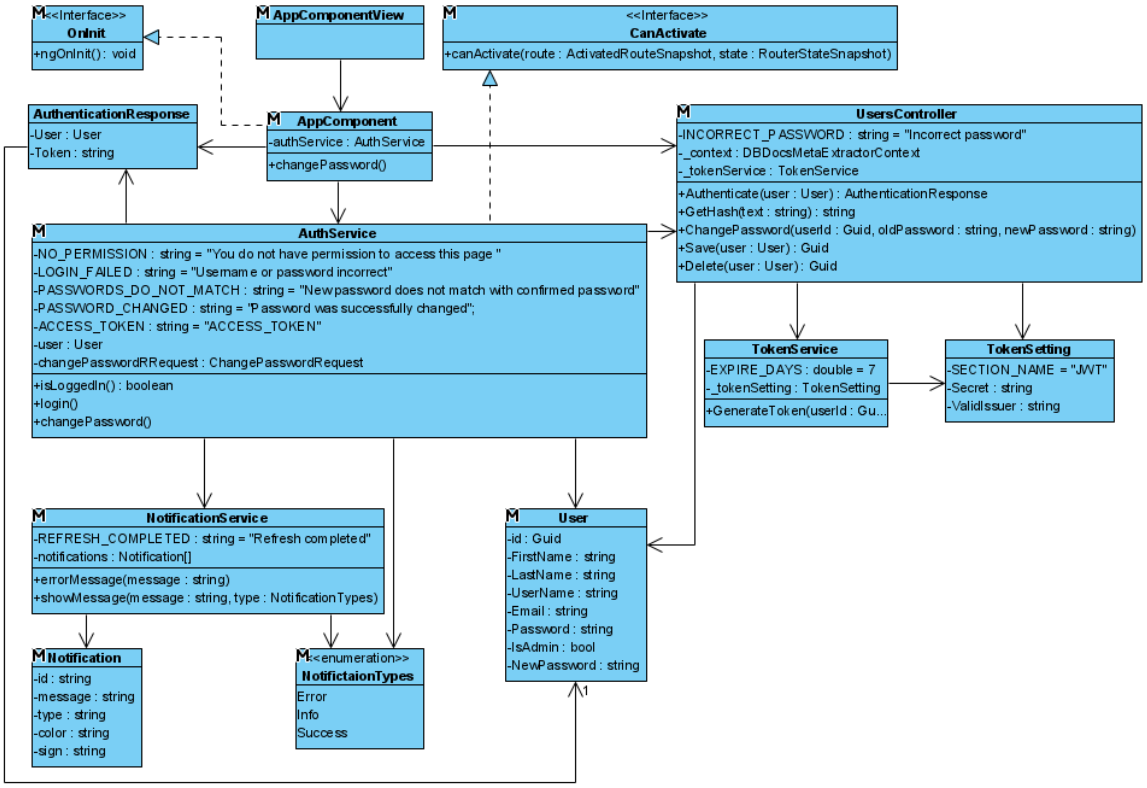


Figure 28: Log In Design Class Diagram

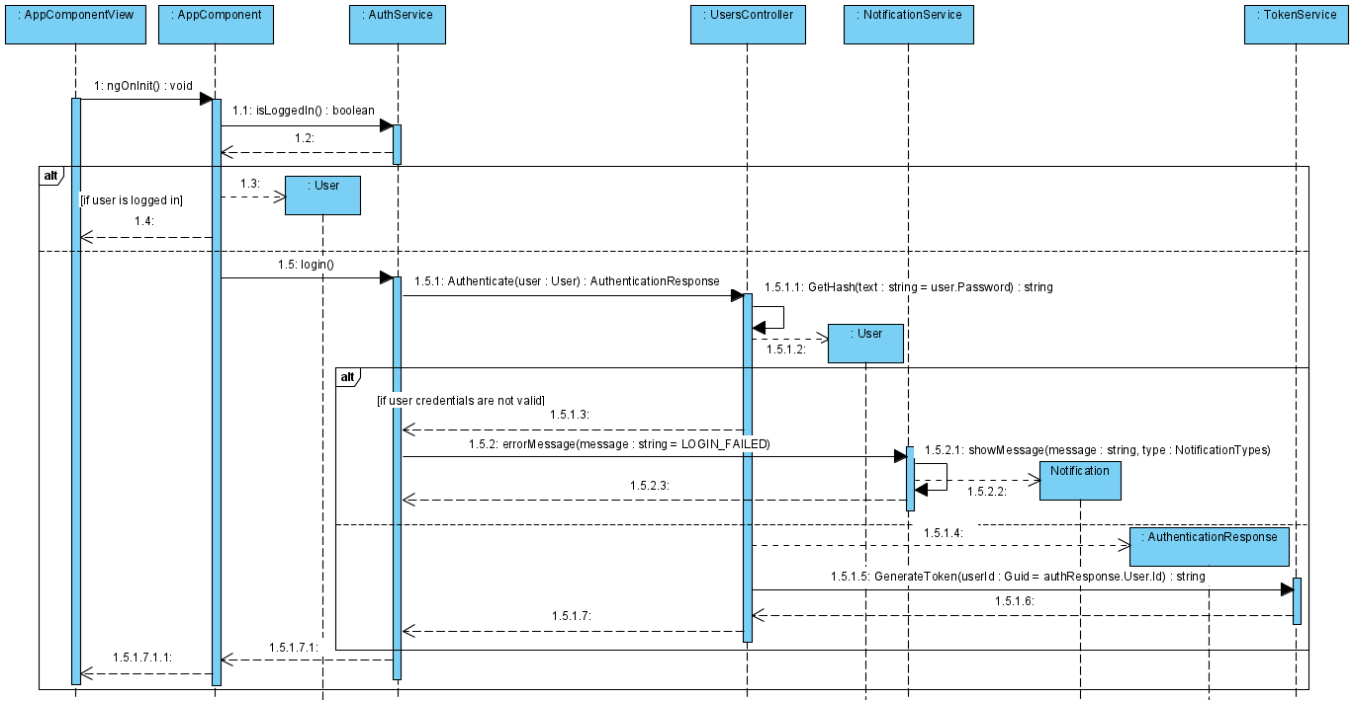


Figure 29: Log In Design Sequence Diagram

4.2.1.2 UC2

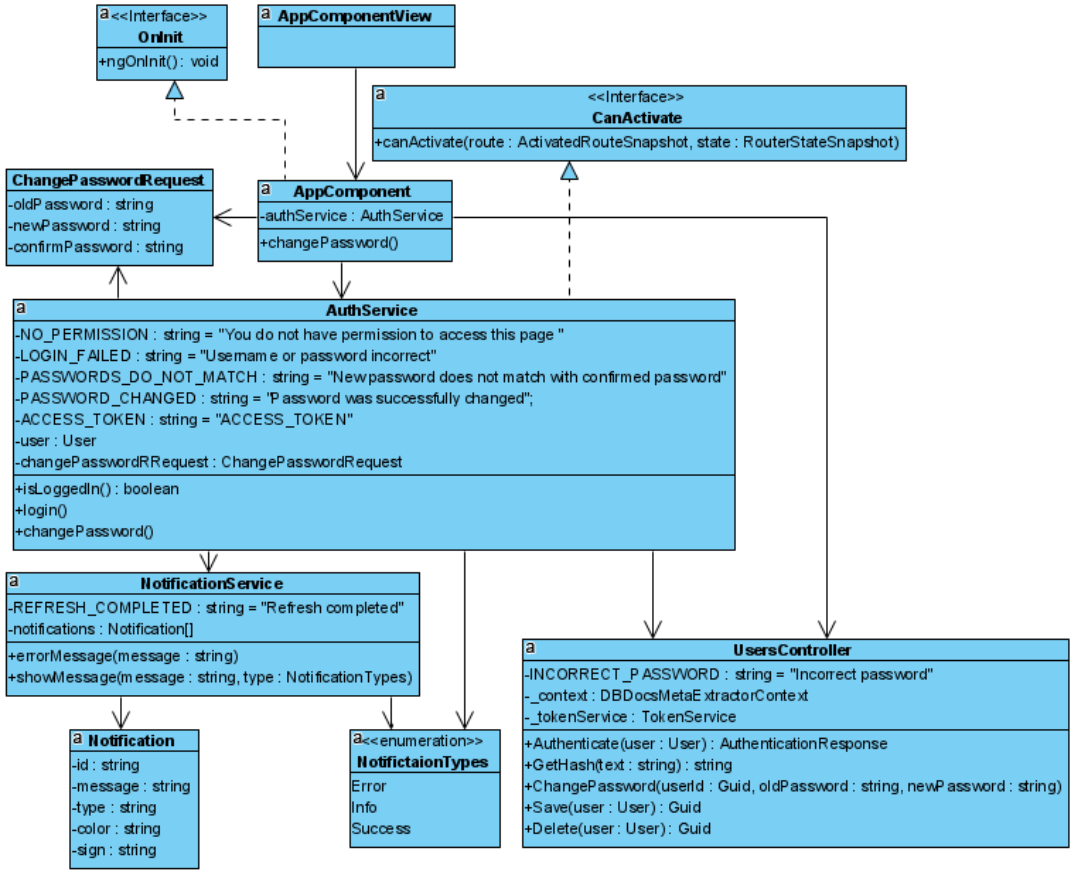


Figure 30: Change Password Design Class Diagram

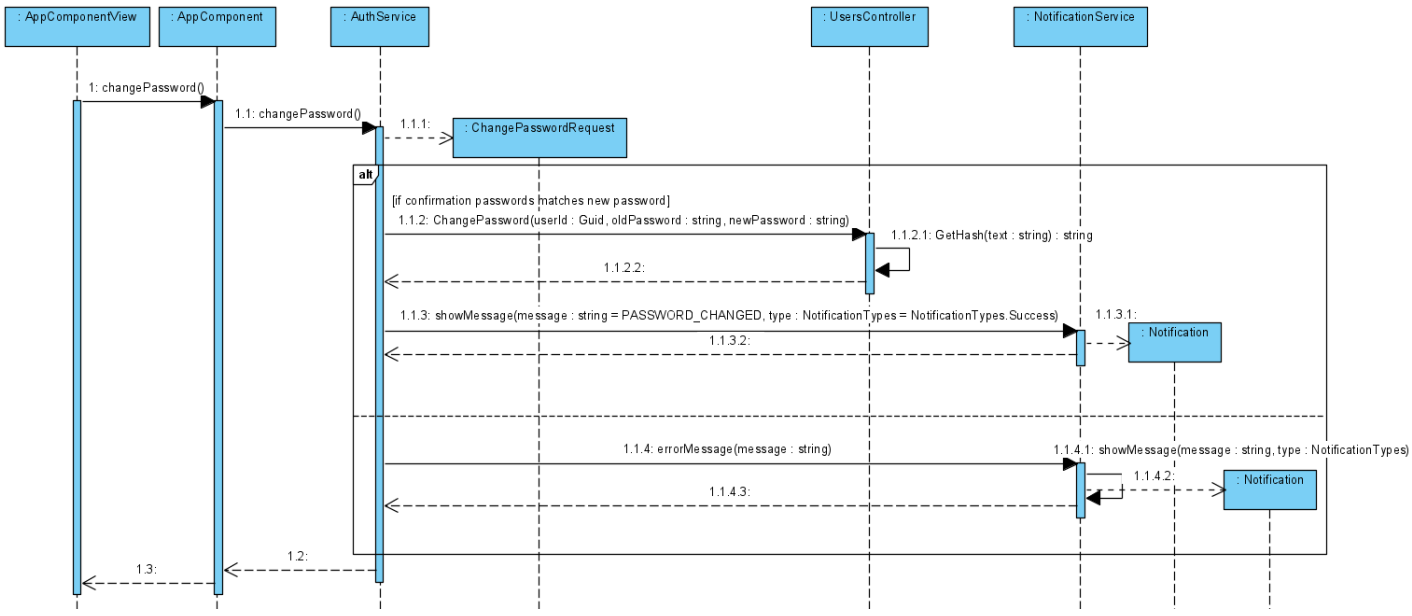


Figure 31: Change Password Design Sequence Diagram

4.2.1.3 UC3

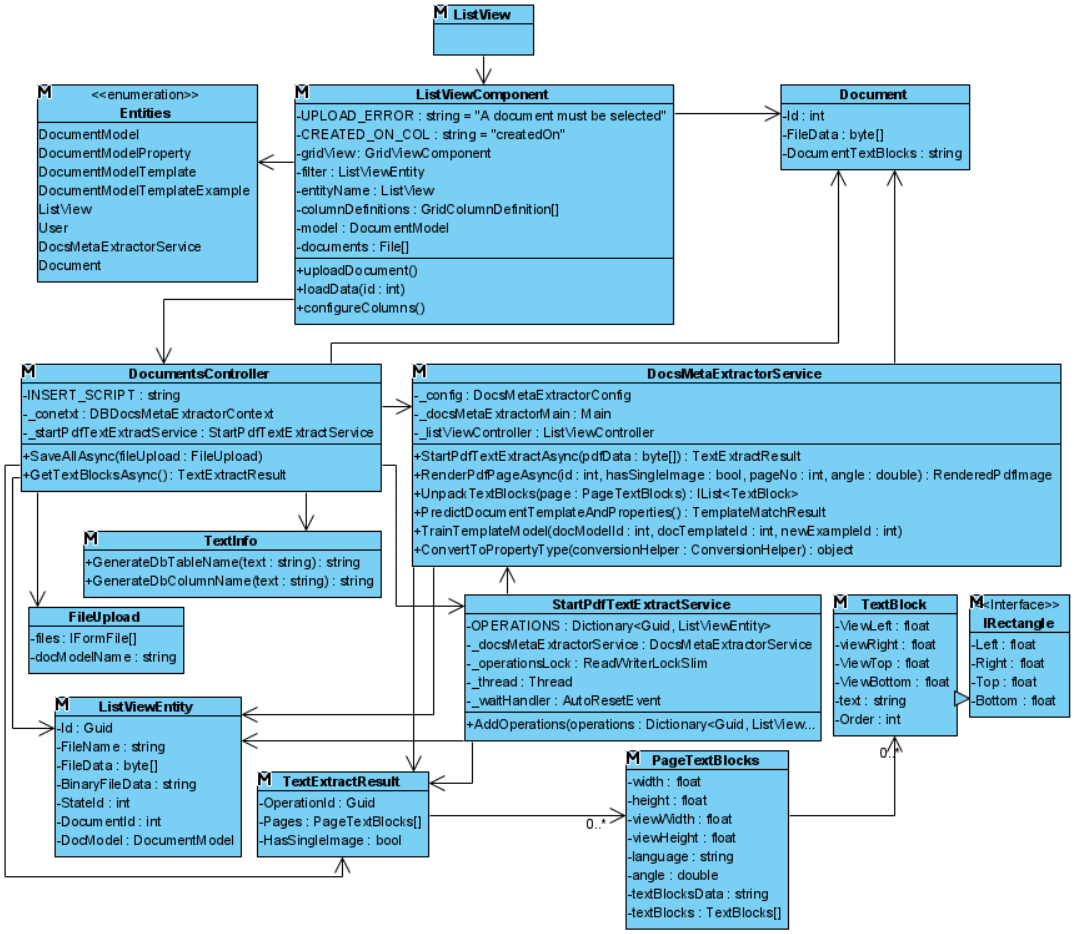


Figure 32: Upload Documents Design Class Diagram

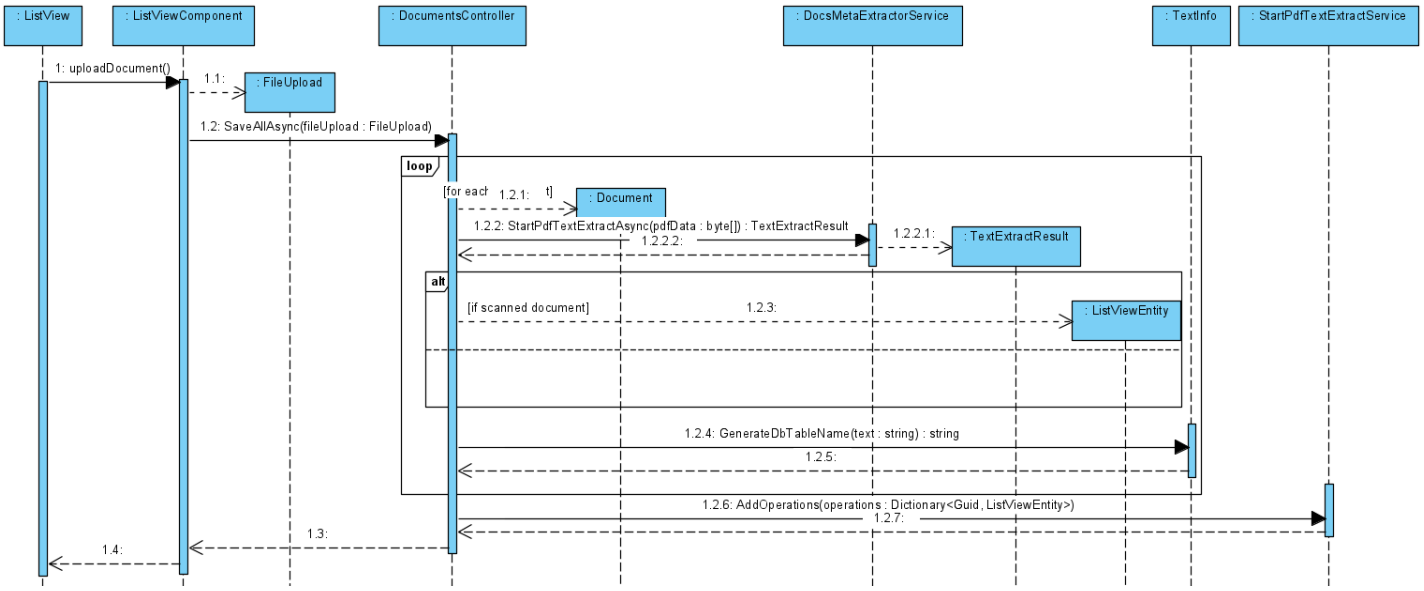


Figure 33: Upload Documents Design Sequence Diagram

4.2.1.4 UC4

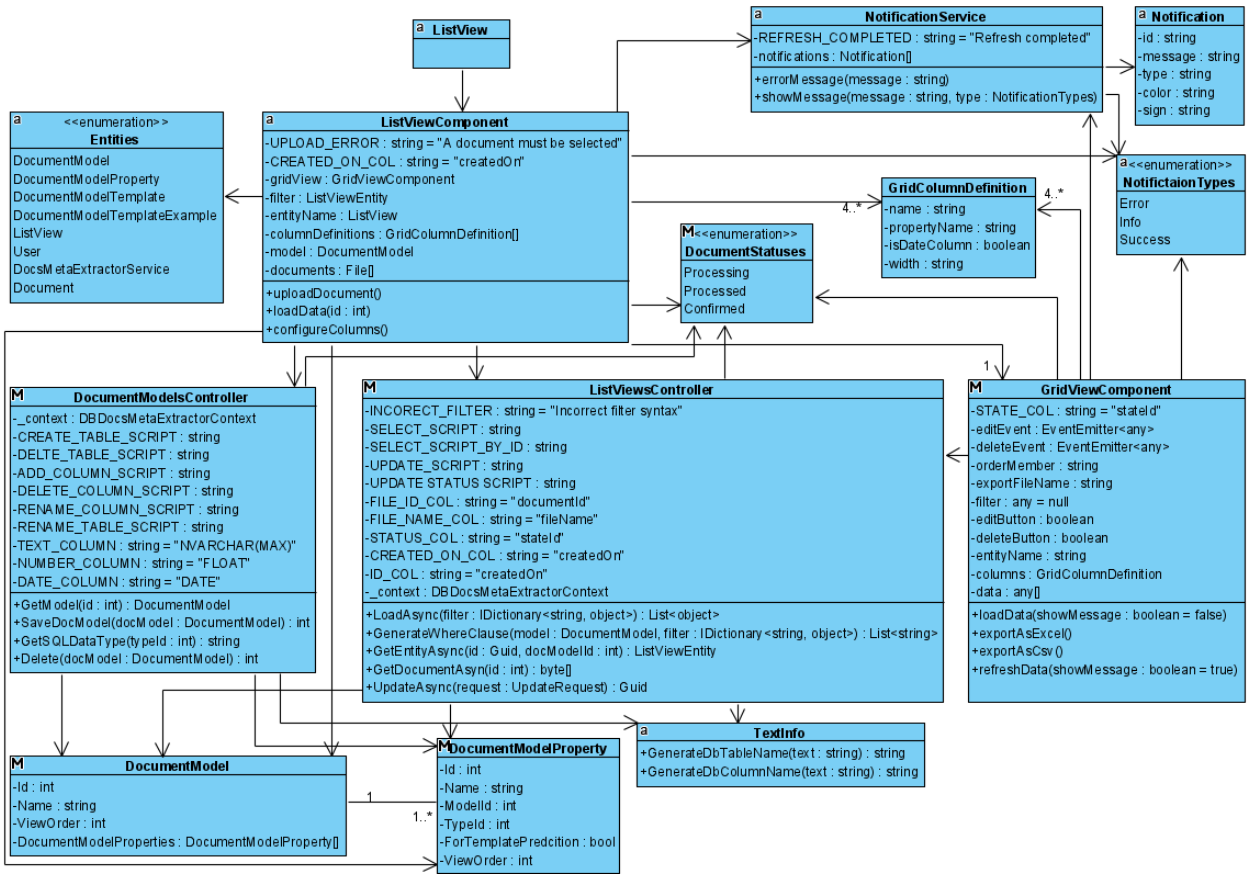


Figure 34: View Documents Design Class Diagram

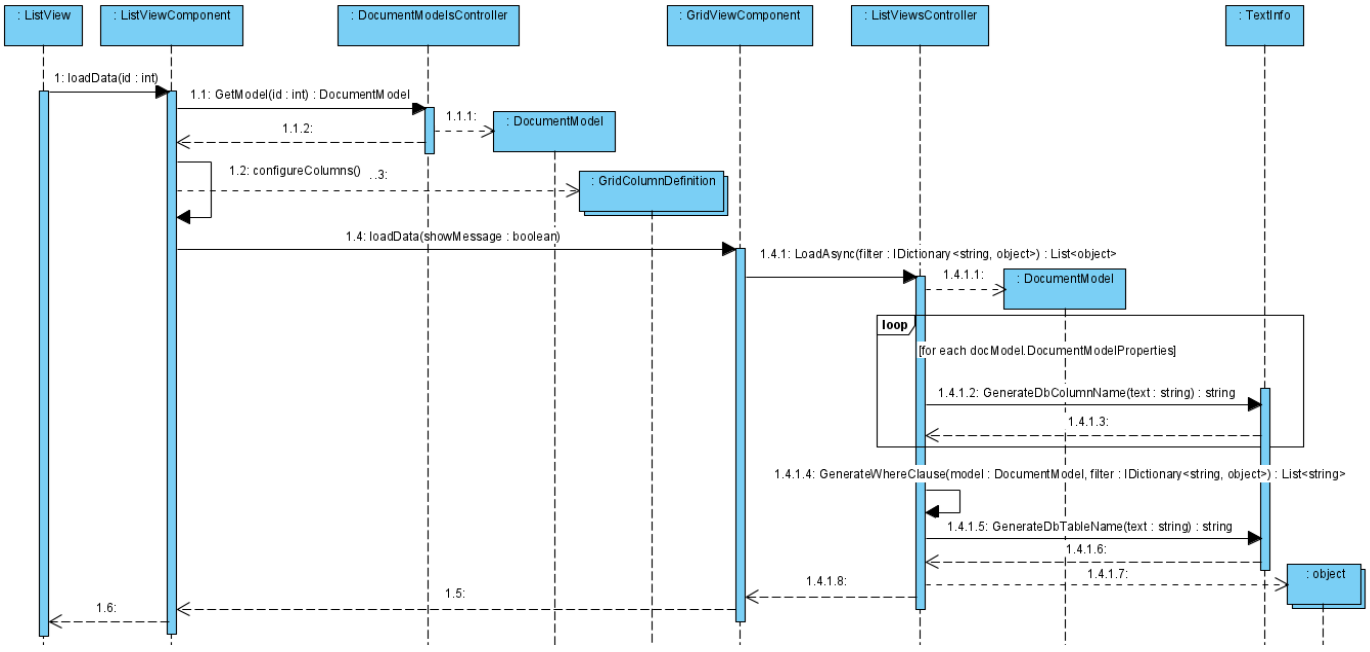


Figure 35: View Documents Design Sequence Diagram

4.2.1.5 UC5

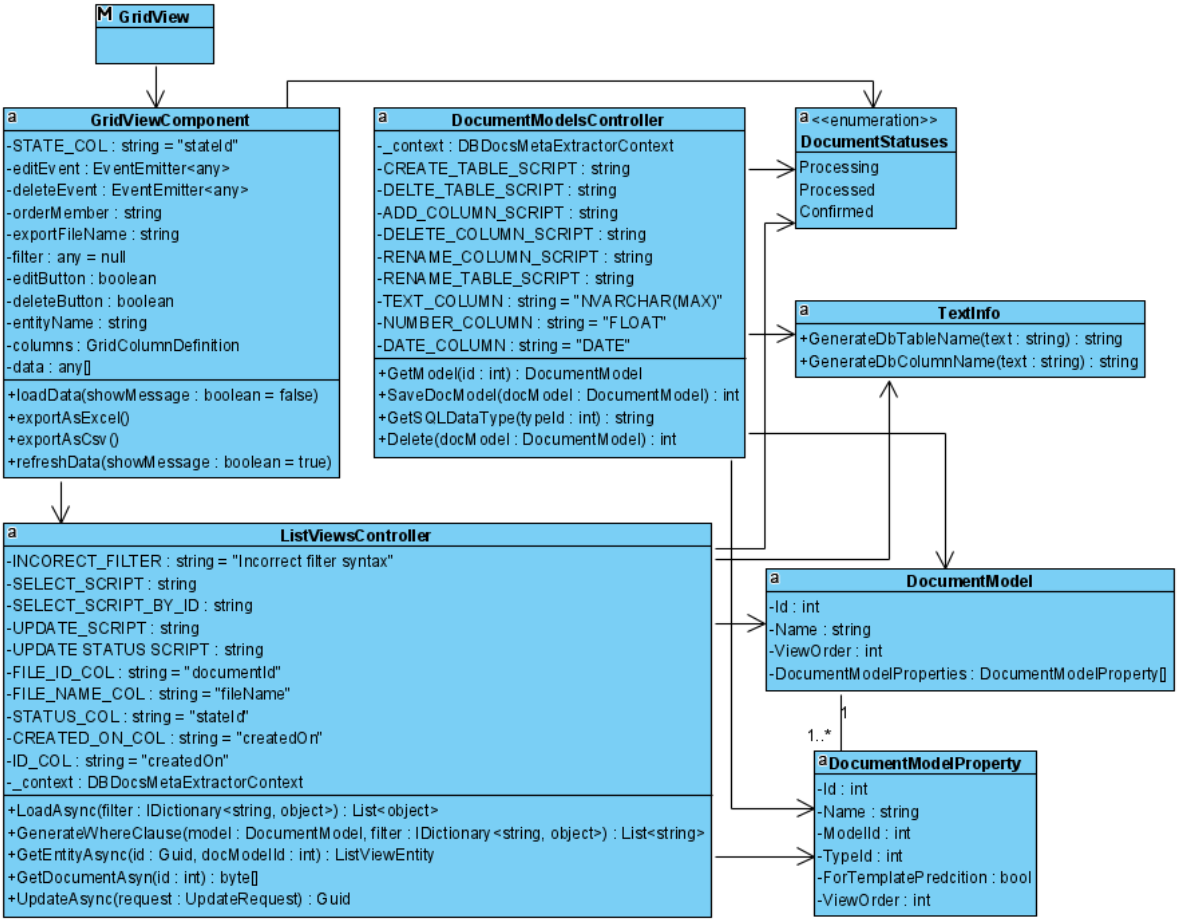


Figure 36: Filter Data Design Class Diagram

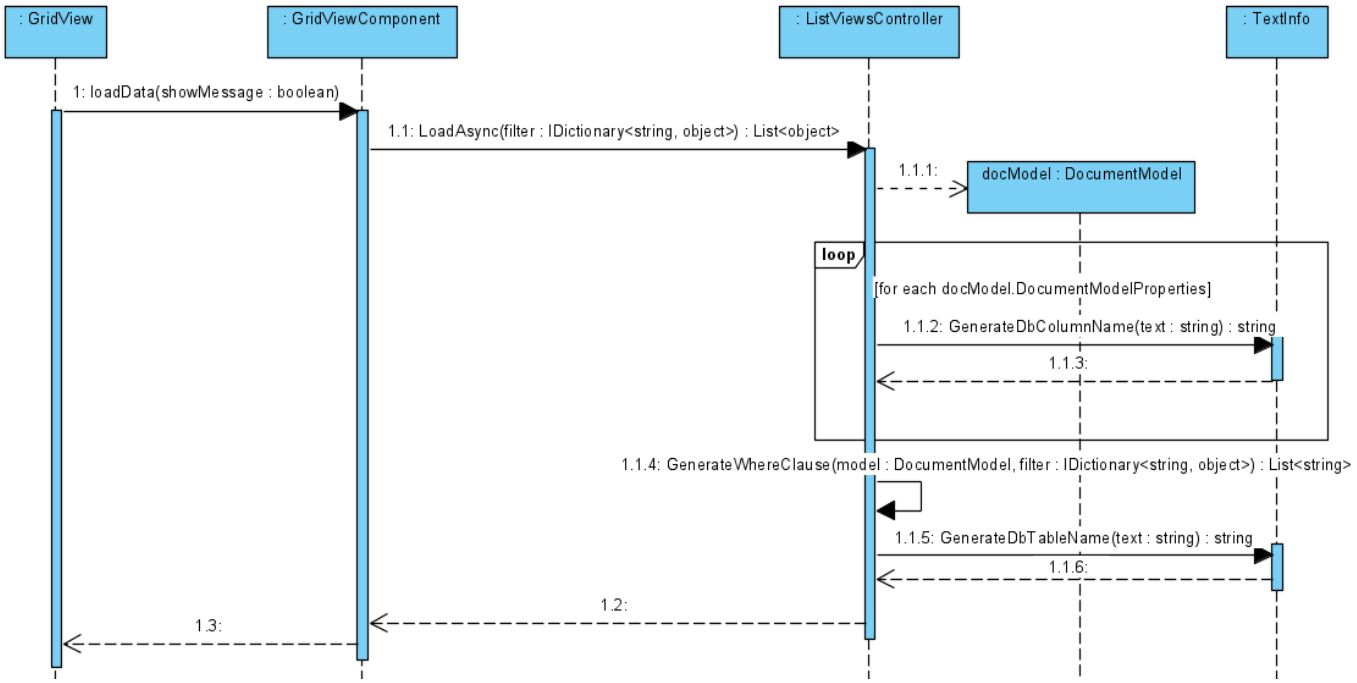


Figure 37: Filter Data Design Sequence Diagram

4.2.1.6 UC7

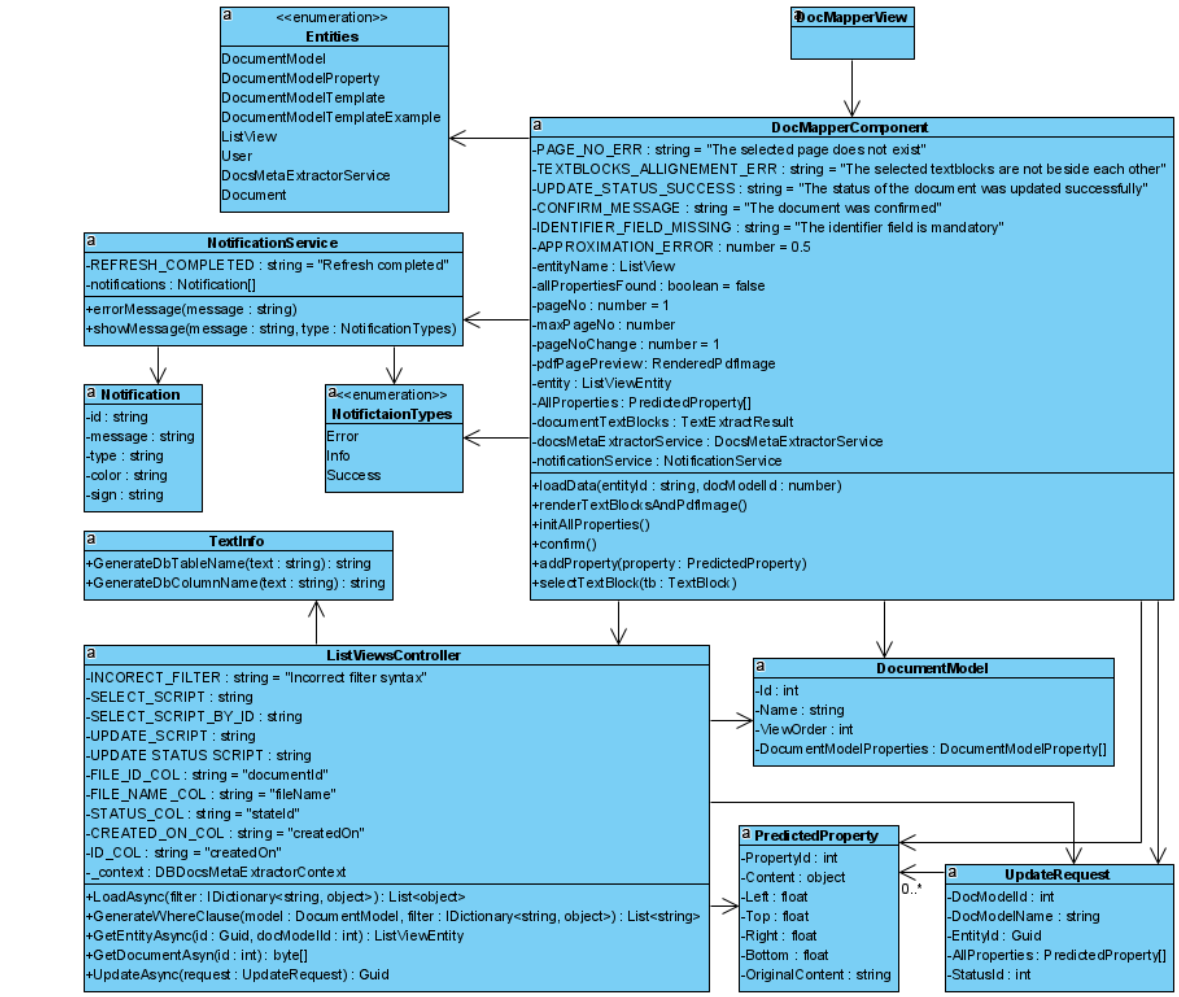


Figure 38: Validate Document Design Class Diagram

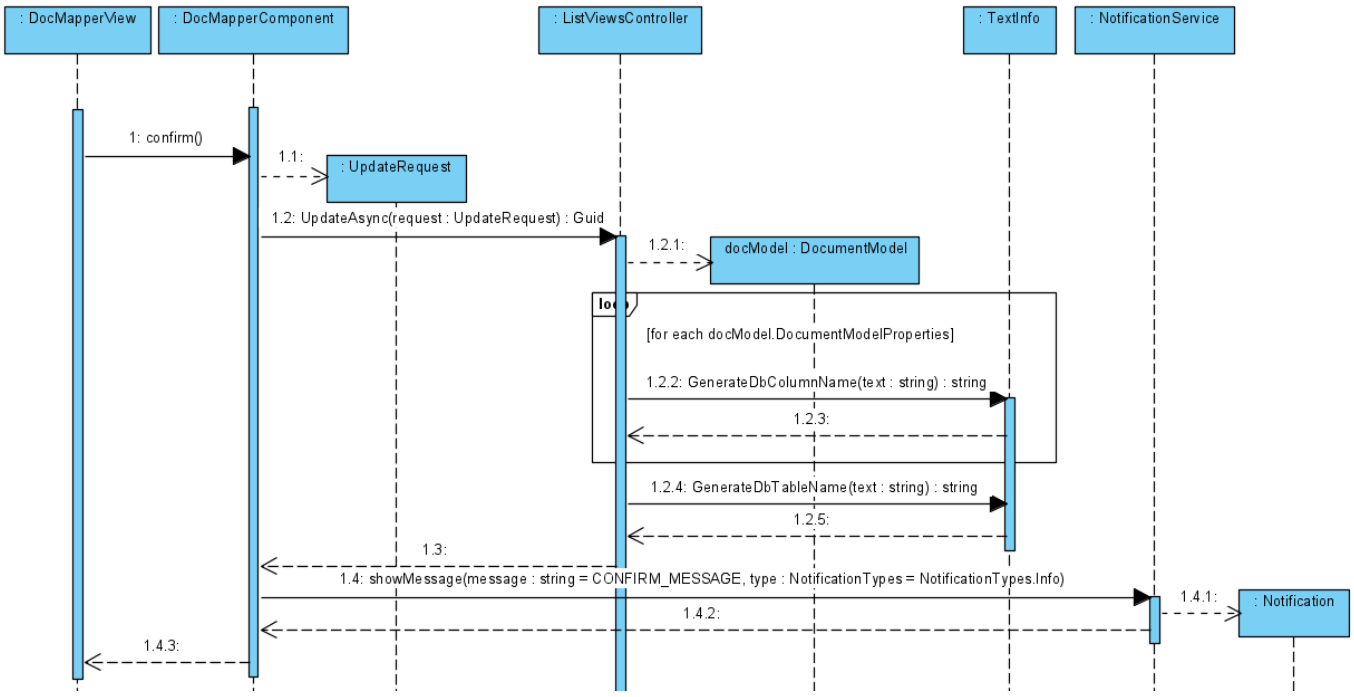


Figure 39: Validate Document Design Sequence Diagram

4.2.1.7 UC8

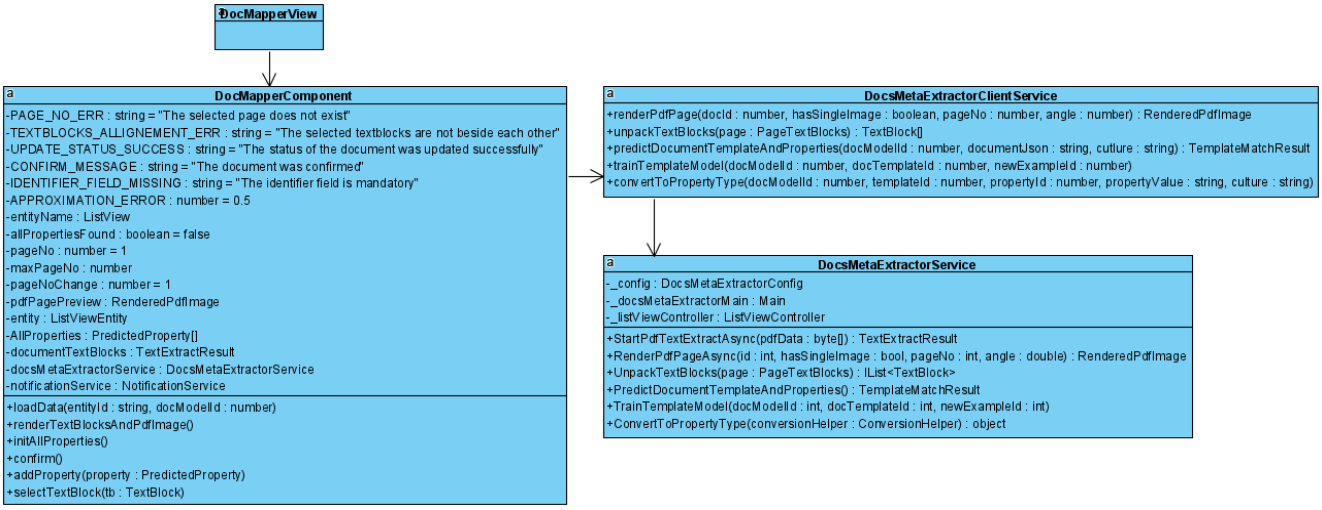


Figure 40: Train Template Design Class Diagram

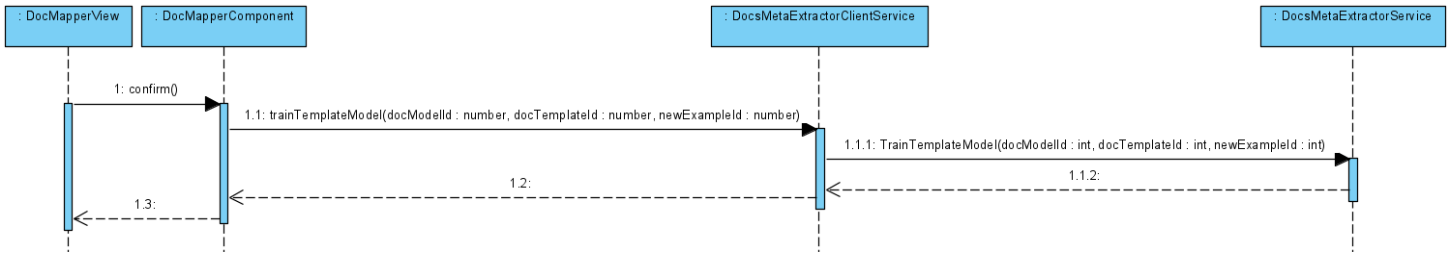


Figure 41: Train Template Design Sequence Diagram

4.2.1.8 UC9

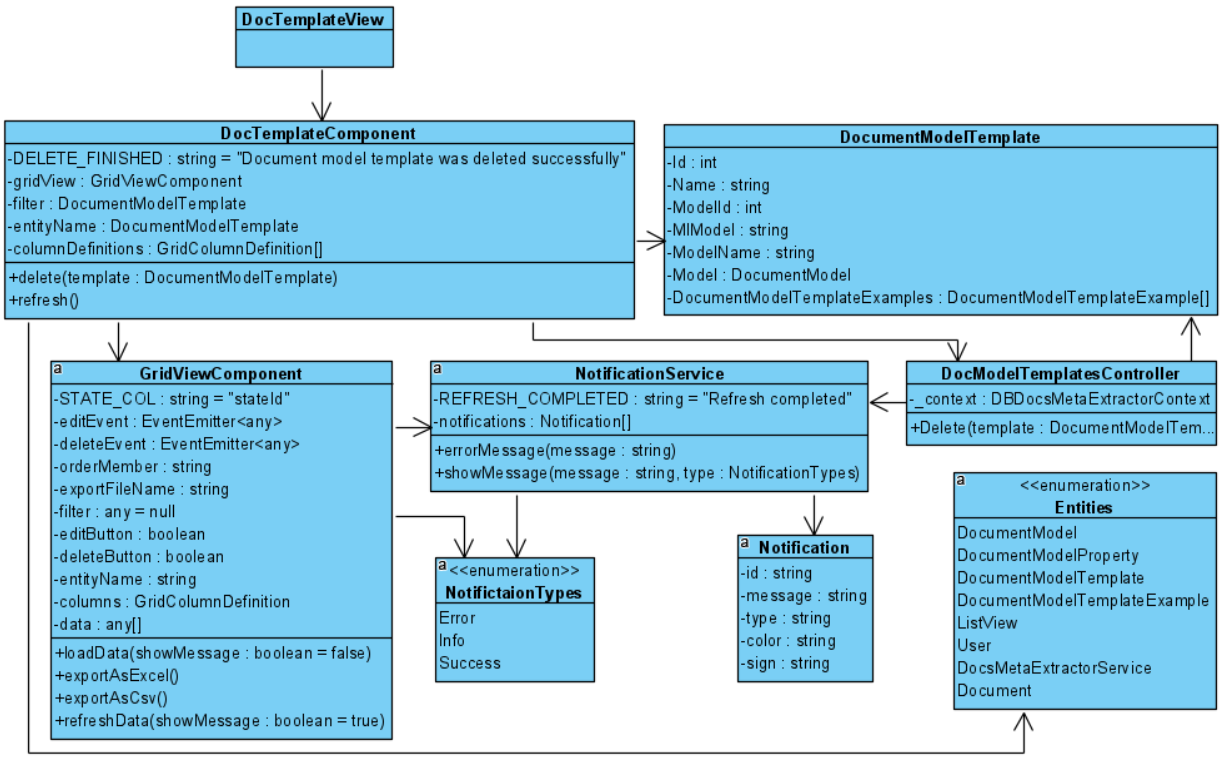


Figure 42: Delete Template Design Class Diagram

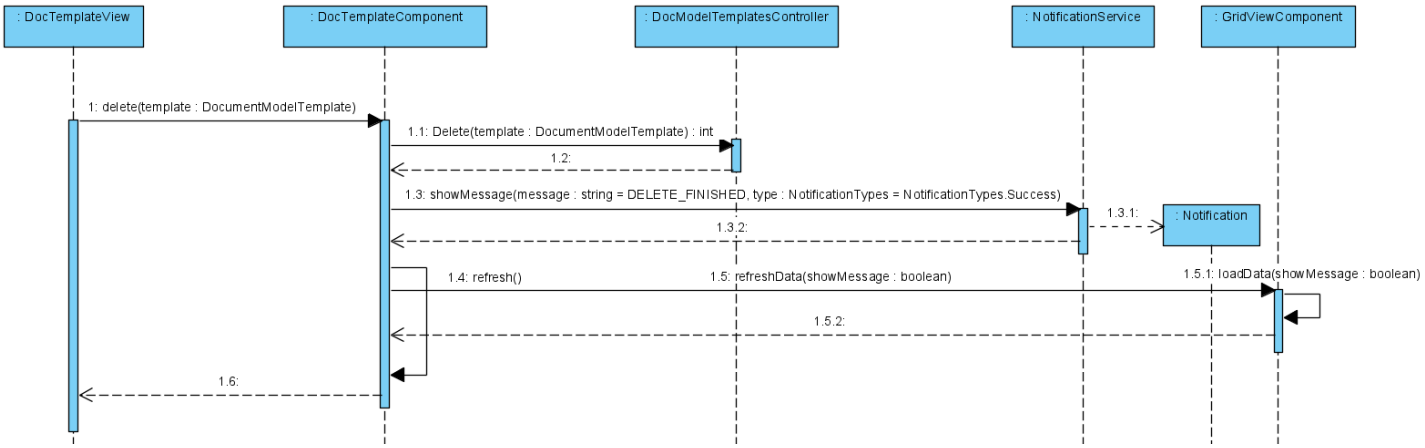


Figure 43: Delete Template Design Sequence Diagram

4.2.1.9 UC10

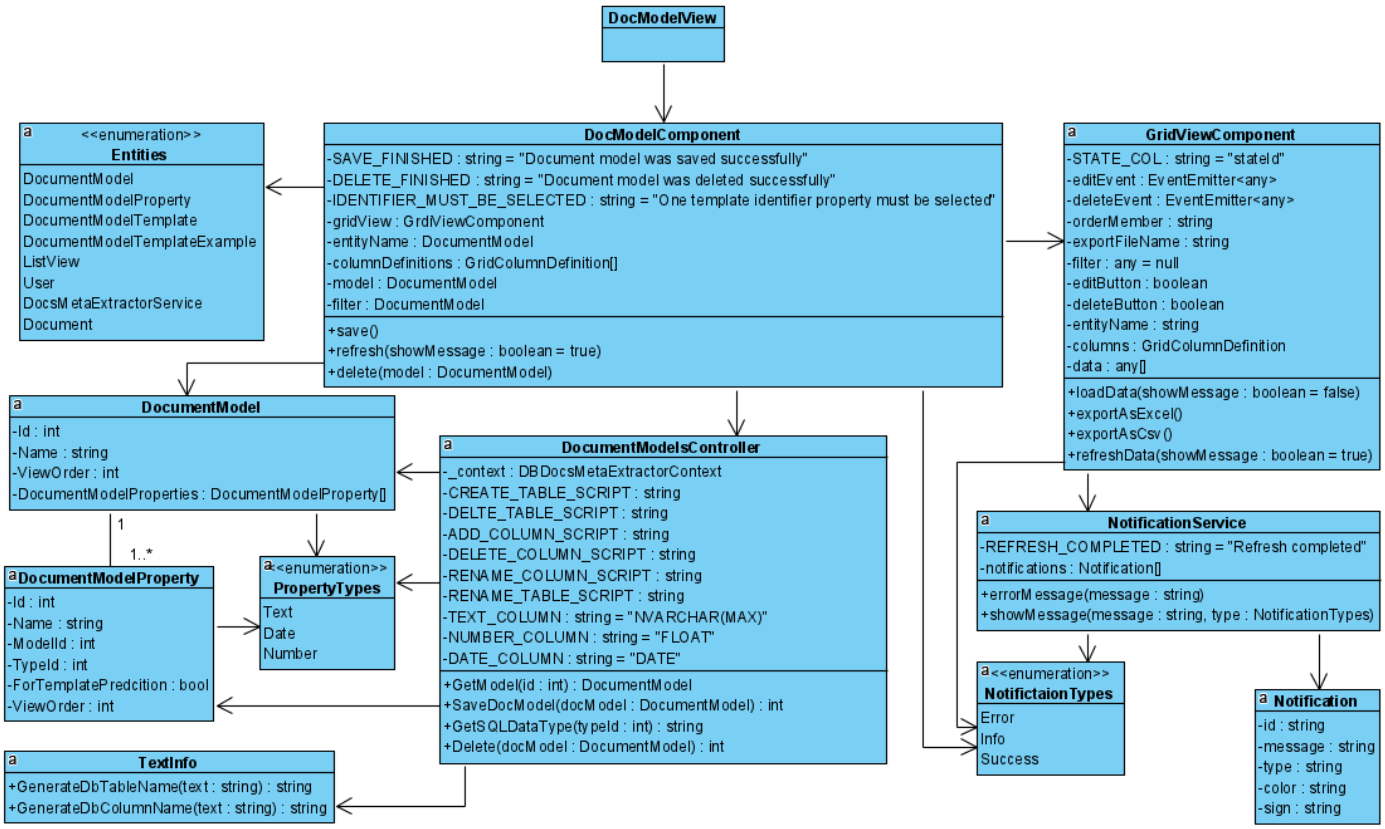


Figure 44: Manage Document Models Design Class Diagram

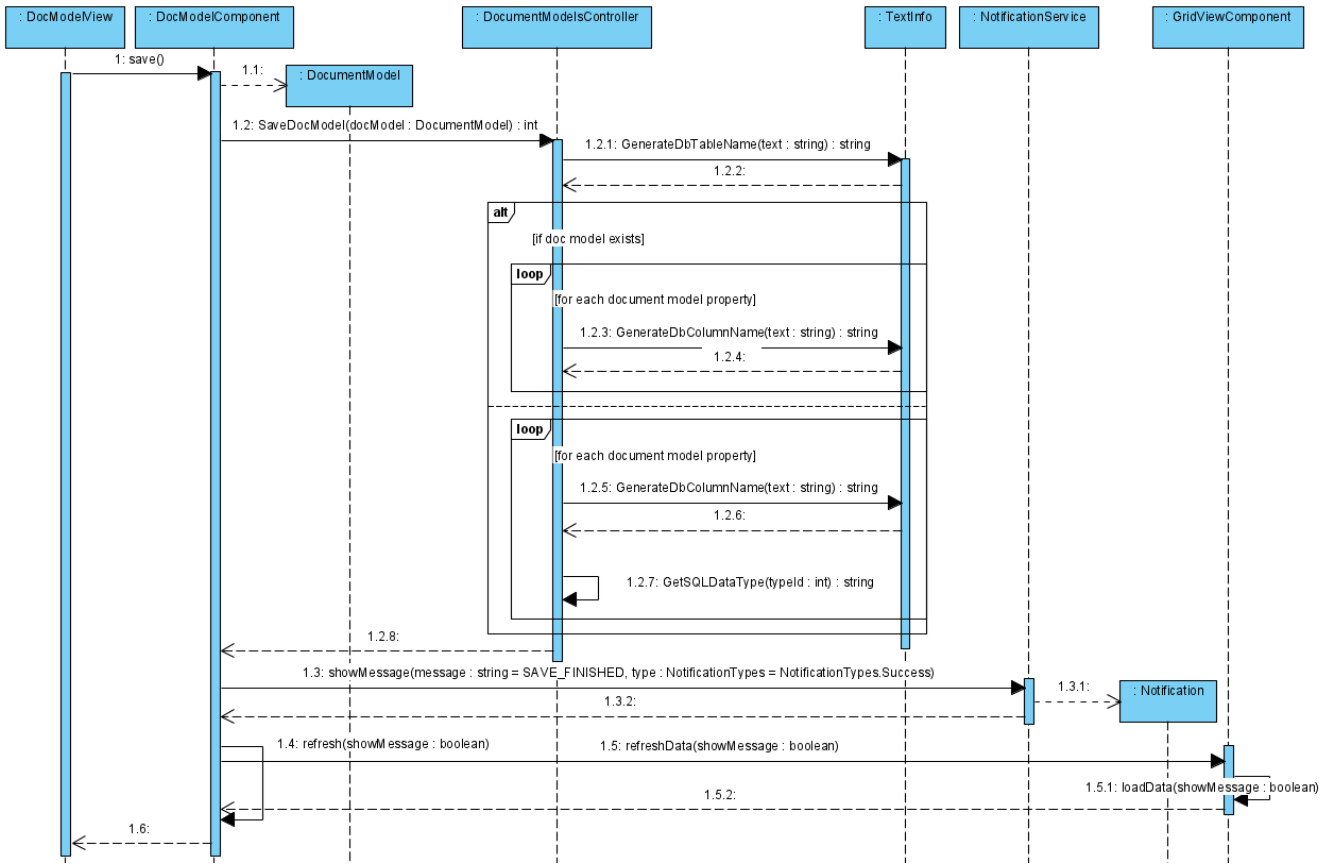


Figure 45: Add and Modify Document Models Design Sequence Diagram

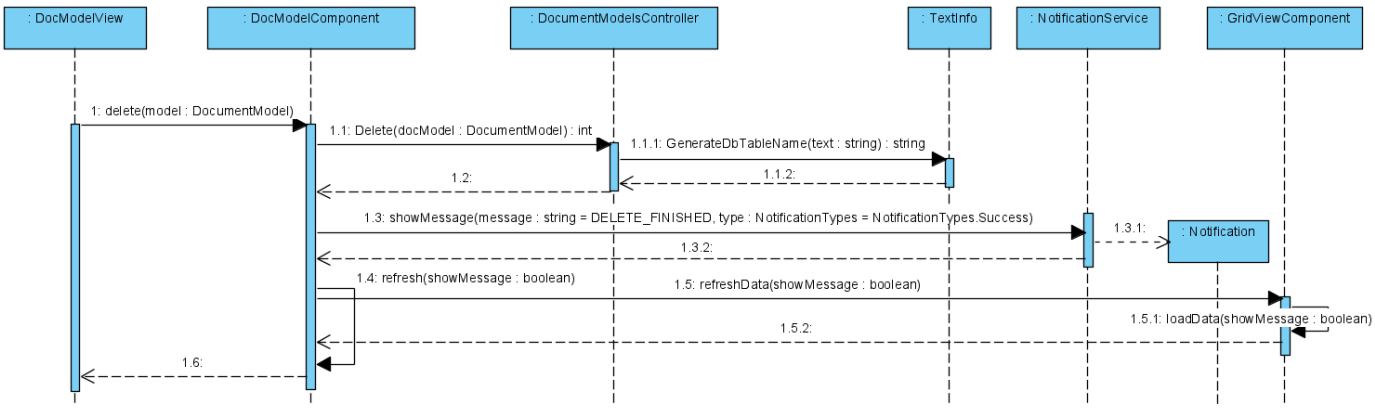


Figure 46: Delete Document Model Design Sequence Diagram

4.2.1.10 UC11

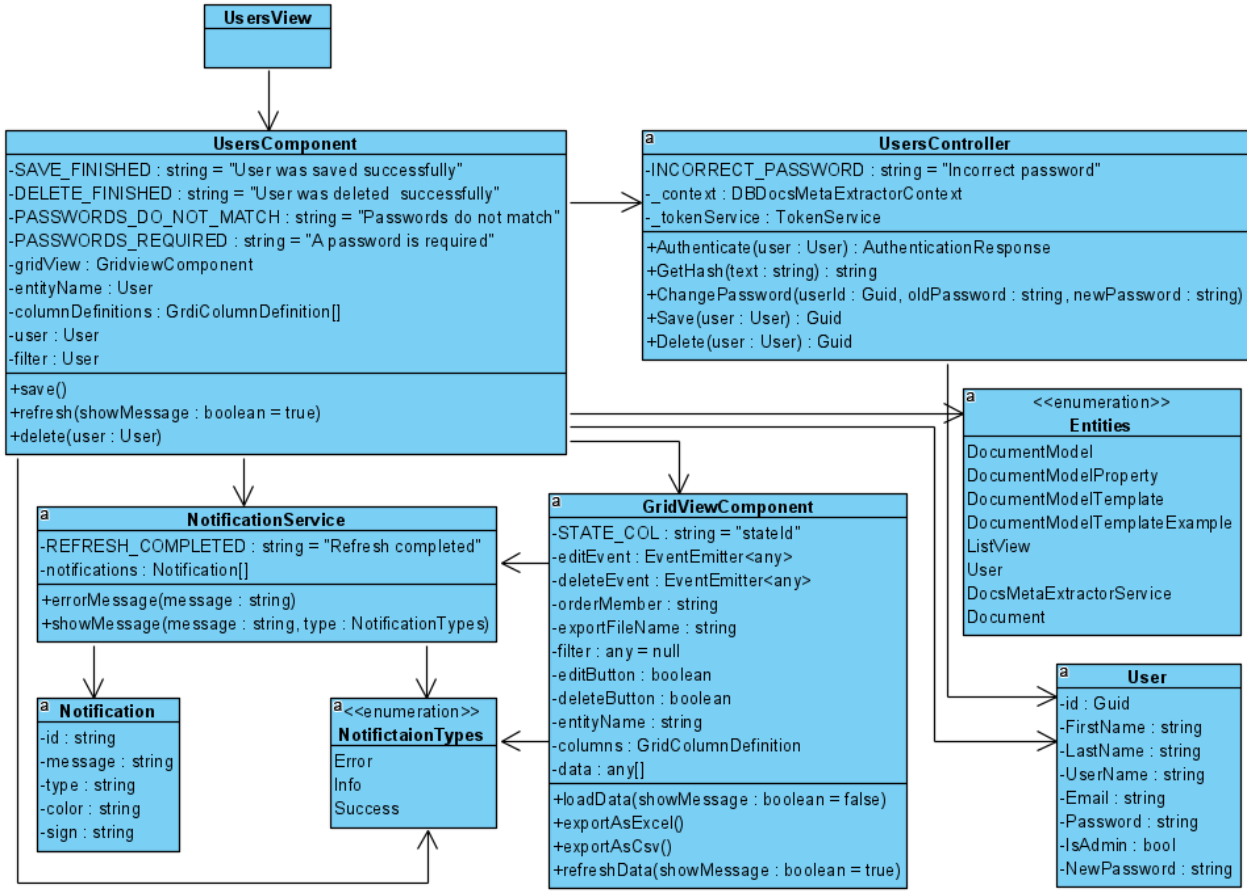


Figure 47: Manage Users Design Class Diagram

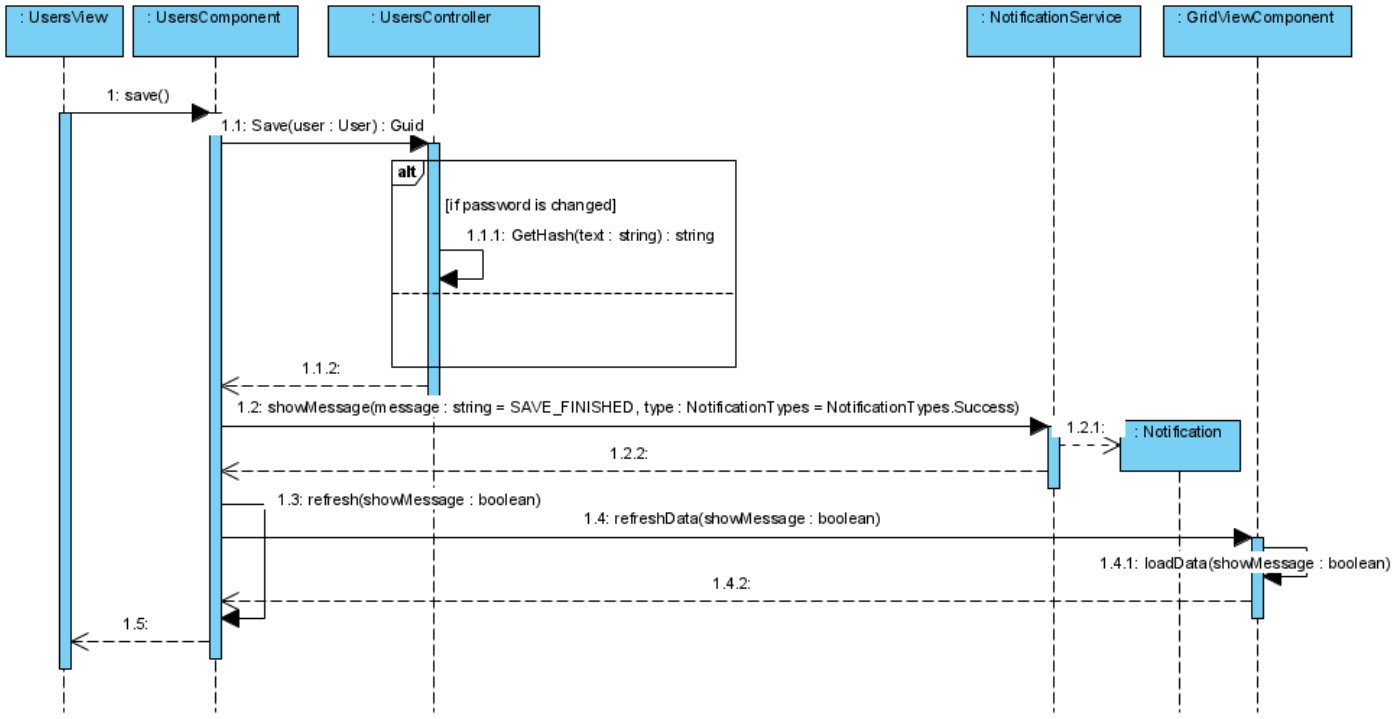


Figure 48: Add and Edit Users Design Sequence Diagram

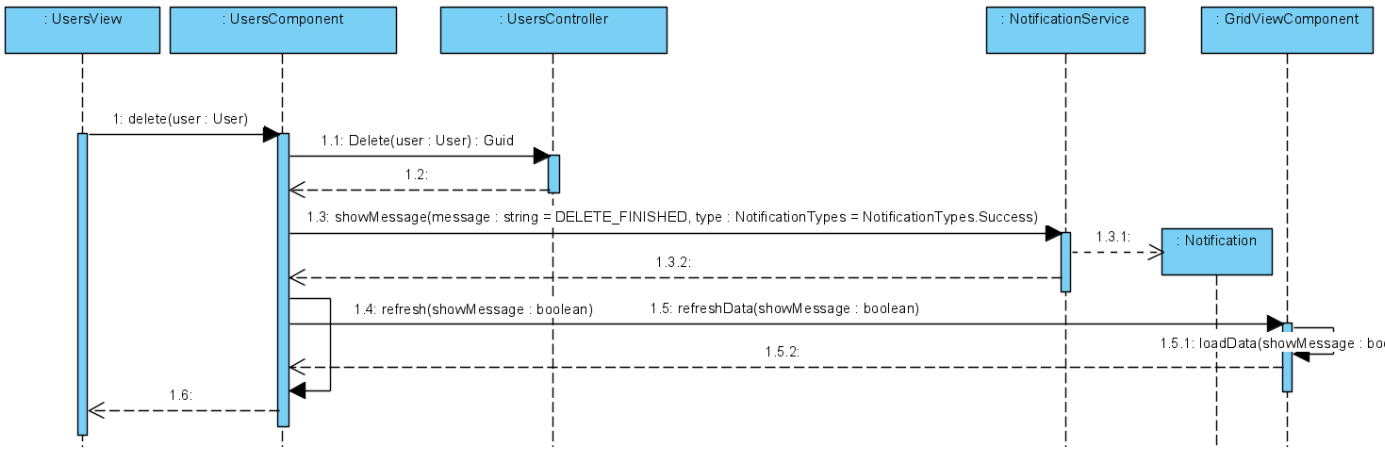


Figure 49: Delete User Design Sequence Diagram

4.2.2 Database structure

The structure of the database can be analyzed from three different point of views. First from the point of the plug-in. To store the training data and the mapped properties the plug-in needs five tables as described in *Figure 50* bellow.

Figure 50: Plug-in Database ERD

The web application in addition to these tables needs three more tables. One for the users, one for all the uploaded documents and one which contains the list of document statuses. Therefore, when deploying the application, the database's structure looks like in *Figure 51*.

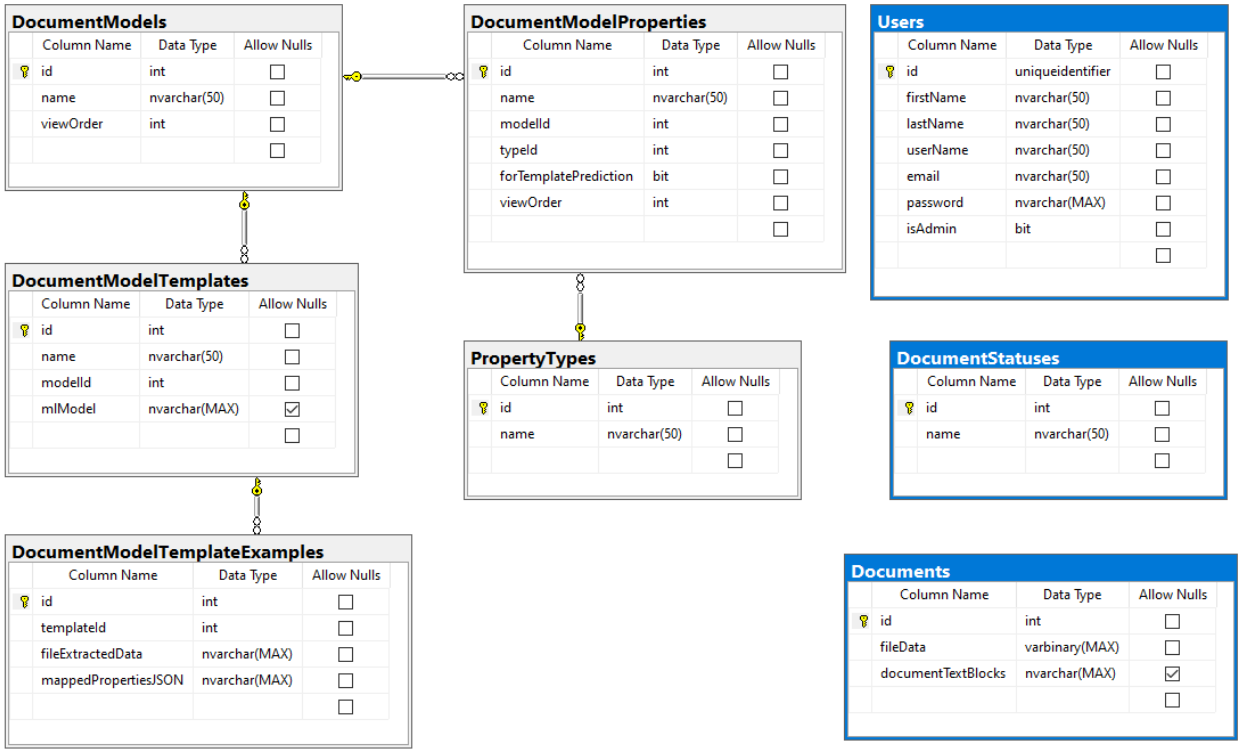


Figure 51: Web App Database ERD at Deployment

For each document model added an associated table will be created dynamically on runtime. These tables by default have five columns:

Column Name	Primary Key	Data Type	Default Value	Foreign Key
id	X	uniqueidentifier		
documentId		int		X - Documents
fileName		nvarchar(MAX)		
createdOn		datetime	GETDATE()	
stateId		int	1	X - DocumentStatuses

Table 18: Default Columns for New Document Model Table

The rest of the columns are generated according to the properties defined for the document model. In *Figure 53* a table is added which represents a document model called *Purchase Orders*, having the following properties:

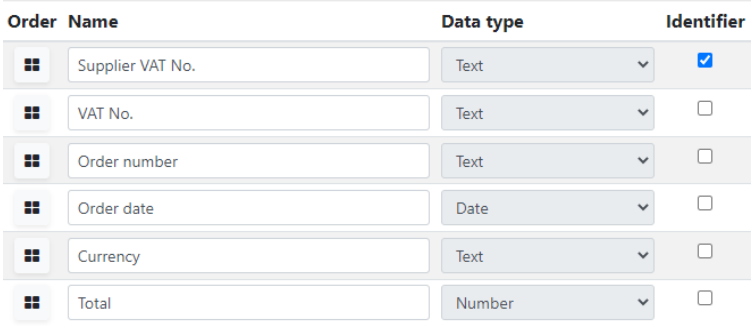


Figure 52: Purchase Orders Custom Properties

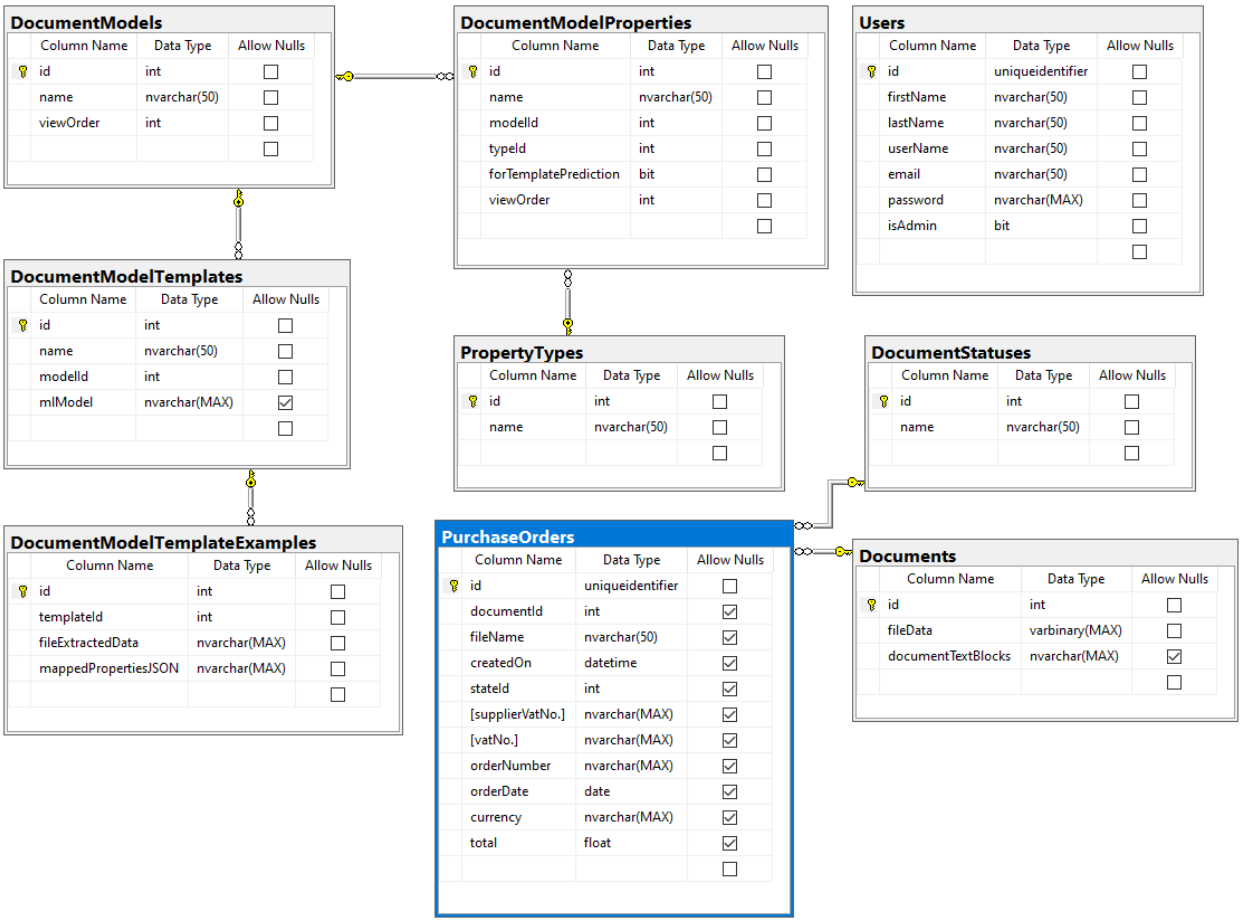


Figure 53: Web App Database ERD at Runtime

5 Implementation

5.1 Technologies

5.1.1 Development

To discuss about the technologies used to implement the solution the web application and the plug-in are separated.

The web application uses Angular 10, TypeScript, Bootstrap4, HTML and CSS in its front-end. TypeScript is a programming language developed by Microsoft. It is a more structured and stricter superset of JavaScript, this means that any JavaScript code can be interpreted in TypeScript as well, but not in the other way. Angular is a framework for building single-page client applications, it has components, modules, directives that ease the development. It works in combination with HTML and TypeScript. Angular makes it possible to import features from several other modules and helps in structuring the code. Bootstrap is an open-source CSS framework, it contains predefined classes, templates, interface components. This offers a lot of help in customizing the user interface (UI).

For the back-end .NET 5.0 is used. It is an open-source Microsoft framework, it was released in November 2020. It combines .NET Core and .NET Framework. It comes with major benefits as no more updates will be made for the other two, only .NET 5.0 is going to be constantly updated. There are already some differences, for example the newer C# versions are supported only by .NET 5.0. Moreover, .NET 5.0 came with Entity Framework (EF) Core 5. EF Core is an object-database mapper for .NET. It provides functions to query, update, change and migrate databases. It is compatible with many database servers, including SQL Server, which is used to store the data.

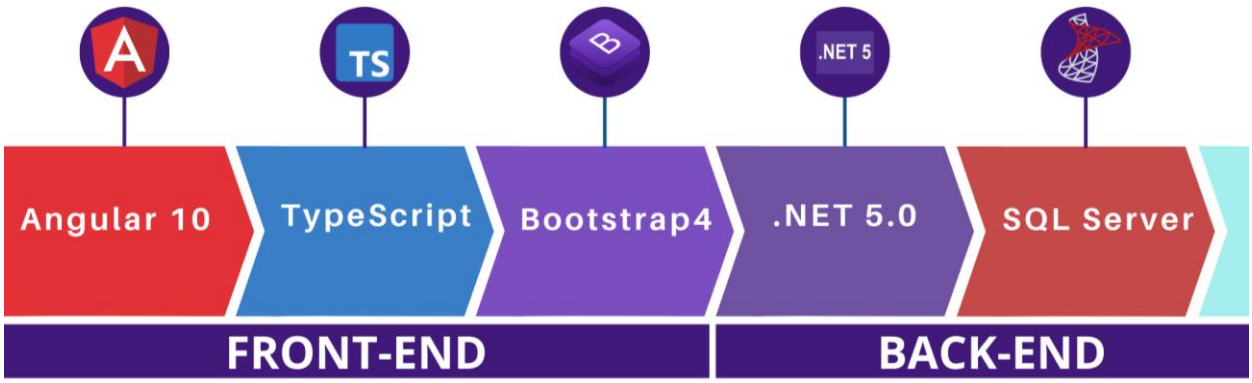


Figure 54: Technologies Used

The plug-in is implemented as a class library, by using .NET Framework, with the assembly name *DocsMetaExtractor*. A class library contains classes, components, functions that can be used by other applications, .NET Framework provides language interoperability, meaning that different programming languages can interact with each other in the same system.

5.1.2 Microsoft Azure Cognitive Services

To interpret documents Azure Cognitive Services are used, these are cloud-based and can be accessed by REST API calls.

To extract text from scanned or captured documents the Computer Vision API is used. This has three services, an OCR, an Image Analysis and a Spatial Analysis. In this solution the Computer Vision Read API is used which is the latest Azure OCR technology with AI. It supports a total of 73 languages, including Romanian, and handwritten text in English. It can extract text from documents with mixed languages and mixed styles. The response is a JSON and it is encrypted and stored temporarily, it can be retrieved by calling the Get Read API. It can process JPEG, PNG, BMP, PDF and TIFF input formats with the maximum size of 50 MB. For PDF and TIFF files the first 2000 pages are processed, which is more than enough for our system. And the images dimensions must be at least 50x50 pixels and at most 10000x10000 pixels. With these in mind the Read API feature costs 1.265 EUR per 1000 transactions for the first million transactions, for more it costs 0.506 EUR per 1000 transactions and supports 10 transitions per seconds (TPS).

The language detection of documents is realized by the Text Analytics API of the Cognitive Services, which provides Natural Language Processing (NLP) features. It requires for input a JSON document with an id field and the content. It can process up to 1000 documents per collection. For each document, the language is returned with a score, between 0 and 1, representing the confidence level. The Language Detection API costs are varying based on the number of requests and represent the amount for 1000 transactions. For the first 500,000 documents is 0.844 Euro, next up to 2.5 million records it is 0.633 EUR, up to 10 million is 0.253 EUR and finally for more than 10 million records is 0.211 EUR. As the number of requests grow the cost are smaller and smaller.

5.2 Plug-in

The plug-in contains the generic functions to process PDF documents, train properties and to predict them in documents. These activities can be divided into four phases, the text extraction, the document properties mapping, the training and the prediction phase.

For each phase there are several functions implemented to satisfy the business requirements. In total there are eight public functions and are distributed into the phases as shown in *Table 19* bellow.

Text Extraction Phase	Properties Mapping Phase	Training Phase	Prediction Phase
Start Pdf Text Extract	**Unpack Text Blocks**	**Get Document Template**	**Predict Document Template and Properties**
Get Pdf Text Extract Result	**Render Pdf Page** **Convert to Property Type**	**Train Template Model**	**Convert to Property Type**

Table 19: Plug-in Functions

It can be observed that the method convert to property type is used in two different phases, however depending in which phase it is called it has different input parameters and works differently, but has the same scope.

The plugin when initialized gets a configuration entity, which provides the connection string to the SQL database and the Microsoft Azure Cognitive Services setup configuration. This is called *DocsMetaExtractorConfig* and contains the following members:

- **visionEndPoint** – endpoint URL for Microsoft Azure Cognitive Services Vision API
- **visionApiKey** – API key for Microsoft Azure Cognitive Services Vision API
- **visionVersion** *(optional)* – target version of Microsoft Azure Cognitive Services Vision API, if not specified the default version will be used
- **textAnalyticsEndpoint** – endpoint URL for Microsoft Azure Cognitive Services Text Analytics API
- **textAnalyticsApiKey** – API key the Microsoft Azure Cognitive Services Text Analytics API
- **overrideLanguage** *(optional)* – language of the documents that will be uploaded, if not specified the language will be detected automatically, by the Text Analytics API
- **connectionString** – connection link to the database, providing the server, database and the user credentials

The configuration data is extracted from the *appsettings.json* file in the web application project. The web application contains a service, called *DocsMetaExtractorService* which at startup initializes and configures the plug-in. Through this service the web application is communicating with the plug-in.

Figure 55: App Settings Structure

In the next sub-chapters, the implementation of the functions is presented according to the four phases.

5.2.1 Text Extraction Phase

In this phase the textual content of a document is returned. Each text block contains the bounding rectangle of the text, the value and the order of it. The order is used to track the blocks with the same text and it is assigned in the apparition order. The result text blocks are binary serialized in order to store and send them efficiently. In the database these serialized text blocks are saved only.

The function *StartPdfTextExtract* is used to get these text blocks and to identify the language. It requires as input the document as a byte array. It checks if the document is a real PDF or a scanned one. For real PDF documents with the help of a library called iText7 (https://itextpdf.com/en) extracts the text and the locations from the metadata and the result is returned synchronously. In the case of scanned documents an internal function is called to prepare and send the request to the Computer Vision Read API and the operation identifier is returned. Later with this identifier the results are requested from the Get Read API, by the *GetPdfTextExtractResult* function.

The algorithm used to identify if a document is scanned or real is simplistically described in the following pseudo-code.

```
FUNCTION StartPdfTextExtract(pdfData)
   pdfDoc = PdfDocument(pdfData)
   firstPage = pdfDoc.getFirstPage()
   pageCount = pdfDoc.GetNumberOfPages()

   IF pageCount == 0 THEN
       RETURN NULL
   ENDIF

   IF firstPage contains an image of the size of the page THEN //scanned document with one
image as a page
       operationId = CALL StartAdvancedPdfOCR(pdfData, CONFIG)
       RETURN operationId
   ELSE IF the area of images IN the first page <= (firstPage area / 0.5) THEN
       pages<- ExtractPdfText(pdfDoc) // real PDF, text extracted from metadata
       IF pages == NULL THEN
           operationId = CALL StartAdvancedPdfOCR(pdfData, CONFIG)
           RETURN operationId
       ENDIF

       DetectLanguage(pages, CONFIG)
       RETURN pages
   ENDIF

   operationId = CALL StartAdvancedPdfOCR(pdfData, CONFIG)
   RETURN operationId
END FUNCTION
```

The OCR service also returns the angle of the document and the facing, by using these the blocks are straightened up. This is very important as documents can be photographed or scanned with a declination angle, or upside down and these need corrections. Also, in order to provide a great experience, the UI needs to be as intuitive and easy to use as possible. With this in mind users should not waste time to figure out how the document is oriented and where are the fields. These transformations can be done automatically.

The text blocks are also normalized between 0-100 in order to be able to calculate the position for each one in the mapping phase. For a better preview, the pages of the document itself are aligned too. Therefore, in the end of this part the text blocks with normalized and aligned coordinates are returned for the uploaded PDF files, for scanned and real ones as well, ready to be displayed on the screen.

5.2.2 Properties Mapping Phase

For this phase three simple functions are implemented in order to provide the UI for the user to map the unmatched properties of a document, namely the *UnpackTextBlocks*, *RenderPdfPage* and *ConvertToPropertyType* functions.

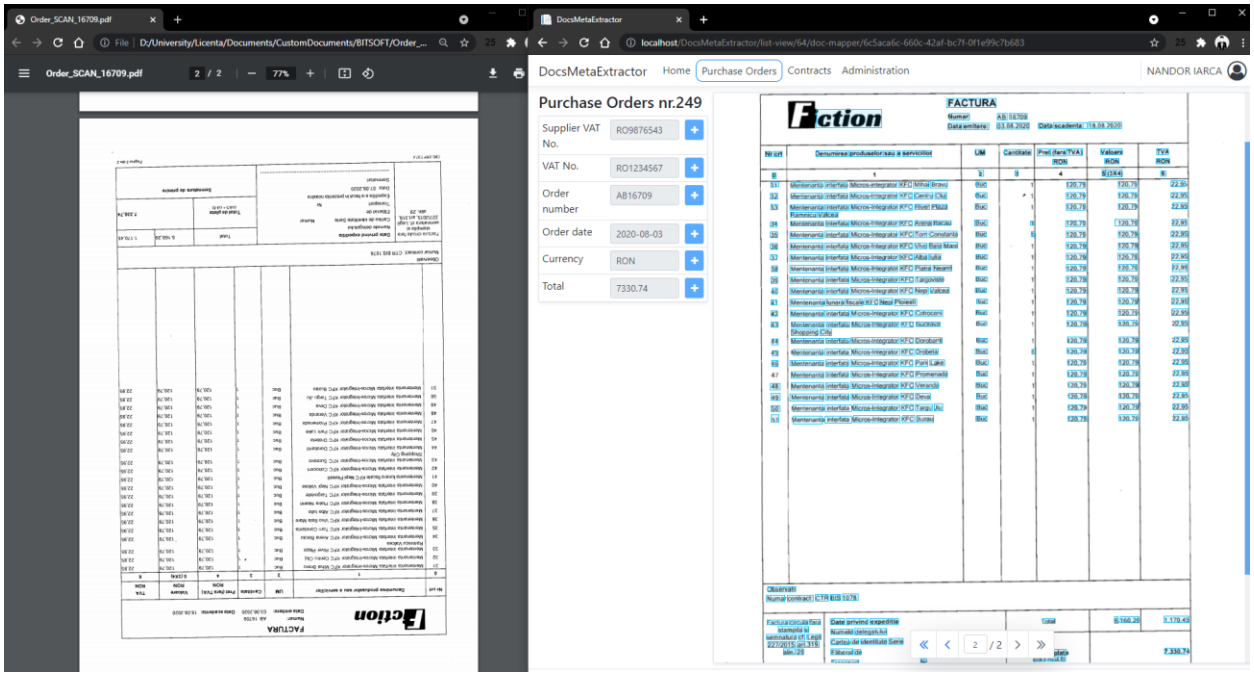


Figure 56: Text Extraction Phase - Straightened Invoice Example

UnpackTextBlock gets as input the binary serialized text blocks string for a page and just simply deserializes them. This helps in achieving a better performance and not to use unnecessarily storage space on client side, as only those text blocks are deserialized what page the user wants to see. For example, in a document of 100 pages the user is interested only in the first two pages. When the user navigates to a page, the system checks if the text blocks were deserialized, if not the plug-in is called, so the memory is not overloaded.

RenderPdfPage is used to render a pdf page as an image to be able to display a preview of it in the application. It uses EVO PDF (https://www.evopdf.com/) library to convert the page to an image represented as a byte array. Moreover, the method gets as input the transformations needed for the document to be straight and it applies them. This can be observed in *Figure 56* as well, the preview image is straight, not only just the text blocks. This function, in the same way as for the serialized text blocks unpacking, only the needed pages are rendered, based on the user's interaction with the application.

Every time the user maps a date or number type the *ConvertToPropertyType* function is called in order to parse the input string to a date or to a double, depending on the property type. This conversion is made based on the culture if it is known, else an invariant culture is defined for it.

5.2.3 Training Phase

In this phase a training is running for a template, where the algorithm structures the data and creates a model which to follow in the prediction process.

The plug-in has its own caching service in order to avoid reading from the database repeatedly, it stores every template for every document model. The *GetDocumentTemplate* is used to get a template associated to a document model. If the template does not exist than it is added, saved in the database and cached.

TrainTemplateModel requires as input the document model id, the template id and the id of the newly uploaded example. For the first two un-trained examples the user must map the fields from the UI by using the previously mentioned text blocks layer. When finished the algorithm receives the mapped properties and tries to identify them. There are two types of properties, fixed and dynamic. For example, on an invoice usually the invoice number, date and company code are in the same position, in contrary the total amount has a dynamic position because it depends on the list of the items bought. The algorithm first checks if a property has a fixed location in all of the examples, if it has than the property is labeled as fixed, and it is added to the model. If the property varies then the algorithm tries to find its label. Two types of labels are taken into consideration, left dynamic labels, which are fixed horizontally but are changing vertically, and top dynamic labels, which are the invers, they are fixed vertically and dynamic horizontally. The algorithm finds a label for the property from which in every example the property has the same distance in every direction of the label. If such a label is found the prediction model is updated. This is done for every property for the document model associated to the template, and for each property the whole example data set, meaning all of the documents of that template, is revised.

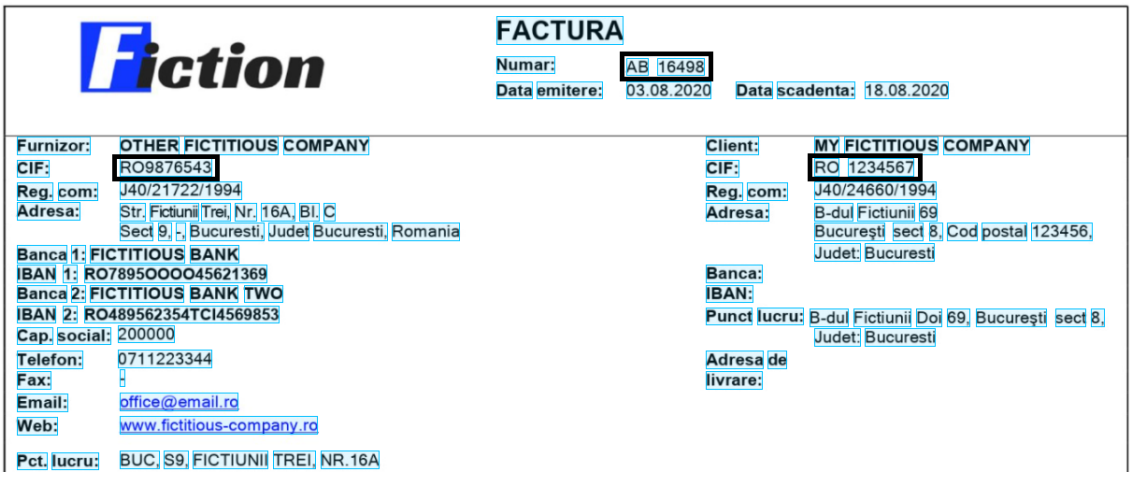


Figure 57: Example of Fixed Properties

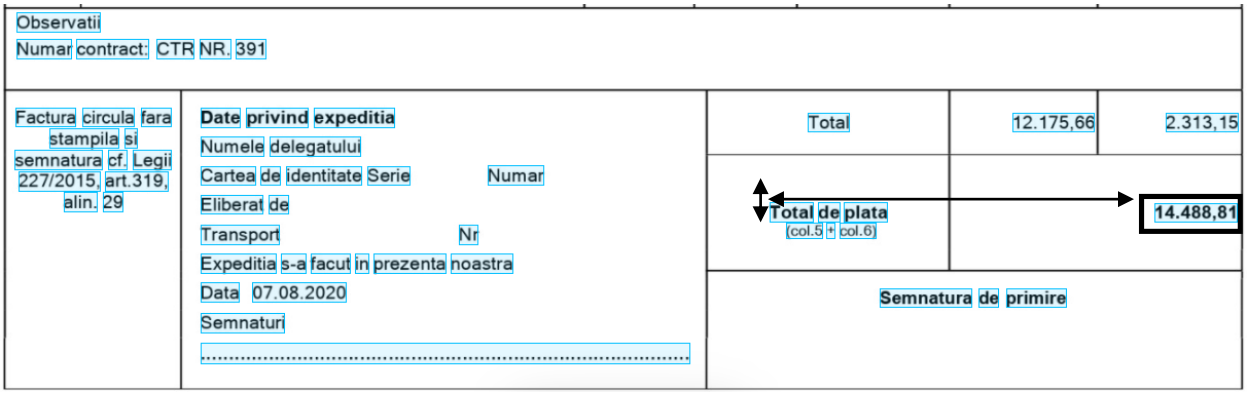


Figure 58: Example of Dynamic Property

Furthermore, scanned documents might not be as easily categorized in a template, because the fields need to be scaled and translated as well. So, if a scanned PDF is zoomed in or out or is it inclined, the coordinates must be scaled and translated not just straightened up. This is why the algorithm before the train scales and translates the coordinate of text blocks according to the first example uploaded, in this way it is certain that the documents of a template will overlay. For this, two fixed key labels are determined from the first page. These labels should have fixed position in each example, they are not allowed to contain numbers, must be longer than three characters and must be as far as possible from each other to get better accuracy. From the coordinates from the first example and the currently new example the horizontal and vertical ratio is computed and the left and top shift. Next, these are applied on the coordinates of the text blocks on each page by following the algorithm presented below:

```
FOR each page IN pages
    FOR each textBlock IN page.textBlocks
        textBlock.ScaledLeft = textBlock.Left* transformations.HRatio - transformations.LeftShift
        textBlock.ScaledTop = textBlock.Top * transformations.VRatio - transformations.TopShift
        textBlock.ScaledRight = textBlock.Right * transformations.HRatio - transformations.LeftShift
        textBlock.ScaledBottom = textBlock.Bottom * transformations.VRatio - transformations.TopShift
    ENDFOR
ENDFOR
```

Moreover, if the training failed the wrong examples are excluded from the database automatically. If there are more than two training examples the new example should be deleted, as the training worked before it was added. If only two documents are in the set both must be deleted as this is the first training and if it failed, with no previous working training, the wrong examples cannot be deduced.

Taken these steps into consideration the core of the algorithm is described in the pseudo code bellow.

```
FUNCTION TrainTemplate(docModelId, templateId, newExampleId)
    model = CALL DataModelCachingService.GetDocumentModel(docModelId)
    trainingExamples = all the examples in the training
    IF trainingExamples.Count< 2 THEN
        RETURN
    ENDIF

    IF NOT DetermineKeyLabels(model, template, trainingExamples, newExampleId) THEN
        Delete new example from training dataset
        RETURN
    ENDIF

    CALL ProcessExamplesLabels(trainingExamples) //get labels type (fixed || dynamic)

    CALL GroupMappedPropertiesFromExamples(model, template, trainingExamples);
        //get fixed and dynamic properties

    IF NOT all the mapped properties could be trained THEN
        IF trainingExamples.Count > 2 THEN
            Delete new example from training dataset
        ELSE
            Delete all two examples from training dataset
        ENDIF

        RETURN
    ENDIF
    model.UpdateTemplateModel(template, template.MlModelObject)

END FUNCTION
```

At the end of this part, all the documents from the template are scaled and translated to create the prediction model used in the upcoming part. The transformation needed to scale PDFs are also saved in this model, so they can be applied directly for new documents.

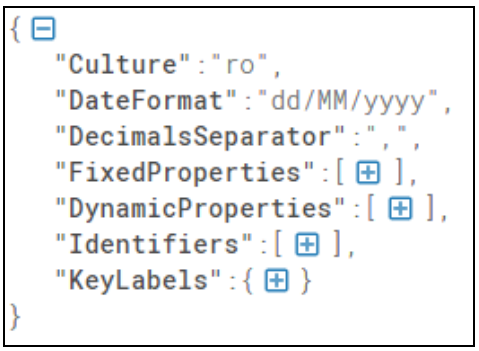


Figure 59: JSON Example of a Trained Model

5.2.4 Prediction Phase

This phase happens when a new document is opened, this is the actual data extraction part. To predict the values from a new document, first the template of it must be found. Every document type has a well-formed structure. In order to identify a document's template a persistent field must be found, which has the same content in every document, this is the identifier of the template and it is present in the trained model. If this field is found the template is found.

The *PredictDocumentTemplateAndProperties* function looks for that field and tries to match with an already existing template, if it cannot find any it can mean that the document is scanned, in which case from every template model the key labels are used to compute the scaling and shifting. If the two labels can be found in the document, in the same way as done in the training phase the text blocks coordinates are transformed. This is applied to the document for each template until it matches with one. If it could not identify the template two situations are present, either a new type of document is uploaded which must be trained first, or we have an existing template, but we must recompute the transformation, which will be done in the training phase. When the template is found, by using its model, the properties are extracted and are converted according to their data type by calling the *ConvertToPropertyType* method. So, dates will be converted from text to date and numbers will be converted from text to numbers. After the extraction and conversion is done the matched fields are returned for the user. Each predicted property containing the original value of the text, the converted value, the id of the property and the bounding coordinates.

The algorithm for matching and predicting fields is presented in the pseudo code bellow.

```
FUNCTION MatchTemplate(modelId, pages, cultureCode)
  cachedModel = CALL DataModelCachingService.GetDocumentModel(docModelId)
  FOR each template IN cachedModel.GetTemplates()
    IF template is trained THEN
      keyLabels = template.MlModelObject.KeyLables
      IF keylabels != NULL THEN
        IF NOT keyLabels.FirstLabel OR keyLabels.SecondLabel exists on the page THEN
          continue to next operation
        ENDIF
        firstLabel = first label from new document
        secondLabel = second label from new document
        hRatio = keyLabels.DistanceH / (secondLabel.Left - firstLabel.Left)
        vRatio = keyLabels.DistanceV / (secondLabel.Top - firstLabel.Top)
        transformations = new DocumentTransformations()
        transformations.HRatio = keyLabels.DistanceH / (secondLabel.Left - firstLabel.Left)
        transformations.VRatio = keyLabels.DistanceV / (secondLabel.Top - firstLabel.Top)
        transformations.LeftShift = firstLabel.Left* hRatio - keyLabels.FirstLabel.Left
        transformations.TopShift = firstLabel.Top* vRatio - keyLabels.FirstLabel.Top

        TransformCoordinates(pages, transformations)

        predictedProperties = FindTemplateProperties(cachedModel, template, pages)

        IF predictedProperties != NULL THEN
          RETURN predictedProperties
        ENDIF
      ENDIF
    ENDIF
  ENDFOR

   RETURN NULL

END FUNCTION
```

The algorithm implemented for the method *FindTemplateProperties* is as follows:

```
FUNCTION FindTemplateProperties(model, template, pages)
  documentProperties = list of predicte dproperties
  identifiers = template.MlModelObject.Identifiers
  FOR each identifier IN identifiers
    predictedProperty = get identifier property from document
    IF predictedProperty == NULL) THEN
      RETURN NULL
    ENDIF
    predictedProperty.PropertyId = identifier.PropertyId
    documentProperties.ADD(predictedProperty)
  ENDFOR
  fixedProperties = template.MlModelObject.FixedProperties
  IF fixedProperties != NULL THEN
    FOR each fixedProperty IN fixedProperties
      predictedProperty = CALL GetPropertyByRectangle(fixedProperty, pages)
      IF predictedProperty == NULL) THEN
        continue to next iteration
      ENDIF
      predictedProperty.PropertyId = fixedProperty.PropertyId
      predictedProperty.Content = CALL ConvertToPropertyType(model.Properties[fixedProperty.PropertyId], template,
predictedProperty.OriginalContent)
      documentProperties.ADD(predictedProperty)
    ENDFOR
  ENDIF

  dynamicProperties = template.MlModelObject.DynamicProperties
  IF dynamicProperties != NULL THEN
    FOR each dynamicProperty IN dynamicProperties
      IF dynamicProperty.Label exists in document THEN
        label = label from document
        IF dynamicProperty.Type == LabelBlockType.Left AND
                ABS(label.ScaledLeft - dynamicProperty.Coordinate) < APPROXIMATION_ERR_FOR_DYNAMIC_LABELS) OR
           (dynamicProperty.Type == LabelBlockType.Top AND
                ABS (label.ScaledTop - dynamicProperty.Coordinate) < APPROXIMATION_ERR_FOR_DYNAMIC_LABELS)) THEN
          rectangle = computed rectangle from the label
          predictedProperty = CALL GetPropertyByRectangle(rectangle, pages);
          IF predictedProperty == NULL) THEN
            continue to next iteration
          ENDIF
          predictedProperty.PropertyId = dynamicProperty.PropertyId
          predictedProperty.Content = CALL ConvertToPropertyType(model.Properties[dynamicProperty.PropertyId],
template, predictedProperty.OriginalContent)
          documentProperties.ADD(predictedProperty)
        ENDIF
      ENDIF
    ENDFOR
  ENDIF

  RETURN documentProperties

END FUNCTION
```

5.3 Web application

DocsMetaExtractor, provides the possibility for users to manage, filter categorize, store, export and label their documents. To understand the features and functionalities implemented going on a walkthrough of the application is presented.

First thing when accessing the application, if the user is not logged in, a login popup appears, if the user is logged in, the home page is opened. There are two types of users, administrators and employees.

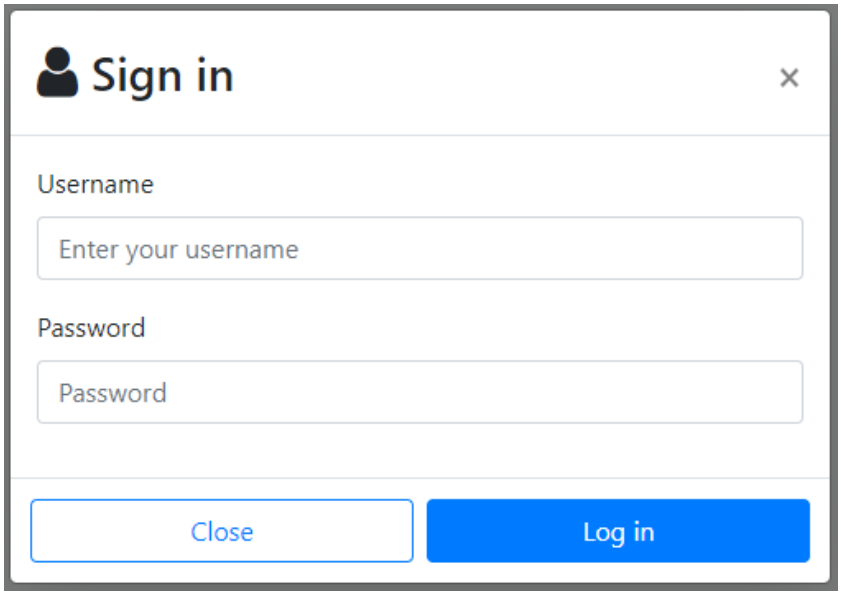


Figure 60: DocsMetaExtractor – Login Popup

5.3.1 Walkthrough for employees

If an employee logs in, the home page will be visible, which contains an overview of the current state of documents. It shows the current number of document models, of templates and of uploaded documents. Functionalities like logging out or changing the current password are available from the navigation bar.

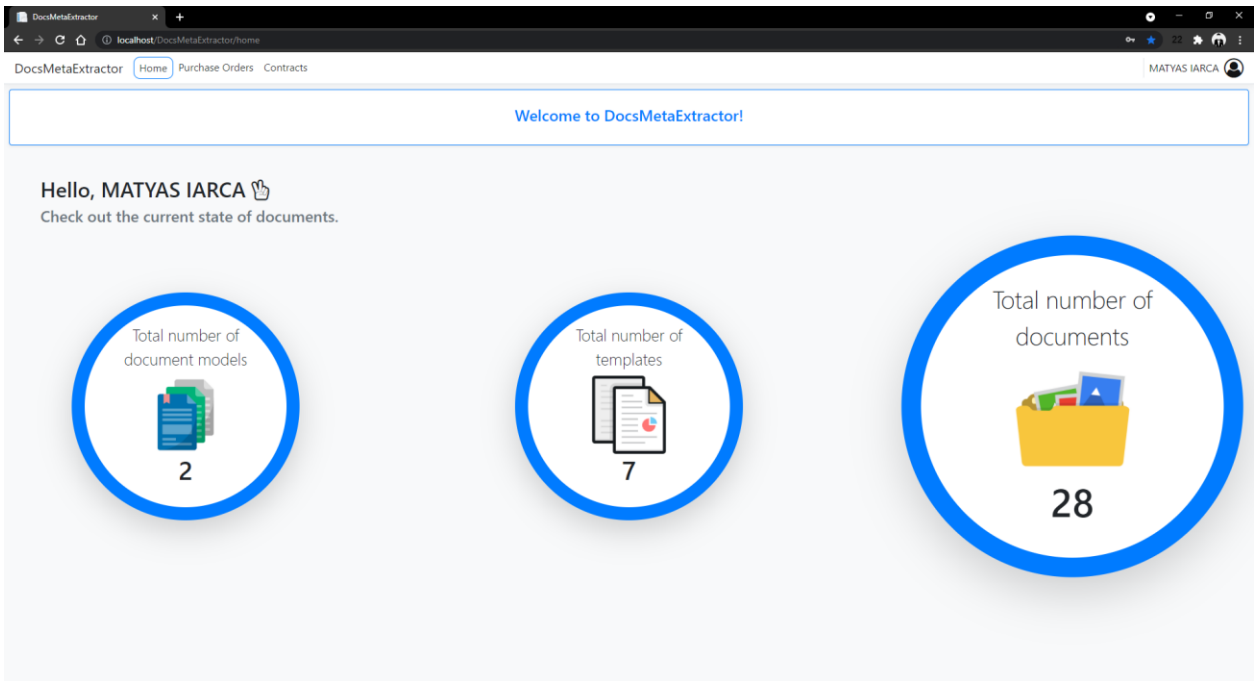


Figure 61: DocsMetaExtractor – Homepage

If the user wants to change his password the following popup will appear:

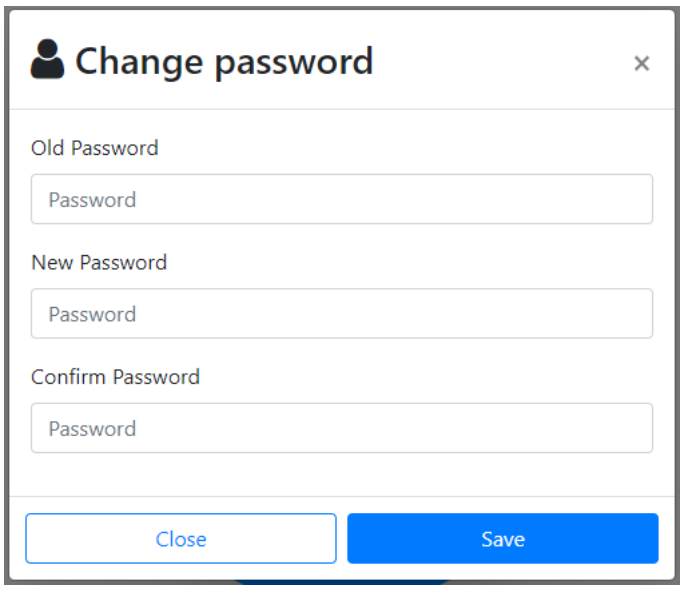
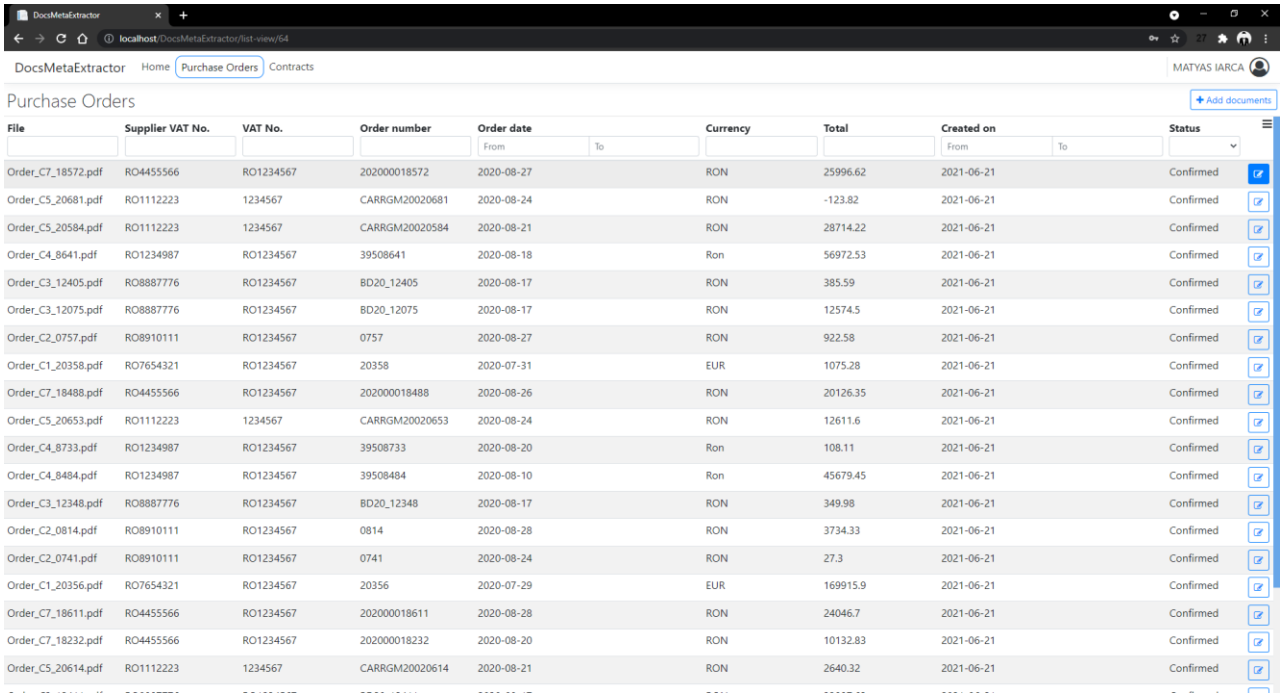


Figure 62: DocsMetaExtractor – Change Password Popup

When the user navigates to a document type from the navbar a centralized list view appears with all the uploaded documents from that type. For each document the name, the uploading date, the status and the properties defined are displayed. From this view also new documents can be added by clicking on the *Add documents* button, multiple documents can be uploaded at the same time to speed up the process. Furthermore, from this list the documents can be opened, with the scope to finalize them by clicking on the pencil icon in the last column.



File	Supplier VAT No.	VAT No.	Order number	Order date	Currency	Total	Created on	Status
Order_C7_18572.pdf	RO4455566	RO1234567	202000018572	2020-08-27	RON	25996.62	2021-06-21	Confirmed
Order_C5_20681.pdf	RO1112223	1234567	CARRGM20020681	2020-08-24	RON	-123.82	2021-06-21	Confirmed
Order_C5_20584.pdf	RO1112223	1234567	CARRGM20020584	2020-08-21	RON	28714.22	2021-06-21	Confirmed
Order_C4_8641.pdf	RO1234987	RO1234567	39508641	2020-08-18	Ron	56972.53	2021-06-21	Confirmed
Order_C3_12405.pdf	RO8887776	RO1234567	BD20_12405	2020-08-17	RON	385.59	2021-06-21	Confirmed
Order_C3_12075.pdf	RO8887776	RO1234567	BD20_12075	2020-08-17	RON	12574.5	2021-06-21	Confirmed
Order_C2_0757.pdf	RO8910111	RO1234567	0757	2020-08-27	RON	922.58	2021-06-21	Confirmed
Order_C1_20358.pdf	RO7654321	RO1234567	20358	2020-07-31	EUR	1075.28	2021-06-21	Confirmed
Order_C7_18488.pdf	RO4455566	RO1234567	202000018488	2020-08-26	RON	20126.35	2021-06-21	Confirmed
Order_C5_20653.pdf	RO1112223	1234567	CARRGM20020653	2020-08-24	RON	12611.6	2021-06-21	Confirmed
Order_C4_8733.pdf	RO1234987	RO1234567	39508733	2020-08-20	Ron	108.11	2021-06-21	Confirmed
Order_C4_8484.pdf	RO1234987	RO1234567	39508484	2020-08-10	Ron	45679.45	2021-06-21	Confirmed
Order_C3_12348.pdf	RO8887776	RO1234567	BD20_12348	2020-08-17	RON	349.98	2021-06-21	Confirmed
Order_C2_0814.pdf	RO8910111	RO1234567	0814	2020-08-28	RON	3734.33	2021-06-21	Confirmed
Order_C2_0741.pdf	RO8910111	RO1234567	0741	2020-08-24	RON	27.3	2021-06-21	Confirmed
Order_C1_20356.pdf	RO7654321	RO1234567	20356	2020-07-29	EUR	169915.9	2021-06-21	Confirmed
Order_C7_18611.pdf	RO4455566	RO1234567	202000018611	2020-08-28	RON	24046.7	2021-06-21	Confirmed
Order_C7_18232.pdf	RO4455566	RO1234567	202000018232	2020-08-20	RON	10132.83	2021-06-21	Confirmed
Order_C5_20614.pdf	RO1112223	1234567	CARRGM20020614	2020-08-21	RON	2640.32	2021-06-21	Confirmed

Figure 63: DocsMetaExtractor – Documents List View

Each column can be filtered according to the type of the values it contains. The status can be selected from a combo box, the dates have a custom filter with date pickers to select an interval, the simple text values are filtered based on whether it contains the text from the filter or not, it is also case insensitive. Numbers can be filtered exactly or by using the symbols: '=', '<=', '>=', '<' or '>', as usually no one is looking for an exact amount. If invalid syntax is used for filtering the system will notify the user.

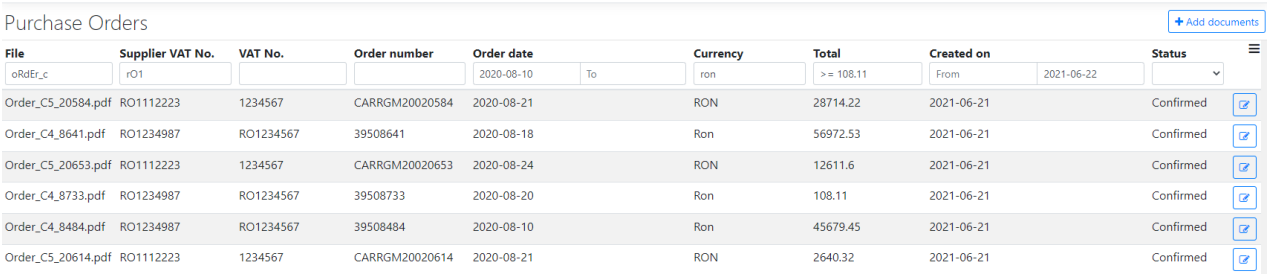


Figure 64: DocsMetaExtractor – Documents List View Filtering Example

The user at any time can refresh the list and can export the filtered data, by clicking on the three bars in the top right corner of the table. There are two formats to export the data, as an excel or as a CSV file, both including a header. The file name is constructed by the name of the document model and the current date.

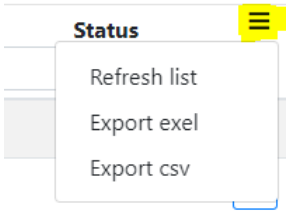


Figure 65: DocsMetaExtractor – Refresh & Export List

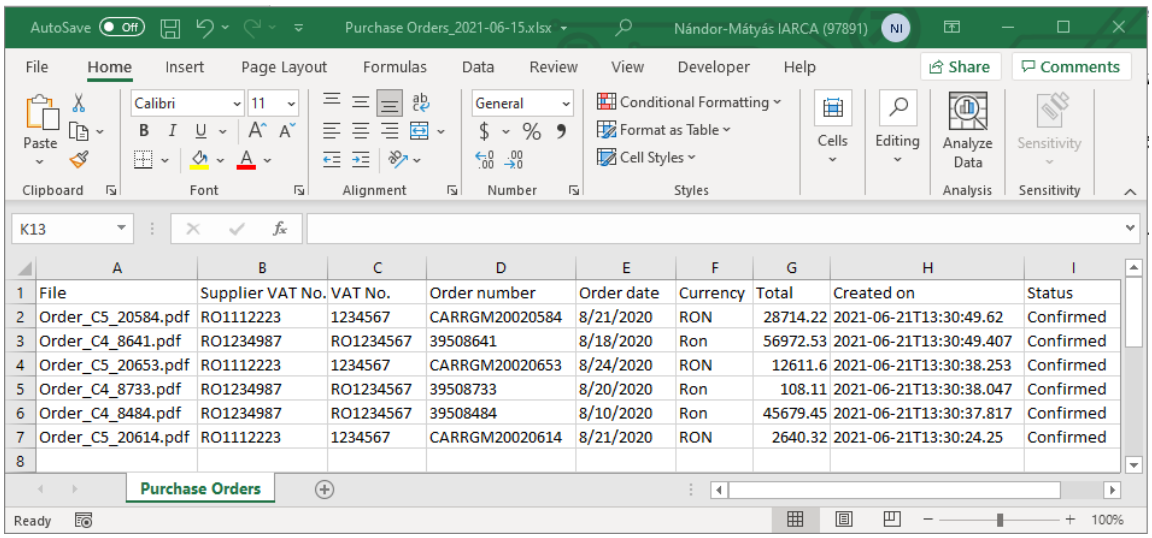


Figure 66: DocsMetaExtractor – Excel Export

There are three statuses, processing, processed and confirmed. A document is in processing if it is a scanned PDF and needs OCR. When the OCR service finished the document will have the status processed, real PDFs by default are processed. Users cannot open unprocessed documents, the system will provide the following notification if trying:

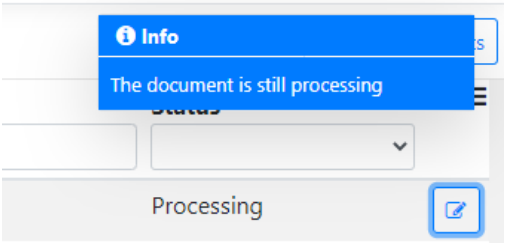


Figure 67: DocsMetaExtractor – Processing Document Notification

When a PDF is processed it can be opened by the user. Here the user can see the prediction results and can map the properties if something is missing or incorrectly extracted. For this a document preview is available, where the user can navigate through the pages of the documents and over each page a layer with selectable text blocks is present. More text blocks can be selected for the same property. Moreover, date and number properties are immediately converted. The first two examples for each template must be fully mapped in order to train the model.

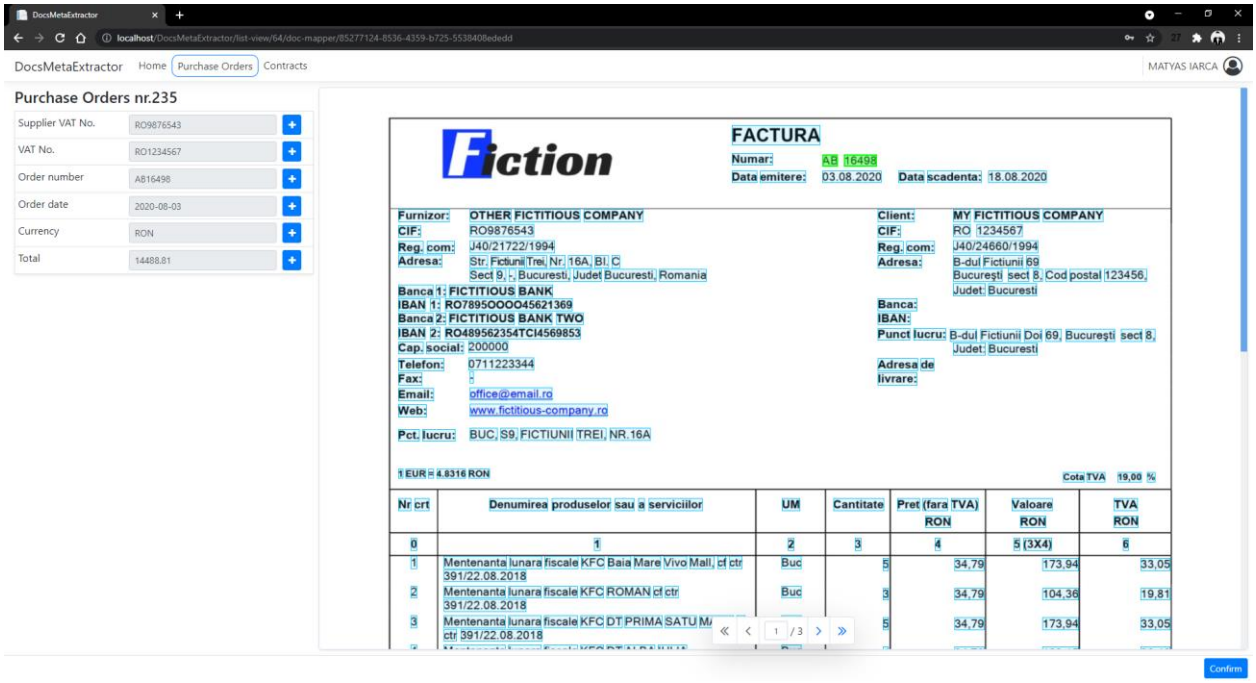


Figure 68: DocsMetaExtractor –Document Properties Mapping

Finally, when the document is confirmed, it can be only viewed, only the pdf preview is available, without the possibility to map properties.

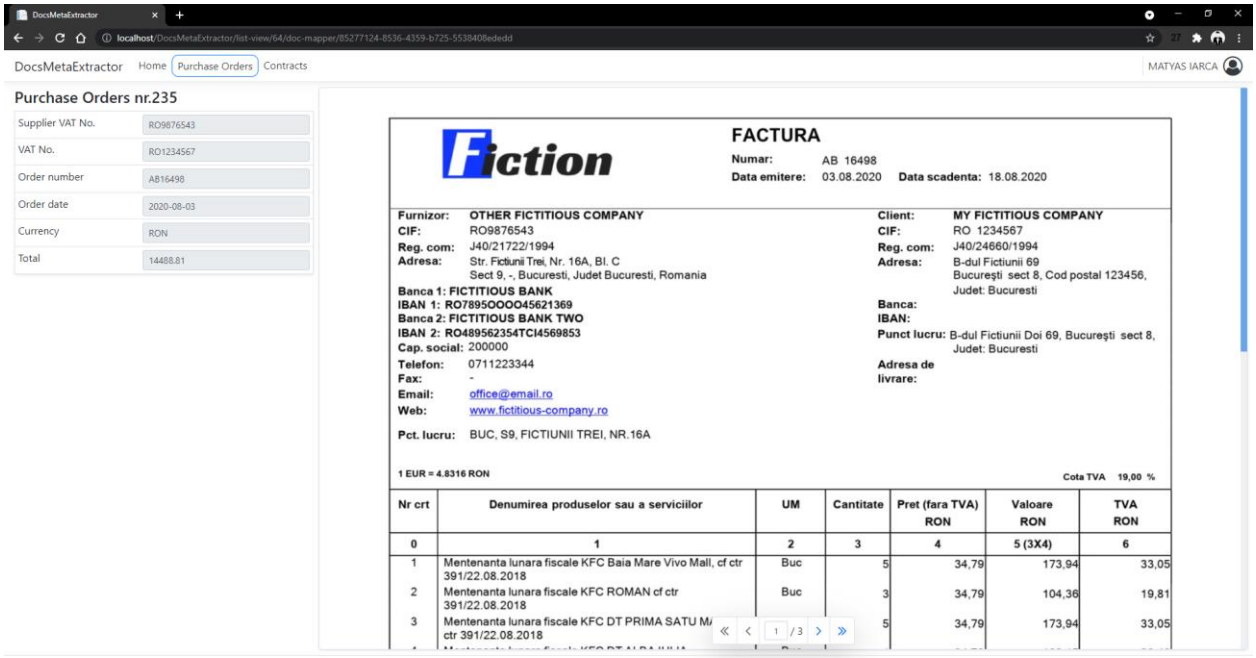


Figure 69: DocsMetaExtractor –Confirmed Document View

5.3.2 Walkthrough for administrators

Administrators can do anything a simple employee can, but they have an extra administration panel available only for them. Here they have three different settings views, the document model, the template and the users administration.

In document model administration the existing document models can be edited, deleted or refreshed, new models can be added, and the order of menu items can be changed by a simple drag and drop functionality.

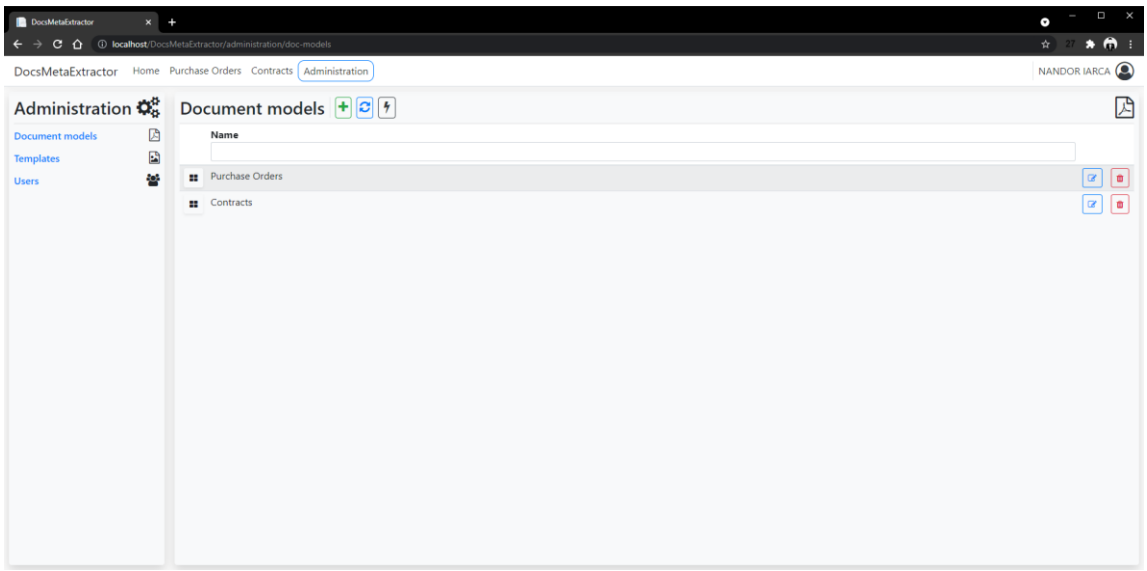


Figure 70: DocsMetaExtractor –Document Model Administration

When adding a new model, a new menu item will be present after refreshing the site data. A table is automatically generated. Each model must have properties and it is mandatory to have an identifier field, like presented in *Figure 71* bellow.

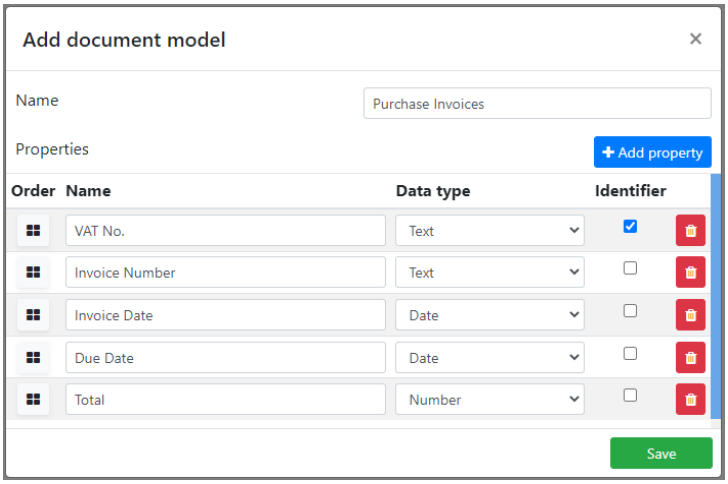


Figure 71: DocsMetaExtractor –Adding New Document Model

The new model at start will be completely empty. Employees will be able to upload documents that correspond to that model.

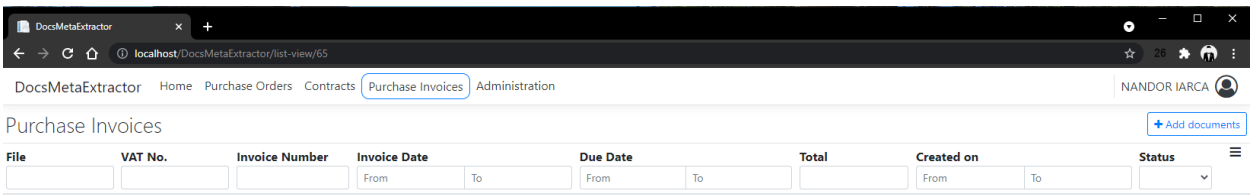


Figure 72: DocsMetaExtractor –New Document Model List

In template administration the currently trained templates are enlisted and can be filtered or deleted. This has the purpose to allow to delete a template if it changes during a refactoring and the old version is erased. This might help when overlapping, duplicate templates are trained, however this might never occur as the training algorithm checks and deletes the wrong examples accordingly. The only time this can happen if documents are inserted directly in the training set database without going through the system.

In the user administration view, new users can be added and existing users can be deleted or edited. Only administrators can add new users. They allocate to them a default password that will be changed by the user when logging in, moreover administrators can change at any time the password of a user.

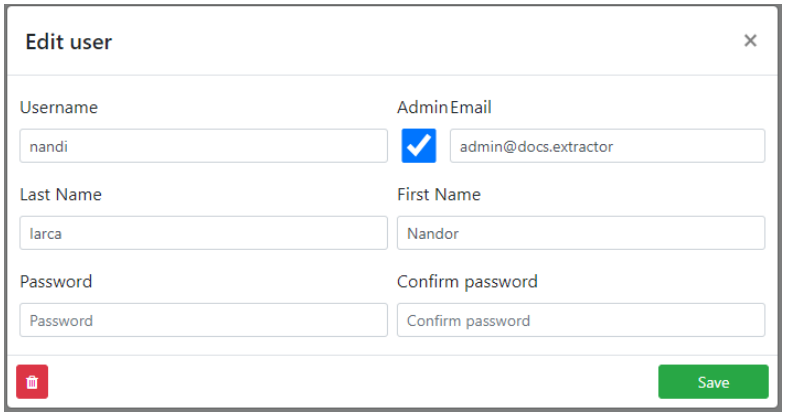


Figure 73: DocsMetaExtractor – Save/Edit User Popup

In addition, the system also can extract handwritten text, but only in English.

Figure 74: Handwritten English Text Recognition

6 Validation

In this part we will focus on the validation of the plug-in, as it is building the core of the whole system. The plug-in is integrated into solutions provided by Arggo Software Development & Consulting (https://arggo.consulting/), a Romanian company established in 2014. It provides several kinds of customized and standard business solutions for companies.

The plug-in is used in a DMS project released on January 1, 2021. This solution provides an automated invoice processing flow for businesses. Currently the application has 938 active users. From the releasing date 38,456 documents were processed in total, both real and scanned PDFs were uploaded. All of these only for one document model as the system is designed only for invoices. An average number of approximately 326 documents are uploaded and predicted on each working day. There are 581 templates currently saved and 2168 training examples for them This means an average of 4 examples per template, but of course there are trained templates with only 2 examples and there are some with more. It can be observed that from the total number of documents only 2168 needed training. This means that for 36288 invoices the prediction worked correctly. This means that the algorithm predicted 94% correct documents.

The overall feedback from clients was satisfying, they liked the idea to automatize their invoice processing, as it can be seen 326 of documents cannot be processed manually a day, only if a big team is available to do the job. In the first weeks when the application was released constant support was needed in order to maintain the flow in production. In this interval several features where modified and added as the volume of data became larger and larger. This is the reason why the caching service was added as well. In this way the database is not overloaded and users can get fast responses. After this period of time, mainly everything that needed fixing was fixed and the project is up and running.

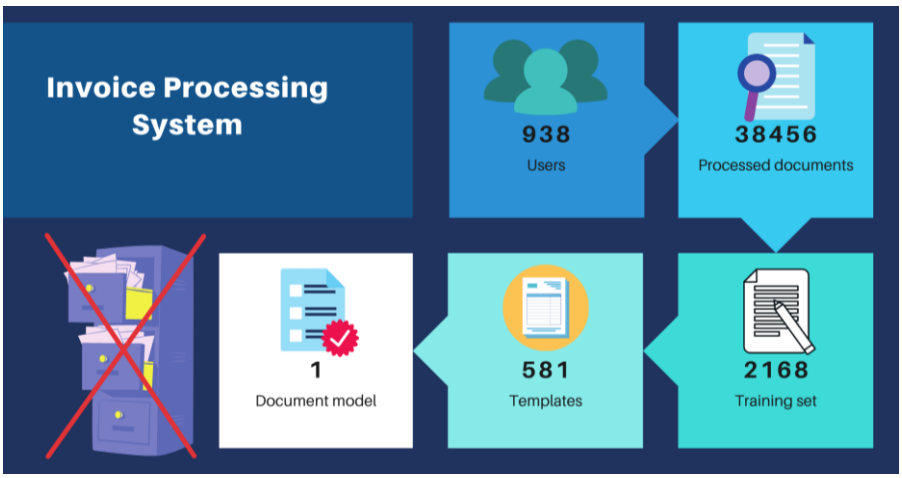


Figure 75: Arggo Consulting's Invoice Processing System Statistics

Conclusions

To sum up, after analyzing the current solution for DMS and automatic data extraction, it can be concluded that DocsMetaExtractor provides something new. It is well suited for business solutions as it does not have the purpose of research, its goal is to make as easy and as fast as possible document managing and data extraction. It is dynamic, changes can be made at any time by the user during runtime, which provides customizability at any point.

Analyzing the users requirements for such a system leaded to the following important aspects:

- provide a system that interacts easily with the user
- complex flows might confuse users, keep everything as simple and clear as possible
- design self-intuitive UI as much as possible
- reduce the processing/waiting time
- make the system scalable as a huge amount of data is uploaded
- do not loose performance when scaling

I implemented the web application by keeping these aspects in my mind in order to get to a real-life useable system.

The plug-in is in its first release, it will need newer versions to expand it for other domains as well, like barcode reading, table extraction or handwritten text. The currently set goals were blissfully achieved after the critical period of the start and by analyzing the feedback of the users new goals will be set to constantly improve and expand it. Currently it is in production with only one document model in the picture, but as it can be observed in the web application it is capable of more. It is not just focused on a single type of document, which is a great advantage for DMSs.

Finally, it can be said that there is a need in this modernizing world for automatization. As technologies evolve and are available for more and more people everything is possible. Business future is automatizing their activities as much as possible, to let humans focus their skills on jobs that only they are qualified to do, these having a higher impact on the overall productivity.

References

[1] Li Liu, Zhiyu Wang, Taorong Qiu, Qiu Chen, Yue Lu, Ching Y. Suen. (2021). Document image classification: Progress over two decades. *Neurocomputing*, 453, 223-240. https://www.sciencedirect.com/science/article/abs/pii/S0925231221006925?via%3Dihub

[2] DocProcess. (2019, August 30). *Cum a reușit un mare retailer din România să reducă rata de eroare a facturilor sub 0.1%*. Softlead. https://softlead.ro/noutati-it-c/cum-a-reusit-un-mare-retailer-din-romania-sa-reduca-rata-de-eroare-a-facturilor-sub-0-1.html

[3] SenseTask. (n.d). *Real benefits for your business*. Retrieved June 16, 2021, from https://sensetask.com/benefits

[4] DocProcess. (2021, May 17). *Mandatory E-invoicing in Saudi Arabia – Are you ready?*. https://doc-process.com/resources/e-invoicing-in-saudi-arabia

[5] DocProcess. (2021, June 10). *Achieve 100% touchless processing with end-to-end e-invoicing*. https://doc-process.com/products_services/dxinvoice

[6] Nanonets. (n.d). *Capture Data from invoices*. Retrieved June 16, 2021, from https://nanonets.com/invoice-ocr

[7] La Laujan, Rohit Mungi, JP Park, Michael Bullwinkle. (May 11, 2021). *Train a custom model using the sample labeling tool*. Microsoft Docs. https://docs.microsoft.com/en-us/azure/cognitive-services/form-recognizer/label-tool?tabs=v2-1

[8] ReadySoft. (n.d). *Invoice Processing Overview*. Retrieved June 16, 2021, from https://readysoft.ro/invoice-processing-overview?lang=en

[9] Peketi Divya, Mahesh Varma, Uma Ratna Mouli, Srinivas, Garima, Nikhil, Vishistha. (2020). Web based optical character recognition application using flask and tesseract. *Materials Proceedings*,https://www.sciencedirect.com/science/article/pii/S221478532038487X?via%3Dihub

[10] Francesco Adamo, Filippo Attivissimo, Attilio Di Nisio, Maurizio Spadavecchia. (2015). An automatic document processing system for medical data extraction. *Measurement*, 61, 88-99. https://www.sciencedirect.com/science/article/abs/pii/S0263224114005016?via%3Dihub

[11] Berke Oral, Erdem Emekligil, Seçil Arslan, Gülşen Eryiğit. (2020). Information Extraction from Text Intensive and Visually Rich Banking Documents. *Information Processing and Management*, 57(6), 102361. https://www.sciencedirect.com/science/article/abs/pii/S0925231221006925?via%3Dihub

[12] Heath Goodrum, Kirk Roberts, Elmer V. Bernstam. (2020). Automatic classification of scanned electronic health record documents. International Journal of Medical Informatics, 144, 104302.https://www.sciencedirect.com/science/article/abs/pii/S1386505620309977?via%3Dihub

[13] Cem Dilmegani. (2021, January 1). *Data Extraction: What it is, Why it matters & Key Features*. AIMultiple. https://research.aimultiple.com/data-extraction/

[14] SenseTask. (n.d). *How it works*. Retrieved June 16, 2021, from https://sensetask.com/howitworks

[15] Nanonets. (n.d). *Automate manual data entry using AI*. Retrieved June 16, 2021, from https://nanonets.com/?&utm_source=nanonets.com%2Fblog%2F&utm_medium=blog&utm_content=Table%20Detection,%20Information%20Extraction%20and%20Structuring%20using%20Deep%20Learning

[16] Nanonets. (n.d). *Invoice Automation for AP Teams*. Retrieved June 16, 2021, from https://nanonets.com/invoice-automation

[17] Nanonets. (n.d). *Digitize documents, receipts and PDFs using OCR & Deep Learning*. Retrieved June 16, 2021, from https://nanonets.com/invoice-automation

[18] Docparser. (n.d). *What is Data Extraction? (+3 use-cases & examples)*. Retrieved June 16, 2021, from https://docparser.com/blog/what-is-data-extraction/

[19] Mark Smallcombe. (2020, June 25). *Structured vs Unstructured Data: 5 Key Differences*. Xplenty. https://www.xplenty.com/blog/structured-vs-unstructured-data-key-differences/

[20] Lawtomatoed. (2019, April 7). *Structured Data vs. Unstructured Data: what are they and why care?*. https://lawtomated.com/structured-data-vs-unstructured-data-what-are-they-and-why-care/

[21] Angular. (n.d). *Angular Docs*. Retrieved June 16, 2021, from https://angular.io/docs

[22] Microsoft. (n.d). *Azure Form Recognizer documentation*. Microsoft Docs. Retrieved June 16, 2021, from https://docs.microsoft.com/en-us/azure/cognitive-services/form-recognizer/

[23] Microsoft. (n.d). *Computer Vision documentation*. Microsoft Docs. Retrieved June 16, 2021, from https://docs.microsoft.com/en-us/azure/cognitive-services/computer-vision/

[24] Microsoft. (n.d). *Text Analytics API documentation*. Microsoft Docs. Retrieved June 16, 2021, from https://docs.microsoft.com/en-us/azure/cognitive-services/text-analytics/

[25] Microsoft. (n.d). *Azure Cognitive Services documentation*. Microsoft Docs. Retrieved June 16, 2021, from https://docs.microsoft.com/en-us/azure/cognitive-services/

[26] Microsoft. (n.d). *ASP.NET documentation*. Microsoft Docs. Retrieved June 16, 2021, from https://docs.microsoft.com/en-us/aspnet/core/?view=aspnetcore-5.0

[27] Microsoft. (n.d). *.NET documentation*. Microsoft Docs. Retrieved June 16, 2021, from https://docs.microsoft.com/en-us/dotnet/fundamentals/

[28] Microsoft. (2021, February 2). *Architect Modern Web Applications with ASP.NET Core and Azure*. Microsoft Docs. https://docs.microsoft.com/en-us/dotnet/fundamentals/

[29] TypeScript. (n.d). *TypeScript Documentation*. Retrieved June 23, 2021, from https://www.typescriptlang.org/docs/

[30] Bootstrap. (n.d). *Documentation*. Retrieved June 23, 2021, from https://getbootstrap.com/docs/4.0/getting-started/introduction/

[31] Visual Paradigm. (n.d). *Documentations*. Retrieved June 16, 2021, from. https://www.visual-paradigm.com/support/documents/

[32] IText. (n.d). *API documentation*. Retrieved June 16, 2021, from https://itextpdf.com/en/resources/api-documentation

[33] EVO Pdf, (n.d) *User Guide*. Retrieved June 16, 2021, from https://www.evopdf.com/help/html_to_pdf/index.aspx

Glossary

DMS	Document Management System
PDF	Portable Document Format
OCR	Optical Character Recognition
AI	Artificial Intelligence
ML	Machine Learning
REST	Representational State transfer
API	Application Programming Interface
ERP	Enterprise Resource Planning
XML	Extensible Markup Language
FTE	Full-Time Equivalent
BEA	Business Ecosystem Automation
CRM	Customer Relationship Management
RPA	Robotic Process Automation
CSV	Comma-Separated Values
JSON	JavaScript Object Notation
SDK	Software Development Kit
JPG	Joint Photographic Group
BMP	Bitmap
PNG	Portable Network Graphics
TIFF	Tag Image File Format
MB	Megabyte
HTTPS	Hypertext Transfer Protocol Secure

URL	Uniform Resource Locator
EDI	Electronic Data Interchange
DIP	Document Image Processing
BOVW	Bag of Visual Words
AP	Accounts Payable
UC	Use Case
MTBF	Mean Time Between Failures
IIS	Internet Information Services
JWT	JSON Web Token
ERD	Entity Relationship Diagram
HTML	Hyper Text Markup Language
CSS	Cascading Style Sheets
UI	User Interface
EF	Entity Framework
TPS	Transactions Per Second
NLP	Natural Language Processing
SQL	Structured Query Language