POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

# The Design of a Data Lake architecture for the healthcare use case: problems and solutions

Laurea Magistrale in Computer Science and Engineering - Ingegneria Informatica

**Author:** Sara Parente

**Advisor:** Prof. Letizia Tanca

**Co-advisor:** Prof. Marco Gribaudo

**Academic year:** 2020-2021

## 1. Introduction

In the last years, due to the ever increasing number of devices connected to the Internet and to the continuous technological growth, the generation of data of any type has also increased. This growth has given rise to the phenomenon of Big Data. The new characteristics associated to this data have made the need of reorganizing business processes and the need of developing new solutions increasingly evident. Above all, it has emerged the need of new systems that allow to exploit new opportunities that came with Big Data and that allow, also, to handle the constantly increasing complexity of data management.
In the healthcare field the term "Big Data" is used to describe all information generated from the digital technologies that collect patients' records. In the near future, physicians will have to deal with terabytes of data. Gaining clinical insight from such a large volume of data is an impossible expectation to place upon humans. An answer to this problem is to make Electronic Health Record not only records of patient-physician interactions and digital medical records, but also diagnostic aids. Another important aspect is to make this huge volume of data available and easily accessible by physicians in order to make the diagnostic process much faster.

## 2. Aim of the thesis

The aim of the thesis is to design a system that can efficiently handle healthcare Big Data. The project involves the adoption of a Data Lake that connects many Istituti di Ricovero e Cura a Carattere Scientifico [1] (IRCCS) in order to have a central repository for all connected data and, therefore, create a rich dataset.
The Data Lake should be the only entry point for all types of data, which are stored in their raw format. More in detail, the objective is to design a system that integrates traditional data and multimedia data. This can be achieved only by developing an architecture that can handle all types of data. Furthermore, the proposed system should be scalable and it should allow to quickly access every type of data. This can be achieved by extracting structural, semantic and process metadata that can make retrieving of data easier and can help the query proccess.
Since in the healthcare domain data is very het-

[1] Institute for Treatment and Research

erogeneous, as it spans from lab reports to genomic data, we have decided to restric our analysis to medical images, e.g, x-rays, CTs and PETs. So, the project proposes a Data Lake architecture that can handle this type of data and its metadata.

## 3. Overview of the proposed architecture

The literature [7] offers multiple examples of data lake architectures. These models, however, are too generic and do not take into considerations important aspects, such as data modeling and metadata menagement. The objective of the thesis is to define a comprehensive data lake architecture that best fits the studied use case.
The proposed architecture aims at handling medical Big Data coming from heterogeneous sources in different formats, including multimedia and, particularly, medical images. A high level component view of the system is presented in Fig. 1.
From Fig.1, it can be observed that we have different data sources external to the Data Lake. Data is ingested from these sources by the Data Ingestion component and is permenantely stored in the Raw Data Storage. The Ingestion module is also responsible for the generation of the metadata on the ingested dataset. Then, based on the type, data goes under different kinds of procesing. In this phase other metadata can be generated, e.g. metadata on the processes. Data obtained from the processing steps is then stored in a Database. The Query Engine, instead, is the component that executes queries against data in that database to provide answers to the front-end application. Lastly, the Metadata Management System handles and stores the metadata generated by the other components.

Data Lake architectures are usually classified in Data Pond Architectures and Zone Architectures [6]. Zone architectures assign data to a zone according to their degree of refinement. Pond architectures, instead, represent the Data Lake as a set of data ponds and each data pond can be seen as a part of the Data Lake which handles a specific type of data. For this specific use case a Zone Architecture has been chosen because data should go through different levels of refinement and, also, there is the need to keep a copy of data in its raw format. More in detail, the proposed architecture is composed by four different zones: Transient Landing Zone, Raw Zone, Stage Zone, and Sandbox. It is important to highlight that metadata management spans all the zones as metadata can be produced at any stage.
In particular, the **Transient Landing Zone** is represented by the Data Ingestion module and it is the zone where data lands when extracted from the data sources and pushed to the Data Lake. In the analyzed use case images are collected in micro-batches and pushed into the Transient Landing Zone. The **Raw Zone** comprises the Raw Data Storage and it is the zone where data is stored indefinetely in its native raw format. This zone covers a big role in the healthcare environment because it is likely that images in raw format are used multiple times to make different kinds of analysis. The **Stage Zone** can be mapped to the two modules, Image Analysis and Other Processing and it is the zone where data is landed for preparation and processing. In our case, this is the layer where features are extracted from the images. Lastly, the **Sandbox Zone** corresponds to the Database component. It is the zone where data scientists and researchers can build analytical models, discover associations and patterns within the data.

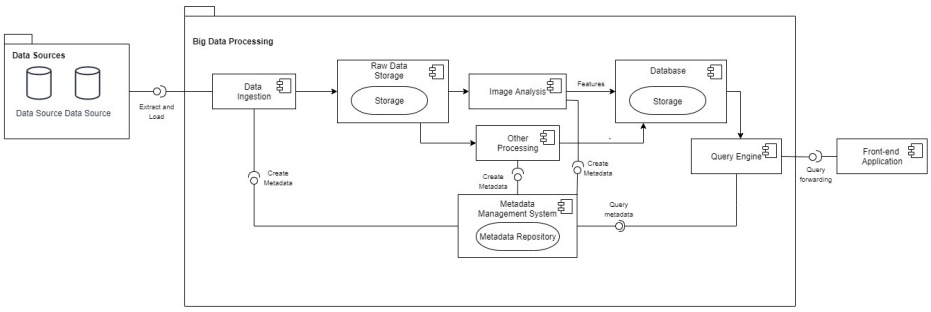


Figure 1: Component diagram of the Data Lake architecture.

### 3.1. Metadata Management System

As already mentioned, metadata management is a crucial aspect of the Data Lake architecture as it allows full exploitation of data's value. Since metadata is still data, it needs to be managed in a right way that allows to obtain all the possible value from it, an important issue is the model used to represent this metadata.
The literature proposes different metadata models [5] [4] [3] [8]. These models, however, do not suit the healthcare use case. First of all, they mostly focus on structured and semi-structured data. Moreover, none of them allows to describe to which zone the data belong, which is an important aspect to understand the level of refinement. Lastly, these models do not take into consideration the level of granularity. Therefore, we conducted further researches and found a model that can fit the use case. Handling metAdata maNagement in Data LakEs (HANDLE) [1] has been developed with the intention of creating a model that can handle all use cases. This model can be divided into two main parts, the core model and three extensions. The scope of the core model is to define all elements and the relations that are needed to model metadata. The extensions, instead, represent the zones, the levels of granularity and the categoriazion topics.
Fig.2 describes a slightly modified model of HANDLE using an Entity Relationship schema. The modification that has been applied is the removal of the categorization property because it describes the context of metadata, e.g. operational, technical, business, and it does not apply to the examined application.

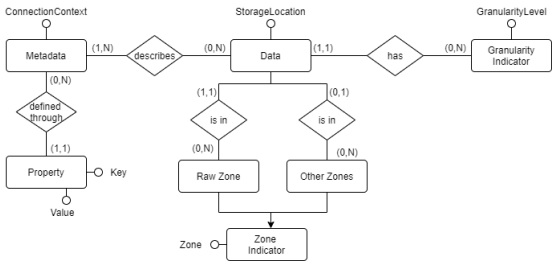


Figure 2: HANDLE Entity Relationship model

The schema in Fig.2 shows that, to each piece of data zero or more metadata can be attached. Each one of these metadata is described by a set of properties. The ConnectionContext attribute, instead, provides a description of the information contained in the metadata element. Data is also described through a granularity indicator and a zone indicator. It is important to underline that each data has a link to the Raw Zone also when it belongs to other zones. For what concerns the attributes represented in the schema, the StorageLocation gives the path to the data element, and each property is defined through a key-value pair. The Granularity Indicator entity allows to collect metadata at different levels of granularity, e.g. single image, collection of images, part of an image. The ZoneIndicator entity provides the information about the location of a specific data element inside the Data Lake.
In [1] also a possible implementation for this model is proposed. Since flexibility is a key aspect in metadata management, the authors have opted for a NoSQL technology. In particular, the chosen model is a graph database.

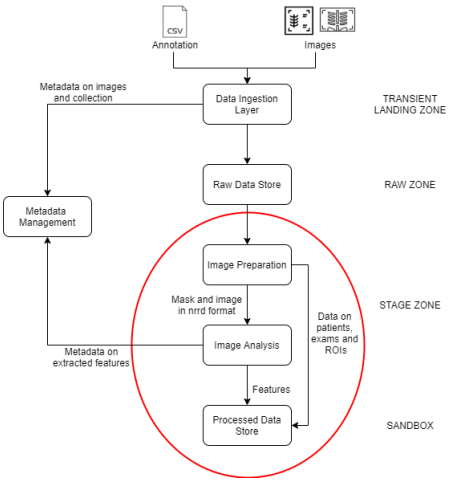


Figure 3: Example of data flow

## 4. Data Flow

The Data flow describes the path that system's information take from external sources through processes and data stores inside the Data Lake. The Data Lake ingests data, irrespective of its format, into a Big Data store. Metadata is decoupled from its data and stored independently. Then, data goes through a series of steps that can have either the function of processing or that of storage.
The schema in Fig.3 represents the data flow for medical images. The input is a set of images in Digital Imaging and Communication in Medicine (DICOM) format, which is the refer-

ence standard for what concerns medical images, along with related set of annotations in Comma Separated Values (CSV) format. The annotations define the Region Of Interest (ROI), i.e. the part of the image that contains important diagnostic information, on the images and have previoulsy been manually defined by radiologists. The ingested data lands in the transient landing zone and two important actions are performed, the masking of Personally Identifiable Information (PII) and the generation of metadata on both single images and image collection. Then, data is permanently stored in the Raw Data Store. After ingestion and storage there are all the steps that process the image and the related data. We have developed a simple implementations of these stages to better understand which are the inputs and the outputs of these phases and, also, to make an assessment of resource requirements.

### 4.1. Data set

To perform this simple analysis 80 images were taken from the ChestX-ray8 database [9]. The images contained in this data set are chest x-rays. The ChestX-ray8 database is composed by 108 948 images which are frontal-view x-ray. These images were taken from a totality of 32 717 patients. Typically the dimension of an x-ray image are 3000x2000, but these dimensions are a challenge for hardware computing capacity. Therefore, in ChestX-ray8 images are extracted from the DICOM file and resized as 1024x1024 bitmap images. This process does not result in losing of detail contents because intensity ranges are rescaled using window settings stored in the DICOM header file.
As part of the ChestX-ray8 database, a small number of images with pathology are provided with hand labeled bounding boxes. In the labeling process only 983 images were analyzed. For each one of these images a board-certified radiologist identified the region interested by a disease. The identified ROI has then be registered in an eXtensible Markup Language (XML) file. If an image has more than one ROI, each one of them is stored in a different XML file. As previously mentioned these ROIs are provided in a CSV file.
In the thesis only images that have at least one ROI were used.

### 4.2. Image preparation

The first step of this phase is the extraction of information about the patient, the case and the exam from the DICOM files. This data is stored in a relational table using MySQL.
The second step is the extraction of data concerning the ROI from the CSV file and its storage in a relational table. In our case the ROI identifies the region from which it is possible to identify a chest disease, e.g. pneuomonia, pneumothorax, atelectasis.
The third step of this phase is the creation of the mask. A mask is a binary image formed by zero and non-zero values. When the mask is applied to a grayscale image, the pixels that have a zero value in the mask are set to zero also in the output image. The other pixels, instead, are unchanged. So, the mask allows to consider only the ROI of the image. We have implemented this task using MATLAB.
The fourth, and last, step is image transformation using SimpleITK. In this step both the DICOM image and the mask, which is in Joint Photographic Experts Group (JPEG) format, are transformed into Nearly Raw Raster Data (NRRD) format. This transformation is needed as the following step does not accept neither DICOM nor JPEG formats.

### 4.3. Image analysis

At this stage the features are extracted from images. This analysis is performed using Pyradiomics. It takes as input the image and the mask both in NRRD format and returnes an ordered dictionary as output. In addition to the features, also metadata is generated in this step, for example the version of Pyradiomics, the original image spacing etc.

### 4.4. Processed data store

This stage provides the storage for both data about patients, exams, ROIs and the features extracted from the images. These information are organized following a relational model. We have choosen this type of model as data has a rigid structure and NULL values are infrequent.

### 4.5. Performance monitoring

As already mentioned, the implementation of the steps of image preparation and features extraction has allowed us to quantify time of exe-

cution, resource usage and the amount of data generated. This represents a big advantage as these numbers helped us understand the requirements that the architecture should satisfy. The tests have been executed using the 80 chest x-rays taken from the ChestX-ray8 database and their ROIs.
The MATLAB function that reads the ROI from the CSV file, opens the image in DICOM format and creates the mask, has been executed 10 times for all the 80 images and the execution time has been recorded. This step takes on average 3,766s, which means that the time needed for the creation of a single mask is on average 47,07ms. For what concerns CPU utilization, instead, the average for this step is 43,3%. Another important aspect is that this code does not take advantage of the GPU.
In the following step, image analysis, features are extracted from medical images. Taking always in consideration the 80 chest x-rays, a total of 10 320 features are extracted which means that for each image 129 features are computed. For what concerns the execution time we have taken advantage of the function offered by Python, *timeit*. This function registers the execution time of a small code snippet and it takes as inputs the code snippet we want to time and the number of times we want to execute it. We have called the function three times passing as *number* parameter: 1, 10 and 100. From execution times, we have estimated that it takes, on average, 202,67ms to extract features from a single image. CPU utilization in this case is 14,4% and it uses also the GPU with a percentage of 4,4%. During the execution of Pyradiomics we were also able to measure memory utilization, more in detail we have executed the code for the 80 images 10 times and register for each execution the peak memory usage. On average the peak is 14,303 MB.
Both the step were run on a PC with 2,90 GHz dual processors CPU.

## 5. Conclusion

The work presented in this thesis is at an early stage of the process of the implementation of a storage system for medical data. The main objective of this work is to design an architecture for a Data Lake that allows efficient storing and fast access to all types of data. In particular, the main focus was on storage of unstructured data, i.e. medical images, which is problematic in traditional systems.
Taking into account the performed work, the main next step is the complete implementation of the system. The most interesting part of the implementation will be certainly the metadata management system. For this part of the system more detailed requirements should be identified by analyzing the different types of data entering the Data Lake and their relationships.
Another important step that sould be made is the analysis of the aspects of data security and data quality. Data security is of great importance in the Data Lake, as it ensures legal conformance, alignment with business objectives, and much more [2]. Data quality, instead, is important to ensure the data's usability and prevent the data lake from turning into a data swamp [2]. When these two aspects are included, also their metadata should be analyzed, e.g. security and quality classification.
Another important step that should be done is to examine how the process of masking PII can be implemented. This process has to be deeply analyzed as the exposure of sensitive data can cause serious damage. It has been selected the masking technique as it can shuffle data columns in different ways so that the masked data looks like the original, in the format and type, but it is no longer sensitive data. Masking is effective because despite it changes all the individual data elements, it still allows to compute aggregate values across an entire database, enabling preservation of the right values within a data set.

## References

[1] Rebecca Eichler, Corinna Giebler, Christoph Gröger, Holger Schwarz, and Bernhard Mitschang. Handle - a generic metadata model for data lakes. In Min Song, Il-Yeol Song, Gabriele Kotsis, A. Min Tjoa, and Ismail Khalil, editors, *Big Data Analytics and Knowledge Discovery*, pages 73–88, Cham, 2020. Springer International Publishing.

[2] Corinna Giebler, Holger Schwarz, Bernhard Mitschang, and Technologie und Web. The data lake architecture framework: A foundation for building a comprehensive data lake architecture. *Datenbanksysteme für Busi-*

*ness, Technologie und Web (BTW 2021) 13.–17. September 2021 in Dresden, Deutschland*, page 351, 2021.

[3] Rihan Hai, Sandra Geisler, and Christoph Quix. Constance: An intelligent data lake system. In *Proceedings of the 2016 international conference on management of data*, pages 2097–2100, 2016.

[4] Alon Y. Halevy, Flip R. Korn, Natasha Noy, Christopher Olston, Neoklis Polyzotis, Sudip Roy, and Steven Euijong Whang. Managing google's data lake: an overview of the goods system. *IEEE Data Eng. Bull.*, 39:5–14, 2016.

[5] Christoph Quix, Rihan Hai, and Ivan Vatov. Gemms: A generic and extensible metadata management system for data lakes. In *CAiSE forum*, volume 129, 2016.

[6] Franck Ravat and Yan Zhao. Data lakes: Trends and perspectives. In *International Conference on Database and Expert Systems Applications*, pages 304–313. Springer, 2019.

[7] Pegdwendé Sawadogo and Jérôme Darmont. On data lake architectures and metadata management. *Journal of Intelligent Information Systems*, 56(1):97–120, Jun 2020.

[8] Pegdwendé N. Sawadogo, Étienne Scholly, Cécile Favre, Éric Ferey, Sabine Loudcher, and Jérôme Darmont. Metadata systems for data lakes: Models and features. In Tatjana Welzer, Johann Eder, Vili Podgorelec, Robert Wrembel, Mirjana Ivanović, Johann Gamper, Mikolaj Morzy, Theodoros Tzouramanis, Jérôme Darmont, and Aida Kamišalić Latifić, editors, *New Trends in Databases and Information Systems*, pages 440–451, Cham, 2019. Springer International Publishing.

[9] Xiaosong Wang, Yifan Peng, Le Lu, Zhiyong Lu, Mohammadhadi Bagheri, and Ronald Summers. Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In *2017 IEEE Conference on Computer Vision and Pattern Recognition(CVPR)*, pages 3462–3471, 2017.

POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

# The Design of a Data Lake architecture for the healthcare use case: problems and solutions

Tesi di Laurea Magistrale in
Computer Science and Engineering - Ingegneria Informatica

Author: **Sara Parente**

Student ID: 928105
Advisor: Prof. Letizia Tanca
Co-advisors: Prof. Marco Gribaudo
Academic Year: 2020-2021

# Abstract

With the growth of data volume and data heterogeneity in the healthcare domain, the need for new types of storage systems has emerged. In particular, there is the need to store different types of data, e.g. lab reports, medical images, genomics data, in a central repository in order to create an organized and rich dataset. Since traditional approaches cannot satisfy these requirements, we opted for the adoption of a Data Lake. A Data Lake is a centralized repository that allows to store structured, semi-structured, and unstructured data at any scale. However, the current Data Lake architectures do not support the specific needs of healthcare data, therefore we proposed an architecture that can fit this use case. The objective was to design a system that can efficiently ingest, store and process medical images, such as x-rays, CTs and PETs, as well as the metadata related to these images. This last aspect plays a very important role in the Data Lake architcture, as it allows full exploitation of the data value. In order to better understand the requirements that the architecture should satisfy, we developed a simple implementation of some steps of the data flow across the Data Lake. The prototype takes as input a series of x-rays and their annotations, performs image preparation and image analysis, and finally stores the results in a relational database. This has allowed us to quantify both the number of features that are generally extracted from a single image and the resource usage during these processes. Moreover, we proposed two cloud solutions that can realize the proposed architecture: the first one takes advantage of services offered by Amazon, while the second one uses services provided by Microsoft. Overall, our work is a step in the right direction to implement a Data Lake that can be used in the healthcare environment and, we believe that, by adding some extensions, a complete solution can be implemented.

**Keywords:** Big Data, Data Lake, Medical Images

# Abstract in lingua italiana

La costante crescita del volume e dell'eterogeneità dei dati in ambito sanitario, ha fatto emergere la necessità di nuovi tipi di storage. In particolare in questo settore vi è la necessità di raccogliere diversi tipi di dati, come referti di analisi, dati genomici e immagini mediche, in un sistema centralizzato che permetta di creare un dataset organizzato e completo. Considerato che gli approcci tradizionali non riescono a gestire in maniera appropriata questo tipo di problema, abbiamo optato per l'utilizzo di un Data Lake. Il Data Lake è una repository centralizzata che permette di salvare su qualsiasi scala dati strutturati, semi-strutturati e non strutturati. La mancata creazione di un'architettura generale per questo tipo di sistema, ci ha indotto a proporre questo lavoro che cerca di dare una soluzione al problema. L'obiettivo è quello di progettare un sistema che efficacemente ingerisca, salvi e processi immagini mediche, come radiografie, PET e TAC, e che allo stesso tempo sia in grado di gestire i metadati associati a tali immagini. Questo ultimo aspetto va analizzato con attenzione quando si progetta il Data Lake perchè permette di sfuttare appieno il valore dei dati. Per comprendere meglio i requisiti che l'architettura deve soddisfare, abbiamo sviluppato una implementazione di alcune parti del Data Lake. Più nel dettaglio, l'implementazione riceve in input una serie di radiografie con le relative annotazioni, esegue gli step di preparazione, le analizza e salva i risultati di quest'ultimo passaggio su un database relazionale. Ciò ha permesso di quantificare sia il numero di features che vengono estratte da una singola immagine, sia l'utilizzo delle risorse durante questi processi. Inoltre, nella tesi vengono proposte due soluzioni cloud che sviluppano l'architettura presentata, la prima soluzione sfrutta i servizi offerti da Amazon, mentre la seconda utilizza servizi forniti da Microsoft. In sintesi, questo lavoro rappresenta il primo passo dell'implementazione di un Data Lake in ambito sanitario e con le dovute estensioni potrà fornire una soluzione adeguata.

**Parole chiave:** Big Data, Data Lake, Immagini Mediche

# Contents

# 1 | Introduction

In the last years, due to the ever increasing number of devices connected to the Internet and to the continuous technological growth, the generation of data of any type has also increased. This growth has given rise to the phenomenon of Big Data. The new characteristics associated to this data have made the need of reorganizing business processes and the need of developing new solutions increasingly evident. Above all, it has emerged the need of new systems that allow to exploit new opportunities that came with Big Data and that allow, also, to handle the constantly increasing complexity of data management. So, in order to satisfy these needs, new scalable and efficient architecture have been designed. The new frameworks born to store and analyze Big Data are mainly based on parallel architectures.

The framework that is currently most used for Big Data storage and management is the Data Lake. However, this type of systems cannot be used without defining an appropriate structure, and governance policies. Without these cautions, there is the risk of losing the value of the data as its volume grows. If this happens, the Data Lake becomes a Data Swamp and access to data is no longer efficient, nor effective.

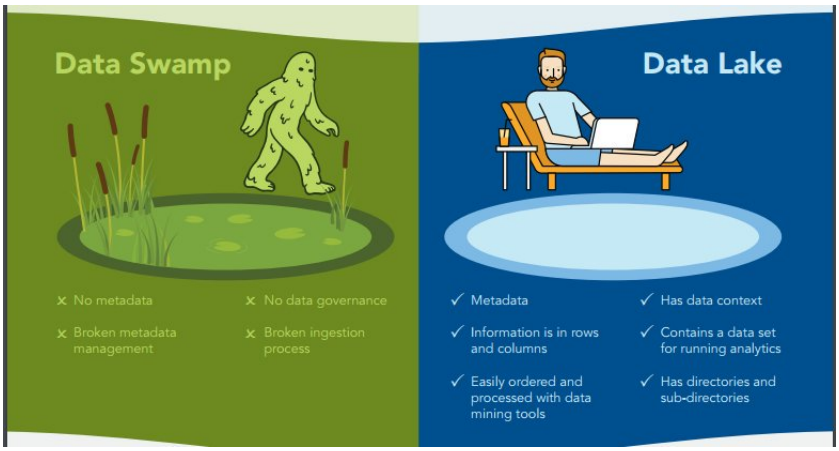


Figure 1.1: Difference between a Data Lake and a Data Swamp [10]

In the healthcare field the term "Big Data" is used to describe all information generated from the digital technologies that collect patients' records. Effective use of Big Data in Healthcare is enabled by the development and deployment of machine learning and artificial intelligence approaches [12]. Machine learning and artificial inteligence algorithms make it possible to unravel the patterns, associations, correlations and causations in complex, unstructured and non-normalized datasets that the Big Data era brings. This allows to perform analysis on datasets as varied as sequences of images or narratives, i.e. patient records, and bringing all these datasets together to generate prediction models, such as response of a patient to a treatment.

In the near future, physicians will have to deal with terabytes of data. Gaining clinical insight from such a large volume of data is an impossible expectation to place upon humans. An answer to this problem is to make Electronic Health Record not only records of patient-physician interactions and digital medical records, but also diagnostic aids. Another important aspect is to make this huge volume of data available and easily accessible by physicians in order to make the diagnostic process much faster.

## 1.1. Aim of the thesis

The aim of the thesis is to design a system that can efficiently handle healthcare Big Data. The project involves the adoption of a Data Lake that connects many Istituti di Ricovero e Cura a Carattere Scientifico [1] (IRCCS) in order to have a central repository for all connected data and, therefore, create a rich dataset. The Data Lake should be the only entry point for all types of data, which are stored in their raw format. More in detail, the objective is to design a system that integrates traditional data and multimedia data, this can be achieved only by developing an architecture that can handle all types of data. The system should also allow users to write queries that use both traditional and multimedia data. Then, results coming from all types of involved data have to be merged to return a unique result. For example, given a medical image there could be the need to look for images similar to it and to filter these images on patients information such as age and sex.

The proposed system should be scalable, and it should allow to quickly access every type of data. This can be achieved by extracting structural, semantic and process metadata that can make data retrieval easier and can help query processing.

[1] Institute for Treatment and Research

Since in the healthcare domain data is very heterogeneous, as it spans from lab reports to genomic data, we have decided to restric our analysis to medical images, e.g, x-rays, CTs and PETs. So, the project proposes a Data Lake architecture that can handle this type of data and its metadata. Another important aspect analyzed in the thesis is the identification of features that can properly represent medical images. The step of feature extraction is really important as it allows to apply machine learning and artificial intelligence algorithms on the images. This phase requires a careful analysis of resource usage as it is the most demanding step in the elaboration of images.

## 1.2. Brief description of the work

The focus of the thesis was to design a storage system that can appropriately handle medical data and, in particular, medical images. First of all, we have identified the best solution for the use case under exam, which is a Data Lake. Then, we have proceeded with the design of the architecture to implement the Data Lake, as the literature does not offer a general framework to implement this type of storage yet. More in detail, we have identified the components needed to extract, load and trasform data, and in this phase the focus was on the modules needed for storing and analyzing medical images.
Another important aspect to consider in this step was metadata management as, if it is not well defined, it can compromise the functioning of the entire system. The next step has been the study of the data flow inside the Data Lake, and for this purpose we implemented of some parts of the flow. This implementation has allowed us to better understand the inputs and outputs of the main components of the architecture and also to quantify resource usage during the phases of image preparation and feature extraction. Thanks to resource usage analysis we were able to fully understand the requirements for the architecture.

Lastly, a brief research on the major cloud service providers has been carried out. Among all available providers, two have been identified as the ones that offer services that best fit the use case under exam: Microsoft and Amazon. For each company, we made a research aimed to identify the needed services and to provide a complete solution for the storage of medical images.

## 1.3. Structure of the thesis

This document is structured as follows:

- Chapter 2 presents the background of the work, explaining the context in which the case under exam is placed. In particular, the concept of Big Data is introduced and, then, two integration-oriented storage paradigms are presented: Data Lakes and Data Warehouses. In this chapter we also made a comparison of the two, in order to better explain the points that have been analyzed in the selection of the type of system. Then, graph databases are biefly introduced as they are used for metadata management and, also, the concept of cloud computing is described as it is a relevant topic when talking about Big Data storage. The last part of the chapter discusses the state of art that is relevant for our work.
- Chapter 3 shows the proposed architecture for the Data Lake. First of all the choice of the Data Lake has been explained. Then, the architecture is presented and a more detailed description of the Data Lake zones is provided. Lastly, we explain and define the aspects that have to be considered in the design of the Data Lake.
- Chapter 4 first provides a more detailed description of the main technlogies proposed for the implementation of the Data Lake. Then, it analyzes the data flow that crosses the system.
- Chapter 5 presents the two cloud solutions. The first one uses services provided by Amazon and the second one exploits services offered by Microsoft.
- Chapter 6 concludes the whole thesis. It provides a brief recap of the work that has been done. In this chapter we also discuss possible future works.

# 2 | Theoretical background

This chapter provides the theoretical foundations required to understand the content of the thesis. More in detail, Section 2.1 presents the concept of Big Data by introducing the 5Vs, Section 2.2 introduces the concept of Data Lake and, Section 2.3 makes a comparison between the Data Lake and the Data Warehouse. Section 2.4 describes graph databases and Section 2.5 briefly introduces the concept of Cloud Computing. Finally, Section 2.6 discusses the state of the art relevant for our work.

## 2.1. Big Data

The literature gives different definition of Big Data [48]. Microsoft defines Big Data as "the process of applying serious computing power – the latest in machine learning and artificial intelligence – to seriously massive and often highly complex sets of information" [39]. From this definition it is clear that the boundary between data and Big Data is not well defined. Defining a limit in terms of concrete storage space or computing power is not an easy task, since these aspects can vary based on the use case and, also, the fast evolution of available technologies continuosly moves the boundaries of Big Data. The Gartner Group, instead, defines Big Data as "Big data is high-volume, high-velocity and/or high-variety information assets that demand cost-effective, innovative forms of information processing that enable enhanced insight, decision making, and process automation" [3].

Gartner's definition introduces the concept of the 3Vs which represents three characteristics that are usually adopted to describe Big Data.

- **Volume** refers to the size of Big Data. The name "Big Data" itself is related to a size which is enormous. The rapid increase of volume is the consequence of an increasingly connected world where new data sources continuously proliferate.
- **Velocity** refers to the speed at which the data is generated. In the Big Data era, data can be created and passed on in real time or near real time. This term can

also refer to the different speed at which different types of data can be generated. As a consequence of this aspect, it is essential to handle in the appropriate way the ingestion phase and the division of the workload.

- **Variety** refers to the nature of data that can be structured, semi-structured and unstructured. Structured data is usually data defined through a schema, this is the easiest type of data to handle. Semi-structured data is semi-organized data, it doesn't conform to the formal structure of data. Data that does not have a structure is unstructured, this type of data cannot be used right away, but usually some processing is needed. However, variety can also refers to the presence of heterogeneous sources.

More recently, two more characteristics have been added to the three original ones.

- **Veracity** refers to the possible lack of quality, i.e., integrity, credibility or accuracy in the data. This term was first used by IBM to highlight that when dealing with Big Data there is always a certain level of uncertainty. So data should always be checked for accurancy before using it for business insights.
- **Value** refers to how useful data is in the decision making process. This is where data analytics comes into play. Without appropriate analytics it is not possible to extract the value out of the Big Data.

Over the years more characteristics were introduced: Viscosity describes the level of correlation in the data, Volatility refers to data durability, Variability addresses the inconsistencies in the data flow and Validity is the ability to find hidden relationships. However, the five main characteristics of Big Data are the ones presented above.

From these characteristics it can be deduced that Big Data has a big potential, but it involves also big challenges [45] (see Figure 2.1).

**Data challenges** are related to the characteristics of the data itself, i.e. data volume, veracity, variety, vlocity, value.

**Process challenges** are related to the new techniques required to handle Big Data. This process is composed by different steps:

- acquisition and storage of new data: the first challenge of big data is the storage. This because traditional approaches are usually not suitable for this type of data.
- Data cleansing: extraction and cleaning of data from a pool of large scale unstruc-

tured data can be very challenging.

- Aggregation and integration: with Big Data, data is available in large volumes and is modeled using different representations. This aspect introduces the challenge of integrating these data sources in order to extract knowledge and support the decision making process.
- Data analysis and modelling: once data has been integrated the next step is the analysis and modelling of Big Data. Although both Big Data systems and traditional Data Warehouses have the same goal of delivering business value by analyzing data, the difference is in the analytics and in the organization of data. The old ways of data modelling can no longer be used because of the unprecedented need of storage capacity and computing power.
- Data Interpretation: is the step that makes data understandable for final users. The growth of unstructured data has affected the way people process and interpret this raw data.

**Management challenges** are related to the aspects of privacy, security and governance.

- Privacy: with Big Data how to preserve the privacy has become one of the most important challenges. There is an increasing fear of inappropriate use of personal data especially when combining data from multiple sources [35].
- Security: this type of challenges is related to the distributed nature of Big Data that makes the data more vurnerable to attack.
- Data Governance includes all processes and technologies needed to manage and protect data assets in order to guarantee correct, complete, secure and discoverable data. It is an important aspect as it helps maintain the value of data as a key organizational aspect.

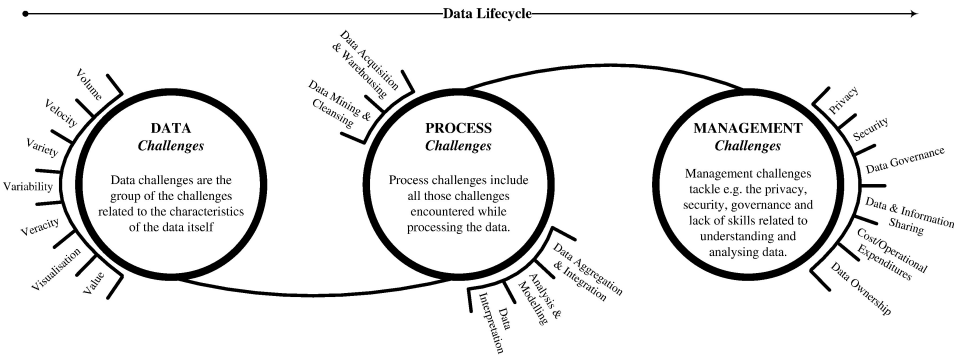


Figure 2.1: Big Data Challenges [45]

## 2.2. Data Lake

As previously explained, Big Data has created a lot of concerns about how to store it and protecting it. As an answer the concept of Data Lake has been introduced. In [31] the Data Lake is defined as "a logical view of all data sources or dataset, in their raw format, available and accessible by data scientist or statistician to find new insight.

- A data lake is governed by a metadata sources index to guarantee the data quality.
- A data lake is controlled by rules, tools and processes to guarantee the data governance.
- A data lake is limited to data scientist or data statistician access to guarantee data security, data privacy and compliance.
- A data lake access all type of data.
- A data Lake has a logical and physical design"

The Data Lake can be seen as a central repository where data is stored without a predefined schema.

From the definition of the Data Lake it is evident that this type of systems can efficiently handle Big Data. Metadata helps exploiting data's value, data variety is preserved as the Data Lake access all type of data and the rules, tools and processes that guarantee the data governance help managing veracity.

There are different benefits that come with the adoption of a Data Lake:

- **Data is stored in native format**, the Data Lake removes the need for data modeling at the time of ingestion. This step can be done when exploring data for analytics.
- **Scalability**, it can effectiely handle the growing amount of data.
- **Schema flexibility**, the Data Lake applies the *schema on read* paradigm which creates the schema only when reading the data. Since it's not necessary to define the schema before storing the data, bringing new data sources in is much easier.
- **Advanced analytics**, unlike traditional approaches, a data lake excels at utilizing the availability of large volumes data along with deep learning algorithms to recognize items of interest that will power real-time decision analytics.

- **Ingestion from different sources**, it can ingest multi-structured and massive datasets from disparate sources. This means that the Data Lake can store literally any type of data such as multimedia, binary, XML, logs, and so on.

### 2.2.1. Data lake architectures

The literature classifies Data Lake architectures into Pond architectures and Zone architectures [23].

**Pond architecture**, in [29] the Data Lake is represented as a set of data ponds. Each data pond can be seen as a part of the Data Lake which handles a specific type of data. In [29] five data ponds have been identified:

- Raw Data Pond handles newly ingested data, i.e. raw data. It represents a transit zone as data is then conditioned and passed to another pond. This pond is not associated with any metadata system.
- Analog Data Pond contains data characterized by a very high frequency of measurements, e.g. log files, IoT data.
- Application Data Pond is the pond dedicated to data generated by software applications. So, this data is usually structured data coming from Relational Database Management Systems (RDBMS). In this pond data is integrated, transformed and prepared for analysis.
- Textual Data Pond handles unstructured, textual data.
- Archival Data Pond comprises data that is not actively used, but may be needed in the future. The data contained in this pond may be generated by the analog, textual or application pond.

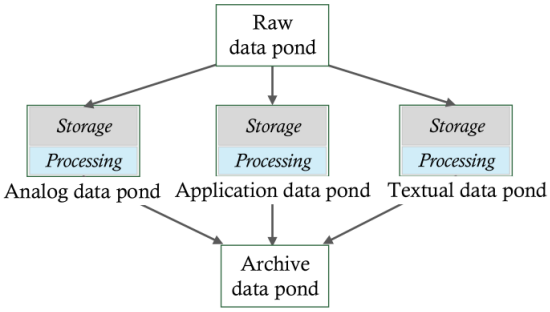


Figure 2.2: Pond architecture schema [42]

**Zone architecture** assigns data to a zone according to its degree of refinement. For example, in [24] six zones have been identified:

- Landing zone, is the first zone of the Data Lake. Data can come into the Data Lake in batches or streams.
- Raw zone is the zone where data is kept in its raw format. The data will live here until it is operationalized.
- Harmonized Zone: data is passed to this zone in a demand-based manner. When data passes to this zone it is not deleted from the raw zone, the harmonized zone contains a copy of the data. In this zone data coming from different sources is integrated in a common schema regardless of its structure.
- Distilled Zone increases the efficiency of following analysis by preparing the data accordingly. It is the first zone that divides data based on the use cases.
- Explorative Zone represents the place where data scientists can explore and analyze data. If results of applied analysis need to be stored, they can be sent back to the distilled zone.
- Delivery Zone, contains small subsets of data which are tailored to specific applications. The main difference between this zone and the explorative zone is that the delivery zone supports users with little knowledge on data analytics.

However, this is just an example of a zone architecture. This type of architectures can have different numbers of zones and the also the characteristics of each zone may differ.

The main difference bewteen the two architectures is that in the pond architecture raw data is deleted when transferred to other ponds. In the zone one, instead, a copy of the data is kept in each zone. This may represent a drawback because multiple copies of data are availble in the system and this can generate difficulties in menaging data lineage.

### 2.2.2. Data lake challenges and problems

Despite all the benefits, Data Lakes are still not much utilized. This is because there are still many challenges and research gaps [22].

- **Data Lake architecture**: at the moment no generally accepted architecture is available. This makes it necessary to investigate and compare the existing alternatives to extract similarities. Moreover, only the conceptual organization of data in zones or ponds is analyzed, there is no architecture that includes also the modeling

aspect or the infrastructure one.

- **Data Lake governance**: the new requirements of flexibility and open access come into conflict with traditional governance approaches, i.e. Data Warehouses. Therefore, the necessity of a comprehensive governance concept which should take into consideration all different types of data has emerged.
- **Comprehensive strategy**: there is also a lack for what concerns the comprehensive design and the realization strategy. The strategy should consider interdependencies between different data lake aspects, such as data lake architecture and data lake modeling, and should combine all aspects into one comprehensive and systematic data lake concept.

## 2.3. Data lake vs Data warehouse

The reader may wonder why we have opted for a data lake instead of more traditional approaches. So, in this section the concept of data warehouse is briefly introduced and, then, the main differences between this type of storage and the data lake are presented in Table 2.1.

Data Warehouses are not a new concept, they were developed in the late 1980s and evolved over time. The Gartner Group defines the Data Warehouse as "a storage architecture designed to hold data extracted from transaction systems, operational data stores and external sources. The warehouse then combines that data in an aggregate, summary form suitable for enterprisewide data analysis and reporting for predefined business needs" [4]. The four main characteristics of a data warehouse are:

- **Subject-oriented**, it usually provides information on a topic, e.g. sales, inventory, promotion, rather than company operations.
- **Time-variant**, the time horizon is significantly longer than the time horizon of an operational system.
- **Non volatile**, meaning that data in this type of system are accessed and loaded, but not frequently updated.
- **Integrated**, data coming from heterogeneous sources are converted into a unified schema.

The Data Warehouse can be seen has a 3-tier architecture:

- Bottom tier is usually composed by a data warehouse server, normally a relational database system, which collects, cleanses, and transforms data from multiple data sources through Extract-Transform-Load (ETL) processes.
- Middle tier consists of an On-Line Analytical Processing (OLAP) server which enables fast query speeds.
- Top tier is composed by a front-end user interface or a reporting tool, which enables end users to conduct ad-hoc data analysis on their business data.

As mentioned above the Data Warehouse is an OLAP system. This type of systems is suited to perform multidimensional analysis at high speeds on large volumes of data. OLAP systems are usually prefered when where is the need to run data mining alghoritms, intelligence applications and complex analytical calculations. They are also used for business reporting functions.

Since On-Line Transaction Processing (OLTP) systems and OLAP systems satisfy different requirements, two different data models are needed. The Entity-Relationship model, used to represent OLTP applications, is not sufficient to represent multidimensional data. To solve this problem another conceptual model was developed. The **data cube** is the model used to conceptualize data in the data warehouse [17]. The cube is composed by cells that represent a measure based on a set of dimensions. It is thus important to introduce the basic concepts on which data warehouses are built.

- **Fact** represents a business measure. It is a concept which is relevant for decisional processes.
- **Measure** represents a numerical property of the fact.
- **Dimension** describes "who, what, when, how, and why" associated with the event [15].

The operations that are usually performed on a data cube are:

- roll-up, aggregates data at a higher level;
- drill-down, de-aggregates data at a lower level;
- slice and dice, apply selections and projections which are used to reduce data dimensionality;
- ranking, sorts data based on a predefined criteria.

- pivoting, selects two dimensions to re-aggregate data.

In a nutshell, the Data Warehouse stores data that is needed for business and analytics and this data is stored following a unique model.

After this introduction to data warehouses, it is more evident that this type of systems differ, under many aspects, from Data Lakes. Table 2.1 summarizies the main differences.

| | **Data Lake** | **Data Warehouse** |
|---|---|---|
| **Storage** | All data is stored irrespective of the structure. Data is kept in its raw form and is only modified when it is used. | Data is cleansed and transformed at the moment of extraction. |
| **Data Ingestion** | Ingests all types of data, i.e. structured, semi-structured and unstructured | Stores structured data in a predefined schema. |
| **Data Timeline** | It can store data as long as there is enough space. In this way it stores not only data that is being used, but also data that may be needed in future. | ETL processes makes it impossible to store data that is not being used. |
| **Type of users** | It is generally used by users that perform in-depth analysis, i.e. data scientists | Since it is structured and easy to use, it is a good fit for operational users. |
| **Processing Time** | Since users can access data before it has been cleaned, modified or structured, it is possible to get fast results. | It allows to query data with predefined schema. Changes to data structure require more time. |
| **Schema** | The schema is defined after data has been ingested. This makes ingestion simpler, but the effort required in the processing phase is higher. | The schema is defined before ingestion. So, at ingestion time more effort is required, but it offers better performance and integration in the processing phase. |
| **Processes** | Data Lake usually uses Extract-Load-Transform (ELT) processes. | It generally uses ETL processes. |

| | | |
|---|---|---|
| **Data Modification** | Since there is no predefined structure data access and modification are easier. Moreover, data modifications can be applied quickly. | The predefined structure makes data manipulation more difficult and more time consuming. |

Table 2.1: Differences between Data Lake and Data Warehouse.

However, despite all the differences, the Data Lake and the Data Warehouse are often used toghether. Usually, the data warehouse is built on top of the data lake, as shown in Fig.2.3.

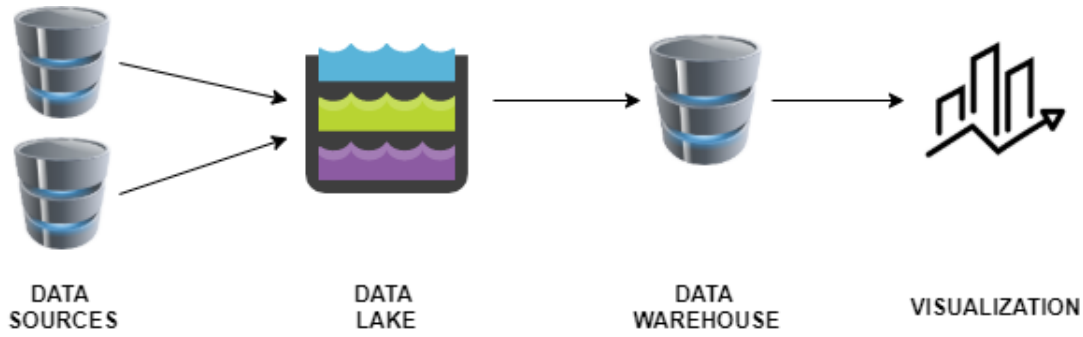


Figure 2.3: Schema of a system in which both a data lake and a data warehouse are used

As depicted in Fig.2.3, this type of systems are usually composed by a number of different data sources which produce data. This data is ingested by the data lake that acts as a staging area for the data warehouse. Then, there is the data warehouse that contains data in a structured form. The structure depends on the type of analysis that is performed. Finally there are all the tools that allow to do analytics, it is important to underline that these tools allow users to visualize data structured in the data warehouse.

## 2.4. Graph database

Traditional database systems for storage have been based on the relational model. These are widely known as SQL databases named after the language they were queried by [30]. However, with the constant growth of volume and heteogeneity of data, relational databases proved their powerless, as they are not flexible and scalable enough to manage this data. To overcome these problems a new kind of databases has been introduced, NoSQL databases. The term "NoSQL" was first coined by Carlo Strozzi in 1998 to name

his lightweight, open-source relational database that did not expose the standard SQL interface [21]. The term was later reintroduced by Eric Evans in 2009 to label the emergence of a growing number of nonrelational, distributed data stores that often did not attempt to provide atomicity, consistency, isolation and durability guarantees that are key attributes of classic relational database systems [21]. Recently, the term has assumed another meaning, "*Not Only SQL*", which underlines that for some use cases the relational model is not the best solution, but for many other cases this model is still the best option.

The main characterististics of NoSQL databases are [41]:

- Flexibility: since this type of databases is schema-less, adding and removing entities and relationships is easier.
- Scalability: NoSQL databases deal with billions of entities by scaling horizontally, i.e. adding resources when needed.
- Availability: since data is replicated on many nodes, it is always available.
- Fault-tolerance: NoSQL databases are generally distributed over multiple nodes. In this way if one node fails, the data that was stored on that node can still be obtained from other nodes thanks to replication.

**Graph databases** are a type of NoSQL databases. Formally, a graph is a collection of nodes and the relationships that connect them. Graphs represent entities as nodes and the ways in which those entities relate to the world as relationships [40]. The most popular form of graph model is the *labeled property graph* which has the following characteristics [40]:

- it contains nodes and relationships.
- Nodes have properties which are represented as key-value pairs.
- Nodes can be labeled with one or more labels which help grouping nodes.
- Relationships are directed, and always have a start node and an end node.
- Relationships can have properties.

In contrast to other database management systems where connections between entities have to be inferred using mechanisms like foreign keys, in graph databases relationships represent the core of the model. These relationships allow data in the store to be linked together directly and, in many cases, retrieved with one operation. In this way querying relationships is fast because they are permanently stored in the database.

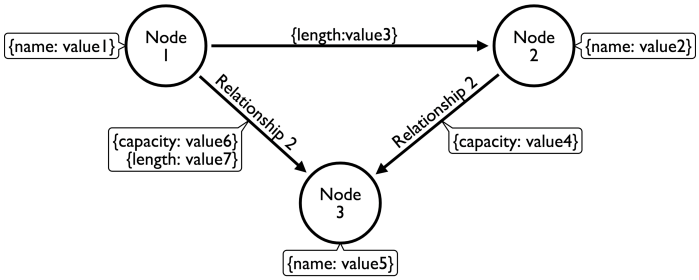


Figure 2.4: Representation of the elements that compose a graph database [5]

Fig. 2.4 shows a simple representation of a graph database. The circles represent the nodes and the arrows represent connections. It is important to underline that links between nodes have a direction. Also, both nodes and relationships are described through a set of properties which are key-value pairs and are shown between curly brackets.

## 2.5. Cloud Computing

The exponential growth of data volumes has made cloud computing essential. In order to analyze this huge amount of data it is essential to rely on Cloud Computing. NIST defines cloud computing as "a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" [33]. So, cloud computing can be seen as a series of technologies enabling data to be processed using resources distributed over the network. Usually, providers of these services charge the user of a cost which is proportional to the use of the service. All offered infrastructures are managed in a way that is transparent to the user that does not have to install anything.

The main benefits of cloud computing are:

- Costs: this is one of the main benefits of cloud computing. It helps saving substantial capital cost as it does not need any physical hardware investments. Also, there is no more the need of trained personnel to maintain the hardware, the buying and managing of equipment is entirely done by the cloud service provider.
- Scalability: the cloud substantially reduces the problem of sizing the Big Data environment. Thanks to the elasticity of cloud computing there is no longer the need of over-provisioning to cope with peak loads. The possibility of scaling resources allows to have the right number of components needed for a specific task. In this

way there is the certainty of always having the sufficient number of resources and at the same time there is no longer the issue of underutilized resources.

- Reliability: this is one of the biggest benefits of Cloud hosting. The cloud reduces backup costs and increases ease of recovery in the case the system is affected by a fault. This is possible because data is replicated inside the provider's network.
- High speed: another important aspect is how quickly a Big Data infrastructure can be deployed. In particular the service known as Infrastructure-As-Code allows to define an infrastructure by simply writing some code.
- Collaboration: by using cloud computing collaboration becomes easier. It is possible to work on a project from different locations as cloud computing allows access from everywhere.
- Unlimited store capacity: in the cloud storage capacity is almost limitless. It is always possible to extend the storage capacity and the costs are pretty low.

Summing up, cloud computing allows to develop a complete and fault tolerant infrastructure in really short times. In addition, infrastuctures based on cloud computing are extremely flexible and scalable and can grow dynamically to accomodate specific needs.

The combination of cloud computing and Big Data brings many advantages inside the organization. First of all, the high speed in data processing allows organizations to perform real time analysis. This type of analysis makes the decision making process much faster and more accurate. Another big advantage resides in the adoption of the cloud storage server which is the central component that stores all data. By storing data in this server there is no more the issue of managing security and privacy aspects as generally the provider and the client sign a Service Level Agreement that guarantees a certain level of security.

## 2.6. State of art

### 2.6.1. Metadata management system

The literature proposes different metadata models. One of this is the Generic and Extensible Metadata Management System (GEMMS) which extracts metadata from the sources and manages the structural and semantical information in an extensible metamodel [37]. In particular, GEMMS extracts data and metadata from heterogeneous sources, stores this metadata in an extensible metamodel, enables the annotation of the metadata with

semantic information, and provides basic querying support. In GEMMS data is represented as key-value pairs, and it is possible to attach to each key-value pair annotations that are usually represented as a Unified Resource Identifier (URI) pointing to an ontology element. The architecture is composed by the *Media Type Detector* which identifies the type of files that compose the dataset, the *Extractor* that creates a specific instance of the *Praser* and the Parser that analyzes the internal structure of the dataset and it extracts metadata. It is importnat to highlight that the Parser is built for a specific type of file and, therefore, when a new type of file is detected it is necessary to create a new parser that can handle the new type.

Another proposed model is GOODS [27], an architecture developed by Google to manage metadata inside the Data Lake. In this case the catalog contains metadata created from datasets that are combined with those generated by production pipelines and those generated by the analysis of the dataset content. This metadata allows to infer relations between datasets and provide a unified view of the Data Lake to users and services. Inside the catalog, related datasets can be grouped into clusters that become first-class entities to which other metadata is associated. Since comparing each pair of datasets is not a scalable solution, GOODS works in *post-hoc* mode and it discovers relationships between datasets by analyzing the signals of the underlying infrastructure. Relationships between datasets can be inferred from already extracted metadata regarding, for example, the structure and the provenance. The discovered relationships are stored in a knowledge graph.

Constance [26] is another architecture for metadata management. There are two main components in its architecture, as shown in Fig.2.5. The first one is the Structural Metadata Discovery which extracts structural metadata, and the second one is the Semantic Metadata Matching which extracts semantic metadata. The produced output is a graph that represents metadata elements and their relationships.

In particular, Constance:

- extracts metadata and stores them in an extensible and flexible model
- makes annotations on the dataset with semantic information
- provides a graphical interface that allows to monitor the metadata management process

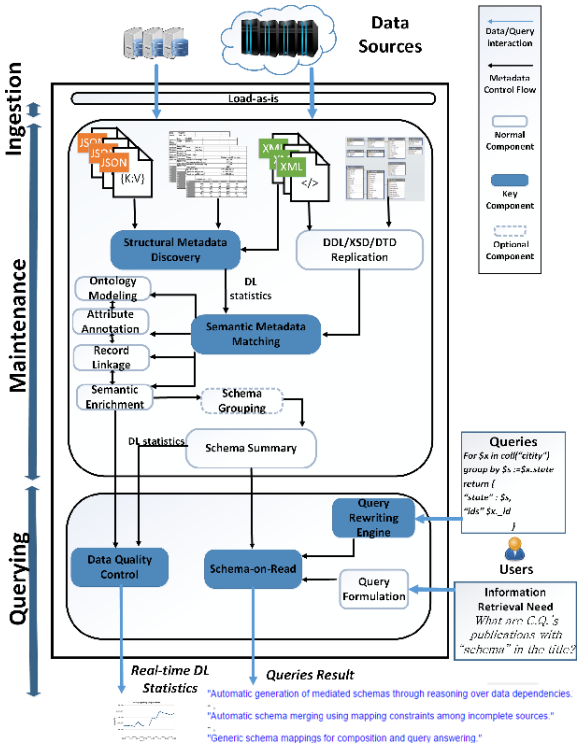


Figure 2.5: Constance Architecture [26]

MEtadata for DAta Lakes (MEDAL) [44] uses the concepts of hypergraph, nested graph and attributed graph to reprent metadata. Each object is described by a hypernode which contains various elements such as properties, versions and so on. These hypernodes can be linked. This model classifies metadata into:

- intra-object metadata represent those characteristics that are associated with a single object of the Data Lake. In particular, MEDAL considers representations, versions, transformations and attributes, such as the origin of the object, as characteristics.
- Inter-object metadata describe relationships between objects. There are two main types of links: the similarity link expresses the level of likeness between objects and the parenthood link is used to keep track of lineage information.
- Global metadata are pieces of data that offer a context layer which can be useful during data processing and analysis. This metadata is not related to a specific object, but it usually describes the entire Data Lake.

#### 2.6.1.1. HANDLE

However interesting, the models described above do not suit the healthcare use case. First of all, they mostly focus on structured and semi-structured data. Moreover, none of them allows to describe to which zone the data belong, which is an important aspect to understand the level of refinement. Lastly, these models do not take into consideration the level of granularity, with the exception of MEDAL that allows to define two levels of granularity. We conclude that none of these methods is flexible enough for the examined use case.

Therefore, we conducted further researches and found a model that can fit the use case. Handling metAdata maNagement in Data LakEs (HANDLE) [18] has been developed with the intention of creating a model that can handle all use cases. As this is the chosen model for our use case a more detailed description is given in Chapter 3 when the entire data lake architecture is presented.

### 2.6.2. DICOM

The medical images that are ingested into the Data Lake from Picture Archiving and Communication Systems (PACS) are in the Digital Imaging and Communication in Medicine (DICOM) format. The DICOM specifies a non proprietary data interchange protocol, digital image format, and file structure for biomedical images and image-related information [13]. It represents the reference standard for what concerns medical images. It has been developed to facilitate the interoperability between software and equipment of different vendors. DICOM defines the rules for communication, visualization and storage of medical information.

# 3 | A general overview of the proposed Architecture

The literature [42] offers multiple examples of data lake architectures. These models, however, are too generic and do not take into considerations important aspects, such as data modeling and metadata menagement. The objective of the thesis is to define a comprehensive data lake architecture that best fits the studied use case. First, we explain when it is right to choose a Data Lake. Then, Section 3.1 presents an overview of the proposed architecture. Sections 3.1.1, 3.1.2, 3.1.3, 3.1.4 analyze the zones that compose the Data Lake. The Metadata Management System is explained in Section 3.2. Finally, Section 3.3 presents the aspects that have to be defined in order to design a comprehensive data lake.

In order to explain why the Data Lake has been selected, it is necessary to understand the limitations of the Data Warehouse that make it a non-feasible solution for the studied use case.

- The valuable data about the patients and medications is digitized and saved as Electronic Health Records which represent medical records of patients. This type of information comprises structured, semi-structured and unstructured data.
- Data Warehouses cannot address the issues related to scalability. To pull the data into it for further processing, data should go through the procedure of data preprocessing namely ETL. ETL processes are costly in terms of time.
- To provide the best solutions, data from diverse sources needs to be analyzed, but Data Warehouses cannot handle data from multiple sources effectively.

For the above mentioned reasons it is evident that the Data Warehouse does not represent a valid solution. Therefore, it has been opted for a Data Lake solution.

A Data Lake provides a unified location for all relevant data generated by the healthcare system, serving as repository for structured data drawn from traditional databases and unstructured data, such as patient images, lab reports, pathology, genomics and clinical notes.

Before diving into the proposed architecture, some important considerations should be made.

- The Data Lake paradigm is still very vague, therefore appropriate architectures and data governance techniques need to be defined, otherwise the Data Lake could easily became a Data Swamp.
- Consequently one of the first objectives is the definition of a metadata management system. In particular, the model chosen for storing the metadata should be flexible and scalable.

## 3.1. Overview

As already mentioned, the study is related to healthcare, with particular focus on medical images. Therefore, the proposed architecture aims at handling medical big data coming from heterogeneous sources in different formats, including multimedia. A high level component view of the system is presented in Fig. 3.1.

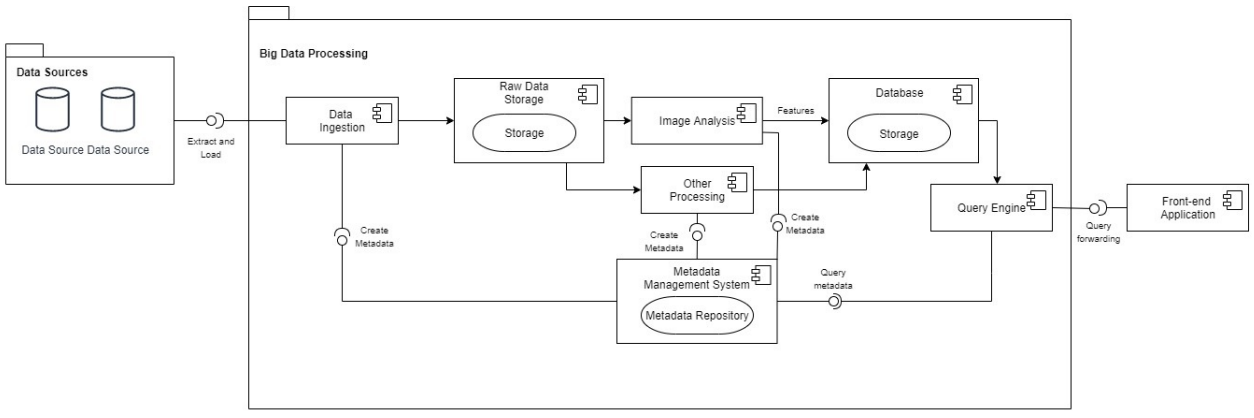


Figure 3.1: Component diagram of the Data Lake architecture

From Fig. 3.1, it can be observed that we have different data sources external to the Data Lake. Data is ingested from these sources by the Data Ingestion component and it is permenantely stored in the Raw Data Storage. The Ingestion module is also responsible for the generation of the metadata on the ingested dataset. Then, based on the type,

data goes under different kinds of procesing. In this phase other metadata can be generated, e.g. metadata on the processes. Data obtained from the processing steps is then stored in a Database. The Query Engine, instead, is the component that executes queries against data in that database to provide answers to the front-end application. Lastly, the Metadata Management System handles and stores the metadata generated by the other components.

As mentioned in Section 2.2.1 Data Lake architectures are usually classified in Data Pond Architectures and Zone Architectures [38]. For this specific use case a Zone Architecture has been chosen because data should go through different levels of refinement and, also, there is the need to keep data in raw format. In the architecture presented in Fig.3.1 four different zones can be identified:

- Transient Landing Zone, which is represented by the Data Ingestion module.
- Raw Zone, that comprises the Raw Data Storage.
- Stage Zone, which can be mapped to the two modules, Image Analysis and Other Processing.
- Sandbox, corresponds to the Database component.

It is important to highlight that metadata management spans all the zones as metadata can be produced at any stage. Two more aspects that span all zones are data quality, privacy, and security which are not represented here because they are out of the scope of this work.

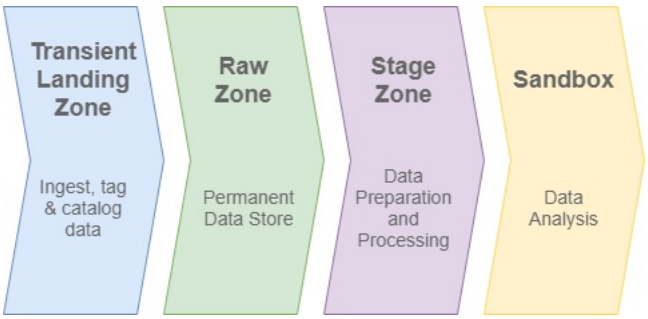


Figure 3.2: Data Lake's Zone

### 3.1.1. Transient Landing Zone

The Transient Landing Zone is the zone where data lands when extracted from the data sources and pushed to the Data Lake. Unlike ETL, usually, the Data Lake ingests data in raw format without applying expensive transformations. At this stage the first metadata is extracted and saved into a unified model.

In the use case we analyzed, medical images come from PACS, which is a medical imaging system used primarily in healthcare organizations to securely store and transmit electronic images and clinical reports. These images are collected in micro-batches and pushed into the Transient Landing Zone. In addition to metadata generation, which will be widely discussed in Chapter 4, in this layer the masking of Personally Identifiable Information (PII) is performed. As already mentioned, this last aspect is out of our scope here.

### 3.1.2. Raw Zone

The Raw Zone is the zone where data is stored indefinetely in its native raw format. The most important property of this zone is that it is considered as "the single source of truth" as it keeps the data in its original form. It is usually implemented as a file based storage with a well defined organization of files and directories. This last point is crucial because an unorganized storage transforms the Data Lake into a Swamp.

This zone covers a big role in the healthcare environment because it is likely that images in raw format are used multiple times to make different kinds of analysis.

### 3.1.3. Stage Zone

The Stage Zone is where data is landed for preparation and processing. This zone contains tools that provide some increased value over the pure raw data of the Raw Zone. The key aspect of this layer is that applies transformations that are needed by data scientists in order to make sense out of the data. The new data generated by this stage are organized following the model that best suits the subsequent analysis processes.

In Fig.3.1 the Stage Zone is composed by two modules, the "Image Analysis" module and the "Other Processing" module. This is to highlight that the main focus of this work is on medical images. In particular, in this layer some features are extracted from the images, as it will be better explained in the next chapter. In this processing phase new metadata is generated regarding the extracted features and the applied transformations.

### 3.1.4. Sandbox

The Sandbox is the zone where data scientists and researchers can build analytical models, discover associations and patterns within the data. In the healthcare case this zone is the zone where researchers can access data. Results produced in this zone can be brought back to the raw zone for later re-use.

For example, in the context of medical image analysis, the features extracted in the Stage Zone are mined to extract classification patterns and clustering criteria.

## 3.2. Metadata Management System

As said in the previous sections, metadata management is a crucial aspect of the Data Lake architecture as it allows full exploitation of data's value. Metadata is usually used to further explain all aspects of the data such as structure, meaning of the content, quality, lifecycle and many others. Since metadata is still data, it needs to be managed in a right way that allows to obtain all the possible value from it. An important issue is the model used to represent this metadata. A good model should allow to manage all metadata management use cases, from use cases specific to the considered application to the lineage information. In order to support any metadata management use case, the model must be very flexible in its ability to assimilate metadata [18]. Therefore, the requirement that it has to satisfy is modeling the metadata as flexible as possible. In order to achieve this flexibility, the model should meet the following conditions:

- Metadata can be stored in the form of objects, properties and relationships.
- The number of metadata object for each use case is unlimited.
- Each metadata element can have an arbitrary number of properties.
- Metadata objects can be linked.

As already mentioned in Section 2.6.1.1, HANDLE is the chosen model to represent metadata in the healthcare environment. This model can be divided into two main parts, the core model and three extensions. The scope of the core model is to define all elements and the relations that are needed to model metadata. The extensions, instead, represent the zones, the levels of granularity and the categoriazion topics. The model fulfills four important requirements for metadata management:

1. propose a model that is as flexible as possible

2. introduce the concept of zones
3. introduce the concept of levels of granularity
4. incorporate categorization using labels

Fig.3.3 describes a slightly modified model of HANDLE using an Entity Relationship schema. The modification that has been applied is the removal of the categorization property because it describes the context of metadata, e.g. operational, technical, business, and it does not apply to the examined application.

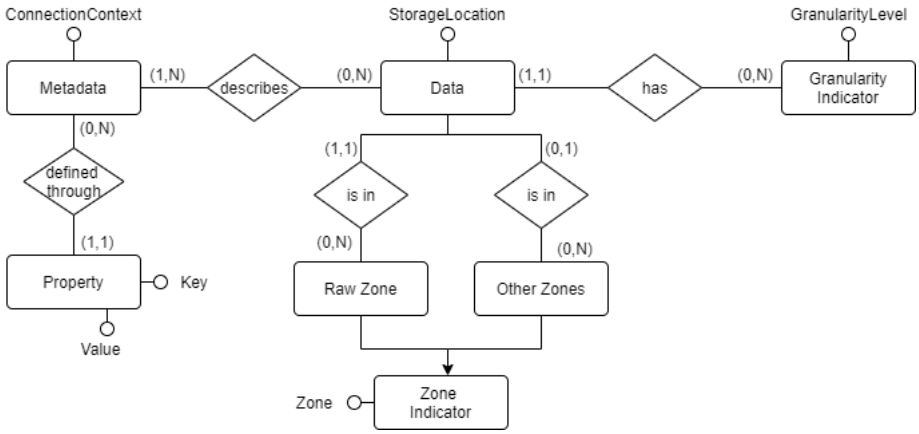


Figure 3.3: HANDLE Entity Relationship model

The schema in Fig.3.3 shows that, to each piece of data zero or more metadata can be attached. Each one of these metadata is described by a set of properties. The ConnectionContext attribute, instead, provides a description of the information contained in the metadata element. Data is also described through a granularity indicator and a zone indicator. It is important to underline that each data has a link to the Raw Zone also when it belongs to other zones as the Raw Zone always stores a copy of the data. For what concerns the attributes represented in the schema, the StorageLocation gives the path to the data element, and each property is defined through a key-value pair.

The Granularity Indicator entity allows to collect metadata at different levels of granularity. Of course, these levels depend on the structure of the data. In the examined use case, there are mainly two types of data, unstructured and structured data. For what concerns structured data, it is relational data and therefore the granularity levels are two, table and tuple. For the images, i.e. unstructured data, the levels are the entire image, the Region Of Interest (ROI) and the collection of images.

The ZoneIndicator entity provides the information about the location of a specific data element inside the Data Lake. The Raw Zone entity is designed to be the central ZoneIndicator, as data that is stored in any of the other zones will have a corresponding data element in the raw zone, making the raw zone the most stable reference [18]. The zone attribute can be simply a string that represents the name of the zone or it can be an enumeration. As previously said, in the examined use case the zones are four, Transient Landing, Raw, Stage and Sandbox.

In [18] also a possible implementation for this model is proposed. Since flexibility is a key aspect in metadata management, the authors have opted for a NoSQL technology. In particular, the chosen model is a graph database. This for two main reasons:

- On of the main characteristics that a metadata model must have is flexibility. Graph databases do not have a predefined schema and this makes them much more flexible than relational databases.
- In many use cases metadata are strictly correlated with each others and graph database can efficiently handle many relations.

In Fig.3.4 a simple example of a graph database modeled following HANDLE is presented. As said in previous sections, images are ingested in mini-batches and these batches are represented as an Image Collection with associated metadata. Then, for each one of the images in the batch other metadata, specific of the single image, is generated. It is important to highlight that both the zone and the granularity level are represented.

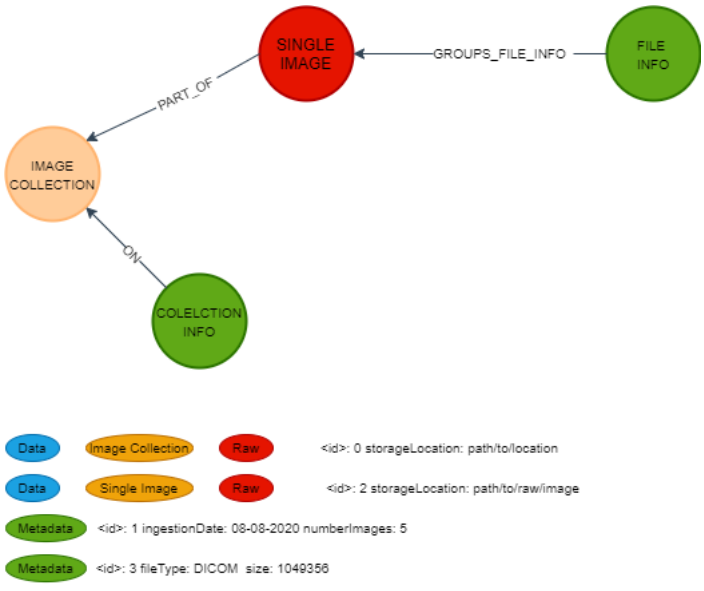


Figure 3.4: Example of a graph database for metadata management

## 3.3. Definition of Data Lake's characteristics

So far, no foundamental data lake architecture has been defined in the literature yet, and this makes it difficult to design and implement this kind of storage. While the literature offers data lake architectures [28] [43] [38] that are too generic to be implemented, the commercial products offer only few of the features that would be needed. Most papers only define data ingestion, data organization and data storage and leave out other important aspects such as metadata management and data modeling.

The Data Lake Architecture Framework [25] is presented as a high-level guide for building a comprehensive data lake. This framework defines nine aspects that should be defined in order to design a complete data lake.

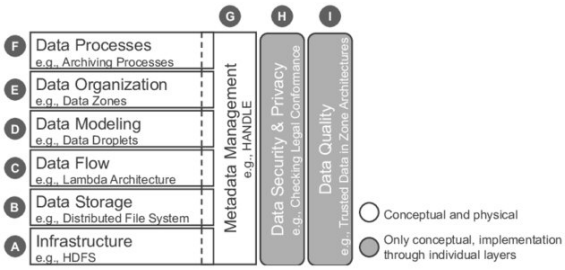


Figure 3.5: Aspects of the Data Lake Architecture Framework [25]

In our work we concentrate on seven aspects out of the nine proposed, as Data Security & Privacy and Data Quality are out of the scope of the thesis.

The first aspect to analyze is the **infrastructure** which considers all concepts related to the physical creation of the Data Lake: both storage systems and needed tools are determined, also the choice between on-premise and the cloud is part of this activity. In the healthcare case we have chosen the following storage systems:

- Hadoop Distributed File System for the raw images
- MySQL for the features extracted from images and for other information such as patients generalities, exams types and case studies.
- The Neo4j graph-based system for metadata management.

For what concerns the tools, we chose:

- MATLAB for image preparation.
- SimpleITK for image transformation

- Pyradiomics for features extraction.

For what concerns the choice between implementing the system on-premise or implementing it on the cloud, the cloud has been selected.

The second aspect is **data storage** which analyzes the types of tools and system used for the storage and processing of the data. It is important to highlight that, in contrast to the infrastructure aspect, in this case no specific tools are selected, but only the categories, e.g. file system, relational database, NoSQL database. So, in the analyzed use case a file system is used to store medical images, relational databases are used for extracted features and data about patients, exams and cases and a NoSQL database is used for metadata management.

The third aspect to be analyzed is **data flow**. In general, it focuses on the architecture and interaction to support the two ways in which data moves inside the Data Lake: streaming data and batch data. As already mentioned, in this work we consider data that comes in mini-batches.

The fourth aspect is **data modeling**. It analyzes how the different types of data are modeled inside the Data Lake. The goal of this step is to explain the types of data present in the Data Lake, highlight the relationships between data, illustate how these data can be grouped, and organize attributes and formats. Basically, data modeling concerns the representation of the structure and the content of the data, it analyzes how to store and access them. In this case images are kept in their raw format, while features, information about patients, exams and cases, are modeled using the relation model.

The fifth aspect to consider is **data organization**. This aspect describes the conceptual structure of the Data Lake. This is a crucial step as it strongly affects the other aspects: it infleunces the data modeling because, if for example a zone architecture is chosen, usually the zones have standardized data models, meaning that all data in the zone are modeled following specific rules; it also influences the data storage aspect, as each data model requires a different type of storage. As already explained the chosen organization is a four-zone architecture.

The sixth aspect is **data processes**. It considers all concepts related to data movement and data processing. In [25], data processes are classified in the two categories of processes for data lifecycle management and processes for data pipelining. Data lifecycle

management processes handle data from the creation to its release. Data pipelining processes, instead, focus on ingestion, processing and movement of data. An example of data pipelining processes is the Extract-Transform-Load process. In the examined use case, data processes are data pipeling processes used to describe how the data moves from one zone to another. In particular, the type of these processes is Extract-Load-Transform, as data is transformed after being ingested in the Data Lake.

The seventh and last aspect analyzed in this work is **metadata management**. As already argued in previous sections, this is a critical aspect as a well designed metadata management will prevent the Data Lake from becaming a data swamp. [25] divides this aspect into two sub-aspects: *metadata as enabler* and *metadata as a feature*, the first one defined as the metadata that describes, for example, the zone of data or when data was created. The latter, instead, describes the additional functionalities that metadata can provide, for example semantics. This classification however does not fit the healthcare use case as these two categories are not specific enough for the metadata that we have identified, so another classification for metadata has been considered [14]:

- *metadata on input dataset* describes the characteristics of the entire dataset, e.g. number of instances, number of attributes, dimension of the dataset.
- *Lineage metadata* can be seen as metadata that describes the steps used to derive a specific piece of data
- *Definitional metadata* is that metadata related to the meaning of data. Taxonomies, integration schemas and vocabularies belong to this category.

# 4 | Design of the application

In this chapter we explain the technologies proposed in Section 3.3, and give a better description of the data flow. In particular, Section 4.1 provides a brief explanation of the format of medical images. Section 4.2 presents the technologies proposed for the use case, Section 4.3 explains the data flow of the Data Lake and gives a brief overview of the dataset used for feature extraction.

## 4.1. DICOM

As already mentioned in Section 2.6.2 medical images are usually in DICOM format. The abstraction used by DICOM to model information and services is based on the object-oriented paradigm. The first concept is the concept of Information Object Definition (IOD) which specifies information related to real world objects. It represents a class of objects that share the same characteristics. Each IOD is composed by a set of entities, e.g. patient, image, acquisition system. Despite the multitude of available entities, the base entities of the DICOM model are four:

- Patient, includes personal data of the patient, e.g. name, birth date, sex.
- Case, comprises the characteristics and the modality of the exam, e.g. description, comments, date.
- Series, each exam is described by a collection of images, e.g. number, date, time, operator's name. A series is associated with exactly one case.
- Image, attributes of the pixels that compose the image, e.g. number, type, patient orientation.

Each IOD can be identified by a Unique IDentifier (UID) which is a unique code. The IOD is formed by a set of data elements, and each data element is composed by:

- a tag which identifies the attribute.
- The Value Representation describes the type and the value of the attribute. It is

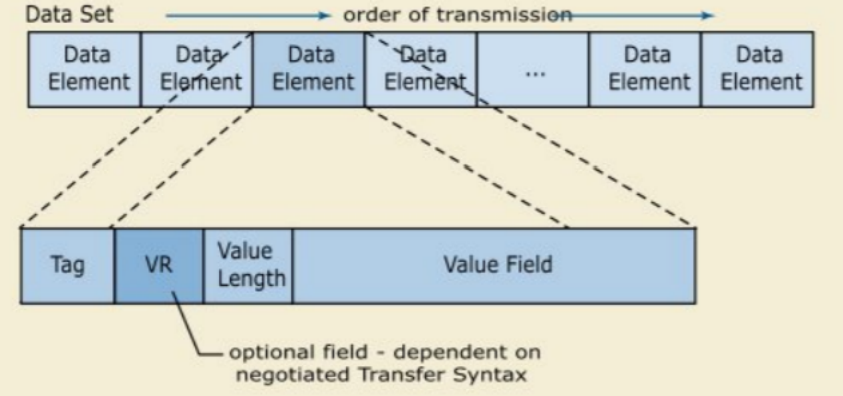


Figure 4.1: Structure of a Data Element [9]

represented by a code composed of two caharacters, e.g. US for "Unsigned Short", CS for "Coded String".

- Value length is the number of bytes needed to represent the value.
- Value field is the actual value of the data element.

When a diagnostic exam is performed, a series of bi-dimensional representations are produced; these correspond to the transversal sections of the anatomic region scanned by the acquisition machine. Each DICOM file is composed by one or more images and information concerning the patient, the exam, the image dimensions etc. All this information is structured following the abstarctions described above. The file format has also an header that precedes the data. This header is composed by a 128 byte preamble followed by a 4-byte prefix. The scope of the header is to facilitate access to both images and other data of the DICOM file.

## 4.2. Technologies

### 4.2.1. Hadoop Distributed File System

Hadoop Distributed File System (HDFS) is a file system developed to guarantee reliability and scalability [36]. It can handle both structured and unstructured data and manage huge volumes. The idea behind HDFS is to consider hardware failures as usual events and not exceptions. Depending on the needs, an HDFS instance can be composed by thousands of nodes and, therefore, it is impossible to think that at a given time instant there isn't an available node. Its most important characteristic is the portability across heterogeneous software and hardware platforms. Another main advantage is that it helps to reduce network congestion and improves system performance as it moves computation

near the storage.

The architecture of the file system is a master-slave architecture. In each cluster there is a single master, called NameNode and a variable number of slaves which are called DataNode. The NameNode handles access requests to the files. It is responsible for opening, closing and renaming of directories. DataNodes manage creation, deletion and replication of data. DataNodes are coordinated by the NameNode.

### 4.2.2. Pyradiomics

Pyradiomics is an open-source Python package used for the extraction of feature from medical images [7]. It supports feature extraction for both 2D and 3D images and it allows to compute single values for a ROI or to generate features maps. It currently supports seven classes of features:

- first-order statistics describe the distribution of pixel intesities in the ROI through commonly basic metrics.
- Shape: this group of features contains descriptors of the 2D or 3D size and shape of the ROI. It does not depend on the gray-level intensity distribution of the ROI.
- Grey Level Co-occurrence Matrix (GLCM) is a matrix of size $N_g$x$N_g$ where $N_g$ is the number of discrete intensity levels in the image. The GLCM functions characterize the texture of an image by calculating how often pairs of pixel with specific values and in a specified spatial relationship occur in an image.
- Gray Level Run Length Matrix (GLRLM) represents gray level runs. Gray level runs can be defined as the number of consecutive pixel that have the same gray level value.
- Gray Level Size Zone (GLSZM) describes grey level zones of an image. A gray level zone represents the number of connected pixels that share the same gray level intensity.
- Grey Level Dependence Matrix (GLDM) computes gray level dependencies in an image.
- Neighbouring Gray Tone Difference Matrix (NGTDM) asseses the difference between a gray value and the average gray value of its neighbours within a given distance.

The extracted features are returned in an ordered dictionary. In addition to the features

the result also contains some additional information about the extraction such as the version of Pyradiomics, original image spacing etc.

Moreover, Pyradiomics is implemented in a modular way. The *Featureextractor* module is the central one and provides the extraction pipeline as well as it handles interaction with other modules. The *feature classes* are, instead, implemented in separate modules.

### 4.2.3. SimpleITK

SimpleITK is a programming interface to the algorithms and data structures of the Insight Toolkit (ITK). It can be integrated with multiple programming languages among which Python and, therefore, it can be used in conjunction with Pyradiomics. In SimpleITK images are multi-dimensional and can be labelmap, scalar or complex value. The region in physical space occupied by the image is defined by:

- *Origin*, indicates the location in the world coordinate system of the voxel with all zeros indexes.
- *Spacing* is the distance between pixels along all dimensions.
- *Size* represents the number of pixels for each dimension.
- *Direction cosine matrix* descibes the direction of the axes corresponding to the matrix column.

In the examined use case, SimpleITK is used to transform the DICOM images into the Nearly Raw Raster Data (NRRD) format, as Pyradiomics does not take DICOM format as input.

## 4.3. Data flow

As already explained in Chapter 3 the data flow analyzes the way in which data moves inside the Data Lake. Data flow describes the path that system's information take from external sources through processes and data stores. The Data Lake ingests data, irrespective of its format, into a big data store [46]. Metadata is decoupled from its data and stored independently. Then, data goes through a series of steps that can have either the function of processing or that of storage.

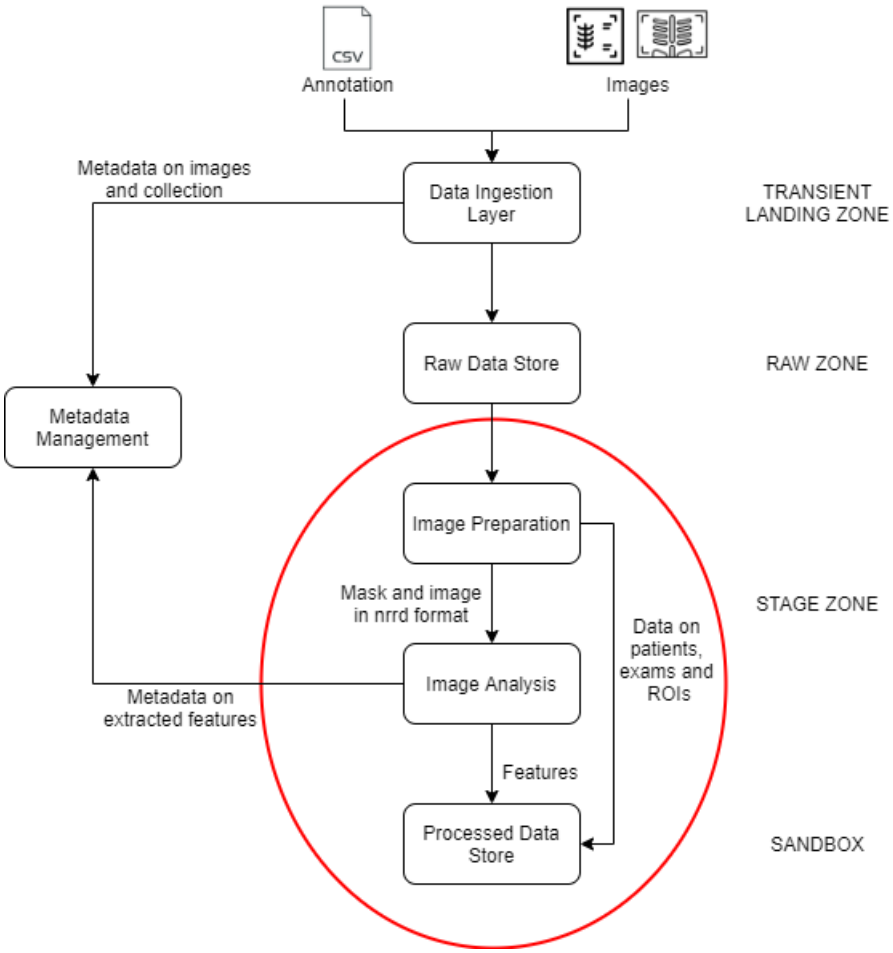


Figure 4.2: Example of data flow

The schema in Fig.4.2 represents the data flow of the use case analyzed in this thesis. The input is a set of images in DICOM format along with related set of annotations in Comma-Separated Values (CSV) format. The annotations define the ROI of the images and have previoulsy been manually defined by radiologists. As already said, the ingested data lands in the transient landing zone. In this step two important actions are performed, the masking of PII and the generation of metadata on both single images and image collection. Then, data is permanently stored in the Raw Data Store.

After ingestion and storage there are all the steps that process the image and the related data. In Fig.4.2, this part is highlighted as it represents the steps that are explained in the remaining part of the chapter. Also, we have developed a simple implementation of these stages to better understand which are the inputs and the outputs of these phases and, also, to make an assessment of resource requirements.

### 4.3.1. Dataset

To perform this simple analysis 80 images were taken from the ChestX-ray8 database [47]. The images contained in this dataset are chest x-rays. This because this type of exam is one of the most commonly accessible radiological examinations for screening and diagnosis of many lung diseases.

The ChestX-ray8 database is composed by 108 948 images which are frontal-view x-ray. These images were taken from a totality of 32 717 patients. Typically the dimension of an x-ray image are 3000x2000, but these dimensions are a challenge for hardware computing capacity. Therefore, in ChestX-ray8 images are extracted from the DICOM file and resized as 1024x1024 bitmap images. This process does not result in losing of detail contents because intensity ranges are rescaled using window settings stored in the DICOM header file.

As part of the ChestX-ray8 database, a small number of images with pathology are provided with hand labeled bounding boxes. In the labeling process only 983 images were analyzed. For each one of these images a board-certified radiologist identified the region interested by a disease. The identified ROI has then be registered in an eXtensible Markup Language (XML) file. If an image has more than one ROI, each one of them is stored in a different XML file. As previously mentioned these ROIs are provided in a CSV file.

In the thesis only images that have at least one ROI were used.

### 4.3.2. Image preparation

The first step of this phase is the extraction of information about the patient, the case and the exam from the DICOM files. This data is stored in a relational table using MySQL. More in detail, the information extracted in this phase are:

- Data about the image
  - width
  - height
  - modality, e.g. sx, dx, frontal
- Data about the patient
  - patientID

  - age
  - sex
- Data about the exam
  - body part
  - exam description

The second step is the extraction of data concerning the ROI from the CSV file and its storage in a relational table. In our case the ROI identifies the region from which it is possible to identify a chest disease, e.g. pneuomonia, pneumothorax, atelectasis. For each ROI the following attributes are entered:

- the reference to the image
- the x-coordinate of the origin
- the y-coordinate of the origin
- width
- height

The third step of this phase is the creation of the mask. A mask is a binary image formed by zero and non-zero values. When the mask is applied to a grayscale image, the pixels that have a zero value in the mask are set to zero also in the output image. The other pixels, instead, are unchanged. So, the mask allows to consider only the ROI of the image. This task is accomplished using MATLAB which provides a function to generate the mask:

$$roipoly(I, c, r); \tag{4.1}$$

This function allows to define a polygonal region of interest on the image and it returns a binary image that represents the mask. In function (4.1), I represents the input image, c is the vector of the x-coordinates of the vertices of the ROI and r is the vector of the y-coordinates of the vertices of the ROI.

The fourth, and last, step is image transformation using SimpleITK. In this step both the DICOM image and the mask, which is in Joint Photographic Experts Group (JPEG) format, are transformed into NRRD format. This transformation is needed as Pyradiomics does not accept neither DICOM nor JPEG formats.

It is important to highlight that images obtained in this step are not permanently stored in the Data Lake, i.e. in the Raw Zone. This decision was made for two main reasons, first when an image is reprocessed it is unlikely that the reprocessing is performed on the same image, usually some transformations are applied. The second reason is that storing the produced image can be more expensive than re-run the process that has generated it.

### 4.3.3. Image analysis

At this stage the features are extracted from images. This analysis is performed using Pyradiomics. It takes as input the image and the mask both in NRRD format and returnes an ordered dictionary as output. In the simple implementation developed for this we computed the following classes of features: first order, shape, GLCM, GLRLM, GLSZM, GLDM and NGTDM. The different modules for the extraction of the mentioned classes of features are executed one at the time to facilitate the insertion of features in the relational database. Also in this stage metadata is generated, for example the version of Pyradiomics, the original image spacing etc.

### 4.3.4. Processed data store

This stage provides the storage for both data about patients, exams, ROIs and the features extracted from the images. These information are organized following a relational model. We have choosen this type of model as data has a rigid structure and NULL values are infrequent. This part has been implemented using MySQL.

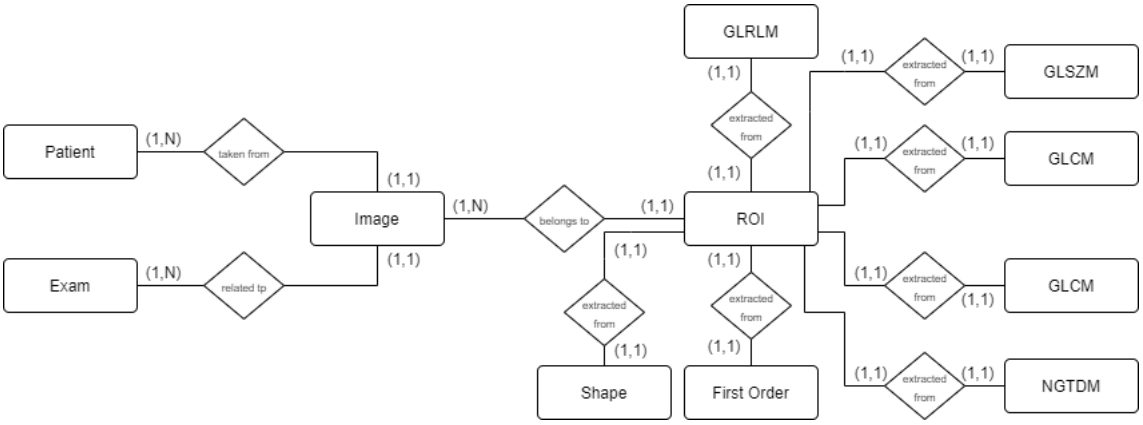


Figure 4.3: Entity-Relation diagram

Summing up, each image is related to a patient and is taken for a specific exam. Then, an image has one or more ROI associated to it. On the ROIs feature extraction algorithms are applied and the aforementioned features are extracted: first order, shape, GLCM,

GLRLM, GLSZM, GLDM and NGTDM. In Fig.4.3 the attributes on the entities are not shown as the high number of attributes would have made the image not readable.

## 4.4. Performance monitoring

As already mentioned, the implementation of the steps of image preparation and features extraction has allowed us to quantify time of execution, resource usage and the amount of data generated. This represents a big advantage as these numbers helped us understand the requirements that the architecture should satisfy. The tests have been executed using the 80 chest x-rays taken from the ChestX-ray8 database and their ROIs.

The step of image preparation was implemented in MATLAB and it creates the masks of the images. The function that reads the ROI from the CSV file, opens the image in DICOM format and creates the mask, has been executed 10 times for all the 80 images and the execution time has been recorded, as shown in Table 4.1. This step takes on average 3,766s, which means that the time needed for the creation of a single mask is on average 47,07ms. For what concerns Central Processing Unit (CPU) utilization, instead, the average for this step is 43,3%. Another important aspect is that this code does not take advantage of the Graphics Processing Unit (GPU).

| | **Execution time** |
|---|---|
| 1 | 3,595s |
| 2 | 4,026s |
| 3 | 3,785s |
| 4 | 3,474s |
| 5 | 4,011s |
| 6 | 3,545s |
| 7 | 3,762s |
| 8 | 3,848s |
| 9 | 3,595s |
| 10 | 4,018s |

Table 4.1: Execution times of image preparation step.

The following step, image analysis, is instead implemented in Pyradiomics. In this phase features are extracted from medical images. Taking always in consideration the 80 chest x-rays, a total of 10 320 features are extracted which means that for each image 129

features are computed. More in detail, the number of features extracted for each class is:

- first order: 19 features
- shape: 10 features
- GLCM: 24 features
- GLRLM: 16 features
- GLSZM: 16 features
- GLDM: 14 features
- NGTDM: 5 features

The remaining 35 features represent metadata such as the version of Pradiomics, the version of Numpy, image original dimensionality and many others.

For what concerns the execution time we have taken advantage of the function offered by Python, *timeit*. This function registers the execution time of a small code snippet and it takes as inputs the code snippet we want to time and the number of times we want to execute the specified code. We have called the function three times passing as *number* parameter: 1, 10 and 100. The results are presented in Table 4.2.

| **Number of times** | **Execution time** |
|---|---|
| 1 | 15,805s |
| 10 | 159,379s |
| 100 | 1 691,373s |

Table 4.2: Execution times of image analysis step.

From execution times presented in Table 4.2, we have estimated that it takes, on average, 202,67ms to extract features from a single image. CPU utilization in this case is 14,4% and it uses also the GPU with a percentage of 4,4%. During the execution of Pyradiomics we were also able to measure memory utilization, more in detail we have executed the code for the 80 images 10 times and register for each execution the peak memory usage. On average the peak is 14,303 MB.

Both the step were run on a PC with 2,90 GHz dual processors CPU.

### 4.4.1. Observations

The first thing to observe from collected data is that the number of features extracted from an image is always constant. This rigid structure justifies the choice of using a relational database to store features.

Looking at CPU utilization, the values fit in the range of normal CPU usage. There is, however, a pretty high difference between the value registered in the image preparation phase and the value registered in the image analysis step. This difference is probably due to the fact that Pyradiomics uses also the GPU while MATLAB does not.

Also for what concerns the execution times there is a quite high difference between the step of mask creation and the step of features extraction. Even though the time needed for processing an image in Pyradiomics is not really high, when feature extraction is performed on a large number of images execution time can reach pretty high values. This is the case, for example, of running the code 100 times for the 80 images that has taken about 28 minutes. So, this phase could probably benefit from a more powerful system.

# 5 | Cloud Solutions

As already mentioned in Chapter 2 cloud computing is a big support for handling Big Data. The market offers many commercial solutions to govern the Data Lake, basically all the big computer companies provide some services that help in the implementation of the Data Lake in the cloud. In fact, for the use case of this thesis, two solutions have been identified as possible implementation of medical image storing in the cloud. The first one, explained in section 5.1, takes advantage of the services offered by Amazon. The second one, presented in section 5.2, uses services provided by Microsoft.

## 5.1. Amazon

Amazon provides a series of cloud computing services known as Amazon Web Services (AWS). AWS is a platform of web services that offers solutions for computing, storing, and networking at different levels of abstarction [49]. These web services are accessible via the Internet and can be used by both machines and humans through a User Interface. Of course, these services are designed to work toghether, in this way a user can replicate its existing local network setup or can design a new architecture from scrap. The cost model applied by Amazon for these services is pay per use.

In [20] an architecture that uses AWS to apply Artificial Intelligence alghoritms to medical images is presented. Some modifications have been applied to the mentioned architecture to accomodate the needs of the analyzed use case. The overall solution is presented in Fig. 5.1

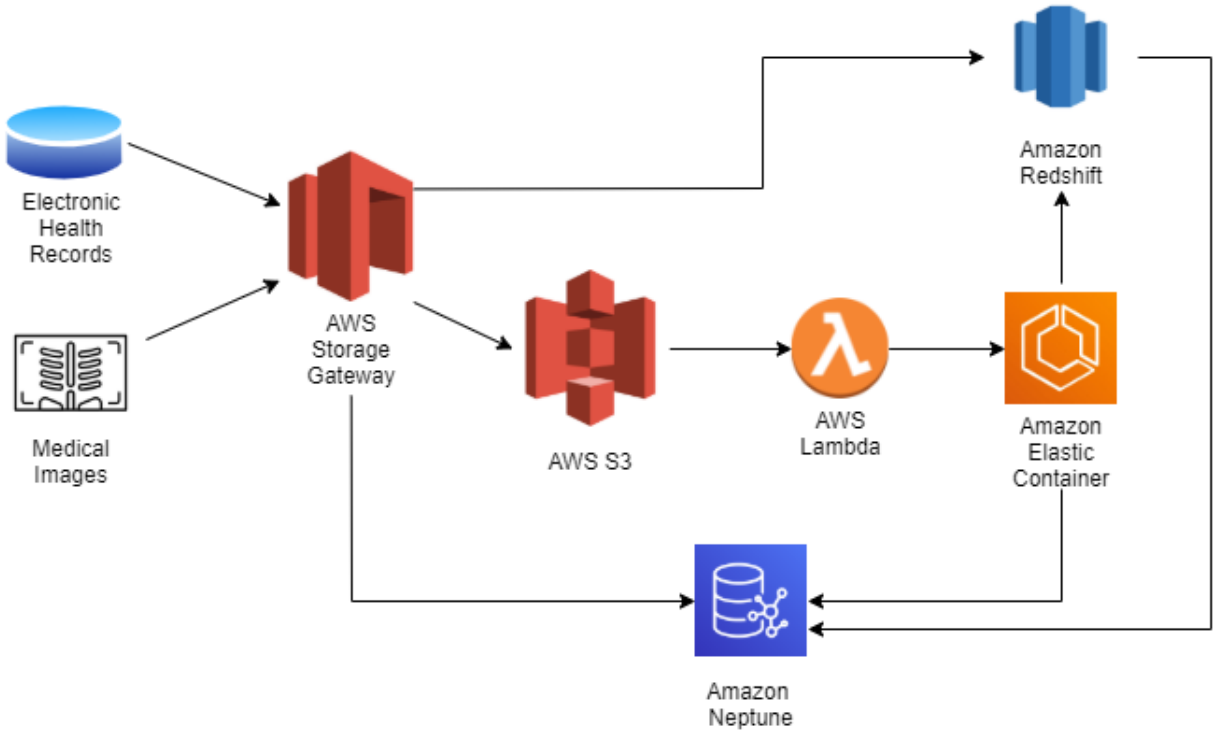


Figure 5.1: Cloud solution that uses services provided by AWS

The components used in Figure 5.1 are:

- **AWS Storage Gateway** is a hybrid storage service that enables on-premises applications to seamlessly use AWS cloud storage [32]. Applications connect to this service through a virtual machine or hardware gateway appliance using storage protocols. The gateway then connects to the storage services. AWS Storage Gateway provides an optimized data transfer mechanism with bandwidth management, automated network resilience and efficient data transfer.
- **AWS S3** is a secure, durable, and extremely low-cost storage service for data archiving and long-term backup [32]. It is an object storage and it provides management features so that access to data can be optimized, organized, and configured in order to meet specific business, organizational, and compliance requirements. Amazon S3 offers multiple storage classes designed for different use cases. For example, for frequently accessed data it offers S3 standard, for infrequently accessed data S3 One Zone-IA is available and to store data at the lowest costs there is S3 Glacier.
- **AWS Lambda** is a serverless computing service that allows you to run code without provisioning or managing servers, create a cluster sizing logic based on workloads, maintain event integrations or manage runtime environments [6]. It executes code

for any type of application without having to worry about administration management. The user simply uploads the code and Lambda automatically assigns the processing power and executes the code according to the request or incoming events for any traffic scale.

- **Amazon Elastic Container Service** is a fully managed container orchestration service that helps you easily deploy, manage, and scale containerized applications [2]. With Amazon Elastic Container Service there is no more the need to install the container orchestration software, scale clusters or schedule execution
- **Amazon Redshift** is a fully managed, petabyte-scale data warehouse service in the cloud [8]. It allows to execute queries that come from or are directed to the data lake. With Amazon Redshift, it is possible to easily instantiate a cluster of nodes on which data is uploaded. Other important features are that it offers fast query performance and it executes query written in SQL.
- **Amazon Neptune** is a fast, reliable, fully-managed graph database service that makes it easy to build and run applications that work with highly connected datasets [32]. It offers a high performance graph database engine that can effectively manage billions of relationships and allows to query the graph with very low latency. Amazon Neptune is fully-managed, in this way the user no longer has to worry about database management tasks as hardware provisioning, software patching, setup, configuration, or backups.

The solution proposed in Fig.5.1 takes into consideration both medical images and Electonic Health Records for completeness, but the main focus is on the processing and storage of the images.

Obviously, there is a binding between the cloud solution and the architecture proposed in Chapter 3. The Transient Landing Zone is represented by AWS Storage Gateway, the Raw Zone corresponds to AWS S3, the Stage Zone can be mapped to Amazon Elastic Container, the Sandbox Zone corresponds to Amazon Redshift and, then, Metadata Management is handled by Amazon Neptune. The AWS Lambda, instead, has the role of triggering the image analysis algorithms when a specific number of new images are availble in AWS S3. It is also important to underline that the solution was designed basing on the choices made in section 3.3 where the tools and technologies used in the Data Lake have been selected.

## 5.2. Microsoft

Cloud services offered by Microsoft can be found on the Azure platform. Basically, Azure is a public cloud computing with solutions including Software as a Service, Platform as a Service and Infrastructure as a Service. These services are used for analytics, virtual computing, storage, networking and many others. To support cloud applications and data, Windows Azure has five types of components [16]. The first type of components are those used for storage that allow to store binary and structured data in the cloud. Then, there are computing components which run application in the cloud. The third category is fabric controller that deploys, manages, and monitors applications, it also handles updates to system software throughout the platform. The fourth category is the Content Delivery Network which speeds up global access to binary data in Windows Azure storage, this is possible because Azure maintains cached copies of that data around the world. Lastly, there are connect components that allow to create connections between on-premises computers and Windows Azure applications.

The proposed architecture is a slightly modified version of the solution proposed in [1].

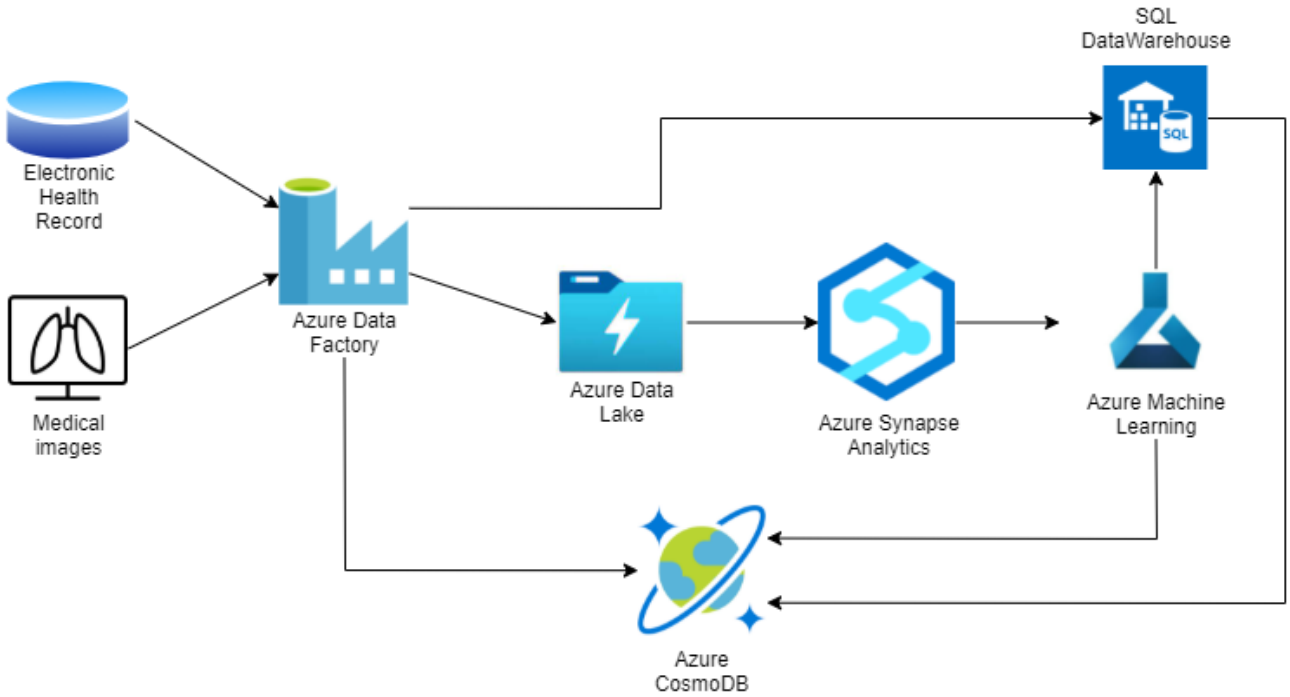


Figure 5.2: Cloud solution that uses services provided by Azure

The components used in Figure 5.2 are:

- **Azure Data Factory** is a cloud service for ETL or ELT processes and for data integration processes. It is composed by a series of interconnected systems that provide an end-to-end platform. With Azure Data Factory it is possible to create and manage workflows based on data that can come from different archiving systems.
- **Azure Data Lake** is a set of capabilities dedicated to big data analytics [11]. It provides a low-cost, tiered storage with high availability and, also, disaster recovery capabilities. It is designed to service multiple petabytes of information while sustaining hundreds of gigabits of throughput. An important aspect of Azure Data Lake is the hierarchical namespace, it allows to organize objects and files into a hierarchy of directories that makes data access more efficient. In this way, operations such as renaming or deleting of a directory, become single atomic metadata operations.
- **Azure Synapse Analytics** is an analysis service that accelerates the time needed to get detailed information about data warehouses and big data systems. It is composed by the best technologies for integration data pipelines and for ELT and ETL processes. It also allows to explore, prepare, manage and distibute data for machine learning algorithms.
- **Azure Machine Learning** provides an interactive visual workspace to easily build, test and iterate predictive analysis models [34]. It does not require any programming, algorithms can be built by visually connecting datasets and modules. However, if a needed functionality is not availble, it can be implemented in Azure Machine Learning by writing Python or R code.
- **SQL DataWarehouse** is a cloud based, scale-out database capable of processing massive volumes of data, both relational and nonrelational [19]. A big advantage of SQL DataWarehouse is that it supports many different programming languages, tools and frameworks, including non-Microsoft software. It is based on SQL Server relational database engine and it easily integrates with the other tools that compose the system. It is composed by a Control node, different Compute nodes, Storage and a service called Data Movement Service which manages the data movement between the nodes. The Control node manages and optimizes queries and coordinates data movement and computation required to run parallel queries. The Compute nodes are, instead, SQL databases that store the data and process queries. So, when new data is inserted, it is distributed to these nodes and when it is requested these nodes run queries in parallel. After processing, each compute node returns its results to

the control node that aggregates these results and produce the final result.

- **Azure CosmoDB** can be described as a database service based on multiple models, with transparent scaling features that replicate organizational data. It is a NoSQL database with really low latency and automatic scalability. As a fully managed service, with Azure Cosmos DB users no longer have to worry about database administration tasks, the service itself manages all the tasks such as the updating and patching of applications. Also, it handles capacity management with cost-effective automatic scalability and serverless options that meet application needs in terms of capacity.

As in the case of the solution using AWS, also in this case there is a mapping between the cloud solution and the architecture proposed in Chapter 3. The Transient Landing Zone is represented by Azure Data Factory, the Raw Zone corresponds to Azure Data Lake, the Stage Zone can be mapped to both Azure Synapse Analytics and Azure Machine Learning, the Sandbox Zone corresponds to SQL DataWarehouse and then Metadata Management is handled by Azure CosmoDB. In particular, Azure Synapse can take unstrctured data, such as medical images, and can feed these data into machine learning algorithms.

# 6 | Conclusions

The work presented in this thesis is at an early stage of the process of the implementation of a storage system for medical data. The main objective of this work is to design an architecture for a Data Lake that allows efficient storing and fast access to all types of data. In particular, the main focus was on storage of unstructured data, i.e. medical images, which is problematic in traditional systems.

The first step of the process has been the analysis of the types and formats of the system's input. More in detail, the scope of this phase was to understand the various types of medical images, how they are structured and what type of analysis is made on this data.

Once these aspects were clear, the next step has been the design of an architecture, as a reference architecture does not yet exist for Data Lakes. First of all, it has been selected a zone architecture because it better satisfies the needs of this specific use case. Then, the components needed for the various stages of data processing and storage have been identified. Once the list of components was complete, the connections between these components have been analyzed. During this phase particular attention has been paid to the metadata management system as its design is crucial for the correct functioning of the whole system. In particular, first the various types of data have been re-analyzed to understand what kind of metadata could be generated and then a solution to model this metadata has been searched. Among the different metadata management systems offered by the literature, HANDLE [18] has been identified as the one that best fits the use case under exam. So, the model proposed in [18] has been studied and small modifications have been applied to adapt it to the healthcare use case.

The following step was to analyze the different aspects that characterize the Data Lake. This is an important phase in the design process because it defines all the features that should be studied in order to develop a complete Data Lake. In this stage the main tools and technology for processing and storage of data were picked.

The next step was to better understand the data flow inside the Data Lake. For this scope the specific case of images was analyzed. More in detail, it has been considered the case in which the inputs are represented by a set of medical images and a CSV file that contains the ROIs of the images. To better understand the steps of the data flow, a simple implementation of the main phases was developed. This demo takes as input 80 x-rays of the chest and their ROIs. Then, it extracts from the DICOM file information related to patients, exams and cases and from the CSV the ROIs. Data extracted in this phase are stored in relational tables. After that, MATLAB was used to create the masks of the images based on the ROIs. Next, both images and masks are converted using SimpleITK into NRRD file as the following step requires this type of data. Once x-rays and their masks are in NRRD format, they can be passed to the module that extracts features from images which was implemented using Pyradiomics. The features extracted by Pyradiomics are also stored in relational tables.

The final step of this project was to identify cloud solutions that can be used to implement this system. Among all cloud providers, two have been identified as the ones that offer services that best fit the healthcare case, Amazon and Microsoft. Therefore, researches have been carried out two identify for both provider the set of services that could be used to develop the Data Lake and two solutions have been presented.

In summary, the Data Lake has been identified as the best solution for the problem under exam. A high level view of the architecture of the system has been proposed and the requirements that this architecture should staisfy have been identified. Lastly, two posible cloud solutions have been described.

## 6.1. Future work

Taking into account the performed work, the main next step is the complete implementation of the system. Ideally this implementation will take advantage of cloud services and, therefore, it will realize one of the solutions proposed in Chapter 5. The most interesting part of the implementation will be certainly the metadata management system. For this part of the system more detailed requirements should be identified by analyzing the different types of data entering the Data Lake and their relationships.

Another important step that sould be made is the analysis of the aspects of data security and data quality. Data security is of great importance in the Data Lake, as it ensures legal conformance, alignment with business objectives, and much more [25]. Data

quality, instead, is important to ensure the data's usability and prevent the data lake from turning into a data swamp [25]. When these two aspects are included, also their metadata should be analyzed, e.g. security and quality classification. However, these aspects are conceptual aspects and many tools are already available to manage them.

Another important step that should be done is to examine how the process of masking PII can be implemented. This process has to be deeply analyzed as the exposure of sensitive data can cause serious damage. It has been selected the masking technique as it can shuffle data columns in different ways so that the masked data looks like the original, in the format and type, but it is no longer sensitive data. Masking is effective because despite it changes all the individual data elements, it still allows to compute aggregate values across an entire database, enabling preservation of the right values within a dataset.

# Bibliography

[1] Approfondimenti clinici con microsoft cloud for healthcare. URL https://docs.microsoft.com/it-it/azure/architecture/example-scenario/mch-health/medical-data-insights.

[2] Amazon elastic container service. URL https://www.amazonaws.cn/en/ecs/.

[3] Gartner glossary - big data, . URL https://www.gartner.com/en/information-technology/glossary/big-data.

[4] Gartner glossary - data warehouse, . URL https://www.gartner.com/en/information-technology/glossary/data-warehouse.

[5] The property graph model of graph databases. URL https://subscription.packtpub.com/book/big_data_and_business_intelligence/9781786466143/1/ch01lvl1sec12/the-property-graph-model-of-graph-databases.

[6] Aws lambda. URL https://aws.amazon.com/it/lambda/.

[7] Welcome to pyradiomics documentation! URL https://pyradiomics.readthedocs.io/en/latest/.

[8] Caratteristiche di amazon redshift. URL https://aws.amazon.com/it/redshift/features/?nc=sn&loc=2&dn=1.

[9] 7 - the data set, 2013. URL http://dicom.nema.org/dicom/2013/output/chtml/part05/chapter_7.html.

[10] Getting rescued from the data swamp, July 2019. URL https://www.getlore.io/blog/getting-rescued-from-the-data-swamp.

[11] Introduction to azure data lake storage gen2, 2021. URL https://docs.microsoft.com/en-us/azure/storage/blobs/data-lake-storage-introduction.

[12] R. Agrawal and S. Prabakaran. Big data in digital healthcare: lessons learnt and recommendations for general practice. *Heredity*, 124(4):525–534, 2020.

[13] W. D. Bidgood Jr, S. C. Horii, F. W. Prior, and D. E. Van Syckle. Understanding and using dicom, the data interchange standard for biomedical imaging. *Journal of the American Medical Informatics Association*, 4(3):199–212, 1997.

[14] B. Bilalli, A. Abelló, T. Aluja-Banet, and R. Wrembel. Towards intelligent data analysis: The metadata challenge. In *IoTBD*, pages 331–338, 2016.

[15] Brandyli. Data warehouse: Facts, dimensions, and star schema, 2020. URL `https://medium.com/@brandyli1103/data-warehouse-facts-dimensions-and-star-schema-6181c64ae51a`.

[16] D. Chappell. Introducing windows azure. Technical report, David Chappell and Associates, 2010. URL `https://www.idt-inc.com/wp-content/uploads/sites/4755/2017/05/IntroducingWindowsAzureFinal_5_5_2011_9_55_46_AM.pdf`.

[17] A. Datta and H. Thomas. The cube data model: a conceptual model and algebra for on-line analytical processing in data warehouses1a related paper introducing this model was presented at the workshop on information and technology (wits), atlanta, ga, december 1997.1. *Decision Support Systems*, 27(3):289–301, 1999. ISSN 0167-9236. doi: https://doi.org/10.1016/S0167-9236(99)00052-4. URL `https://www.sciencedirect.com/science/article/pii/S0167923699000524`.

[18] R. Eichler, C. Giebler, C. Gröger, H. Schwarz, and B. Mitschang. Handle - a generic metadata model for data lakes. In M. Song, I.-Y. Song, G. Kotsis, A. M. Tjoa, and I. Khalil, editors, *Big Data Analytics and Knowledge Discovery*, pages 73–88, Cham, 2020. Springer International Publishing. ISBN 978-3-030-59065-9.

[19] P. J. Ferreira, A. de Almeida, and J. Bernardino. Data warehousing in the cloud: Amazon redshift vs microsoft azure sql. In *KDIR*, 2017.

[20] G. Fu, U. Ratan, and E. Bas. Building a medical image search platform on aws, Oct 2020. URL `https://aws.amazon.com/it/blogs/machine-learning/building-a-medical-image-search-platform-on-aws/`.

[21] S. George. Nosql–not only sql. *International Journal of Enterprise Computing and Business Systems*, 2(2), 2013.

[22] C. Giebler, C. Gröger, E. Hoos, H. Schwarz, and B. Mitschang. Leveraging the Data Lake - Current State and Challenges. In *Proceedings of the 21st International Conference on Big Data Analytics and Knowledge Discovery (DaWaK 2019)*, 2019. doi: 10.1007/978-3-030-27520-4_13.

[23] C. Giebler, C. Gröger, E. Hoos, H. Schwarz, and B. Mitschang. Leveraging the data

lake: Current state and challenges. In C. Ordonez, I.-Y. Song, G. Anderst-Kotsis, A. M. Tjoa, and I. Khalil, editors, *Big Data Analytics and Knowledge Discovery*, pages 179–188, Cham, 2019. Springer International Publishing. ISBN 978-3-030-27520-4.

[24] C. Giebler, C. Gröger, E. Hoos, H. Schwarz, and B. Mitschang. A zone reference model for enterprise-grade data lake management. In *2020 IEEE 24th International Enterprise Distributed Object Computing Conference (EDOC)*, pages 57–66, 2020. doi: 10.1109/EDOC49727.2020.00017.

[25] C. Giebler, H. Schwarz, B. Mitschang, and T. und Web. The data lake architecture framework: A foundation for building a comprehensive data lake architecture. *Datenbanksysteme für Business, Technologie und Web (BTW 2021) 13.–17. September 2021 in Dresden, Deutschland*, page 351, 2021.

[26] R. Hai, S. Geisler, and C. Quix. Constance: An intelligent data lake system. In *Proceedings of the 2016 international conference on management of data*, pages 2097–2100, 2016.

[27] A. Y. Halevy, F. R. Korn, N. Noy, C. Olston, N. Polyzotis, S. Roy, and S. E. Whang. Managing google's data lake: an overview of the goods system. *IEEE Data Eng. Bull.*, 39:5–14, 2016.

[28] B. Inmon. *Data Lake Architecture: Designing the Data Lake and avoiding the garbage dump*. Technics publications, 2016.

[29] B. Inmon. *Data Lake Architecture: Designing the Data Lake and Avoiding the Garbage Dump*. Technics Publications, LLC, Denville, NJ, USA, 1st edition, 2016. ISBN 1634621174.

[30] K. Kline, D. Kline, and B. Hunt. *SQL in a nutshell: a desktop quick reference guide*. " O'Reilly Media, Inc.", 2008.

[31] C. Madera and A. Laurent. The next information architecture evolution: The data lake wave. In *Proceedings of the 8th International Conference on Management of Digital EcoSystems*, MEDES, page 174–180, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450342674. doi: 10.1145/3012071.3012077. URL `https://doi.org/10.1145/3012071.3012077`.

[32] S. Mathew. Overview of amazon web services aws whitepaper. Technical report, Amazon, 2021. URL `https://d1.awsstatic.com/whitepapers/aws-overview.pdf`.

[33] P. M. Mell and T. Grance. Sp 800-145. the nist definition of cloud computing. Technical report, Gaithersburg, MD, USA, 2011.

[34] S. Mund. *Microsoft Azure Machine Learning explore predictive analytics using step-by-step tutorials and build models to make prediction in a jiffy with a few mouse clicks*. Packt Publ, Birmingham-Mumbai, June 2015. ISBN 978-1-78439-079-2.

[35] T. Nasser and R. Tariq. Big data challenges. *J Comput Eng Inf Technol 4: 3. doi: http://dx. doi. org/10.4172/2324*, 9307(2), 2015.

[36] A. Oussous, F.-Z. Benjelloun, A. A. Lahcen, and S. Belfkih. Big data technologies: A survey. *Journal of King Saud University-Computer and Information Sciences*, 30 (4):431–448, 2018.

[37] C. Quix, R. Hai, and I. Vatov. Gemms: A generic and extensible metadata management system for data lakes. In *CAiSE forum*, volume 129, 2016.

[38] F. Ravat and Y. Zhao. Data lakes: Trends and perspectives. In *International Conference on Database and Expert Systems Applications*, pages 304–313. Springer, 2019.

[39] W. REDMOND. The big bang: How the big data explosion is changing the world, 2013. URL `https://news.microsoft.com/2013/02/11/the-big-bang-how-the-big-data-explosion-is-changing-the-world/`.

[40] I. Robinson, J. Webber, and E. Eifrem. *Graph databases: new opportunities for connected data*. " O'Reilly Media, Inc.", 2015.

[41] B. SADEG and C. DUVALLET. To have an idea on nosql databases. *International Journal of Computer (IJC)*, 35(1):1–18, Sep. 2019. URL `https://ijcjournal.org/index.php/InternationalJournalOfComputer/article/view/1465`.

[42] P. Sawadogo and J. Darmont. On data lake architectures and metadata management. *Journal of Intelligent Information Systems*, 56(1):97–120, Jun 2020. ISSN 1573-7675. doi: 10.1007/s10844-020-00608-7. URL `http://dx.doi.org/10.1007/s10844-020-00608-7`.

[43] P. N. Sawadogo and J. Darmont. On data lake architectures and metadata management. *J. Intell. Inf. Syst.*, 56(1):97–120, 2021. doi: 10.1007/s10844-020-00608-7. URL `https://doi.org/10.1007/s10844-020-00608-7`.

[44] P. N. Sawadogo, É. Scholly, C. Favre, É. Ferey, S. Loudcher, and J. Darmont. Metadata systems for data lakes: Models and features. In T. Welzer, J. Eder, V. Podgorelec, R. Wrembel, M. Ivanović, J. Gamper, M. Morzy, T. Tzouramanis, J. Dar-

mont, and A. Kamišalić Latifić, editors, *New Trends in Databases and Information Systems*, pages 440–451, Cham, 2019. Springer International Publishing. ISBN 978-3-030-30278-8.

[45] U. Sivarajah, M. M. Kamal, Z. Irani, and V. Weerakkody. Critical analysis of big data challenges and analytical methods. *Journal of Business Research*, 70:263–286, 2017. ISSN 0148-2963. doi: https://doi.org/10.1016/j.jbusres.2016.08.001. URL https://www.sciencedirect.com/science/article/pii/S014829631630488X.

[46] B. Stein and A. Morrison. The enterprise data lake: Better integration and deeper analytics. *PwC Technology Forecast: Rethinking integration*, 1(1-9):18, 2014.

[47] X. Wang, Y. Peng, L. Lu, Z. Lu, M. Bagheri, and R. Summers. Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In *2017 IEEE Conference on Computer Vision and Pattern Recognition(CVPR)*, pages 3462–3471, 2017.

[48] J. S. Ward and A. Barker. Undefined by data: A survey of big data definitions. *ArXiv*, abs/1309.5821, 2013.

[49] A. Wittig and M. Wittig. *Amazon Web Services in Action*. Manning Publications Co., USA, 1st edition, 2015. ISBN 1617292885.

# List of Figures

# List of Tables

# List of Acronyms

**AWS** Amazon Web Services

**CPU** Central Processing Unit

**CSV** Comma-Separated Values

**DICOM** Digital Imaging and Communication in Medicine

**ELT** Extract-Load-Transform

**ETL** Extract-Transform-Load

**GLCM** Grey Level Co-occurrence Matrix

**GLDM** Grey Level Dependence Matrix

**GLRLM** Gray Level Run Length Matrix

**GLSZM** Gray Level Size Zone

**GPU** Graphics Processing Unit

**HANDLE** Handling metAdata maNagement in Data LakEs

**HDFS** Hadoop Distributed File System

**IOD** Information Object Definition

**JPEG** Joint Photographic Experts Group

**NGTDM** Neighbouring Gray Tone Difference Matrix

**NRRD** Nearly Raw Raster Data

**OLAP** On-Line Analytical Processing

**OLTP** On-Line Transaction Processing

**PACS** Picture Archiving and Communication Systems

**PII** Personally Identifiable Information

**ROI** Region Of Interest

**UID** Unique IDentifier

# Ringraziamenti

Giunta alla fine del mio percorso di studi, vorrei ringraziare tutte le persone che mi sono state accanto e mi hanno supportato durante questi anni e, soprattutto, nell'ultimo periodo.

Inizio, quindi, ringraziando la Prof.ssa Letizia Tanca e il Prof. Marco Gribaudo per la disponibilità, per il supporto e per i preziosi consigli che mi hanno fornito durante questi mesi di lavoro.

Vorrei proseguire ringraziando i miei fedeli compagni del Poli: Fede, Gio, Michi, Rullo, Manu, Jack, Fra e Pigo. Senza di voi questo percorso non sarebbe stato lo stesso, grazie per le innumerevoli partite a carte e gli altrettanti caffè che hanno reso le giornate al Poli un po' meno pesanti. Grazie per i viaggi, gli aperitivi, i sushi, le giornate a Gardaland, le grigliate e tutti gli altri mille momenti che abbiamo condiviso. Grazie per avermi sempre aiutata quando mi sono trovata in difficoltà e per avermi incoraggiata nei momenti in cui pensavo di non potercela fare. Quando ripenserò a questi anni sicuramente il primo pensiero andrà a voi. Vorrei potervi ringraziare singolarmente ma, purtroppo, lo spazio è limitato. Permettetemi, però, di fare un ringraziamento speciale alla Fede che è stata la mia spalla lungo tutto il percorso, nonchè la migliore compagna di partite a scopa.

Non posso poi non ringraziare Sara perchè ormai sono undici anni che ci supportiamo a vicenda. Grazie per avermi sempre sostenuta con tutti i messaggi pre-esame e post-esame.

Vorrei proseguire ringraziando Filippo che mi è sempre stato vicino durante questi anni spronandomi a non mollare mai e regalandomi attimi di spensieratezza con le cene al sushi o al messicano.

Infine il ringraziamento più grande va alla mia famiglia, in particolare ai miei genitori senza i quali non avrei mai potuto intraprendere questo percorso e a mia sorella che ha sempre creduto in me. Grazie per avermi constantemente supportata, per avermi incoraggiata nei momenti di sconforto e per aver condiviso con me la gioia di ogni traguardo raggiunto.