International Hellenic University
Department of Information and Electronic Engineering

«Development of a cloud-based API for retrieving web data»

Thesis paper of
Aristeidis Tsachlaris

Supervisor: Salampasis Michalis, Professor

September 2022

Responsibility Statement: I certify that I am a writer of this thesis and that any help I had to prepare is fully recognized and refers to the thesis. I have also mentioned any sources of which I have used data, ideas or words, whether they refer to exactly either paraphrased. I also confirm that this thesis was prepared by me personally especially for the requirements of the Department of Informatics, Computer and Telecommunications Department of the International University of Greece.

«If there is no struggle, there is no progress»

Abstract

The Theatrical Plays API (TP API) accesses an existing database and exposes the necessary entities and their relations. It uses the Spring Framework using the Kotlin programming language and allows CRUD operations on the entities. Elastic Search is also used to index through specific fields, fetching data more efficiently. The API currently serves the data in three applications. A web-based front end, an Android app and an iOS app. All three applications are their own standalone thesis. I was in charge of coordinating the communication between the people involved in these applications, and I was to meet the needs of each application. The current thesis uses the work produced by three previous theses that extracted information from viva.gr, gathering all the data relating to theatrical plays. Actors, plays and venues were some of the collected entities. The TP API is hosted on the cloud using the AWS Elastic Beanstalk service and the data are stored in AWS Aurora database service.

Acknowledgments

I could not have undertaken this journey without the support of my family. Their continuous encouragement has kept my spirit high all this time. I would also like to extend my sincere thanks to professor Salampasis for his valuable insights.

Contents

Abstract 5
Acknowledgments 6
Contents 7
Table of figures 10
Keywords 11
Chapter 1. Introduction 12
1.1 Thesis overview 12
Chapter 2. Technologies used 14
2.1 Java 14
2.2 Kotlin 15
2.3 Differences of Java and Kotlin 16
2.4 Spring framework 17
2.5 Maven 17
2.6 Elastic Search 18
2.7 Git 18
2.8 Amazon Elastic Beanstalk 18
2.9 HTTP 19
2.10 Rest API 19
2.11 Client-server architecture 19
Chapter 3. Cloud computing 21
3.1 Introduction to Cloud Computing 21
3.2 Advantages and Disadvantages of Cloud Computing 22
3.2.1 Advantages of Cloud Computing 22
3.2.2 Disadvantages of Cloud Computing 23
3.3 Cloud Services Providers 24
3.3.1 Amazon Web Services (AWS) 24
3.3.2 Microsoft Azure 24
3.3.3 Google Cloud Platform (GCP) 25
3.3.4 Alibaba Cloud 25
3.3.5 IBM Cloud 25
3.4 Security Concerns of Cloud Computing 26
3.4.1 Encryption 27
3.4.2 Identity Authentication 28

3.4.3 Data loss prevention (DLP) 28
3.4.4 Security information and event management (SIEM) 29
3.5 Future Trends in Cloud Computing 29
3.5.1 Emergence of Internet of Things (IoT) 30
3.5.2 Emergence of Artificial Intelligence & Machine Learning 31
3.5.3 Emergence of Edge Computing 32
Chapter 4. Databases 34
4.1 Introduction to Databases 34
4.1.1 Database Types 34
4.2 The Database Management System (DBMS) 35
4.2.1 DBMS components 36
4.3 Advantages and Disadvantages of DBMSs 38
4.3.1 Advantages of DBMSs 38
4.3.2 Disadvantages of DBMSs 38
4.4 Database Security 39
4.4.1 Common Threats and Challenges 39
4.4.2 Defense Mechanisms 39
4.5 Amazon Aurora 40
Chapter 5. Project Implementation 41
5.1 Elastic Beanstalk and Spring Boot 41
5.2 Amazon Aurora Configuration 42
5.3 Elastic Search Server 42
5.4 Spring Boot Application 43
5.4.1 Controller Annotation 43
5.4.2 Service Annotation 45
5.4.3 Repository Annotation 46
5.4.4 Java Persistence API 46
5.4.5 Hibernate 47
5.4.6 Persistence Context 47
5.4.7 Hibernate Lifecycle 48
5.4.8 Entity Annotation 49
5.4.9 Id Annotation 50
5.4.10 EmbeddedId Annotation 50
Chapter 6. API Endpoints 52
6.1 Authentication 52
6.2 Person 53

6.3 Image 55
6.4 Role 55
6.5 Venue 55
6.6 Organizer 57
6.7 Production 57
Chapter 7. Future Improvements 61
7.1 Elastic Beanstalk alternatives 61
7.2 Amazon OpenSearch Service 61
Bibliography 62

Table of figures

Figure 1. Cloud Computing deployments 21
Figure 2. Features of Cloud Computing 23
Figure 3. Worldwide market share of leading Cloud providers 26
Figure 4. A Model for Securing Cloud Workloads 27
Figure 5. Importance of Cloud Encryption 28
Figure 6. SIEM Solutions 29
Figure 7. IoT application sectors 31
Figure 8. Machine Learning types 32
Figure 9. Basic Edge Computing architecture 33
Figure 10. Schematic of a DBMS system 36
Figure 11. DBMS components 37
Figure 12. Database Security Mechanisms 40
Figure 13 Elastic Beanstalk - Create application page 41
Figure 14 Elastic Beanstalk - Upload code section 42
Figure 15 Amazon Aurora - Public access 42
Figure 16. Rest Controller example 44
Figure 17. Service class example 45
Figure 18. Repository class example 46
Figure 19. JpaRepository 46
Figure 20. Hibernate Lifecycle 48
Figure 21. Project entity – Role 50
Figure 22. Embeddable annotation 51
Figure 23. Elastic Kubernetes Service deployment flow 61

Keywords

API	Application Programming Interface
AWS	Amazon Web Services
CURD	Create, Update, Read, Delete
DTO	Data Transfer Object
EB	Elastic Beanstalk
JVM	Java Virtual Machine
JWT	JSON Web Token
NPE	Null Pointer Exception
REST	Representational State Transfer
TP API	Theatrical Plays API

Chapter 1. Introduction

1.1 Thesis overview

The current thesis has been created as a subsystem of a pilot project that retrieves, visualizes and performs data analysis on web data. This task was undertaken by a group of students with distinct thesis papers. In total, 7 different thesis papers contributed to achieving this goal. The pilot project consists of three steps. The first step was to scrap theatrical plays data from viva.gr. Three theses undertook this task. Mr. Georgiadis implemented a .NET application, Mr. Fotiou implemented a Python application and Mr. Gkoltis implemented a C# application using the HTML Agility Pack and Selenium libraries. All three theses scrape data from viva.gr, transform them in a standard format and store them in a MySQL database. The next step was to retrieve the data from the database. After performing a migration from the MySQL database to an AWS Aurora database, I created a server using the Spring framework and the Kotlin programming language. The server searches, filters and sorts the data before serving them to a client. Since high availability and consistency is required, the server was deployed to the cloud through the AWS Elastic Beanstalk service. The final step was to implement the clients that present the data to the end user. Three theses were involved in this step. Mr. Dimitriadis created a web application using the React library, Ms Papachristou created an Android application and Mr. Skarlis created an iOS application. The clients make requests to the server and the server responds with the appropriate data.

Communication and teamwork between all members were necessary to successfully implement a project of this scale. A good understanding of the data structure and the underlying relations was needed to start developing the server. The assistance of Mr. Fotiou and Mr. Georgiardi was critical as they created the initial database schema and designed the relations between the tables. Once this information was passed to all the team members, numerous brainstorming sessions followed. The primary focus was to find intuitive ways to present the data. It would not be sufficient to simply fetch the data and project them to the clients. The aim is to modify and visualize the data in a manner that intrigues the end users and makes them engage with the applications. To that end, diagrams, quick search and meaningful filters were implemented.

Cloud services were also utilized to achieve high availability and scalability of the server. AWS Aurora is used to store the data and AWS Elastic Beanstalk is used to deploy and run the server. AWS Aurora is a fully managed relational database service compatible with MySQL and PostgreSQL. It was selected as it provides up to five times better performance that MySQL. Continuous back-ups are produced so in case of disk corruption or other data loss events, recovery is easy and expedient. Region disaster recovery takes less than one minute. It also offers up to fifteen read replicas to scale read operations and reduce traffic to the primary database. Overall is a cost-effective and performant alterative for enterprise-level databases. AWS Elastic Beanstalk is a fully managed service that makes the process of deploying, running and scaling web application effortless. It provides a platform for managing web application with support for multiple programming languages (Java, Javascript) and frameworks (Spring, NodeJS). It simplifies the process of deploying, scaling and monitoring applications by handling the underlying infrastructure and providing a platform for deploying and running applications.

Elasticsearch was also used to minimize search time. It is designed to scale horizontally, which means it can handle large amounts of data and handle a high number of queries without sacrificing performance. It can handle a high number of queries per second, and it has low latency meaning that the response time for each query is minimal. Compared to MySQL, it is considered to be faster, especially for full-text search and other types of data analysis, as it is optimized for analyzing large amounts of data in near real-time.

Apart from gaining technical knowledge, the current thesis has improved the soft skills of all the team members. The project acted as a simulation of a professional work environment. Effective

communication was a key element to the success of the project. Responsibility was shared among all members, so the team had a greater sense of accountability and motivation. We were tasked with producing a satisfying result, so decisions had to be agreed by everyone involved. Different perspectives and ideas were exchanged, and each individual had to make a case for their reasoning. Final decisions had to be comprehended by all team members as understanding of the subsystems was essential to produce the best possible result.

Chapter 2. Technologies used

2.1 Java

Java is a high level, robust, object-oriented, and secure programming language, first developed by *Sun Microsystems* in 1995. Java was originally designed for interactive television, aiming to become a language for digital devices like TVs, but it was too advanced for its time. Instead, it became clear it was best fitted for internet programming. Before its current name, the language was called Oak after an oak tree that stood outside James Gosling's office, the so-called father of Java. Approximately 6 million developers use Java technology and about 5.5 billion devices of all sorts run on Java. These include a variety of applications such as Desktop, Web, mobile and enterprise applications, while Java is also used in game programming, embedded Systems, Robotics, Big Data and Networking. Desktop applications, also known as Standalone applications, are applications that run locally on each device and don't require anything else to work. Some examples may include Notepad, Microsoft Word, Adobe Photoshop, Google Chrome, antivirus, etc. Web applications are applications that are stored on a remote server and create a dynamic page. Known technologies that are used in the creation of web applications in Java are JSP, Spring, Hibernate, etc. Mobile applications are applications for mobile devices and are primarily made by Android and Java ME while enterprise applications are applications that satisfy the needs of organizations, such as banks, schools, governments, etc. They are created by EJB technology and are considered highly secured and efficient. Java also has 4 editions or platforms. Java SE or Java Standard Edition is the minimum requirement for a java application to run and includes Java programming APIs like java.util, java.lang, java.io, etc. Java EE or Java Enterprise Edition is the platform primarily used for the development of enterprise applications and includes technologies such as JSP, EJB, JPA, etc. Finally, Java ME or Java Micro Edition provides a flexible and robust environment for applications running on mobile devices while JavaFX is dedicated to the development of rich internet applications that operate across a diversity of platforms.

Java comes with a plethora of advantages that make users want to stick with it:

- Simplicity

The syntax of Java is straightforward and easy to learn, while the code itself is easily debuggable and less complex than other languages such as C and C++. It is also a high-level programming language as it is a human-readable language.

- Security

Java includes a wide range of defense and security mechanisms, tools and APIs. Java's core security elements focus primarily on high level platform security, cryptography, Authentication and Access Control, Secure Communications and Public Key Infrastructure (PKI)

- Object-Oriented Programming language

Java's nature as an Object-Oriented Programming language enables the reuse of code in other programs and the better management and organization of data and functions.

- Economical

The ease at which Java programs can be executed on any device make their development and maintenance costs very cheap and economical.

- Stability

In comparison to other programming languages, java programs are much more stable and new versions of the language that are released help maintain this stability.

- Platform Independence

Java's compiled code can run on all operating systems making it platform independent thanks to the use of a virtual machine.

- Multithreading

Java possesses the ability of multithreading, meaning that it can enable multiple users at a time without the need of multiple copies of the program running on the computer.

Java has also some drawbacks that are worth mentioning:

- Poor performance

Compared to other languages like C and C++, Java is vastly slower, primarily due to the extra level of compilation and abstraction by the JVM.

- Memory Space

Java also demands significantly more memory space as compared to other languages since Java programs run on top of the JVM.

- GUI

There is a great variety of frameworks such as Swing, JavaFX, JSF, SWT for creating GUI in Java, still they are not well suited for the creation of more sophisticated UI.

- Garbage Collection

Java Programs perform memory management automatically with a process known as Java Garbage Collection. The lack of control over this process by the programmer points out another drawback, since there are no functions like delete() or free() in Java.

2.2 Kotlin

Kotlin is an open source, multi purposed programming language running on JVM, and is primarily used in the development of Android applications. Kotlin was developed by JetBrains in 2010, a Czech software company and was named after Kotlin Island, near ST. Petersburg, Russia. Its first release came in February 2016. The purpose of Kotlin is to become a wide scale object-oriented language and surpass Java, while at the same time be fully interoperable with it to help companies gradually transition to it. Some say that Kotlin is what Java could have looked like if it were designed today.

Kotlin offers a wide variety of advantages:

- Open-Source

Kotlin is that it is an open-source programming language and provides developers with a way to convert existing Java code.

- Interoperability

Kotlin is fully interoperable with Java, runs on JVM and supports many of its libraries, making many Java and Android developers willing to learn Kotlin.

- Security

Kotlin is relatively safe and allows its users to simplify their code for the debugging and operations of Android app development.

- Cost

The adoption of Kotlin comes with no cost it is open source. Additionally, compared with other programming languages, it is easier to learn for developers.

- Clean Syntax

As far as the syntax and clarity of the code is concerned, Kotlin can get things done with only a few lines of code, making it more reliable with fewer errors and bugs. This also allows easier maintenance and ease of reading the code.

- Standard Library Functions

Kotlin offers many standard library functions that make Android mobile application development easier.

- Data classes

Kotlin offers a simple and automated way of creating data classes.

Kotlin also comes with some notable disadvantages:

- Small community of developers

The lack of Kotlin developers make their availability in the market and its learning opportunities minimal.

- Compilation speed

Kotlin has a slower compilation speed than Java

2.3 Differences of Java and Kotlin

Here, are the major differences between Java and Kotlin.

- Null Safety

In Java users are allowed to assign null to any variables, making NullPointerExceptions an annoying issue. In Kotlin, this is not possible since all types of variables are non-nullable.

- Extension Functions

In Java, in order to extend the functionality of an existing class, a new class must be created to inherit the parent class. In Kotlin, there is the possibility of creating extend functions just by prefixing the name of the class to the name of the new function.

- No checked exceptions

Java supports checked exceptions, making the code more robust and with better error handling, in contrast to Kotlin where checked exceptions don’t exist.

- Data classes

In Java, users need to establish the fields to store the data, the constructor, the getter, as well as other functions. On the other hand, Kotlin enables the creation of classes to hold data by including the "data" keyword in the class definition. After that, the compiler auto-generates the constructor and the getter and setter functions for several fields.

- Type inference

In Java, it is necessary to explicitly specify the type of each variable when declaring it, while in Kotlin there is no such need and can be done optionally.

- Functional Programming

Java didn't support functional programming up until Java 8, in contrast to Kotlin which is a mix of both procedural and functional programming language and has many useful methods.

2.4 Spring framework

The Spring framework is an open-source application framework that provides infrastructure support for the development of Java applications and helps developers create high performing applications using old Java objects. A framework can be thought as a large body of predefined code to which developers can add code to solve a problem. In a broader sense, it can be seen as structure where we find the solution of various technical problems. The Spring Framework was first released in 2003 by Rod Johnson and is now hosted by SourceForge. It provides predefined templates, thus lessening the need to write too much code. Its applications are loosely coupled, meaning that their components depend on each other to the least extent practicable. The Spring framework also does not require a server to run the application, making it easier to test it, and it is also non-invasive and lightweight. It also provides powerful abstraction to Java EE specifications such as JPA, JTA, JMS and JDBC. On the downside, Spring suffers from complexity, lacking clear focus and has more than 2400 classes with 49 other tools, making it even more complicated. Developing a Spring application would be quite difficult to learn, given the difficulty of its new programming methods and would also require a lot of XML. The existence of parallel mechanisms can also be considered a disadvantage, since developers can get multiple options, thus creating a point of confusion. Finally, the selection of the wrong mechanism can lead to wrong decisions and cause severe delays in developing the applications.

2.5 Maven

Maven is a well-known open-source build tool that was developed by the Apache Group. It primarily focuses on providing developers an inclusive, maintainable, and plain model for projects, while at the same time helping them build, publish, and deploy projects at once for better project management. It also comes with a set of tools and plug-ins that can interact with the declarative model. It is written in Java but is also used to build projects written in other languages. Maven automates the processes of source code generation and compilation that all applications must undergo, which are by nature a time-consuming concern for all developers. Maven simplifies and standardizes the building process by dealing with Builds, Documentations, Dependencies, Reports, Releases, etc. It is also loaded with a great variety of effective and beneficious features, such as a growing repository of user libraries, the ease of setting up projects and dependency management. It is backwards compatible with previous versions, has a powerful error and integrity reporting and can also be considered extensible. It is also known for simplifying and increasing the performance of the project and building process. Maven is primarily adopted for Java-based projects and is used by more than 4000 companies worldwide, in sectors like computer science, information technology, financial services, etc. Most notable companies using Maven are situated primarily in the United States, like Accenture, JPMorgan Chase & Co, etc. and, as of 2020, 4000 websites were using it.

2.6 Elastic Search

Elasticsearch is a distributed, open-source search and analytics engine built on Apache Lucene and developed in Java. It comes with search, analysis, and storage capabilities of great amounts of data and can accomplish fast search responses, primarily with the use of extensive REST APIs. It is widely used in applications that depend on a search platform, websites that store a lot of content and demand effective searches, and also in enterprise-wide searches. It can also be found in logging and log analytics, security analytics and infrastructure metrics. Elasticsearch also comes with a great number of capabilities. It offers a powerful full-text search engine that permits the combination of many types of searches. It possesses an effective analytical engine that can capably perform the slicing and dicing of numerical data such as application and infrastructure performance metrics. Elasticsearch was also designed with a distributed architecture meaning it can be scaled to a vast number of servers and thus manage a great amount of data. It has a well-documented API and holds client libraries for many programming languages such as Java, JavaScript, PHP, , Python, etc. Well-known companies such as Netflix, eBay, and Walmart use Elasticsearch.

2.7 Git

Git is an open-source distributed version control system, created to efficiently manage projects and organize the work among developers. It was made by Linus Torvalds in 2005 and is the core of well-known services like GitHub and GitLab and a powerful distributed version-control tool for the DevOps. Git is scalable, meaning it can handle any growing number of users with ease, and distributed, effectively cloning the entire repository instead of switching the project to another machine. Other key aspects of Git are security, which is accomplished with the use of the SHA-1 (Secure Hash Function) for the identification and integrity checking of all file objects and commits, and speed, since all its operations are done on the local repository. Git also supports offline working, effectively dealing with online connectivity issues since everything can be done locally.

2.8 Amazon Elastic Beanstalk

The Amazon Elastic Beanstalk service is used to deploy and scale web applications developed in different languages (Java, NodeJS, Python, Go etc). The web application can be hosted on different servers such as Tomcat, Apache and Nginx. It is a simple-to-configure service, and it comes with no additional charge. To begin deployment, the user can upload your code or use a S3 bucket and Elastic Beanstalk will handle the rest. Load balancing, auto-scaling, monitoring and other features are automated. Any AWS resource needed will be created, whilst the user has full access to them.

Benefits of Elastic Beanstalk

- Beginner friendly

Elastic Beanstalk requires minimum configuration. All that is required is the code needed for the application. An S3 bucket can be used to store the code but is optional. Once the code upload is complete, Elastic Beanstalk handles the details of the deployment, such as load balancing, auto-scaling and monitoring. No previous infrastructure or manual resource configuration is needed by the user.

- Increased productivity

Being so easy to use, Elastic Beanstalk saves up on development time. Instead of consuming time and effort to manage and monitor servers, databases, load balances and firewalls, the user can more

productively allocate their time. The underlying platform will also be continuously kept up to date with patches and updates.

- Auto-scaling

Elastic Beanstalk reserves and releases resources based on the incoming traffic. Peaks in traffic are handled easily while resources are released in downtimes to lower expenses.

- Resource control

As mentioned previously, Elastic Beanstalk will create all the necessary infrastructure, but if needed the user can select specific resources that are best suited for their application. The resources can be changed at any point, giving the user full control over the application.

2.9 HTTP

HTTP stands for Hypertext Transfer Protocol and is an application-layer protocol for transmitting hypermedia documents and files on the World Wide Web. Communication is usually achieved through TCP/IP sockets and can be supported by a mixture of network configurations. HTTP is a protocol primarily used to transfer the hypertext from the client end to the server end. When it was first designed in 1991, its main purpose was to get the html document and send it to the client, not supporting other media types. Its continuous evolution meant that new features were introduced by the years, making it the best way to move data rapidly and accurately on the web. Communication between clients and servers is accomplished via requests and responses. Initially, a client sends an HTTP request to the web. After that, a web server receives the request and runs an application to process the request. Then, the server returns an HTTP response to the browser and finally the client receives the response. HTTP is connectionless, meaning that both client and server know about each other during their current request and response only. HTTP is also stateless, so, if the connection is closed, client and server must reconnect once more from the very start. Large quantities of HTTP requests can also be used in a malicious way to mount an attack, most notably DoS and DDoS attacks.

2.10 Rest API

Rest API (or REpresentational State Transfer API) is an application programming interface that defines a set of constraints to be used for the creation of web services. It is a way of accessing web services in a plain and flexible manner, without the need of processing. It's used to give or take some information from a web service and uses only HTTP requests to access data. That data can be used for the reading, updating, creating, and deleting of operations. REST consumes less bandwidth, so it's more efficient for internet usage. First defined in 2000 by computer scientist Dr. Roy Fielding, REST provides a comparably high level of adaptability and freedom for developers.

2.11 Client-server architecture

A client-server architecture is a network application that breaks down tasks and assignments between clients and servers that are linked by the same network. In a conventional client-server architecture, multiple users' devices are connected to a central server via an Internet connection or other network. The client sends a request for data, and the server accepts and accommodates the request, sending the data packets back to the user. This model is also called a network computing model or client-server network. It offers a higher level of processing and increases the effectiveness of workgroup empowerment, remote network management, and market-driven business. Client-server architecture

provides the exact framework that most organizations need nowadays to face the challenges of an ever-evolving IT age. Some real-life application examples of this architecture are email, file, and web servers.

Some advantages of the Client-server architecture are listed below:

- It is a centralized system, meaning it keeps all the data and its controls in one place.
- It provides scalability and organization of high level.
- It is cost-efficient and easily maintainable.
- It allows data recovery.
- It allows load-balancing.
- It permits different platforms to share resources.

The most notable disadvantages of the Client-server architecture are listed below.

- The users are vulnerable to viruses since the network consists of linked clients and servers.
- It is vulnerable to DoS attacks.
- Data packets can be spoofed during transmission.
- It is expensive to start up.
- The setup is prone to Man in the Middle (MITM) attacks.

Chapter 3. Cloud computing

3.1 Introduction to Cloud Computing

Cloud computing refers to the delivery of computing services, such as databases, networking, software, and analytics over the Internet so that users do not need to consider the physical location of servers and storage. In other words, it could be said that Cloud computing essentially provides a suitable on-demand network from which users can gain access to a shared pool of computing services. Those services have long been called Software as a Service (SaaS) whilst the Cloud can include Infrastructure as a Service (IaaS) and a platform as a Service (PaaS). Software as a service is a way of delivering software applications over the Internet, on demand and typically on a subscription basis. Infrastructure as a Service enables the rent of IT infrastructure, like servers and virtual machines, networks, and operating systems, while Platform as a service refers to the cloud computing services that are responsible for the development, test, and management of software applications. Clouds can be categorized in Public, Private and Hybrid Clouds depending on the range of customers they can offer their services to. Public Clouds are similar to the internet, with users and services from across the World Wide Web having full access to their computing environment. Private Clouds permit access to a selection of users and services, such as organizations of a company, and very much resemble an intranet. Hybrid Clouds also offer their services to a selection of users, but, if needed, can expand to reside on a Public Cloud infrastructure.

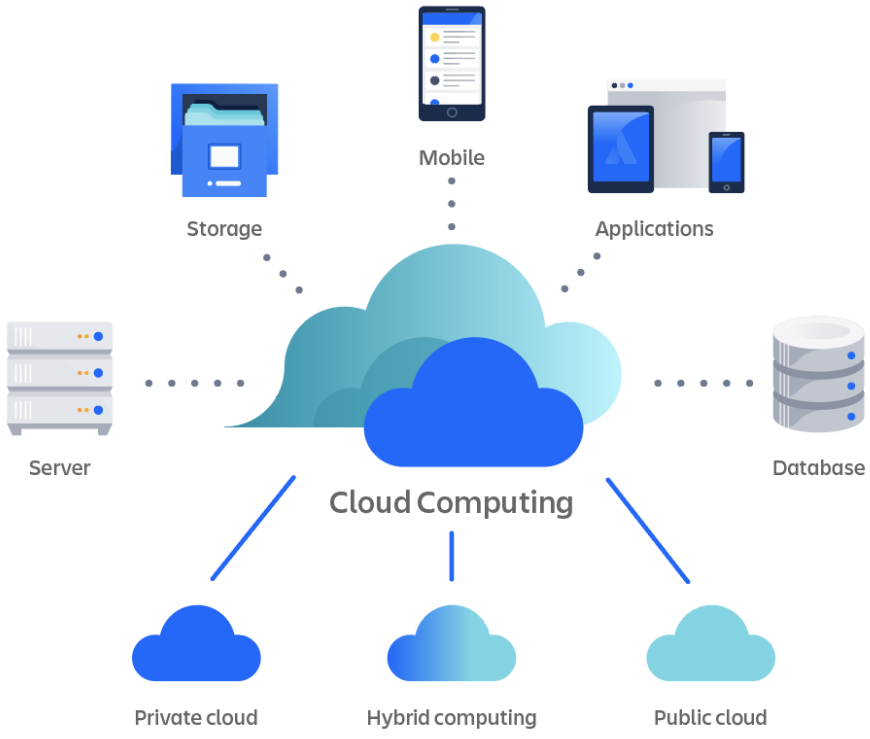

Figure 1. Cloud Computing deployments

3.2 Advantages and Disadvantages of Cloud Computing

Cloud computing is a major shift from the traditional way IT resources are being managed and used and comes with a great variety of benefits but also challenges that need to be faced.

3.2.1 Advantages of Cloud Computing

Cost

Cloud Computing practically eliminates the expense of buying, setting up and maintaining the hardware and software used on the data centers of companies. These would also demand electricity power, cooling, racks of servers and experts to manage them, so Cloud Computing can provide a long-term cost-effective solution. Once in the Cloud, a company's data can easily be accessed and time, money and effort can be saved. The pay-as-you-go system enables companies to get exactly as much storage space as they need, and not get charged for space that they do not. All these factors result in lower costs and higher long-term returns.

Security

Many cloud providers can help companies protect their data, apps, and infrastructure by offering a wide variety of security mechanisms and policies. In contrast to conventional systems, where security is just one of many concerns that IT experts must attend to, a cloud host's job is to carefully monitor security all day long, providing much needed protection. Many times, it is also safer to keep sensitive information offsite, since a high percentage of data thefts occurs internally. Approximately 9 out of 10 businesses can validate the fact that they saw an improvement in security after switching to the cloud, the key to which is the encryption of data being transmitted over networks and stored in databases. Encrypted data are less prone to hackers and unauthorized personnel.

Flexibility & Mobility

Modern day businesses only have a limited amount of time to focus and divide between all their responsibilities. Computer and data-storage issues can be really demanding, so relying on an outside organization to manage and maintain all the IT infrastructure can free up more time and effort. Cloud computing does just that and offers a great degree of flexibility and freedom. Cloud computing also provides mobile access to data via smartphones and devices. Personnel that have busy schedules or live far from their office can use this this feature to stay up to date with their co-workers and clients. Mobility is thus one of the greatest priorities for organizations and companies tend to expand to cloud usage for this reason.

Collaboration & Competition

Cloud computing makes collaboration an effortless process. Team members can gain access and share information with ease and security across a cloud-based performance. Companies that have adopted and implemented a cloud-based solution also feel that this technology has given them a competitive edge over those that have decided to keep everything local, something that is being proven daily with the rising popularity of cloud services.

Recovery & Updates

There is no way a company can predict or prevent disasters, by they can help speed up their recovery. Cloud-based services provide fast data recovery for all types of emergency scenarios, like natural

disasters, power outages, etc. Cloud-based applications can also automatically refresh and update themselves. In contrast, an IT department would have to perform a wide scale manual update which would require valuable time and money to be spent on outside IT consultation.

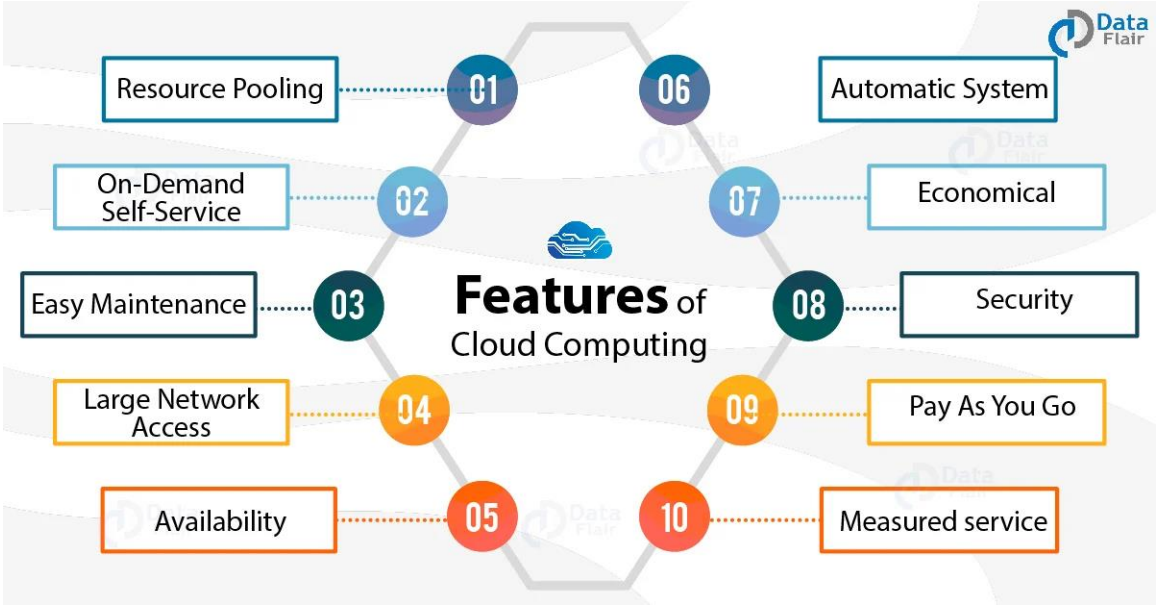

Figure 2. Features of Cloud Computing

3.2.2 Disadvantages of Cloud Computing

Understanding the costs

Cloud computing can help reduce the overall costs in many areas, but it is important to carefully plan and analyze which systems should be moved to the cloud and which systems should remain on-premises. Moving to cloud data centers is an easy process, but that is not always the case to another cloud supplier or back to an on-premises server. This movement can get quite expensive, and the terms can often side with the cloud supplier. It is thus critical to cover the timelines, fines, and process, before entering a contract.

Limited control

Businesses that move to the cloud often find themselves worried about not having enough control over the service. The fact is that the infrastructure is owned and managed by the service provider. If ownership is the most crucial factor regarding a company's stored data, a public cloud may not be the best option. Decreased control is a common disadvantage for companies considering the switch to the cloud. The provider's end-user license agreement (EULA) can help explain what limits the provider can place on the use of the deployment.

Vendor lock-in

Cloud providers often promise that cloud services are easy to use and integrate. But businesses may find it difficult to switch to a different vendor with a different platform and might run into complications. In case of improper handling, data may be exposed to unnecessary vulnerabilities. Profound differences

between systems makes it complex and expensive for applications to be reconfigured. Correctly understanding what services and capabilities the vendor is providing will help avoid getting locked-in.

Internet reliance

Cloud computing is completely reliant on the internet. If the internet connection goes down, access to data stored in the cloud is lost for the duration of the outage. An internet interruption will not compromise the data stored in the cloud. Also, during network rush hours, the cloud might increase congestion and reduce your internet performance. Good providers avoid this issue through scheduling, but many companies that have not invested in high internet bandwidth and speeds might be affected significantly none the less. Backups and restores also depend heavily on internet use and since moving data to the cloud involves significant communication latency, it could prove an issue. Individual files and folders might not be a problem but restoring a whole server could be.

3.3 Cloud Services Providers

There are a handful of well-known, major public cloud companies, the most notable of which being Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform (GCP), Alibaba Cloud and Oracle Cloud.

3.3.1 Amazon Web Services (AWS)

AWS Cloud is the most used cloud computing services. AWS was launched in 2006 by Amazon.com and was one of the first companies to introduce a pay-as-you-go cloud computing model that provides users with compute, storage or throughput as needed. It offers a variety of use cases such as cloud operations, database migrations, analytics, edge computing and web and mobile development. Enterprises and software developers rely heavily on AWS and make use of its many tools and solutions. Government agencies, institutions and various organizations use AWS services in more than 190 countries. AWS is divided into many different services, each one of which can be configured depending on the user's needs. Approximately 100 services are included in its portfolio and include categories such as Storage databases, Migration, Networking, Development tools, Management, Monitoring, Security, Big data management, Analytics, Artificial intelligence, and Mobile development. AWS leads the worldwide market share of cloud infrastructure service providers with a percentage of 34% out of a total 200-billion-dollar cloud market.

3.3.2 Microsoft Azure

Microsoft Azure is the cloud computing platform provided by Microsoft and allows its users to gain access to its services and resources, provided they have an internet connection and the capacity to connect to the Azure portal. It was launched 4 years after its main competitor, AWS, in 2010 and follows a pay-per-use model. There are 42 Azure data centers around the world, which make up for the highest number of data centers of any cloud platform. Microsoft Azure is used by the majority of United States largest corporations and supports a multitude of programming languages such as Java, Node Js, C#, etc. Just as AWS, Microsoft Azure provides a plethora of services, more than 200, divided into 18 categories. These categories include computing, networking, storage, IoT, migration, analytics, artificial intelligence, machine learning, management tools, developer tools, security, databases, DevOps, and web services. Azure is also used in Application development, hosting, testing, the creation of virtual machines, the collection and storage of metrics, and in virtual hard drives. Microsoft Azure comes second on the worldwide market share of cloud infrastructure service providers with a percentage of 21%.

3.3.3 Google Cloud Platform (GCP)

Google Cloud is a collection of public cloud computing services for the compute, storage and application development offered by Google, first launched in 2008. Software developers, cloud administrators and other IT experts can get access to GCP over the internet or through a network connection. Some of its services resolve around categories like storage, networking, big data, machine learning, IoT, security and developer tools. Google Cloud offers its users powerful tools such as Google Compute Engine, an infrastructure as a service (IaaS) that facilitates workload hosting with VM instances, Google App Engine, a platform as a service (PaaS) that provides developers access to Google's scalable hosting, Google Cloud Storage, a cloud storage platform made to store extensive, unstructured data sets and Google Kubernetes Engine (GKE), a management system for Docker container that runs within Google's public cloud services. Google's cloud platform continues to evolve, adding higher-level services related to the processing and analysis of big data and machine learning. Google Cloud comes third on the worldwide market share of cloud infrastructure service providers with a percentage of 10%.

3.3.4 Alibaba Cloud

Just like the emergence of AWS from Amazon, Alibaba Cloud is a product of the Alibaba Group, the well-known Chinese e-commerce giant. It was founded in 2009 and had its first operation centers in Hangzhou, Beijing, and Silicon Valley. Alibaba Cloud is the biggest public cloud in China, and it's still growing rapidly. According to its latest quarterly report, Alibaba's cloud revenue grew 59% in 2020 alone. Alibaba Cloud offers a variety of infrastructure and application development services and follows the pay-as-you-go pricing model that most cloud providers use. Alibaba has a powerful infrastructure portfolio, which involve computing, database, network, security, and hybrid cloud solutions. Alibaba Cloud comes with a wide range of services. Elastic Compute Service (ECS) is the platform's core IaaS and comes with a scalable compute capacity. Object Storage Service (OSS) is a service that facilitates the storage of large amounts of data in the cloud and allows enterprises to store any type of unstructured data in OSS for their applications and websites. As far as networking is concerned, Alibaba Cloud offers the Virtual Private Cloud (VPC) which enable users to create a private and customizable network dedicated to their cloud accounts. Alibaba Cloud comes fourth on the worldwide market share of cloud infrastructure service providers with a percentage of 5%.

3.3.5 IBM Cloud

IBM Cloud is a suite of cloud computing services from information technology company IBM or International Business Machines Corporation. IBM Cloud used to be known as SoftLayer Technologies, Inc. a cloud computing provider, founded in 2005 that was acquired by IBM in 2013. After its acquisition it changed its orientation from hosting workloads for gaming companies to enterprise workloads. As of 2021, IBM Cloud contains more than 170 services and announced it would achieve net zero greenhouse gas emissions by 2030. There is a great number of IBM cloud services that are grouped into 16 categories, involving AI, machine learning, Automation, Compute, Networking, Storage, Security, Analytics, IoT, etc. The exact cost of IBM Cloud differs depending on factors like resource usage, deployment model, support, etc. As of late 2021 all new accounts are automatically created as pay-as-you-go plans. IBM Cloud comes fifth on the worldwide market share of cloud infrastructure service providers with a percentage of 4%.

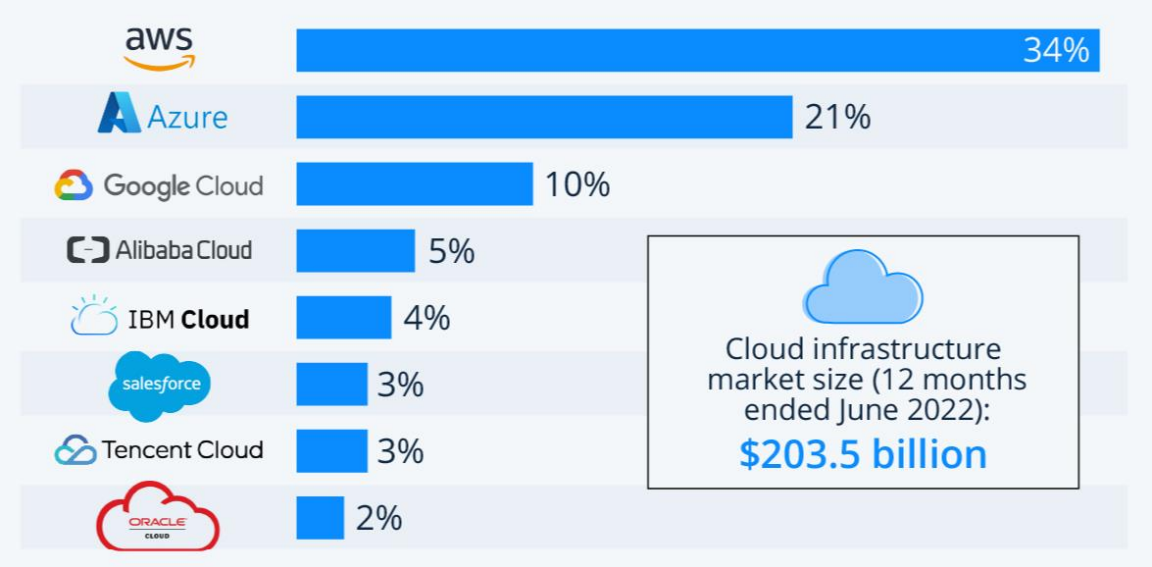

Figure 3. Worldwide market share of leading Cloud providers

3.4 Security Concerns of Cloud Computing

Most organizations nowadays already use cloud computing in one way or another. The worldwide market for public cloud services has grown by a quarter in 2021 alone, so one would argue that cloud security is becoming more and more important on a daily basis. The movement of data and applications to the cloud is always a concern for IT specialists, since there is always the risk of having sensitive data exposed through cyber-attacks or accidental leaks. For this reason, cloud providers are taking measures to guarantee that their clients intellectual property stays protected from unauthorized access.

Figure 4. A Model for Securing Cloud Workloads

3.4.1 Encryption

Cloud encryption uses encryption algorithms to transform a client's data into cyphertext and then save it in the cloud. It basically follows the same procedure as an on-premises encryption, although in the case of cloud encryption, it is the cloud provider that sets the encryption policies and procedures. Depending on the sensitivity of a customer's information, cloud encryption capabilities tend to vary greatly. Encryption is one of the most important defense mechanisms Cloud computing has to offer and is mandated in many regulatory standards, such as Federal Information Processing Standards (FIPS), Federal Information Security Modernization Act (FISMA), Accountability Act (HIPPA). Amazon Web Services (AWS), Microsoft Azure and Google Cloud provide data-at-rest cloud encryption which means that encryption and decryption processes are handled in the background, so users do not need to bother.

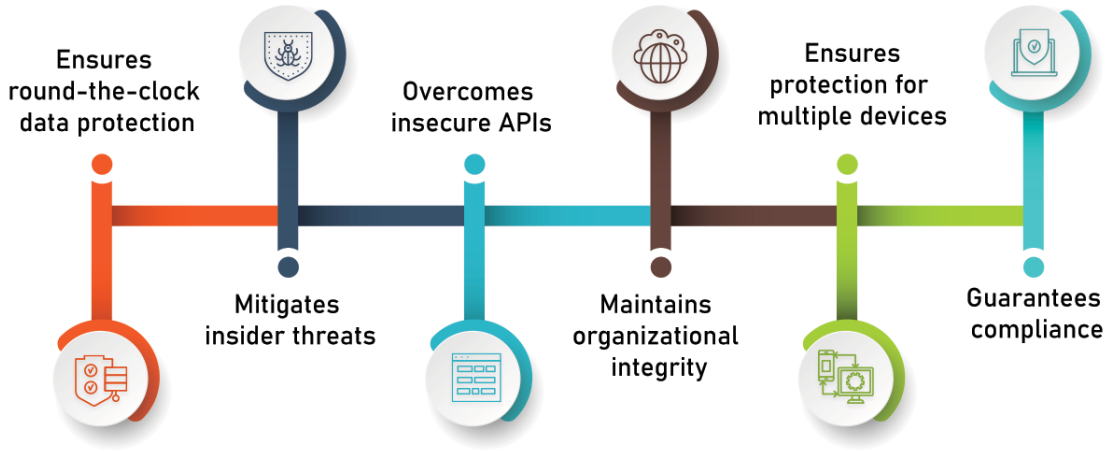

Figure 5. Importance of Cloud Encryption

3.4.2 Identity Authentication

Authentication is the process of determining whether someone is who he is claiming to be. Identity authentication is usually applied when someone requests access to a system. The most standard form of access control method is, the known to all, use of passwords. Despite being mainstream and widely adopted, passwords are vulnerable to brute force hacking. Although some policies increase the length and complexity of passwords, modern day processing power can easily hack passwords of considerable size. Cloud identity and cloud authentication are the cloud-based solutions to a company's identity and access management demands. They are composed of two main services, Authentication as a Service (AaaS) and Identity as a Service (IDaaS). AaaS grants multi-factor authentication, single sign-on, and password management in the cloud and enables companies to trace password usage and password requirements. IDaaS is an application delivery model that permits its users to connect and make use of the various identity management services from the cloud. Major tech companies like Google, Facebook, and Apple provide cloud identity.

3.4.3 Data loss prevention (DLP)

Data loss prevention (DLP) services provide all necessary tools and services designed to ensure the security of regulated cloud data, with the use of a combination of remediation alerts, encryption, and other measures. Also known as data leak prevention and extrusion prevention, DLP is widely implemented as part of a company's plan for overall data security and ensures that sensitive data remains behind a network firewall. In order to maintain regulatory compliance, organizations tend to update their data storage and retention policies by adopting a DLP strategy. Remote working has played a crucial role in the adoption of DLP, since cyber-attacks have become more common and sophisticated. It is estimated that 9 out of 10 organizations implemented at least some form of integrated DLP as of 2021, in comparison to 5 out 10 in 2017. Most DLP software products focus primarily on blocking actions that go against company policy. For example, an employee that would attempt to upload a business file to a consumer cloud storage service, would have his permission denied. DLP software can also prevent employee computers from reading and writing to USB drives, thus avoid unauthorized copying. Nowadays DLP software uses machine learning-based artificial intelligence with the purpose of improving its detection and blocking mechanisms.

3.4.4 Security information and event management (SIEM)

Security information and event management (SIEM) serves the purpose of automating threat monitoring, detection, and response in cloud-based environments. It uses artificial intelligence (AI)-driven technologies and provides IT teams the ability to auspiciously apply their network security protocols while, on the same time, react quickly to any potential threats. Organizations can collect, store, and analyze security information and alert their security teams to possible attacks. Cloud-based SIEM or SIEM-as-a-Service provides IT teams with greater comfort, flexibility, and power when managing threats, both on-premises and in the cloud. It is possible to constantly monitor all devices, servers, users, and applications anywhere on the network, receive real-time alerts on security incidents and take risk analysis to the next level. Cloud SIEM provides companies a great number of benefits in comparison to the on-premises solutions, some of which are its speed of deployment, its lack of need for expertise and its cost effectiveness in total.

Figure 6. SIEM Solutions

3.5 Future Trends in Cloud Computing

Global spending on public cloud products is rising rapidly, with an annual rate of approximately 20% and it is projected to reach 600 billion dollars by 2023. As far as cloud services are concerned, spending is anticipated to reach 30% year-over-year growth for Infrastructure as a Service (IaaS), 27% for Desktop as a Service (DaaS) and 26% for Platform as a Service (PaaS). PaaS spending will probably increase to $109.6 billion. The growing trend of remote working is urging companies to invest more capital on their cloud migration efforts, with an estimated 2.6 billion dollar being spent on yearly basis. IT spending towards public cloud solutions by 2025 will reach 51%, compared to 41% in 2022 and application software spending will reach 66% from 58% in 2022. By 2025, IT spending on public cloud services will surpass traditional IT spending. A transformation of such scale will likely bring tremendous changes in the technological world, but as we have seen previously in this chapter, Cloud Computing faces some limitations, the most notable of which being its implementation cost, its lack of control, its internet

reliance, and its vendor lock-in dependance. It is thus important to examine the emergence of some new future trends that could ease these problems and help facilitate this transformation.

3.5.1 Emergence of Internet of Things (IoT)

The Internet of Things (IoT) refers to a group of devices that are linked together using an Internet connection. The hub for this collection sends and collects data using the Internet, helps devices make decisions and remembers patterns for actions to be carried out automatically. Since IoT has both real-time and historical data stored, it can be used as an effective guidance tool and provide decision-making instructions to devices, and control certain of their actions. Some of its core application sectors are Agriculture, Energy, Finance, Healthcare, Manufacturing, Retail, Transportation and Logistics. IoT is an automated cost-effective technology and can be used in combination with Cloud Computing, to enable its users to perform computing tasks using services from the Internet. The mutual use of the Internet of Things and cloud technologies will likely alleviate the problems of storing, processing, and accessing large amounts of data. The benefits of combining these two technologies into an IoT cloud infrastructure could lead to an extensive internet-based network that stores data from IoT devices and applications. IoT Cloud Computing can provide many connectivity options, meaning that people can use a great number of devices, like phones, tablets, and laptops, to gain access to cloud computing resources. Developers can use IoT cloud computing on-demand and access it without special permission, just a simple Internet access. It is also a fast and flexible way of expanding storage space, edit software settings, and it is also possible to provide deep computing power and storage. As far as security is concerned, Cloud solutions could provide companies with reliable authentication and encryption protocols. Finally, IoT cloud computing can be very advantageous because customers pay-as-they-go, meaning that costs vary depending on use. All in all, it is crucial for the cloud architecture to be well-designed and focused on key aspects such as reliability, economy, security, and performance optimization if the combination of IoT and Cloud Computing is to succeed.

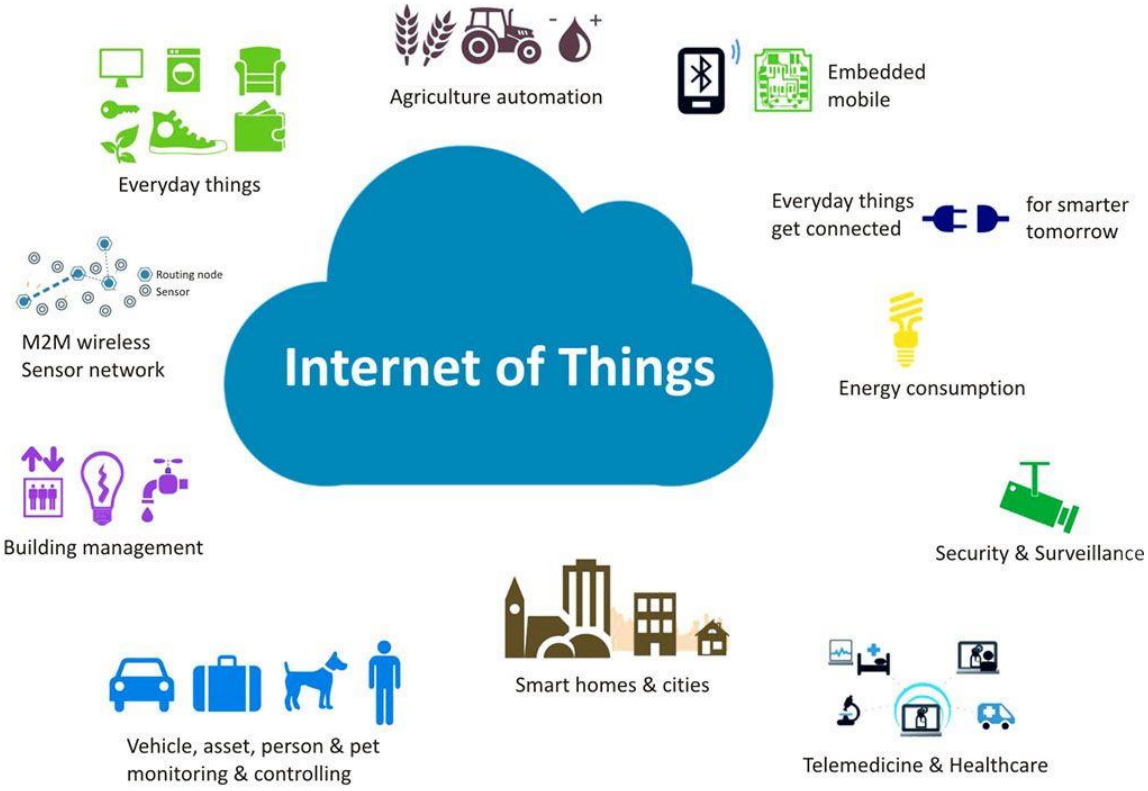

Figure 7. IoT application sectors

3.5.2 Emergence of Artificial Intelligence & Machine Learning

Artificial intelligence and machine learning are rapidly progressing into enterprise applications in various areas and there are encouraging signs that this will also happen in the cloud. Over the past few years, Amazon, Google, and Microsoft have invested significantly in artificial intelligence (AI) and machine learning, proceeding to major reorganizations that place AI strategically in their core structures. The Cloud is the ideal place for enterprises to experiment with machine learning capabilities and scale up, since it can provide access to smart capabilities without the requirement of advanced skills in artificial intelligence. AWS, Microsoft Azure, and Google Cloud Platform already offer a range of machine learning options. However, building sophisticated machine learning models in-house is risky, since training real-world models demands large compute clusters. Bringing sophisticated machine learning capabilities to enterprise applications is challenging and requires specialized skills and computational and special-purpose hardware. Problems like these can be solved by cloud computing and thus make it possible for companies to adopt machine learning capabilities in an easier way. Bursty AI and machine learning workloads can capitalize on the cloud's pay-per-use model since companies can leverage the speed and power of GPUs for training without investing in the hardware. The cloud also makes intelligent capabilities accessible without requiring advanced knowledge in data science. About a quarter of modern tech companies have some experience with AI or machine learning. Cloud providers like AWS, Microsoft Azure, and Google Cloud Platform could help increase these numbers by offering many options for implementing intelligent features in enterprise applications.

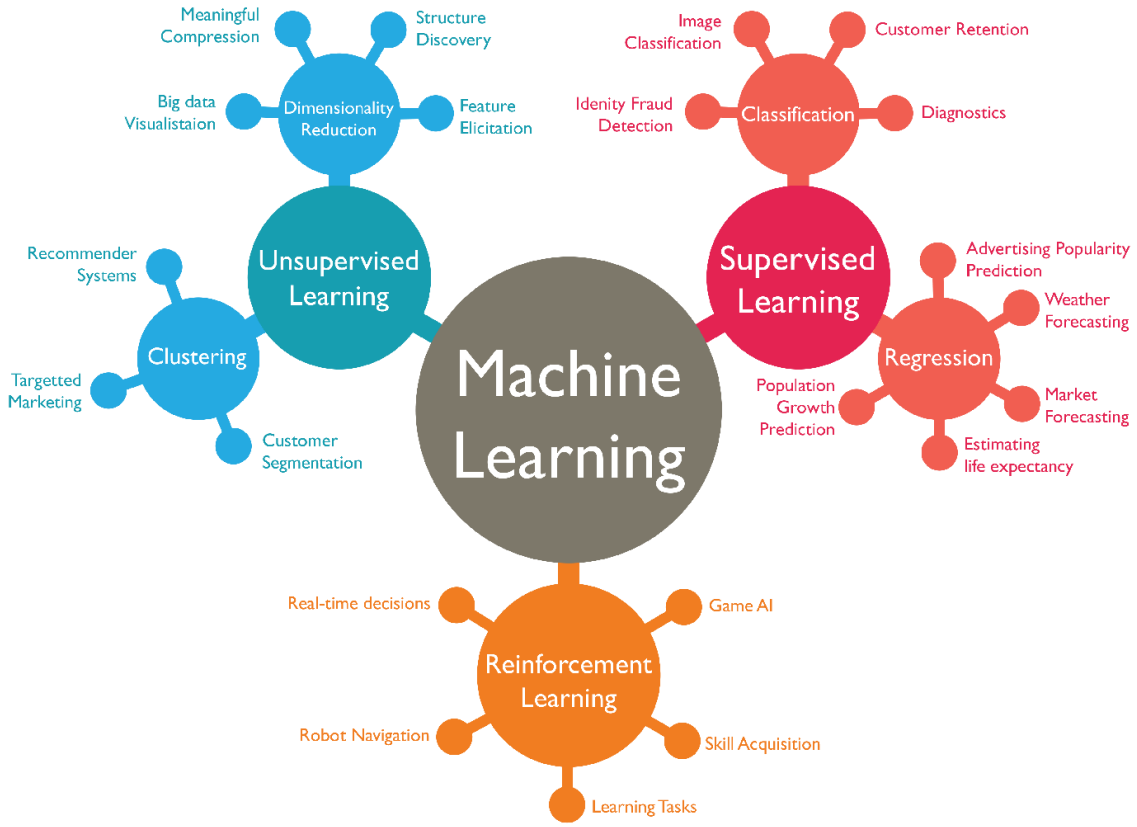

Figure 8. Machine Learning types

3.5.3 Emergence of Edge Computing

In a cloud computing system, the data gets stored in the cloud or the local server but considering the large amount of data generated by IoT devices, the cost of cloud storage can rise significantly. Also, due to limitations in bandwidth and the distance of cloud data centers, data exchanges might happen with a longer than expected response time. By the end of 2021, it was projected that there were 25 billion connected devices of all sorts, producing tremendous amount of data. This quantity is too large and requires a lot of storage and bandwidth demands. Simultaneous attempts by numerous devices to reach the cloud can lead to a severe fall of performance. It is thus crucial to adopt new technologies, such as edge computing, to facilitate processing and put smaller pressure on the network. Edge computing is basically a distributed computing paradigm that takes computation and data storage closer to the data sources. When edge device computations are performed in combination with cloud technology, it is known as cloud edge computing or edge cloud. Edge cloud is a promising new way of combining the storage capacities of cloud computing with the data gathering potential of edge computing. Some data storage could be done in the cloud, while the most necessary processes could run at the edge, in a more responsive and robust way. Thus, the data collected on the cloud can be used for improved data analytics, research, and drive innovation. Network connectivity plays a crucial part in accomplishing this and edge devices must be able to function isolated from the rest of the network. The combination of these two technologies could also improve the edge device's security and privacy.

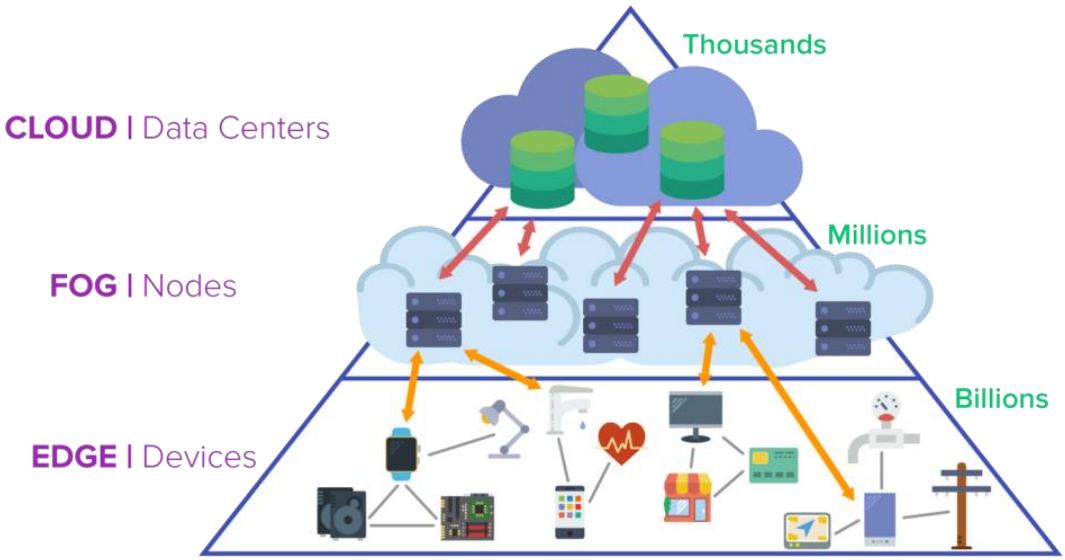

Figure 9. Basic Edge Computing architecture

Chapter 4. Databases

4.1 Introduction to Databases

A database is a shared collection of structured information, or logically related data, usually stored in a computer system and designed to be used by many departments and users of an organization. A database is often controlled by a database management system (DBMS) and has come a long way since its inception in the early 1960s. The original systems that were used to manipulate and store data were called navigational databases, with notable examples being the hierarchical and network databases. Even though these early models were simple, they suffered from inflexibility so, by the 1980s, relational databases became popular, followed by object-oriented databases in the 1990s. In more recent years, the rapid growth of the internet, and of its subsequent needs for greater speed and processing, saw the rise of NoSQL databases, while modern trends like cloud and self-driving databases are breaking new ground every day. Databases are often characterized not only as an organization's operational data but also a description of this data, known as the system catalog. A database can also be defined as logically related meaning that, by analyzing the information stored inside it, entities, attributes, and relationships can be identified.

4.1.1 Database Types

There exists a variety of types of databases, some of whom are listed below.

- Relational databases

The organization of items in a relational database is made with the help of set of tables with columns and rows. They are the most flexible and adequate way of accessing structured information.

- Object-oriented databases

Just like in object-oriented programming, an object-oriented database is represented using objects. Some of its features are Query language (for finding objects and retrieving data from the database), Transparent Persistence (for data manipulation) and ACID Transactions (so that all transactions are complete without conflicting changes).

- Distributed databases

A distributed database consists of two or more files located in different sites. Portion of the database may be stored on multiple computers, located in the same physical location, or scattered over entirely different networks.

- Data warehouses

A data warehouse is repository of data collected by various operational systems, specifically designed for fast query and analysis. Business analysts and data scientists can analyze and get access to data from different sources using data warehouses. Thus, data warehousing is commonly a part of a wider data management strategy.

- NoSQL databases

A NoSQL, or nonrelational database, is used to store and manipulate unstructured and semi structured data and are widely popular since web applications became more and more prominent.

- Graph databases

A graph database, or semantic database, stores, queries and modifies network graphs.

- Multimodel database

Multimodel databases supports multiple data models and combines them into one single database. This means they can accommodate various data types.

- Cloud databases

A cloud database is a pool of data that resides on a private, public, or hybrid cloud computing platform.

- Document/JSON database

Document databases store data in JSON format instead of columns and rows. They were designed to store, manage, and retrieve document-oriented information.

- Self-driving databases

Self-driving databases, or autonomous databases, are the most innovative type of database so far. They are cloud-based and use machine learning for a variety of functions like security, updates, backups, and other routine management tasks.

4.2 The Database Management System (DBMS)

A DBMS is a system software for creating, defining, and managing databases. A database usually requires a comprehensive database software program to serve as an interface between the database and its end users. This way, users can update and manage the organization and optimization of information. It is also easier to supervise and control the databases, thus facilitating the processes of performance monitoring, tuning, and backup and recovery. Some popular database software or DBMSs are MySQL, Microsoft Access, Microsoft SQL Server, and Oracle Database.

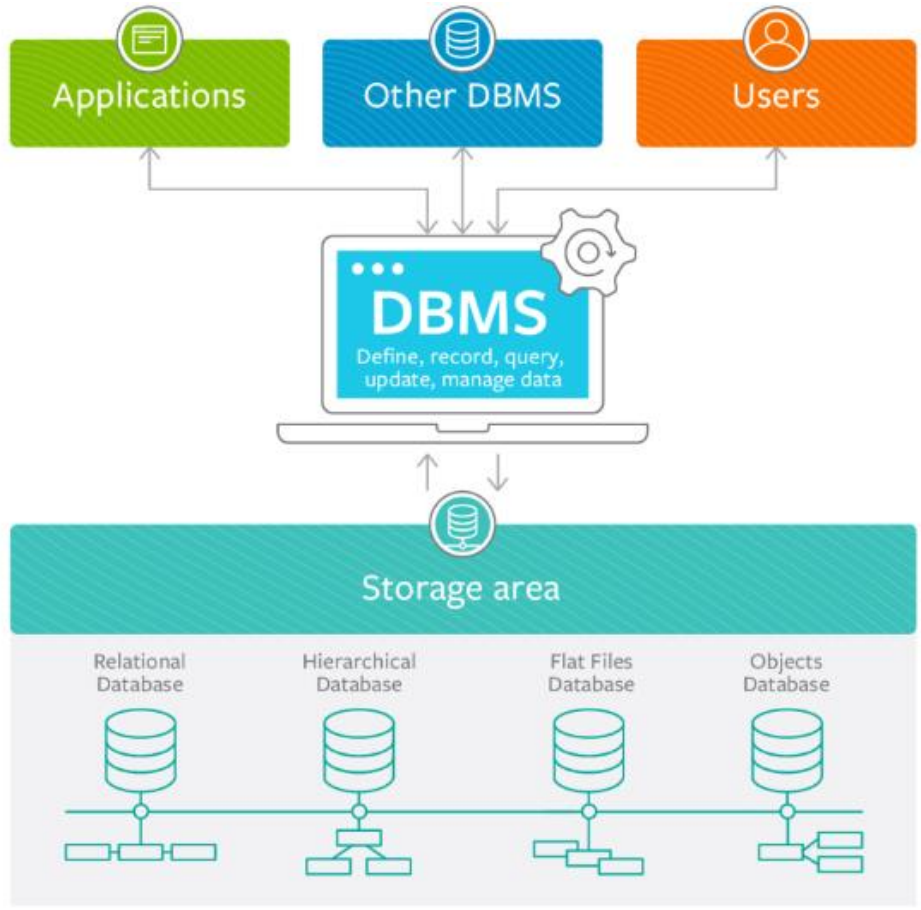

Figure 10. Schematic of a DBMS system

4.2.1 DBMS components

A Database Management System consists of a multitude of integrated components that provide a sophisticated and consistent environment for creating, accessing and modifying data in databases. These components are listed below.

- Storage engine.

The storage engine is used to store data. The DBMS must interface with a file system at the operating system (OS) level to store data. It can store data or interface with the actual data at the file system level by making use of additional components.

- Metadata catalog.

Also called a database dictionary, a metadata catalog acts as a repository for all database objects created. Upon their creation, the DBMS immediately registers information about them in the metadata catalog, which in turn is used verify user requests for data. Database objects, schemas, programs, security, performance, and communication are just some of the information included inside the metadata catalog.

- Database access language.

A database access language is an API used to access the data in a DBMS and can also be used to create database objects and secure and authorize access to the data. SQL is an example of a database access language.

- Optimization engine.

The main task of an Optimization engine is to parse database access language requests and turn them into actionable commands for accessing and modifying data. It also provides insights into the performance of the database in terms of optimizing the database itself and queries.

- Log manager.

The DBMS records all data changes into a record known as the log, and the log manager task is to ensure that log records are made efficiently and correctly. Data integrity is ensured with the use of the log manager, and it can also create backups and run recoveries with the help of data utilities.

- Data utilities.

Data utilities are made to control and manage database activities and consist primarily of reorganization, backup, recovery, integrity check, loading and unloading data, and repairing the database.

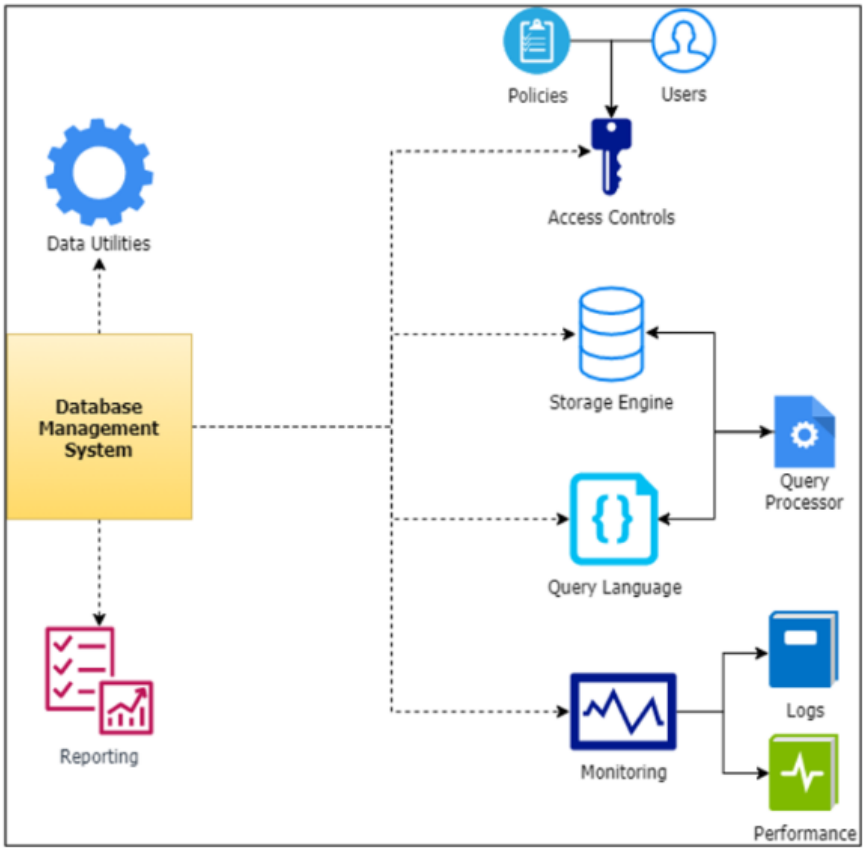

Figure 11. DBMS components

4.3 Advantages and Disadvantages of DBMSs

Employing a database management system has a lot of clear and promising advantages. Unfortunately, there are also disadvantages. Some of the advantages and disadvantages of DBMS are examined below.

4.3.1 Advantages of DBMSs

The advantages of DBMS are as follows:

- Data exchange

In DBMS, authorized database users can exchange data. Each user has individual access privileges to the database, which is readily available to the admin. He has the power to add users to the database.

- Data protection

When it comes to data privacy and security standards, a database management system (DBMS) has a very high security level, protecting all precious information from unauthorized access.

- Data integrity

By unifying multiple files into a single file, DBMS allows data integrity, thus facilitating the decrease of data duplicity, integration, and redundancy.

- Data backup

Data loss is a major concern for all organizations. DBMS solves this problem by automatically taking backup and recovery of the database.

- Multiple users

DBMS allows users to access the same database simultaneously without any conflicts.

4.3.2 Disadvantages of DBMSs

The disadvantages of DBMS are as follows:

- Complexity

A DBMS is quite a complex piece of software. In order to take full advantage of it, it is important for database designers, administrators, and developers to understand this functionality. Failure to do might seriously undermine the functionality of an organization.

- Cost

The cost of DBMS varies significantly depending on many factors, but it is usually required to have sophisticated hardware, software, and highly skilled personnel to maintain it. Training, licensing, and regulation compliance costs can be substantial.

- Size

The steady increase of the amount of data that is stored in a database can lead to many problems. Huge amounts of data can lead to database systems not providing good results and not running efficiently. That's why size is another disadvantage of the DBMS.

- Performance

Performance is also an important drawback of DBMSs, especially for smaller organizations and companies where the speed of the database systems are lower.

4.4 Database Security

Security is a critical aspect of a database's functionality and longevity. Misconfigurations, vulnerabilities, and carelessness are just some of the causes that can lead to attacks and breaches in the system. It is thus important to examine the most common threats and challenges faced by databases, as well as some promising defense mechanisms.

4.4.1 Common Threats and Challenges

- Insider threats

An insider threat is a security threat coming from a source with privileged access to the database, such as a malicious attacker, a negligent employ, or an infiltrator. Insider threats are considered one the most common causes of database security breaches and tend to happen when too many employees hold privileged user access credentials.

- Human error

Accidents, weak passwords, password sharing, and other unwise user behaviors make up half of all reported data breaches.

- SQL/NoSQL injection attacks

The insertion of arbitrary SQL or non-SQL attack strings into database queries served by web applications or HTTP headers can be a serious threat. Organizations need follow secure web application coding practices and to avoid these attacks.

- Denial of service (DoS/DDoS) attacks

In a denial of service (DoS) attack, the attacker floods the database server with so many requests that the server can no longer function properly and sometimes even crashes.

- Malware

Malware can exploit vulnerabilities and cause damage to the database may and can arrive via any endpoint device connecting to the database's network.

4.4.2 Defense Mechanisms

- Database monitoring

Scanning the database from breaches with the use of monitoring software allows users to react to potential attacks. Escalation protocols can also be used to keep sensitive data safer in case of an attack as well as organizing cybersecurity penetration tests.

- Firewalls

Firewalls are the first wall of defense for preventing malicious access attempts, with the most known to secure a network being a packet filter firewall, a stateful packet inspection and a proxy server firewall.

- Data encryption protocols

Encryption is crucial when moving or storing sensitive user information, thus setting up data encryption protocols can have a significant effect on dealing with a breach. In case of loss, the information that was stollen remains safe.

- Regular backups

Backups need to be created regularly to prevent the loss of sensitive information from malicious activity or data corruption. The backup should also be stored and encrypted in a separate server to be recoverable.

- User authentication

Compromised passwords are the cause for the majority of database breaches and has to do with the human-error aspect of creating weak passwords. A multi-factor authentication process can be used to fight this problem.

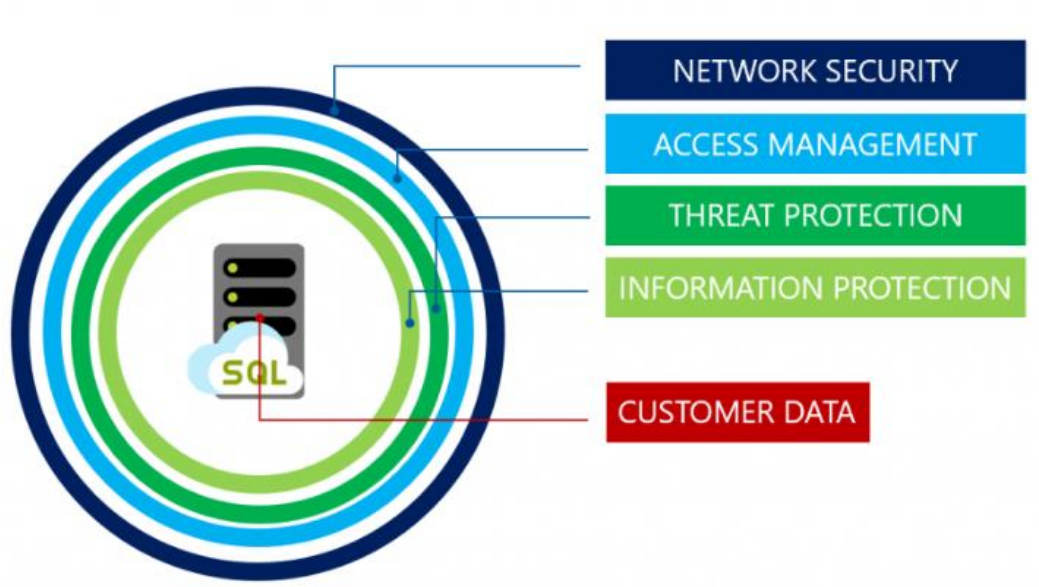

Figure 12. Database Security Mechanisms

4.5 Amazon Aurora

One of the most worth mentioning databases currently out on the market is Amazon Aurora. Amazon Aurora is an affordable cloud based relational database compatible with MySQL and PostgreSQL, that combines the performance and availability of common enterprise databases with the integrity and cost-effectiveness of open-source databases. Management of an Aurora database is automated and so is the backing up of its data. It is designed for high performance and availability at global scale, and provides high level security continuous backups, serverless compute, and integrations with other AWS services. It is also characterized by its high level of scalability, availability, durability, and cost-effectiveness. It is used in sectors like Enterprise Applications, Software as a Service (SaaS) Applications and Web and Mobile Gaming.

Chapter 5. Project Implementation

5.1 Elastic Beanstalk and Spring Boot

To deploy a Spring Boot application in Elastic Beanstalk, all that is needed is the bundled code. In our use case, we used maven to manage all the necessary dependencies and build the code. Maven offers us a straight-forward solution to bundle together the code and the dependencies with the command:

```
mvn clean package
```

After running the command, a jar file is generated in the target folder. We are now ready to start using Elastic Beanstalk.

After logging in to the AWS Management Console, we open the Elastic Beanstalk service. We choose to create an application and fill in the required fields. In our case, we used the Java 11 Platform. We continue by uploading the generated jar file and proceed by creating the application. Elastic Beanstalk will upload and deploy the application. Once the application is up, the service will run health checks at various times to ensure its smooth operation.

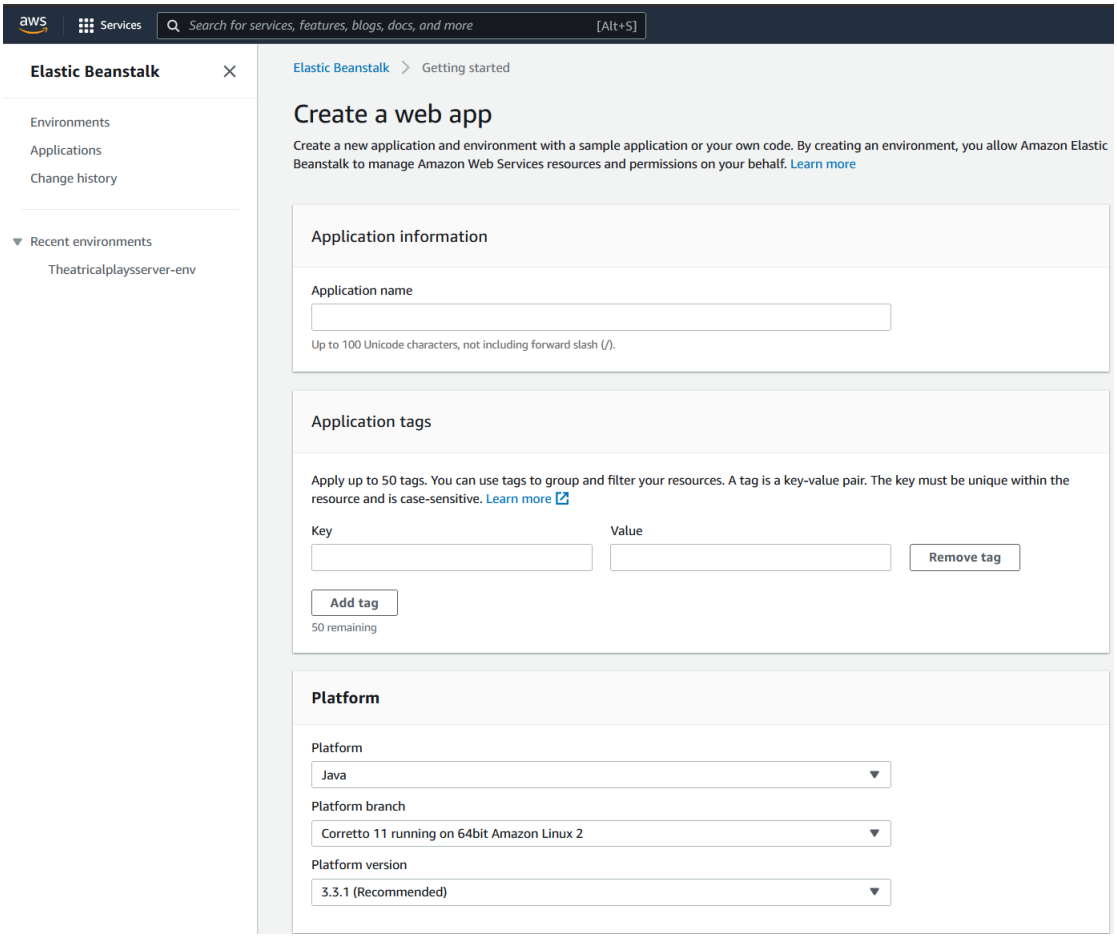

Figure 13 Elastic Beanstalk - Create application page

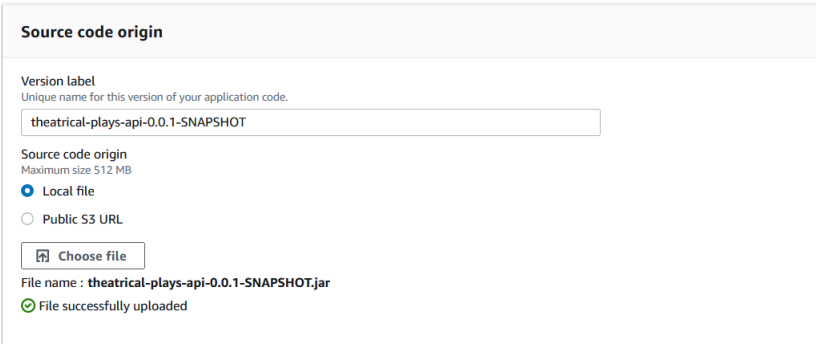

Figure 14 Elastic Beanstalk - Upload code section

With few simple steps we have managed to deploy our application, as Elastic Beanstalk handles everything under the hood. If we wanted, we could have specified different resources to be used such as a database, CPU, hard drive, load balancer and others.

5.2 Amazon Aurora Configuration

To create an Aurora database, we go to the Amazon Relation Database Service (RDS) and click to create a new database. We are presented with different options, Amazon Aurora, MySQL, Oracle, and others. We choose the Aurora option, and we can select between a MySQL or a PostgresSQL compatible database. Be extremely careful when inputting the username and password, as we will need them later to access the database. Another important option is whether we want to make our database public. If we choose not to, then only resources inside the selected VPC can connect to the database. For our use case, we enable public access to the database.

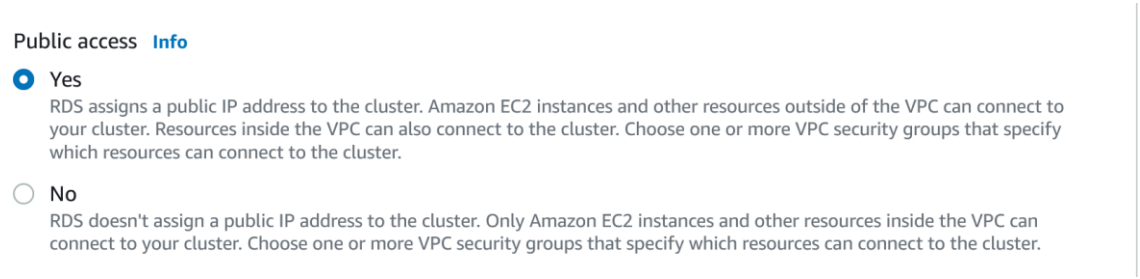

Figure 15 Amazon Aurora - Public access

Once we create the database and we have configured the right accesses, our Spring Boot application can access the database. We should first run any scripts needed for the initialization of the tables and insert any pre-existing data.

5.3 Elastic Search Server

To be able to utilize the benefits of ElasticSearch, we must run an instance of the ElasticSearch server. We can either download a zip file from the official site or we can pull an ElasticSearch image from dockerhub. We opted for the second option. Docker must be installed on our machine and then we run the command

```
docker run --name es01 --net elastic -p 9200:9200 -p 9300:9300 -e
"discovery.type=single-node" -it
docker.elastic.co/elasticsearch/elasticsearch:7.17.4
```

Once the service is running, we have to provide our Spring application with the ElasticSearch url. We add the below line to the application.properites file

```
elasticsearch.url=localhost:9200
```

5.4 Spring Boot Application

To initialize our Spring project, we head to https://start.spring.io/. There we can select all the configuration for our project. We can select the build tool (Maven or Grandle), the programming language (Java, Kotlin or Groovy), Spring boot version and some other settings. We also choose which dependencies (external libraries) to use in our project. After we have chosen the project configuration, we generate the project and now we have our Spring Boot application.

5.4.1 Controller Annotation

Using the @RestController annotation, we mark the class as ready to receive web requests. It is the entrance point to our application. We can also specify the request URL by utilizing the @RequestMapping. The @RequestMapping is also used at the class level. Spring also provides us with different annotations for the different HTTP methods. We have @PostMapping used for HTTP Post requests, @GetMapping used for HTTP Get requests etc. These annotations are used at the method level and we can also specify a different URL for each method.

```java
@RestController
@RequestMapping("people")
public class PersonController {

    @Autowired
    private PersonService personService;

    @PostMapping
    public APIResponse createPerson(@RequestBody CreatePersonDTO requestDTO) {

        personService.createPerson(requestDTO);

        return new APIResponse();
    }

    @GetMapping
    public APIResponse getPeople() {
        GetPeopleDTO response = personService.getPeople();

        return new APIResponse(response);
    }

    @GetMapping("{id}")
    public APIResponse getPerson(@PathVariable int id) {
        GetPersonDTO response = personService.getPerson(id);

        return new APIResponse(response);
    }

    @GetMapping("{id}")
    public APIResponse deletePerson(@PathVariable int id) {
        personService.deletePerson(id);

        return new APIResponse();
    }

    @GetMapping("search")
    public APIResponse searchPeople(@RequestParam String q) {
        SearchPeopleDTO response = personService.searchPeople(q);

        return new APIResponse(response);
    }
}
```

Figure 16. Rest Controller example

In lines 14 and 22 we have not defined a specific mapping for these methods. The URL for these methods is the one we have defined in the @RequestMapping annotation. We should also take into consideration that we cannot have two same endpoints with the same URL. Spring will throw an error if we attempt to.

In lines 29 and 36 we are expecting any integer number. This way we can create dynamic URLs creating a more robust API. The URL for these methods will be /people/{id}. The {id} parameter can be any number.

In line 15 we have used the @RequestBody annotation alongside a Data Transfer Object (DTO). With the help of the DTO, we define the body of the request we expect. We can require some fields to be present in the request and we can validate the fields.

5.4.2 Service Annotation

Services classes are conventionally being called by the controller. They are used to separate the business layer from the API interface. To mark a class as a service class simply annotate it with @Service. Any business functionalities will be implemented here. Validations and mappings can also be performed inside a service class.

```java
@Service
public class PersonService {

    @Autowired
    private PersonRepository personRepository;

    @Autowired
    private PersonMapper personMapper;

    @Transactional
    public void createPerson(CreatePeopleDTO requestDTO) {
        Person person = new Person();
        person.setFullName(requestDTO.getName());
        person.setAge(requestDTO.getAge());

        personRepository.save(person);
    }

    @Transactional(readOnly = true)
    public GetPeopleDTO getPeople() {
        final List<Person> people = personRepository.findAll();

        return personMapper.map(people);
    }

    @Transactional(readOnly = true)
    public GetPersonDTO getPerson(int id) {
        final Person person = personRepository.findById(id)
                                              .orElseThrow(RuntimeException::new);

        return personMapper.map(person);
    }

    @Transactional
    public void deletePerson(int id) {
        personRepository.deleteById(id);
    }
```

Figure 17. Service class example

Every method in the service class is annotated with @Transactional. With @Transactional we define the scope of a single database transactional. If an error is being thrown, any changes to the database will not be persisted. This is especially helpful when we are making a series of database updates in a single transaction. If one update fails, then a rollback will occur.

5.4.3 Repository Annotation

Once the service class has made the necessary validations and has prepared the data, it will make a repository call. The Repository annotation is used in classes that handle CRUD operations on database tables. Repositories provides as with build-in methods to retrieve and update data.

```
@Repository
public interface PersonRepository extends JpaRepository<Person, Integer> {
}
```

Figure 18. Repository class example

We have a variety of build-in repositories to choose from. For our project we chose JpaRepository as it provides us the necessary methods for executing CRUD operations, as well as methods to paginate our results.

```
@NoRepositoryBean
public interface JpaRepository<T, ID> extends PagingAndSortingRepository<T, ID>, QueryByExampleExecutor<T> {
    List<T> findAll();

    List<T> findAll(Sort sort);

    List<T> findAllById(Iterable<ID> ids);

    <S extends T> List<S> saveAll(Iterable<S> entities);

    void flush();

    <S extends T> S saveAndFlush(S entity);

    <S extends T> List<S> saveAllAndFlush(Iterable<S> entities);

    /** @deprecated */
    @Deprecated
    default void deleteInBatch(Iterable<T> entities) { this.deleteAllInBatch(entities); }

    void deleteAllInBatch(Iterable<T> entities);

    void deleteAllByIdInBatch(Iterable<ID> ids);

    void deleteAllInBatch();

    /** @deprecated */
    @Deprecated
    T getOne(ID id);

    T getById(ID id);

    <S extends T> List<S> findAll(Example<S> example);

    <S extends T> List<S> findAll(Example<S> example, Sort sort);
}
```

Figure 19. JpaRepository

5.4.4 Java Persistence API

When we are developing an application, we are coding in the programming language of our choice, but the data are stored and updated using SQL. To bridge the gap between the database and our application we are using Object-Relation Mapping (ORM). ORM is the concept of being able to write queries using

the object-oriented paradigm of our programming language, instead of using SQL to interact with our database.

Java Persistence API (JPA) is one possible approach to ORM. JPA enables the developer to work directly with objects instead of SQL statements. It allows the developer to create, read, update and delete from relational databases using Java objects.

JPA though does not provide the implementation. Its role is to set a number of guidelines to standardize the above operations. The JPA jar only contains interfaces. So JPA cannot be used on its own. The implementation of the JPA guidelines is provided by a JPA provider. So the developer uses a JPA provider to describe the operations using Java, and the provider translates the java code to SQL statements. Hibernate, Eclipse Link and TopLink are some of the available JPA providers. In our project we use Hibernate.

5.4.5 Hibernate

Hibernate is the most popular JPA provider. It is an open-source, ORM tool that equips our application with a mapping mechanism between Java objects and database tables. Hibernate Query Language (HQL) is the object-oriented version of SQL, which we use to generate database queries. One of the benefits of using HQL, is that we operate independently of the database engine. Were we to change the database from PostgreSQL to MySQL, we would also have to update the queries to conform to the engine's standards. With HQL we avoid this issue altogether.

5.4.6 Persistence Context

For Hibernate to be able to create, update and delete database rows, it needs to be able to track all the changes happening to the data. Persistence Context is used for that purpose. It is a staging area between our application and the data store. It is a first-level cache where we store all the entities we fetched from the database, or all the entities that are to be saved in the database. Persistence context keeps track of all the changes we make to these entities and is responsible to synchronize the changes back to the database. It operates in a transaction level, and any changes made inside a transaction are marked, and when the transaction is complete, these changes are transferred to our database. We can access the Persistence Context, both with JPA and Hibernate. JPA has the Entity Manager class and Hibernate has the Session class.

5.4.7 Hibernate Lifecycle

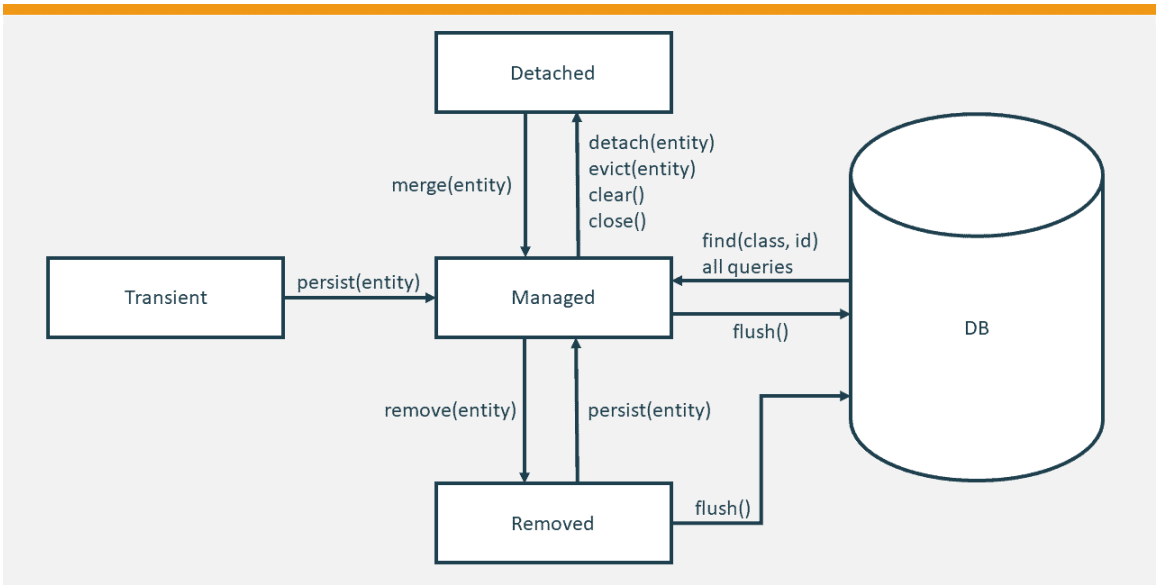

Figure 20. Hibernate Lifecycle

To be able to manage the changes, Hibernate uses four states to mark the entities.

- Transient

A newly created entity is labeled as transient. Hibernate is not yet aware of its existence. A transient entity is a simple Java object not connected to the database. Subsequently, any changes happening to the entity, will not affect any SQL rows. We can affiliate the entity with a row after creating the required connections. A simple example of a transient entity would be instantiating a new entity via its constructor.

```
Person person = new Person();
person.setFirstName("John");
person.setLastName("Smith");
```

- Managed

For an entity to be persisted to the database, it needs to be in the managed state. Any updates happening to a managed entity, will also be persisted to the database. We can mark an entity as managed using functions like persist(), save(), update()

```
Person person = new Person();
person.setFirstName("John");
person.setLastName("Smith");
entityManager.persist(person);
```

- Detached

An entity that was previously managed, but currently is not, is in the detached state. It can be considered as a simple Java object that its' values correspond to a database row. The difference with a managed entity is that any updates happening to it, will not be persisted to the database. Before detaching an entity, the persistence context should be flushed to avoid losing any pending changes

```
entityManger.detach(person);
```

- Removed

Once an entity has been marked as removed, its corresponding database row will be deleted. However, the delete action will not happen immediately. Just before the database connection closes, Hibernate will issue a DELETE statement to remove the row.

```
entityManger.remove(person);
```

5.4.8 Entity Annotation

The Entity annotation is used to map our Java classes to database tables. It is placed at the class level. Each entity corresponds to a database table. The fields of the class represent the columns of the table. By default, the name of the class is the same as the name of the table in snake case. To select a different table name, we can use

```
@Table(name = "table_name")
```

The same can be achieved for the columns' name using the Column annotation above each field. Once a Java object has been annotated with Entity, the Persistence Context will start monitoring and managing the updates. Of course, we should first call the relevant methods, to mark the entity as managed.

```
@Entity
@Table(name = "roles")
class Role {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "ID", nullable = false)
    var id: Int? = null

    @Column(name = "Role", nullable = false)
    var role: String? = null

    @Column(name = "SystemID", nullable = false)
    var systemID: Int? = null

    @Column(name = "timestamp", nullable = false)
    var timestamp: Date? = null
    override fun toString(): String {
        return "Roles{" +
                "ID=" + id + '\'' +
                "role=" + role + '\'' +
                "systemID=" + systemID + '\'' +
                "timestamp=" + timestamp + '\'' +
                '}'
    }
}
```

Figure 21. Project entity – Role

5.4.9 Id Annotation

As database tables must have a primary key column, an entity object must have an id field. To mark a field as id, we simply annotate it with @Id. One annotation that goes hand with hand with @Id is the @GeneratedValue. With @GeneratedValue we can define how we want to auto-increment our primary key. The simplest generation type to use is Identity. It relies on an auto-incremented database column and lets the database generate a new value with each insert operation.

5.4.10 EmbeddedId Annotation

If we want to create a more complex primary key, we can use the @EmbeddedId annotation. First, we create a new class having the fields we want to use in the primary key. We annotate this class with @Embeddable.

```java
@Embeddable
public class PersonId {

    private String firstName;

    private String lastName;

    public PersonId(String firstName, String lastName) {
        this.firstName = firstName;
        this.lastName = lastName;
    }

    //setters and getters
}
```

Figure 22. Embeddable annotation

And on the entity class, we add our id class and annotate it with @EmbdeddedId.

Chapter 6. API Endpoints

All responses from the server are being encapsulated within the ApiResponse container object. It consists of three fields: data, errors and status. The data field contains the requested data. The errors field contains any possible errors occurred during the request. When the errors field is empty the request was successful. The status field contains the HTTP status code. Below is a response example.

```
{
   "data": "Some data",
   "errors": null,
   "status": "OK"
}
```

6.1 Authentication

All endpoints require user authentication. The users are split into two authorities, admin or user. Admins can fetch, create, update or delete a resource, while users can only fetch a resource. Authentication is implemented using JSON Web Tokens (JWT) as credentials. On login, the server first checks if the provided username and password are valid. If they are, the server creates an encrypted token (JWT) holding all the necessary user details for authentication. The JWT can hold the user email, user id, their authorities and anything else that is needed. The client keeps the JWT in storage, and on every request adds the JWT on the Authorization header. The server then decrypts the message and if it is valid and the user has the necessary authority, it allows the request to procced. Otherwise, access is denied.

Register new user request

This request is used to create a new user.

URL	POST /api/users/register
Parameters	email (body variable). The email of the user
	password (body variable). The password of the user
	authorities (body variable). The authorities of the user (ADMIN | USER)
Response	void

Validate JWT token request

This request is used to create a new user.

URL	POST /api/users/validate
Parameters	token (request parameter). The JWT to validate
Response	void

Login request

This request is used to create a new user.

URL	GET /api/users/login

Parameters	email (request parameter). The email of the user
	password (request parameter). The password of the user
Response	Header -> Authorization: Bearer JWT

6.2 Person

Get person request

This request is used to retrieve a person and their image by the provided identifier.

URL	GET /api/people/{id}
Parameters	id (path variable). The identifier of the person to retrieve
Response	PersonDTO {id: Int, fullName: String, image: String}

Get people request

This request is used to retrieve all people. Pagination is optional.

URL	GET /api/people/
Parameters	page (request parameter). The index of the page to return, optional.
	size (request parameter). The size of the page, optional.
Response	Page<PersonDTO> {id: Int, fullName: String, image: String}

Get people by role request

This request is used to retrieve all people. Pagination is optional.

URL	GET /api/people/role
Parameters	value (request parameter). The role provided to filter the results
	page (request parameter). The index of the page to return, optional.
	size (request parameter). The size of the page, optional.
Response	Page<PersonDTO> {id: Int, fullName: String, image: String}

Get people by the first letter of their name request

This request is used to retrieve people filtered by the provided letter. Pagination is optional.

URL	GET /api/people/letter
Parameters	value (request parameter). The letter provided to filter the results
	page (request parameter). The index of the page to return, optional.
	size (request parameter). The size of the page, optional.

Response	Page<ProductionRoleDTO> {productionId: Int, title: String, url: String, producer: String, mediaURL: String, duration: String, description: String, role: String}

Get production and role by person id request

This request returns all the productions a person participated in and the corresponding role. Pagination is optional.

URL	GET /api/people/{id}/productions
Parameters	id (path variable). The identifier of the person
	page (request parameter). The index of the page to return, optional.
	size (request parameter). The size of the page, optional.
Response	Page<PersonDTO> {id: Int, fullName: String, image: String}

Search people by field request

This request is used to search people by the provided query

URL	GET /api/people/search
Parameters	q (request parameter). The query to search. (example: q=fullName~Giorgos,id:1920)
	page (request parameter). The index of the page to return, optional.
	size (request parameter). The size of the page, optional.
Response	Page<PersonDTO> {id: Int, fullName: String, image: String}

Get photos by person id request

This request is used to retrieve a person and their image by the provided identifier.

URL	GET /api/people/{id}/photos
Parameters	id (path variable). The identifier of the person
Response	Set<ImageDTO> {id: Int, imageURL: String, personId: Int}

Create new person request

This request is used to create a new person.

URL	POST /api/people
Parameters	fullName (body variable). The full name of the person
Response	CreatePersonResponseDTO {newUserId: Int, existingUserId}

If the person does not exist (full name is unique), newUserId holds the new auto-generated id, otherwise it is null. If the person already exists, then existingUserId holds the id of the person.

6.3 Image

Create new image request

This request is used to create a new image and associate with a person.

URL	POST /api/images
Parameters	imageURL (body variable). The full name of the person
	personId (body variable)
Response	CreateImageResponseDTO {imageId}

Delete image request

This request is used to delete an image.

URL	DELETE /api/images/{imageId}
Parameters	
Response	void

6.4 Role

Create new role request

This request is used to create a new role.

URL	POST /api/roles
Parameters	role (body variable). The name of the role
Response	CreateRoleResponseDTO {newRoleId: Int, existingRoleId: Int}

If the role does not exist (role name is unique), newRoleId holds the new auto-generated id, otherwise it is null. If the role already exists, then existingRoleId holds the id of the role.

Delete role request

This request is used to delete a role.

URL	DELETE /api/roles/{roleId}
Parameters	
Response	void

If the role is used in any contributions an exception will be thrown.

6.5 Venue

Get venue by the provided id request

This request is used to retrieve a venue by the provided identifier.

URL	GET /api/venues/{id}

Parameters	id (path variable). The identifier of the venue to retrieve
Response	VenueDTO {id: Int, title: String, address: String}

Get venues request

This request is used to retrieve all productions. Pagination is optional.

URL	GET /api/venues
Parameters	page (request parameter). The index of the page to return, optional.
	size (request parameter). The size of the page, optional.
Response	Page< VenueDTO > {id: Int, title: String, address: String }

Get productions by venue id request

This request is used to retrieve all productions located in the given venue

URL	GET /api/venues/{id}/productions
Parameters	id (path variable). The identifier of the person
	page (request parameter). The index of the page to return, optional.
	size (request parameter). The size of the page, optional.
Response	Page<ProductionDTO> {productionId: Int, title: String, url: String, producer: String, mediaURL: String, duration: String, description: String, role: String}

Create/Update venue

This request is used to create a new venue or update an existing one.

URL	POST /api/venues
Parameters	title (body variable). The title of the venue
	address (body variable). The address of the venue
Response	CreateVenueResponseDTO {newVenueId: Int, existingVenueId: Int}

If the venue does not exist (title is unique), newVenueId holds the new auto-generated id, otherwise it is null. If the venue already exists, then existingVenueId holds the id of the venue and the address is updated.

Delete venue request

This request is used to delete a venue.

URL	DELETE /api/venues/{venueId}
Parameters	

Response	void

If the venue is used in any events an exception will be thrown.

6.6 Organizer

Create/Update organizer request

This request is used to create a new organizer or update an existing one.

URL	POST /api/venues
Parameters	name (body variable). The name of the organizer
	address (body variable). The address of the organizer
	town (body variable). The town of the organizer
	postcode (body variable). The post code of the organizer
	phone (body variable). The phone of the organizer
	email (body variable). The email of the organizer
	afm (body variable). The afm of the organizer
	doy (body variable). The doy of the organizer
Response	CreateOrganizerResponseDTO {newOrganizerId: Int, existingOrganizerId: Int}

If the organizer does not exist (afm is unique), newOrganizerId holds the new auto-generated id, otherwise it is null. If the organizer already exists, then existingOrganizerId holds the id of the organizer and their properties are updated.

Delete organizer request

This request is used to delete an organizer.

URL	DELETE /api/organizers/{organizerId}
Parameters	
Response	void

If the organizer is related with any productions, an exception will be thrown.

6.7 Production

Get production request

This request is used to retrieve a production by the provided identifier.

URL	GET /api/productions/{id}
Parameters	id (path variable). The identifier of the production to retrieve
Response	ProductionDTO {productionId: Int, title: String, url: String, producer: String, mediaURL: String, duration: String, description: String, role: String}

Get productions request

This request is used to retrieve all productions. Pagination is optional.

URL	GET /api/productions/
Parameters	page (request parameter). The index of the page to return, optional.
	size (request parameter). The size of the page, optional.
Response	Page<ProductionDTO> {productionId: Int, title: String, url: String, producer: String, mediaURL: String, duration: String, description: String, role: String}

Get latest productions request

This request is used to retrieve all productions sorted their event data

URL	GET /api/productions/latest
Parameters	page (request parameter). The index of the page to return, optional.
	size (request parameter). The size of the page, optional.
Response	Page<ProductionDTO> {productionId: Int, title: String, url: String, producer: String, mediaURL: String, duration: String, description: String, role: String}

Get people by production id request

This request returns all the people participating in a production and their role

URL	GET /api/productions/{id}/people
Parameters	id (path variable). The identifier of the person
	page (request parameter). The index of the page to return, optional.
	size (request parameter). The size of the page, optional.
Response	List<PersonRoleDTO> {id: Int, fullName: String, image: String, role: String}

Get events by production id request

This request returns all the events linked to a production

URL	GET /api/productions/{id}/events
Parameters	id (path variable). The identifier of the person
	page (request parameter). The index of the page to return, optional.
	size (request parameter). The size of the page, optional.

Response	List< EventVenueDTO> {eventId: Int, date: Date, image: String, pricaRange: String, venueId: Int, title: String, address: String}

Search productions by field request

This request is used to search productions by the provided query

URL	GET /api/productions/search
Parameters	q (request parameter). The query to search.
	page (request parameter). The index of the page to return, optional.
	size (request parameter). The size of the page, optional.
Response	Page<ProductionDTO> {productionId: Int, title: String, url: String, producer: String, mediaURL: String, duration: String, description: String, role: String}

Create/Update production

This request is used to create a new production or update an existing one.

URL	POST /api/productions
Parameters	title (body variable). The title of the production.
	description (body variable). The description of the production.
	URL (body variable). The production URL from viva.gr.
	producer (body variable). The name of the producer.
	mediaURL (body variable). Can be a youtube video, a twitter ad etc
	duration (body variable). The duration of the production
	organizerId. The id of the organizer
Response	CreateOrganizerResponseDTO {newProductionId: Int, existingProductionId: Int}

If the production does not exist (title is unique), newProductionId holds the new auto-generated id, otherwise it is null. If the production already exists, then existingProductionId holds the id of the production and its properties are updated.

Create event request

This request is used to create a new event and associate with a production.

URL	POST /api/productions/{id}/events
Parameters	venueId (body variable). The venue id that hosts the event.
	dateEvent (body variable). The date of the event.
	priceRange (body variable). The price of the event.
Response	void

Delete event request

This request is used to delete an event associated with a production.

URL	DELETE /api/productions/{id}/events/{eventId}
Parameters	
Response	void

Create contribution request

This request is used to create a new contribution and associate with a production.

URL	POST /api/productions/{id}/contributions
Parameters	personId (body variable). The person id that contributes to the production
	roleId (body variable). The role id associated with the person for the specific production.
	subrole (body variable). The secondary role of the person.
Response	void

Delete contribution request

This request is used to delete a contribution associated with a production.

URL	DELETE /api/productions/{id}/contributions/{contributionsId}
Parameters	
Response	void

Chapter 7. Future Improvements

7.1 Elastic Beanstalk alternatives

Elastic Beanstalk, although an easy-to use service, does not offer great flexibility around deployments. Deploying in EB is a time-consuming process. The entire zipped application has to be uploaded to AWS and then is to be downloaded to each targeted machine and then finally deployed. Troubleshooting is also difficult since everything happens under the hood. If a company has a dedicated DevOps department, then alternatives such as Qovery or Amazon Elastic Kubernetes Service, would be a better fit.

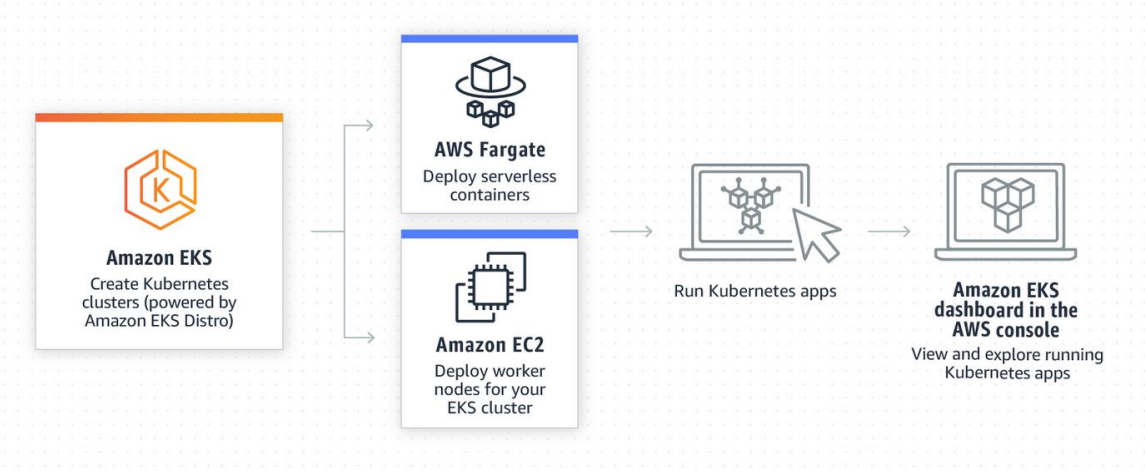

Figure 23. Elastic Kubernetes Service deployment flow

7.2 Amazon OpenSearch Service

Currently we have not configured ElasticSearch to AWS as it requires a more complex process. It is not as easy as running a container, as we can do in a remote server. Instead, Amazon offers the OpenSearch service, a project based on older version of ElasticSearch and Kibana. This has happened due to licensing disputes between the Elastic organization and AWS. Keep in mind that OpenSearch does not provide the full benefits that ElasticSearch does.

Bibliography

Books

[1] Baun, C., Kunze, M., Nimis, J., & Tai, S. (2011). Cloud Computing. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-20917-8

[2] Sehgal, N. K., & Bhatt, P. C. P. (2018). Cloud Computing. Springer International Publishing. https://doi.org/10.1007/978-3-319-77839-6

[3] Cloud Computing. (2010). In N. Antonopoulos & L. Gillam (Eds.), Computer Communications and Networks. Springer London. https://doi.org/10.1007/978-1-84996-241-4

[4] Cloud Computing. (2013). In Z. Mahmood (Ed.), Computer Communications and Networks. Springer London. https://doi.org/10.1007/978-1-4471-5107-4

[5] THOMAS M. CONNOLLY, CAROLYN E. BEGG, Database Systems A Practical Approach to Design, Implementation, and Management, Fourth Edition, UNIVERSITY OF PAISLEY, ISBN 0 321 21025 5

Publications

[6] Khalil, I., Khreishah, A., & Azeem, M. (2014). Cloud Computing Security: A Survey. In Computers (Vol. 3, Issue 1, pp. 1–35). MDPI AG. https://doi.org/10.3390/computers3010001

[7] Lascano, J., (2008), JPA implementations versus pure JDBC, Conference: Congreso de Ciencia y Tecnología de la ESPE-2008, At Quito, Ecuador

Internet sites

[8] https://www.techtarget.com/

[9] https://www.geeksforgeeks.org/

[10] https://www.javatpoint.com/

[11] https://www.w3schools.com/

[12] https://www.redhat.com/

[13] https://azure.microsoft.com/

[14] https://www.nibusinessinfo.co.uk/content/disadvantages-cloud-computing

[15] https://www.apriorit.com/dev-blog/549-authentication-as-a-service#:~:text=Authentication%20as%20a%20Service%20(AaaS,from%20various%20devices%20and%20networks.

[16] https://cybersecurity.att.com/blogs/security-essentials/cloud-based-siem

[17] https://imaginovation.net/blog/8-sectors-benefit-from-iot-development-in-2021/

[18] https://www.statista.com/chart/18819/worldwide-market-share-of-leading-cloud-infrastructure-service-providers/

[19] https://www.vogella.com/tutorials/JavaPersistenceAPI/article.html?fbclid=IwAR0EPpaPvm_bebanoGQ2xxanAbcqVR8v85FZZZrLtP7ZaqiOoO7y4p-1WAA

[20] https://thorben-janssen.com/entity-lifecycle-model/?fbclid=IwAR14sToogCdb66fMgVYCT0L-R3ZjkWGmUZf95lq4b5Xznaae1EDaDHFg-bU