Illinois Institute of Technology

School of applied Technology

Universidad Politécnica
de Madrid

Escuela Técnica Superior de Ingenieros Informáticos

Master of Information Technology and Management

Data Analytics and Management

Máster Universitario en Ingeniería Informática

Research Project

Factor: Web Repository of Transcribe Online Classes

Author: Pablo Ángel Álvarez Fernández

Project director IIT: Jeremy Hajek

Project director UPM: Damiano Zanardini

Chicago, 05/2020

Abstract

With a proliferation of Cloud based Speech-to-Text Services, from the major Cloud providers as well as several Opensource Speech-to-Text projects available, it can be difficult to decide where to start and how to make use of these technologies relating to the handling of recording artifacts that are the by-product of Online Education.

The fact that so many resources are available means that the computing, and technical barriers for applying speech recognition algorithms has decreased to the point of being a non-factor in the decision to use Speech-to-Text services. Other barriers such as price, time, and access to the services source code (software freedom) now can be factored into the decision of which platform to use.

This case study provides a beginning to developing a test-suite and guide to compare Speech-to-Text libraries and their out of the box accuracy while integrating these libraries into a single test application. Our initial test suite employs two models a Cloud model employing AWS S3 using AWS Transcribe while the on-premises Opensource model relies on Mozilla’s DeepSpeech. We present our findings and recommendations based on the criteria discovered.

Moreover, beyond comparisons of Speech-to-Text libraries we also investigate the artifacts produced by high quality Speech-to-Text and examine and report what new features become available in relation to text transcripts. The decision based on our requirements will be used to deploy a full-stack educational system. This system aims to develop a service that will facilitate student learning and help teachers create publications derived from their own classes, improving learning in general.

In order to deliver this test-suite, we also conducted research into the latest web development technologies to accomplish with emphasis on the security in order to produce a reliable and secure development process and to provide open access to this proof of concept for further testing and development.

Finally, the Kubernetes technology will be investigated to create a cluster and organize the services as an auto-scale single product.

Keywords

Speech-To-Text, Algorithms, FLOSS, Kubernetes, DeepSpeech, Cloud, Subtitles, Automatic Speech Recognition, Test Suites, Full-stack, Node, Angular, Mongo, Google Oauth 2.0, AWS, Containers, Pod, Replicas, Docker.

Resumen

Con la proliferación de servicios de voz a texto basados en la nube, de los principales proveedores Cloud, así como la aparición de varios proyectos de código abierto, puede resultar difícil decidir por dónde empezar y cómo hacer uso de estas tecnologías relacionadas con el manejo de artefactos de grabación que son el subproducto de la educación en línea.

El hecho de que haya tantos recursos disponibles significa que las barreras informáticas y técnicas para aplicar los algoritmos de reconocimiento de voz han disminuido hasta el punto de dejar de ser un factor en la decisión de utilizar los servicios de voz a texto. Otras barreras como el precio, el tiempo y el acceso al código fuente de los servicios (libertad de software) pueden ahora tenerse en cuenta en la decisión de qué plataforma utilizar.

El presente caso de estudio es el comienzo de la elaboración de un conjunto de pruebas y una guía para comparar las bibliotecas de voz a texto y su precisión inmediata, integrando al mismo tiempo esas bibliotecas en una sola aplicación de prueba. Nuestro conjunto de pruebas inicial emplea dos modelos: un modelo basado en herramienta cloud que emplea AWS S3 usando AWS Transcribe, mientras que el modelo de código abierto ejecutado en la red interna se basa en DeepSpeech de Mozilla. Presentamos nuestros hallazgos y recomendaciones basados en los criterios descubiertos.

Además, más allá de las comparaciones de las bibliotecas de Voz a Texto, también investigamos los artefactos producidos por la alta calidad de Voz a Texto y examinamos e informamos de las nuevas características disponibles en relación con las transcripciones de texto. La decisión basada en nuestros requisitos se utilizará para desplegar un sistema educativo completo. Este sistema tiene como objetivo desarrollar un servicio que facilite el aprendizaje de los estudiantes y ayude a los profesores a crear publicaciones derivadas de sus propias clases, mejorando el aprendizaje en general.

Con el fin de entregar este conjunto de pruebas, también llevamos a cabo una investigación sobre las últimas tecnologías de desarrollo web para lograr, con énfasis en la seguridad, producir un proceso de desarrollo fiable y seguro y proporcionar un acceso abierto a esta prueba de concepto para su posterior prueba y desarrollo.

Finalmente, se investigará la tecnología de Kubernetes para crear un cluster y organizar los servicios como un único producto automáticamente auto escalable.

Keywords

Speech-To-Text, Algorithmos, FLOSS, Kubernetes, DeepSpeech, Cloud, Subtítulos, Reconocimiento Automático de Voz, Test Suites, Full-stack, Node, Angular, Mongo, Google Oauth 2.0, AWS, Containers, Pod, Replicas, Docker.

Contents

1 Introduction 1

1.1 Motivations 1

1.1.1 E-Learning 1

1.1.2 Transcriptions 2

1.1.2.1 Methods 3

1.2 Large-Scale applications 4

1.2.1 Challenges 4

1.3 Application architecture patterns 5

1.3.1 Monolithic architecture 5

1.3.2 Microservice architecture 6

1.4 Objectives 7

1.4.1 Gantt 9

2 Speech-To-Text 13

2.1 Selection 14

2.2 Analysis Factors 14

2.3 AWS Transcribe Methodology 14

2.4 DeepSpeech Methodology 16

3 Design and Infrastructure 18

3.1 Requirements 18

3.2 Requests processing 19

3.2.1 Kubernetes stream 19

3.2.2 Angular stream 20

3.2.3 Node stream 22

4 Deployment and Implement 23

4.1 Angular 23

4.1.1 Models 23

4.1.1.1 Router Module 23

4.1.1.2 Globals 24

4.1.2 View 26

4.1.2.1 Navbar 26

4.1.2.2 Footer 27

4.1.2.3 Log In 28

4.1.2.4 Student 29

4.1.2.5 Professor 30

4.1.2.6 Admin 31

4.1.2.7 Edit Video 31

4.1.2.8 Show Video 32

4.1.3 Controller 33

4.1.3.1 Authentication guard 33
4.1.3.2 HTTP service 33
4.1.3.3 File service 34
4.1.3.4 Video phrase pipeline 34
4.2 Node 35
4.2.1 Auth 35
4.2.2 DeepSpeech integration 37
4.2.3 STT Method Comparation 37
4.2.4 Selection Criteria 38
4.2.5 Results 39
4.3 MongoDB 40
4.4 Testing 41
5 Reliable Delivery Process 43
5.1 Deployment 43
5.2 Kubernetes 44
5.2.1 Infrastructure as code 46
5.2.2 Final Resource Infrastructure 47
5.3 Jenkins 48
6 Conclusions 50
6.1 Future Improvements 51
7 Bibliography 52
8 Annex 54
8.1 Node code structure 54

Figures

Figure 1 Task overview Gantt 9
Figure 2 Meeting task 9
Figure 3 Scope stage 10
Figure 4 Research stage 10
Figure 5 Design stage 10
Figure 6 Deployment stage 11
Figure 7 CI/CD and Document drafting 11
Figure 8 Hours distribution Gantt 12
Figure 9 Kubernetes request processing 20
Figure 10 Angular MVC patern 21
Figure 11 Node code structure. Source: [15] 22
Figure 12 SPA Routes 24
Figure 13 Navbar view 27
Figure 14 Professor footer 27
Figure 15 Student footer 28
Figure 16 LogIn Test Suit 28
Figure 17 Hint languages 28
Figure 18 Student home site 29
Figure 19 Video accordion 29
Figure 20 Professor home site 30
Figure 21 Video card professor 30
Figure 22 Admin home site 31
Figure 23 Edit video site 31
Figure 24 Show video view 32
Figure 25 Edit transcripts 32
Figure 26 AWS measures 38
Figure 27 Deepspeech measures 38
Figure 28 Kubernetes cluster 45
Figure 29 Kubernetes cluster description 45
Figure 30 Kubernetes control plane. Source [28] 45
Figure 31 Yaml pod and service Kubernetes 46
Figure 32 Yaml ingress kubernetes 46
Figure 33 Web access to ingress endpoint 47
Figure 34 Pods description, running node detail 47
Figure 35 Console access to ingress endpoint 47
Figure 36 Complete CI/CD pipeline for microservices in azure. Source: [31] . 49
Figure 37 CI/CD pipeline Test Suite 49
Figure 38 Example view video transcriptions 50

Listings

Listing 1 ffmpeg command AWS Transcribe 15
Listing 2 AWS Transcribe JSON transcribe structure 16
Listing 3 Reading router params Angular 24
Listing 4 Global variables 25
Listing 5 Global classes 26
Listing 6 Footer fixed bottom 27
Listing 7 Example HTTP request 34
Listing 8 Mongoose functions for setting and validating a password 36
Listing 9 GRPC service proto description 37
Listing 10 Nested schemas MongoDB 41
Listing 11 Basic Angular component test 42
Listing 12 Basic Node.js test 42
Listing 13 Test Suit IaC Kubernetes 47

1 Introduction

1.1 Motivations

Online education has been in our lives for a while, it has been a good option for people that cannot access to presential education due to time requirements, price or other personal problems. Now it's being an increasing number of web services offering reliable online courses with real certifications like Udemy, coursera or edX. This last one is the first integration between online courses companies and some top university institutions. And each day more companies invest more on online courses to instruct their workers.

This service will both facilitate student learning and help teachers create publications derived from their own classes, enhancing learning. It is focused on IIT students and uses their authentication service, but it could be extrapolated to any university with the necessary video systems, effortless.

I believe that Speech-to-Text technology have become stable at the current time of the writing of this paper. We can find both private and open-source options. So, I will compare them and find the most suitable for the final service.

What is offered to IIT students is the ability to transcribe and search for keywords in their course videos, in addition to being able to export them as class notes.

On the other hand, professors would also increase their performance, when observing the sessions. They can reorganize the subject with more emphasis on the terms that have not been dedicated enough time to them, while reviewing their transcripts. And even paraphrase ideas from the transcripts of those lessons to create a book.

This application is going to emphasize on the security section, since these videos and transcriptions represent the most important assets of the university as a company.

1.1.1 E-Learning

Most of this online learning methods can be sub-divided on two main groups asynchronous and synchronous. While asynchronous learning is an approach centered on the student allowing more schedule flexibility, synchronous learning is a teacher-centered approach allowing more instant interaction with the students and a feeling of class group.

The system deployed follows the asynchronous style as it gives the students independency with unlimited access to the resources shared by the professors anywhere. They also can access to the resources in a linear or freeform way. The negative point its lack of direct interaction with the professor and other students.

David Leaser from IBM shows some e-Learning benefits:

- Reduced cost: the amount of time spent away from work is reduced.
- Efficient: an instructor can support multiple students in multiple locations.
- Globally consistent: e-learning provides a global solution to synchronize worldwide skills.
- Scalable: bandwidth expense and the infrastructure for transmitting data are the key variable costs to scale.
- Trackable: provide tracking mechanisms that record attendance, completion and time spent.
- Convenient: e-learning solutions allow a student to learn at their convenience, unique location and/or during their time preference. Students may pause and resume training.

He also provides method to measure the company's return on investment (ROI) for training, based only on employee productivity gains. "A productivity gain of just three minutes per employee per day could save a company with 1,000 employees at least $240,000 USD." [1]

This project tries to urge universities to claim their place over those profits. Any medium university has all the resources required to provide e-Learning, they just need to go a step forward and share their knowledge online.

1.1.2 Transcriptions

Indexing and transcribing are extremely useful ways of identifying the main themes of a recording and the approximate point at which they occur in the recording. This reduces the time needed to identify the content. In addition, they help researchers quickly determine if a recording is related to a selected topic.

While persons watch a video, they are using the senses of sight and hearing. The sight is considered an important sense as it is very tied to mobility and give us a huge amount of information of our environment. It is also tied to memory, as our minds capture images to reinforce the rest of the senses. This reinforcement over the hearing sense, help the understanding of the speech adding context as for example hand movement information or the lips movement. However, the most crucial information is sent via the hear sense. You still can understand most of the video closing your eyes. Sometimes it is even the unique source of information as it happens on the radio or static image videos.

What can be achieve by retrieving the transcriptions from their videos, is to show that text synchronized with the image can reinforce the most crucial source of information. As it will be received over two ways. Lastly, the heard sense also receives music or sound effects but in a class case, this is usually considered as noise. The main source of information is the professor voice who gives the speech to their students. Having the support of the transcriptions will also help in noise reduction and with difficulties to heard low sounds.

The Factor test suite has an application initially in the academic higher-learning arena relating to the concept of learning management systems. Most Universities in America use an LMS to contain recorded or ad-hoc class videos for students to watch or re-watch. Universities host additional video on YouTube or other streaming channels. Factor targets the captured audio and video for use in Speech-to-Text conversion in order to produce an internal written corpus for an education institution. This written or transcribed corpus can be presented back to a University body for additional works such as:

- Subtitling of streaming video
- Search engine capability with timestamp correlation (a modern Card Catalog) [2]
- Confidence scoring in closed captioning [3]
- Custom markdown capabilities and dynamic text creation for ebook creation (digital library)
- Public API access for historical research purposes

There is an additional hidden cost of using Speech-to-Text on a large corpus of video. Once a video has end up their useful lifetime, it is too big to be stored indefinitely. What I am proposing is to store only that piece of information that summarizes the file. Taking as an example a video 2:24:49 hours long with a quality of 720p, it will use approximately 1.3 GB of storage size or 1,352,497,909 bytes. While the text transformation would only require a size of 86,542 bytes. This minimization means a decrease of 99.993601% on the long-term storage requirements.

1.1.2.1 Methods

Analyzing financial facts, there are three different strategies while approaching the transcript of a video. [4]

First, use a professional transcription service. Standard rates for a professional North American transcriptionist range from $1.50 to $3 per minute of audio ($90-180 per hour of audio), depending on the audio quality and number of speakers. In addition, transcriptions require a minimum one-day wait. As we will see, this is the system with the highest cost and the slowest. But the quality of the results will be 99% of accuracy.

Secondly, choose software to do it. The price range for this setting would also include the hardware costs to about $1.44 per hour of text to transcribe. The higher the price services represent cloud services, in this case AWS Transcribe. [5] Furthermore, for example AWS gives other advantages such as more languages, no need of expertise training models, options to add vocabulary terms or large variety of file formats.

Thirdly, do it yourself with software and crowdsourcing. Build your own application, run your own Speech-to-Text library and use your audience (students, members, viewers) do work in conjunction with the library to perfect the transcripts and machine errors. This method does not have an upfront or per hour cost outside of acquiring hardware and the cost of electricity to tun the servers. The main disadvantage is the accuracy of the work, it may never be ended or even deliberately introduce mistakes. This model will require extra monitorization of the services.

The application is going to use a mix of second and third methods to create transcriptions on a cost-effective manner, creating a fast first prototype and fix them giving a small task to qualified users.

1.2 Large-Scale applications

Nowadays, as complexity in technology is increasing, you must step back and get an overview of the whole structure before addressing the issue. It is difficult to make a big invest on new large applications so initially these may not afford expensive systems to manage all the complexity. The problem comes while predicting the increase on requests that the application has to take care. If the predictions are too low, the application won't be able to respond to every request, resulting in long waits for users, which will stop using the service and if the predictions are too high the expenses won't meet the revenues.

1.2.1 Challenges

When building a large-scale application, the architect has to deal with some serious challenges upfront or the whole system could collapse. The first challenge is performance, these applications need to handle a large number of requests and users reaching our service simultaneously. This requires high performance to decrease the clients' waiting time on their queries. Another challenge which improves the performance is the scalability, creating more than one process to respond these queries is a way to increase that performance. The problem arises from connecting these independent processes and trying to make them behave as a single application, which is known as abstraction.

This large number of requests also promotes larger issues. They end up producing a large volume of data, with many different types. Because applications not only need to store user information or the product they offer, these applications also track logs or behavior over the time, for example. Another issue came from the fact that we live in a global world, there is a need of availability every day at every time, and these requests can come from multiple locations. The long distance a request travels, the more time it takes. This challenge can be fixed having several nodes in different locations. But that also increases the managing complexity.

Managing these difficulties require the application of some technologies which make our deployment process and posterior maintenance more reliable. [6]

1.3 Application architecture patterns

Most new companies today run their business processes in the cloud. Newer startups and enterprises which realized early enough the direction technology was headed developed their applications for the cloud.

Not all companies were so fortunate. Some built their success decades ago on top of legacy technologies - monolithic applications with all components tightly coupled and almost impossible to separate, a nightmare to manage and deployed on super expensive hardware.

Having a defined structure is very important while creating a Large-scale application. This kind of application has many points of failure and are usually developed by a team of programmers that may not interact every day, but they need to be able to allow standardized connections between the components that form the application. They have to produce a legible code in case another programmer has to follow the deployment or at the end, allow an easier maintenance within the application lifetime. [7]

As Douglas C. Schmidt states on his book Pattern-Oriented Software Architecture, “A pattern is a recurring solution schema to a standard problem in a particular context. Patterns help to capture and reuse the static and dynamic structure and collaboration of key participants in software designs. They are applied to resolve common design and implementation problems.” [8]

What these patterns end up adding to the whole structure are an increased clarity of roles, responsibilities, and relationships to one another.

1.3.1 Monolithic architecture

The applications that are supported over this architecture are mostly called legacy systems. This name is used to describe outdated or obsolete software programs. They mostly use old technologies and standards and may become more unstable as the company introduce new paths.

A monolith requires an expensive investment in hardware. Being a large and unique piece of software that is continuously growing, it has to run on a single system that has to meet your computing, memory, storage and networking requirements. Hardware of such capacity is both complex and expensive.

Since the entire application of the monolith is executed as a single process, the scaling of certain individual features is almost impossible, the monolith must be scaled together. Internally it supports an encrypted number of connections and operations. However, scaling the entire application means manually deploying a new instance of the monolith on another server, typically behind a load balancing device. This means doubling the cost and wasting some computing power.

In addition, during upgrades, patches or migrations of the monolith application, downtime occurs, and maintenance windows must be planned. Because, service interruptions will affect customers. It is true that there are solutions to minimize customer downtime by configuring monolith applications. One such configuration would be the active/passive high availability configuration. Although this can still be a challenge for system engineers. It requires keeping all systems at the same patch level.

If you consider a legacy system, written in older programming languages like Cobol or Assembler, it may be more economical to rebuild it from scratch as a native cloud application. A poorly designed legacy application should be redesigned and rebuilt from scratch following modern architectural patterns for microservices and even containers. Applications closely tied to data warehouses are also poor candidates for refactoring.

However, if the application is in suitable shape, companies can use an incremental refactoring approach. This ensures that new features are developed and implemented as modern microservices. These new pieces of code will be able to communicate with the monolith via APIs. But, the need to add code to the monolith is eliminated. This will slow down the degradation of the monolith. Meanwhile the monolith is slowly fading away while all or most of its functionality is being modernized into microservices. This incremental approach provides a gradual transition from a legacy monolith to a modern microservices architecture and enables the phased migration of application features to the cloud.

Once an enterprise chooses the path of refactoring, there are other considerations in the process. As the choice of which enterprise components should be separated from the monolith to become distributed microservices, a decision must be made on how to decouple the application databases to decrease the data complexity of the application logic, and how to test the new microservices and their dependencies. These are just some of the decisions an enterprise faces during refactoring.

The refactoring phase slowly transforms the monolith into a native cloud application that takes full advantage of the cloud's features, coding in new programming languages and applying modern architectural patterns. Through refactoring, a legacy monolithic application is given a second chance at life: to continue living as a modular system adapted to fully integrate with today's cloud automation tools and services.

1.3.2 Microservice architecture

These legacy apps need to be refactorized to break the highly interconnected structure. It will be turned into some smaller services before been moved to the cloud.

When thinking about a microservice, we can think about each of the pieces that make up a legacy application. These microservices are loosely coupled, each of which performs a specific business function. All the functions grouped together form the overall functionality of the original monolithic application. Microservices are easy to select and group based on color, size, shape, and require minimal effort to relocate when necessary.

The microservices based architecture is supported by the principles of Event Driven Architecture and Service Oriented Architecture (SOA), where complex applications are composed of small independent processes that communicate with each other through APIs over a network. APIs allow access to other internal services of the same application or external services and applications of third parties.

Each microservice is developed and written in a modern programming language, selected to be most suitable for the type of service and its business function. This offers great flexibility in matching microservices to specific hardware when needed, allowing deployments on inexpensive commodity hardware.

While the distributed nature of microservices adds complexity to the architecture, one of the greatest benefits of microservices is scalability. As the overall application becomes modular, each microservice can be scaled individually, either manually or automatically through demand-driven self-scaling.

Continuous improvements and patching processes are other benefits of microservice architecture. There is virtually no downtime or service interruption for customers because upgrades are performed seamlessly, one service at a time, rather than having to recompile, rebuild and restart an entire monolithic application. As a result, companies can develop and deploy new features and upgrades much faster, in an agile approach, having separate teams focusing on different features, thus being more productive and cost-effective.

Management of large-scale applications has changed and evolved in huge steps during the last few years. When hardware level virtualization first gained traction, it quickly became the standard model for application deployment

1.4 Objectives

I have observed these three following problems in our society. One is that since 2007 drastic content shift from printed and book format to digital media leading to social platforms. For example, YouTube has their own media generators on an exclusive basis, but also provide content taken from outside the social network like news, TV shows or music festivals. Another problem is that these platforms control the embodiment of human knowledge. They lack on content freely “text” searchable as you can only search by title or description and the content of these social platform is owned by these single companies. Last, this media is restrictive and don't allow the creation of text-based derivative works.

What I propose to deal with these problems:

1. Speech-to-text technology is now capable and cost-effective to produce text searchable transcripts of digital media.
   i. Through these providers as AWS Transcribe.

2. Project Factor platform allows.
   i. Content freely "text" searchable.
   ii. Maintain ownership of your digital media and speech-to-text searchable transcripts.

3. The community can own this platform and create text-based derivative works.

Project Factor will be able to:

- Produce text searchable transcripts of media.
- Search "text" freely.
- Maintain media ownership.
- Help to create text-based derivative works.
- Offer both Cloud and OpenSource models.

To sum up, the completion of the project consists on the design and deployment of a Large-Scale client-server application. All sustained over a reliable integration and delivery process to provide online classes anywhere, anytime.

The above requirements would demand high scalability and availability, transparently delivered and consistency. With the creation of a tool that tackles the problem, I want to see an increase on performance of students and professors.

To deal with the typical problems of a Large-Scale application, I am going to support the hole application on the Kubernetes engine. As the problem requires to support a big amount of people and resources at the same time, this setup will orchestrate all the containers and will provide the scalability and stability needed.

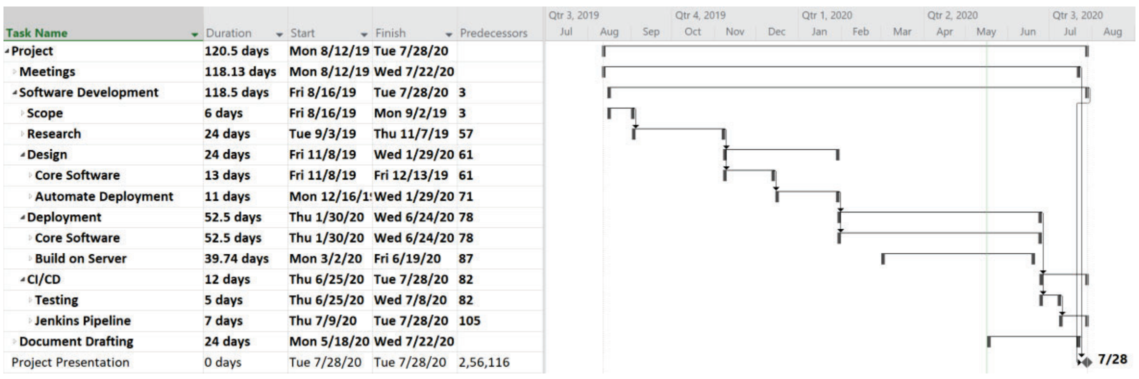
Kubernetes brings software development and operations together by design. By using a declarative way to describe the decoupled infrastructure that constructs to describe how applications are composed, how they interact, and how they are managed.

The transcription service is where the value resides, I make use of the AWS Transcript solution from Amazon to translate the video and store the plain information on the database. As I make use of microservices, it could be changed to run in any other commodity as the open-source platform named DeepSpeech. There will also be a comparation and examples between the execution of both solutions. This service can be abstracted to an endpoint that will take a video from the user and will return the extracted words alongside the timestamps from when the word was said.

1.4.1 Gantt

Project Factor has been developed following the following summary tasks and ends with the milestone regarding to the presentation of the project at Illinois Institute of Technology.

As can be seen in the image bellow, it starts with the weekly meeting of one-hour duration. Then, the development of the application divided into and analysis of the scope of the problem, the further research of new technologies needed to succeed in the system building, the design of the infrastructure according to the needs, followed by the construction of the software and lastly surrounded by good coding practices such as testing and the continuous integration and continuous delivery pipeline.

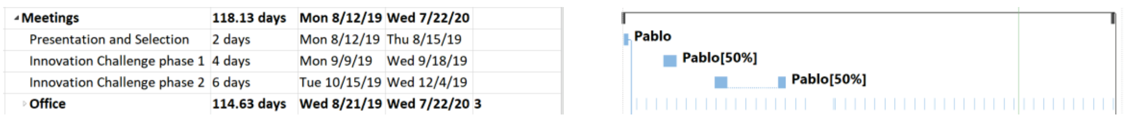


Task Name	Duration	Start	Finish	Predecessors
Project	120.5 days	Mon 8/12/19	Tue 7/28/20	
Meetings	118.13 days	Mon 8/12/19	Wed 7/22/20	
Software Development	118.5 days	Fri 8/16/19	Tue 7/28/20	3
Scope	6 days	Fri 8/16/19	Mon 9/2/19	3
Research	24 days	Tue 9/3/19	Thu 11/7/19	57
Design	24 days	Fri 11/8/19	Wed 1/29/20	61
Core Software	13 days	Fri 11/8/19	Fri 12/13/19	61
Automate Deployment	11 days	Mon 12/16/1	Wed 1/29/20	71
Deployment	52.5 days	Thu 1/30/20	Wed 6/24/20	78
Core Software	52.5 days	Thu 1/30/20	Wed 6/24/20	78
Build on Server	39.74 days	Mon 3/2/20	Fri 6/19/20	87
CI/CD	12 days	Thu 6/25/20	Tue 7/28/20	82
Testing	5 days	Thu 6/25/20	Wed 7/8/20	82
Jenkins Pipeline	7 days	Thu 7/9/20	Tue 7/28/20	105
Document Drafting	24 days	Mon 5/18/20	Wed 7/22/20	
Project Presentation	0 days	Tue 7/28/20	Tue 7/28/20	2,56,116

Figure 1 Task overview Gantt

To further analyze each of these phases, a short subdivision of each of these tasks will be shown.

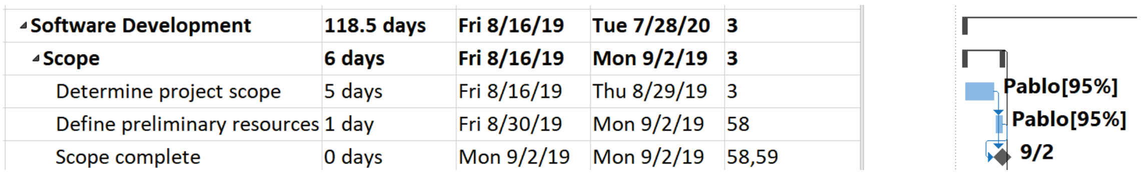
The meetings consider the presentation and selection of the project, the time required to present the project on an innovation challenge celebrated at the IIT and an hourly weekly meeting that has been represented by 0.2h per day or 5% of the daily schedule.



Meetings	118.13 days	Mon 8/12/19	Wed 7/22/20	
Presentation and Selection	2 days	Mon 8/12/19	Thu 8/15/19	
Innovation Challenge phase 1	4 days	Mon 9/9/19	Wed 9/18/19	
Innovation Challenge phase 2	6 days	Tue 10/15/19	Wed 12/4/19	
Office	114.63 days	Wed 8/21/19	Wed 7/22/20	3

Figure 2 Meeting task

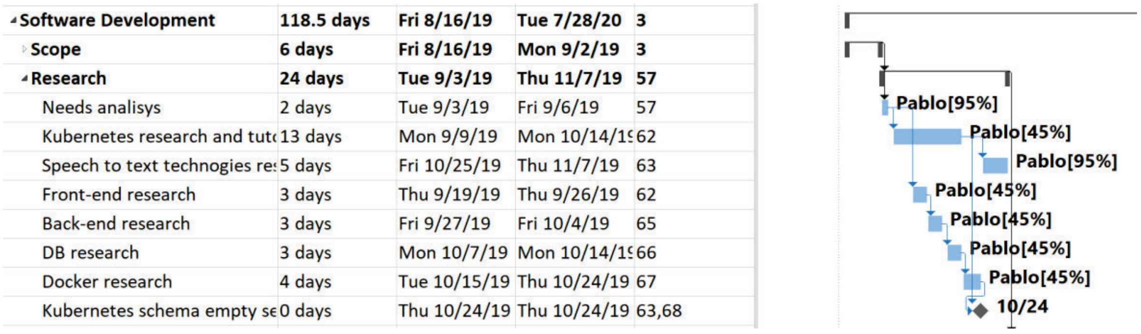
The first tasks while developing a project is determine the scope. I was given a video showing the basic work of another old solution.



Software Development	**118.5 days**	**Fri 8/16/19**	**Tue 7/28/20**	**3**
Scope	**6 days**	**Fri 8/16/19**	**Mon 9/2/19**	**3**
Determine project scope	5 days	Fri 8/16/19	Thu 8/29/19	3
Define preliminary resources	1 day	Fri 8/30/19	Mon 9/2/19	58
Scope complete	0 days	Mon 9/2/19	Mon 9/2/19	58,59

Figure 3 Scope stage

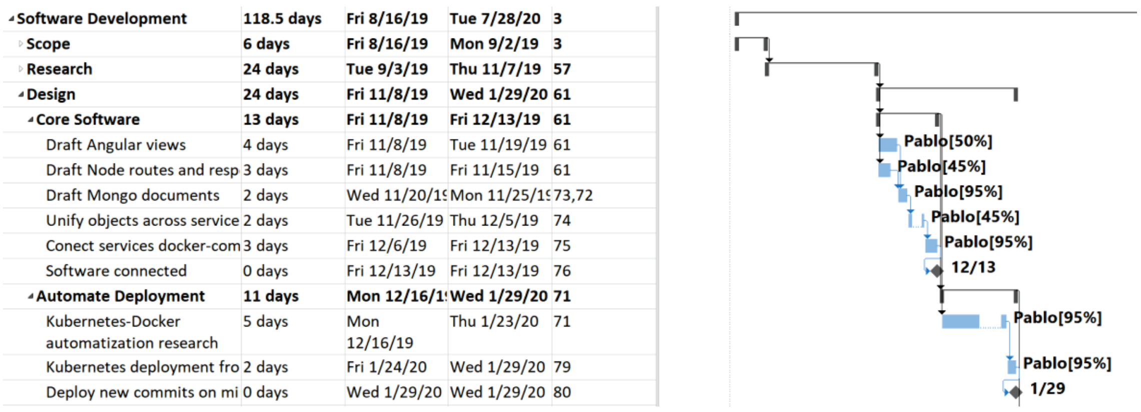
Then I started the research on technologies that I had never used like Kubernetes and Speech to text technologies as the main driver of this innovation. I had a little more background in front-end and back-end frameworks, so that took less time for me to decide. Finally, I has also been my first time working with docker and I spent some time making tutorials.



Software Development	**118.5 days**	**Fri 8/16/19**	**Tue 7/28/20**	**3**
Scope	**6 days**	**Fri 8/16/19**	**Mon 9/2/19**	**3**
Research	**24 days**	**Tue 9/3/19**	**Thu 11/7/19**	**57**
Needs analisys	2 days	Tue 9/3/19	Fri 9/6/19	57
Kubernetes research and tut	13 days	Mon 9/9/19	Mon 10/14/1	62
Speech to text technogies re	5 days	Fri 10/25/19	Thu 11/7/19	63
Front-end research	3 days	Thu 9/19/19	Thu 9/26/19	62
Back-end research	3 days	Fri 9/27/19	Fri 10/4/19	65
DB research	3 days	Mon 10/7/19	Mon 10/14/1	66
Docker research	4 days	Tue 10/15/19	Thu 10/24/19	67
Kubernetes schema empty se	0 days	Thu 10/24/19	Thu 10/24/19	63,68

Figure 4 Research stage

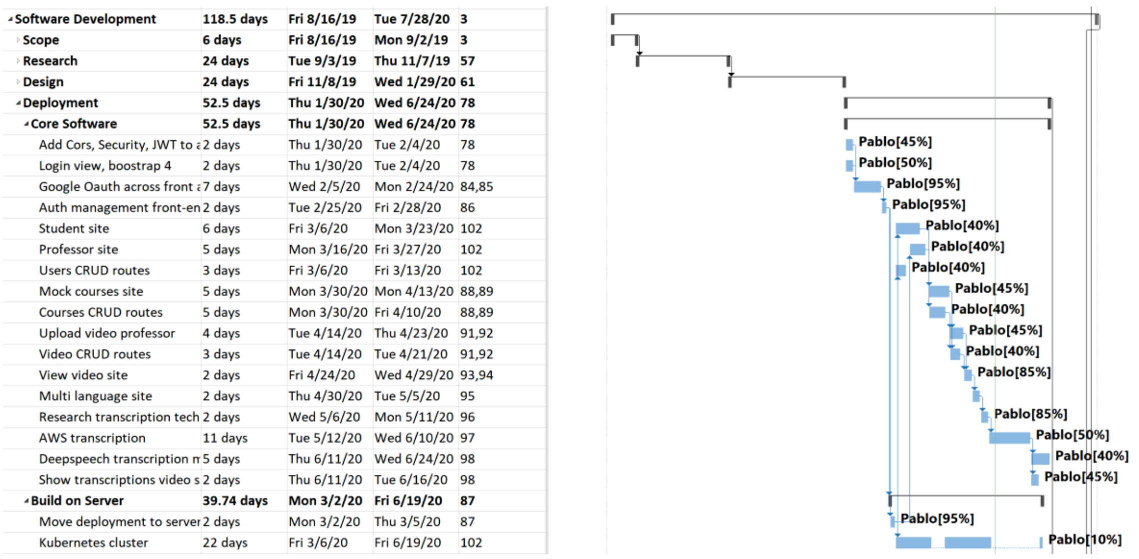
Following with the draft of the front-end views, the back-end routes requirements, the object models and the interconnection of all these pieces. Firstly, with docker-compose and then deploying the git repositories in Docker-hub and then seeing then deployed automatically on Kubernetes.



Software Development	**118.5 days**	**Fri 8/16/19**	**Tue 7/28/20**	**3**
Scope	**6 days**	**Fri 8/16/19**	**Mon 9/2/19**	**3**
Research	**24 days**	**Tue 9/3/19**	**Thu 11/7/19**	**57**
Design	**24 days**	**Fri 11/8/19**	**Wed 1/29/20**	**61**
Core Software	**13 days**	**Fri 11/8/19**	**Fri 12/13/19**	**61**
Draft Angular views	4 days	Fri 11/8/19	Tue 11/19/19	61
Draft Node routes and resp	3 days	Fri 11/8/19	Fri 11/15/19	61
Draft Mongo documents	2 days	Wed 11/20/1	Mon 11/25/1	73,72
Unify objects across service	2 days	Tue 11/26/19	Thu 12/5/19	74
Conect services docker-com	3 days	Fri 12/6/19	Fri 12/13/19	75
Software connected	0 days	Fri 12/13/19	Fri 12/13/19	76
Automate Deployment	**11 days**	**Mon 12/16/1**	**Wed 1/29/20**	**71**
Kubernetes-Docker automatization research	5 days	Mon 12/16/19	Thu 1/23/20	71
Kubernetes deployment fro	2 days	Fri 1/24/20	Wed 1/29/20	79
Deploy new commits on mi	0 days	Wed 1/29/20	Wed 1/29/20	80

Figure 5 Design stage

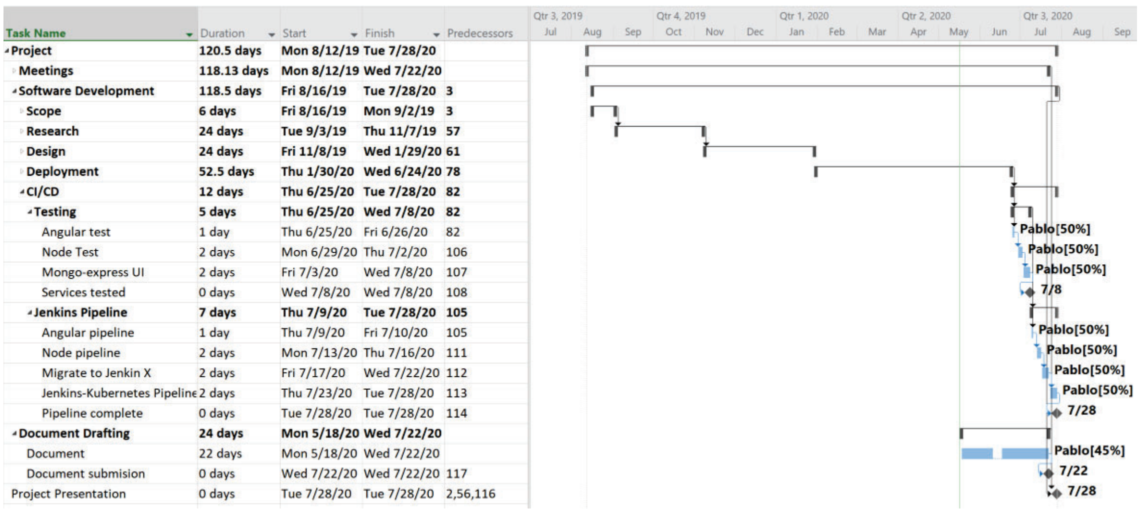
This is the core problem, the design of the software, this could have taken longer but as far as I where more familiarized in these technologies we are not getting deeper here, only show the time was mostly spent on the transcription process.



Software Development	**118.5 days**	**Fri 8/16/19**	**Tue 7/28/20**	**3**
Scope	**6 days**	**Fri 8/16/19**	**Mon 9/2/19**	**3**
Research	**24 days**	**Tue 9/3/19**	**Thu 11/7/19**	**57**
Design	**24 days**	**Fri 11/8/19**	**Wed 1/29/20**	**61**
Deployment	**52.5 days**	**Thu 1/30/20**	**Wed 6/24/20**	**78**
Core Software	**52.5 days**	**Thu 1/30/20**	**Wed 6/24/20**	**78**
Add Cors, Security, JWT to a	2 days	Thu 1/30/20	Tue 2/4/20	78
Login view, boostrap 4	2 days	Thu 1/30/20	Tue 2/4/20	78
Google Oauth across front a	7 days	Wed 2/5/20	Mon 2/24/20	84,85
Auth management front-en	2 days	Tue 2/25/20	Fri 2/28/20	86
Student site	6 days	Fri 3/6/20	Mon 3/23/20	102
Professor site	5 days	Mon 3/16/20	Fri 3/27/20	102
Users CRUD routes	3 days	Fri 3/6/20	Fri 3/13/20	102
Mock courses site	5 days	Mon 3/30/20	Mon 4/13/20	88,89
Courses CRUD routes	5 days	Mon 3/30/20	Fri 4/10/20	88,89
Upload video professor	4 days	Tue 4/14/20	Thu 4/23/20	91,92
Video CRUD routes	3 days	Tue 4/14/20	Tue 4/21/20	91,92
View video site	2 days	Fri 4/24/20	Wed 4/29/20	93,94
Multi language site	2 days	Thu 4/30/20	Tue 5/5/20	95
Research transcription tech	2 days	Wed 5/6/20	Mon 5/11/20	96
AWS transcription	11 days	Tue 5/12/20	Wed 6/10/20	97
Deepspeech transcription m	5 days	Thu 6/11/20	Wed 6/24/20	98
Show transcriptions video s	2 days	Thu 6/11/20	Tue 6/16/20	98
Build on Server	**39.74 days**	**Mon 3/2/20**	**Fri 6/19/20**	**87**
Move deployment to server	2 days	Mon 3/2/20	Thu 3/5/20	87
Kubernetes cluster	22 days	Fri 3/6/20	Fri 6/19/20	102

Figure 6 Deployment stage

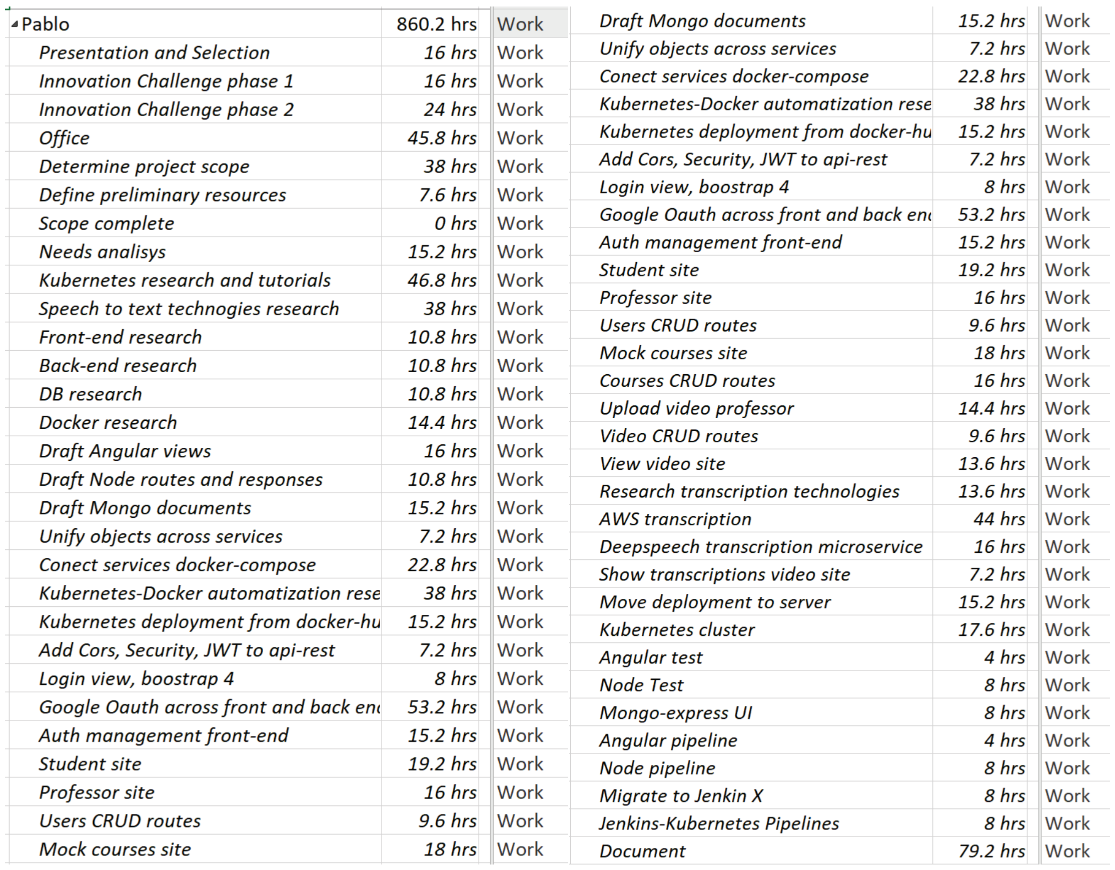
The final days before the project submission I have been working on the hole automatization process alongside with the document drafting.



Task Name	Duration	Start	Finish	Predecessors
Project	**120.5 days**	**Mon 8/12/19**	**Tue 7/28/20**	
Meetings	**118.13 days**	**Mon 8/12/19**	**Wed 7/22/20**	
Software Development	**118.5 days**	**Fri 8/16/19**	**Tue 7/28/20**	**3**
Scope	**6 days**	**Fri 8/16/19**	**Mon 9/2/19**	**3**
Research	**24 days**	**Tue 9/3/19**	**Thu 11/7/19**	**57**
Design	**24 days**	**Fri 11/8/19**	**Wed 1/29/20**	**61**
Deployment	**52.5 days**	**Thu 1/30/20**	**Wed 6/24/20**	**78**
CI/CD	**12 days**	**Thu 6/25/20**	**Tue 7/28/20**	**82**
Testing	**5 days**	**Thu 6/25/20**	**Wed 7/8/20**	**82**
Angular test	1 day	Thu 6/25/20	Fri 6/26/20	82
Node Test	2 days	Mon 6/29/20	Thu 7/2/20	106
Mongo-express UI	2 days	Fri 7/3/20	Wed 7/8/20	107
Services tested	0 days	Wed 7/8/20	Wed 7/8/20	108
Jenkins Pipeline	**7 days**	**Thu 7/9/20**	**Tue 7/28/20**	**105**
Angular pipeline	1 day	Thu 7/9/20	Fri 7/10/20	105
Node pipeline	2 days	Mon 7/13/20	Thu 7/16/20	111
Migrate to Jenkin X	2 days	Fri 7/17/20	Wed 7/22/20	112
Jenkins-Kubernetes Pipeline	2 days	Thu 7/23/20	Tue 7/28/20	113
Pipeline complete	0 days	Tue 7/28/20	Tue 7/28/20	114
Document Drafting	**24 days**	**Mon 5/18/20**	**Wed 7/22/20**	
Document	22 days	Mon 5/18/20	Wed 7/22/20	
Document submision	0 days	Wed 7/22/20	Wed 7/22/20	117
Project Presentation	0 days	Tue 7/28/20	Tue 7/28/20	2,56,116

Figure 7 CI/CD and Document drafting

This result on the following hourly distribution numbers.



Pablo	860.2 hrs	Work
Presentation and Selection	*16 hrs*	Work
Innovation Challenge phase 1	*16 hrs*	Work
Innovation Challenge phase 2	*24 hrs*	Work
Office	*45.8 hrs*	Work
Determine project scope	*38 hrs*	Work
Define preliminary resources	*7.6 hrs*	Work
Scope complete	*0 hrs*	Work
Needs analisys	*15.2 hrs*	Work
Kubernetes research and tutorials	*46.8 hrs*	Work
Speech to text technogies research	*38 hrs*	Work
Front-end research	*10.8 hrs*	Work
Back-end research	*10.8 hrs*	Work
DB research	*10.8 hrs*	Work
Docker research	*14.4 hrs*	Work
Draft Angular views	*16 hrs*	Work
Draft Node routes and responses	*10.8 hrs*	Work
Draft Mongo documents	*15.2 hrs*	Work
Unify objects across services	*7.2 hrs*	Work
Conect services docker-compose	*22.8 hrs*	Work
Kubernetes-Docker automatization rese	*38 hrs*	Work
Kubernetes deployment from docker-hu	*15.2 hrs*	Work
Add Cors, Security, JWT to api-rest	*7.2 hrs*	Work
Login view, boostrap 4	*8 hrs*	Work
Google Oauth across front and back en	*53.2 hrs*	Work
Auth management front-end	*15.2 hrs*	Work
Student site	*19.2 hrs*	Work
Professor site	*16 hrs*	Work
Users CRUD routes	*9.6 hrs*	Work
Mock courses site	*18 hrs*	Work
Draft Mongo documents	*15.2 hrs*	Work
Unify objects across services	*7.2 hrs*	Work
Conect services docker-compose	*22.8 hrs*	Work
Kubernetes-Docker automatization rese	*38 hrs*	Work
Kubernetes deployment from docker-hu	*15.2 hrs*	Work
Add Cors, Security, JWT to api-rest	*7.2 hrs*	Work
Login view, boostrap 4	*8 hrs*	Work
Google Oauth across front and back en	*53.2 hrs*	Work
Auth management front-end	*15.2 hrs*	Work
Student site	*19.2 hrs*	Work
Professor site	*16 hrs*	Work
Users CRUD routes	*9.6 hrs*	Work
Mock courses site	*18 hrs*	Work
Courses CRUD routes	*16 hrs*	Work
Upload video professor	*14.4 hrs*	Work
Video CRUD routes	*9.6 hrs*	Work
View video site	*13.6 hrs*	Work
Research transcription technologies	*13.6 hrs*	Work
AWS transcription	*44 hrs*	Work
Deepspeech transcription microservice	*16 hrs*	Work
Show transcriptions video site	*7.2 hrs*	Work
Move deployment to server	*15.2 hrs*	Work
Kubernetes cluster	*17.6 hrs*	Work
Angular test	*4 hrs*	Work
Node Test	*8 hrs*	Work
Mongo-express UI	*8 hrs*	Work
Angular pipeline	*4 hrs*	Work
Node pipeline	*8 hrs*	Work
Migrate to Jenkin X	*8 hrs*	Work
Jenkins-Kubernetes Pipelines	*8 hrs*	Work
Document	*79.2 hrs*	Work

Figure 8 Hours distribution Gantt

2 Speech-To-Text

As the Gutenberg printing press become a revolution to text media, nowadays, a branch of machine learning known as automated-speech-recognition (ASR) or speech-to-text (STT), represents a dramatic change in the effectiveness and efficiency of transcription systems.

This branch of machine learning has made a big progress over the past years. Even though it has been monopolized by large corporation due to a couple of reasons. The high compute requirements were a high entry barrier and followed by its requirement of large speech data sets with a diverse vocabulary and a variety of accents for each language.

This companies started developing private services applied to their own devices like assistants as Siri, Alexa or Cortana, which improved their neural networks and increased their variety of data sets.

Assistants have been sent to the background and companies has started to sell their extra computing times for these advanced deep-learning neural network algorithms as cloud services like AWS transcribe or Google Speech API. This APIs allows users to send real-time or recorded audio or video files and return the text associated. These services are now offering accuracies up to 95% or higher. [9]

Taking a look at the example provided by Matthew Loxton where she presents itself transcribing a 30-minute audio file, taking several hours. Then comparing three voice to text services transcribing nine videos, two of them based only on machine learning methods with a mean cost per minute of $0.17 with an accuracy of 98% and the third one with a cost of $1.00 per minute and 100% of accuracy mixing both machine and human methods. She founds that machine alone transcriptions only needed edits, of typically for similar-sounding words such as “IV” vs “IB”, or “wake” vs “wait”, and it didn’t make a significative advance to their needs. [10]

With the application of these techniques we will provide universities with indexing and transcribing capabilities. These are extremely useful ways of identifying the main themes of a recording and the approximate point at which they occur in the recording. It would reduce the time needed to identify the content. In addition, they help researchers quickly determine if a recording is related to a selected topic over a big set of media.

As we will be talking about transcriptions played alongside the class video, it will be required to have adequate timestamps to keep the synchronicity between voice and transcription text.

All this done on purpose of accessibility, ability to research and collection of key words as data. A final process will identify the main topics that have been covered to create that indexing over the final transcripts.

2.1 Selection

In order to develop a proper examination of Speech-to-Text libraries the research settled on comparing initially two models, a cloud based closed source service and an on-premises opensource software.

The current major Speech-to-Text cloud platforms considered were:

- Google Speech-to-Text
- Azure Speech-to-Text
- IBM Watson
- Amazon Web Services Transcribe

The two major opensource Speech-to-Text libraries considered were CMU Sphinx and Mozilla DeepSpeech.

Though there are no doubt more libraries available, in order to begin to understand the usefulness of Speech-to-Text as a whole only AWS Transcribe and Mozilla DeepSpeech were selected. This will leave room for additional libraries and comparative research. AWS Transcribe was selected because of the size of AWS and its available computing power. Mozilla DeepSpeech is a relatively new library available not only with promising results, but also the largest crowdsourced corpus of transcribed audio that can be used as a test-suite for any Speech-To-Text library called Common Voice. [11]

2.2 Analysis Factors

In addition to basic computational factors as accuracy, we wanted to examine factors of cost in using these services as well as examine the Software Freedom of each solution. The source and background of software freedom is important. Many people view the ability to control their own data and control their own software/computer as more valuable than pure features and straight cost. In an academic setting where learning is valued above all else, Software Freedom should be considered. The term Software Freedom should not be confused with Privacy which is another issue all together but can be conflated. A research group out of University of Wisconsin-Madison did a similar research project using DeepSpeech, but their focus was on privacy alone. [12]

2.3 AWS Transcribe Methodology

Amazon Transcribe is an automatic speech recognition (ASR) service that makes it easy for developers to incorporate speech-to-text capability into their applications. Using the Amazon Transcribe API, after storing the media in a S3 bucket. The service will return a text file with the transcribed voice.

The application created is supported on a Node.JS and we have made use of the AWS-SDK available on NPM to integrate with it remotely. We can call via API the

AWS servers from our code. It requires two keys, accessKeyId and secretAccessKey for authentication. In the transcription case, it also requires setting a region, because the transcription service is only available in 15 regions worldwide. In our case, the servers were closer to "us-east-1" region, and the AWS S3 bucket used has to be in that same region.

Because of the long transcription times, after finalizing the processing of the video, a response will be sent to the client. This will end the request-response cycle and allow us to submit the transcription job at background.

First, the audio will be extracted as it is the only meaningful piece and this will decrease the transmission requirements, in terms of time and size. In this case, the best audio format will be extracted with ffmpeg, here is an example of which command line provide the best acceptable format. Although, AWS transcribe is prepared to accept many formats.

```
ffmpeg -i video.mp4 -acodec pcm_s16le -ac 1 -ar 8000 audio.wav
```

Listing 1 ffmpeg command AWS Transcribe

It will output an audio file using the codec pcm_s16le, in a single audio channel and with a sampling frequency of 8000Hz.

Then, the output audio will be sent to a S3 bucket, located in the same geographic localization as the consequent launched transcription job. This job name parameter doesn't allow duplicates. Therefore, the timestamp will be added to this parameter for a unique URI as per S3 naming requirements.

This S3 bucket URI will be sent to AWS Transcribe and the transcription job will be loaded and start on the AWS services. In our case, a timeout will trigger every ten seconds and keep it running while the response displays "IN_PROGRESS" as status.

Once the transcription job has finished, the output JSON URI will be stored within the last response and we will be able to access those transcripts via an S3 bucket.

Our application retrieves the transcript and stores it in our server's database and we will delete all the information from the S3 bucket to retain our assets inside our barriers.

At the end of the AWS transcription job, the transcript retrieved will follow the next JSON structure.

```
jobName: {type: String},
accountID: {type: Number},
result: {
   transcripts: {type: String},
   [
       start_time: {type: Number},
       end_time: {type: Number},
       alternatives: [{
          confidence: {type: Number},
          content: {type: String}
       }],
       type: {type: String}
   ]
},
status: {type: String}
```

Listing 2 AWS Transcribe JSON transcribe structure

AWS has the capability to do transcription on single and multiple speaker audio inputs. For the sake of our test-suite we opted for single speaker 5-minute lecture clips.

Here we are using the basic parameters to compare the starting accuracy for both methods. AWS only lacks in a model to avoid active waits. As you can see in the code, we have to trigger a request each ten seconds, asking if the transcription job has been completed.

With the transcript output in our hands, and without much effort, when we look at our results, we can see correct punctuation and text-formatting. The results even formatted “one point two” to “1.2” [2]. Also, the system adds timestamps for each word in the text. So, we will be able to match each word in the text to the corresponding place in the audio file. The next feature is the recognition of multiple speakers and label their voices in the text. And finally, the possibility of providing a word dictionary. it can be seen multiple errors identifying acronyms like SSH, getting outputs as “S s H” or “s s h” that might be solved with this feature.

2.4 DeepSpeech Methodology

For the Free and Opensource model we have deployed DeepSpeech behind a python service in our test suite. In this case we will be using the inference model, which refers to the process of using a trained machine learning algorithm to make a prediction [1]. Secondly, there will be a necessity to continually optimizing our own model. This involves the use of a deep-learning framework such as TensorFlow, which is the one used by DeepSpeech and training datasets.

Application wise, DeepSpeech provide inference methods in their NodeJS SDK. However, the second part, training the dataset can only be done using Python code as this is the case.

The first actions required by this solution are the same. After finalizing the processing of the video, a response will be sent to the client. This will end the request-response cycle and allow us to submit the transcription job at background.

Again, we will need to extract the audio from the video. In this case, this step is fundamental as DeepSpeech do not provide any media transformation before its processing. Then, a gRCP call will be made from the core Node.JS service as the client to the server Python as the microservice that handles the transcriptions. The client only transmits a path as they share access to the same storage.

At the server site, the last model and scorer will be loaded to proceed to the transcription process. The basic DeepSpeech model weights almost 2GB. DeepSpeech as an open-source solution, is only prepared to transcribe small chucks of audio. From now on, DeepSpeech can only transcribe correctly pieces of audio with a length lower than 10 seconds or the quality highly deceases. That's the reason why they provide an example code called "Voice Activity Detector" or VAD.

This was the main missing piece in the starting stages of project that used DeepSpeech in the past. It is used to split the WAV file into smaller chunks process each chuck and finally save a complete transcript. This piece of code detects the silent moments to safely divide the audio. It also returns the number of seconds with sounds in each chuck. It also provides a parameter called "aggressiveness" to smooth the results.

After transcribing each chuck, a complete transcript will be handled to the client. This response will be stored in the database to make it available on next requests.

Among some of the problems of deepspeech is that this "vad_segment_generator" only support 16000Hz input WAV files for now. It also fails if we pass more than one audio channel.

The problem with adopting DeepSpeech as your main Speech-to_Text library is at the moment it is highly American English centered. There are French and Russian pretrained models with other language support just beginning with much training work to follow. DeepSpeech currently lacks the ability to place timestamps in the transcript. This could be easily provide modifying the VAD tool.

3 Design and Infrastructure

The methods presented before will be tested with a test suite designed with the purpose of validating its accuracy. It will also provide a real use case where these technologies would increase value of the assets after their transformation.

The test suit has been chosen to be structured as a web framework, accessed from a user web browser. It is a full-stack solution structured with Angular as the display tier, Express, and Node.js as the application tier and MongoDB as the database tier. [13]

Web-based applications consist of two main components:

1. Client-side, also known as front-end, which includes the browser and the mobile web app client. It will be defined and controlled by Angular and supported on the MVC framework as will be exposed latter.
2. Server-side that includes all back-end functionalities (database, validations, authorizations and authentications). While Node.js provides the fundamental platform for development, it will be supported on Express to define the webserver and in mongoose to access the Mongo database. Notice that this is not a relational database and its main language is JavaScript.

To bring the reader closer to the initial requirements and the way they have been addressed, we first need to explain the movement of data in the different subsystems. This makes it easier to understand the routing of HTTP requests from the front-end to the back-end, all inside subnetwork running in the Kubernetes cluster. Allowing us to generate many versions of each one of the different services working as a unique system.

3.1 Requirements

We need to create a system that:

- Reliable service: The use of Kubernetes will ensure high availability and scalability of services to handle a high number of requests.

- No registration form: we will extract basic profile information thanks to Google Oauth 2.0, with the limitation of using of university Google accounts. This limitation also allows us to avoid fake accounts and to distinguish students from professors.

- Multi-language view: Considering Angular i-18n feature, we can distribute the same views in multiple languages.

- Generate video transcriptions: As presented before, we have chosen the AWS transcribe service to meet this requirement.

- Secure system: Starting over a secure design of the components, there have being inserted guards to guide the correct flow of the requests.

- Create multi-user level accounts: users, admins, superusers.

  - Administrators require:
    - Adding courses to users.
    - Overview of system data.
  - Professors require:
    - Uploading videos.
    - Edit and view videos.
    - Edit and view transcriptions.
  - Students require:
    - View videos.
    - View transcriptions.

3.2 Requests processing

The user interacts with the browser and it will one have one global entry point that will use English language and many language entries points that will redirect the user the selected entry point. These requests will only be allowed to the front-end, not to the back-end.

3.2.1 Kubernetes stream

Once the user sent the request, as we are using a Kubernetes cluster, the request will arrive to the master node. This node will encapsulate the package through the internal flannel network to the ingress service. There, if the route is allowed, the package will be pushed forward to one of the replicated front-end pods, it could be present in any of the slave nodes. Then, the back-end can be found the ingress in the route "/api".

While the back-end and the database will share a link name to make the requests without exiting the flannel network. Finally, we can see a last microservice called transcription service. This can be called from the back-end through gRPC messages, although the request is unidirectional, the transcription service is allowed to respond the request with an object carrying the output transcription. As we are using a cloud service, the transcription service will be the only one allowed to connect Amazon servers.

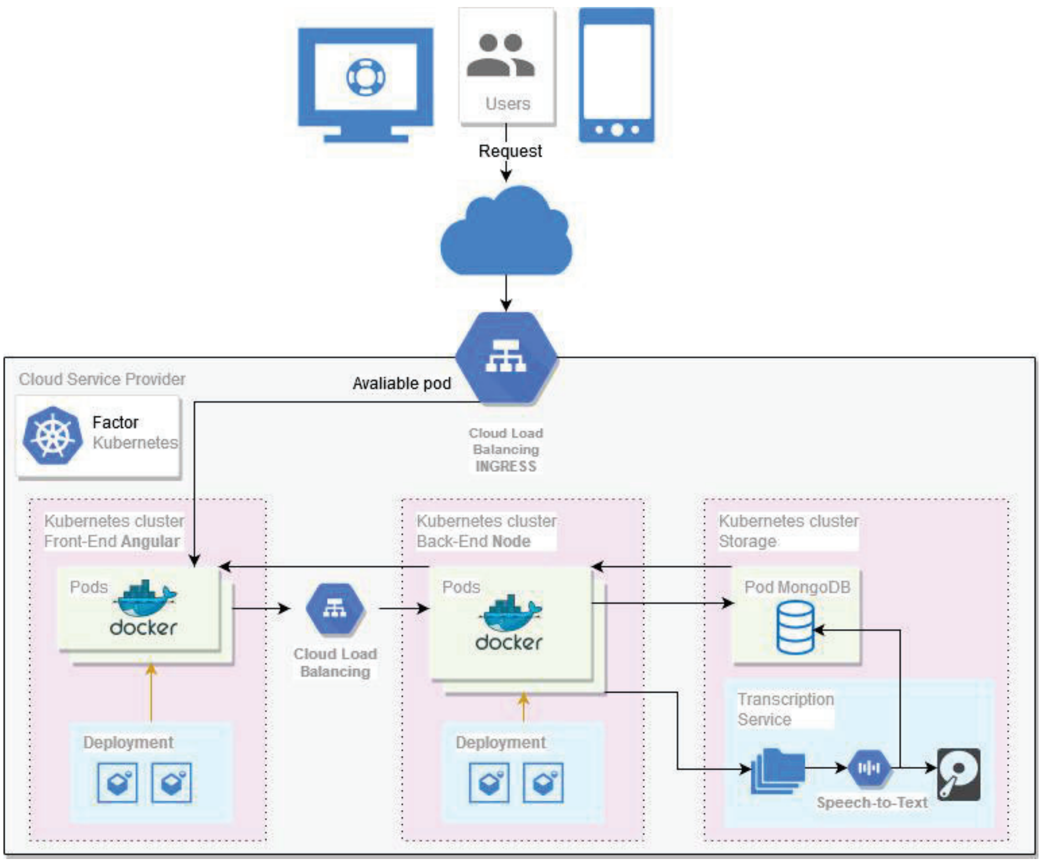


Figure 9 Kubernetes request processing

3.2.2 Angular stream

The present project is part of a set in which the user only sees the front-end, hiding the functioning of the rest of the structure.

Internally, the design of this service consists of reusable modules throughout the application. Looking at the next picture, each of these modules can be part of the following types. On the right we can see the components, they are the elements that implement the necessary functionality along the interfaces. The central modules represent the global interfaces of the system and should be consistent with the types used in the integration with other parts of the project. Finally, the service module implements the application-specific business logic, it doesn’t have a view and contain the most complex decisions.

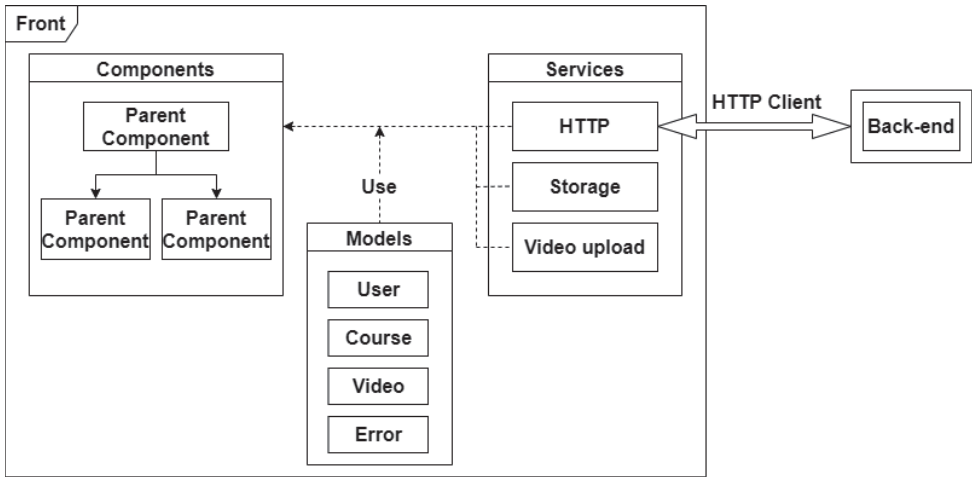


Figure 10 Angular MVC patern

The decision to structure the HTTP service in the front-end came about in order to encapsulate in a single reference all the requests in the direction to the backend. This is based on the singleton pattern.

Due to the use of Angular framework, we ensure the follow of MVC or Model-View-Controller pattern. This is a design pattern composed of three parts that help improve the uniqueness of the components and generate logical separations over the whole code.

In Angular each component forms its own MVC between the dynamically formed view or template, model or HTML and the controller or TypeScript. [14]

The view is the result of relating the data that both share through the binds or links between the template and the controller.

Apart from the MVC at component level, the whole front-end follows this structure.

- Model: This part benefits from Typescript to define the data structure. This will be helpful for programming since, through the definition of global classes, a complete definition of the front-end is unified. At the same time, it allows the static checking of types.

- View: Allows to display the state of the component. Using the bindings or links with pipes, properties and/or events, to link data from the controller that are in turn modelled through the template. Or also, iterations on an array, *ngFor, and visualization conditions, *ngIf.

- Controller: These code sections are not shown to the client, although they offer most of the functionality to change the state of the model. It is important that the internal controller of the component is only in charge of defining the properties and methods available in the template. Therefore, global services explained before should extract duplicated functionalities to discharge each component controller.

3.2.3 Node stream

One of the weak points of employing JavaScript at the server is that it doesn't need any structure, you could run the whole service from a single file. As we are dealing with a REST API and one API receives a request and returns a response, we are going to follow that request from the point where it reaches our application, travels through our layers, and a response is returned by the application. Our core backend service has the following folder structure to differentiate the responsibilities for each part of the code.

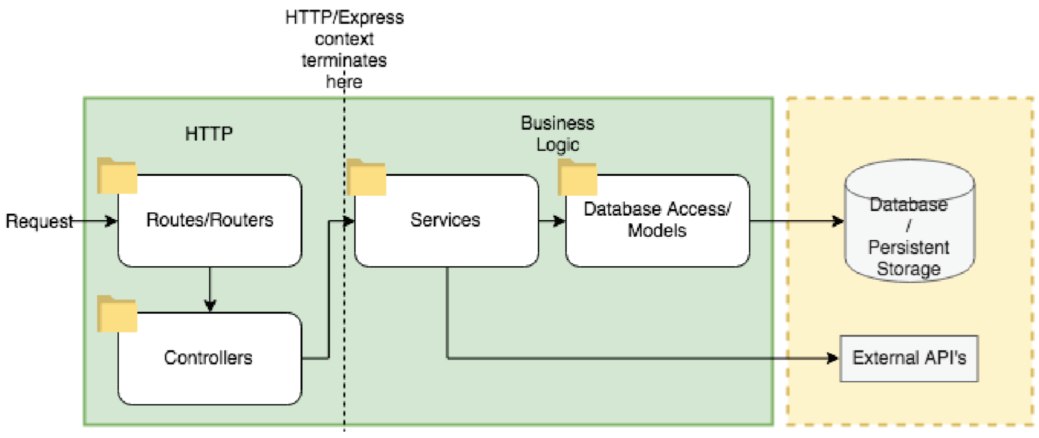


Figure 11 Node code structure. Source: [15]

As James Kolce states in his book, “In a non-trivial application, the architecture is as important as the quality of the code itself. We can have well-written pieces of code, but if we don’t have a good organization, we’ll have a hard time as the complexity increases. There’s no need to wait until the project is half-way done to start thinking about the architecture”. [16]

A big map of the final structure can be seen at the annex. The column at the left represents the HTTP request, response flow. The middle column represents the business logic to extract the audio from the video, following with the extraction of the transcripts using both methods and then compute the similarity. Finally, the last column at the right provide abstractions to external features as is the case with AWS-SDK to the business logic.

4 Deployment and Implement

From here, the way in which the requirements presented in the previous chapter have been fulfilled will be shown. The possible continuation of the project by other people has been considered and the refactoring of the code has been emphasized.

4.1 Angular

As Angular framework focuses its architecture on the use of the design pattern called MVC or model-view-controller, the functionality will be presented according to these existing divisions in the code.

4.1.1 Models

While the whole system communication had been created, we had extracted general designs pieces that are similar across the application. These pieces in Angular are represented as global variables, classes and a router which makes the system accessible to the client.

The TypeScript compiler will warn us if at any time we don't respect these types of data, helping in development time and avoiding hidden errors.

4.1.1.1 Router Module

This element of Angular allows to simulate the navigation of the user inside the web as a SPA or Single-Page-Application. It allows to perform actions as if it was a normal web, but instead of selecting the path of a different file, it only reorganizes the components to be shown in a virtual way.

By loading the only web page with the full weight of the application, it avoids making extra requests to the server. It is true that the first load requires more time. But between the cache system and the deferred loading managed by the router, Angular has managed to eliminate the initial overload. This deferred load comes from the lazy loading concept and consists of postponing the loading of certain pages that are not going to be visited until their route is activated. This ensures that initially the non-visited pages do not increase the loading time.

Each path is composed by an address or path that activates a parent component. Once the user hit one of these paths, activated component shows depending on the Boolean output of the guard routers assigned to that particular route.

These guards control access attempts which meet certain specific conditions for each route. In our case, it mainly checks the authentication level necessary to each route.

In addition, thanks to these routes, we can send data from one component to another as if it were a form. The data that can be used for internal transformations of the component, are preceded by the character ":" and it is possible to access them as shown below.

```
this.router.params.subscribe(params => {
  this.ID = params['id']; // id as route output
}
```

Listing 3 Reading router params Angular

The actual model for the SPA router is the following.

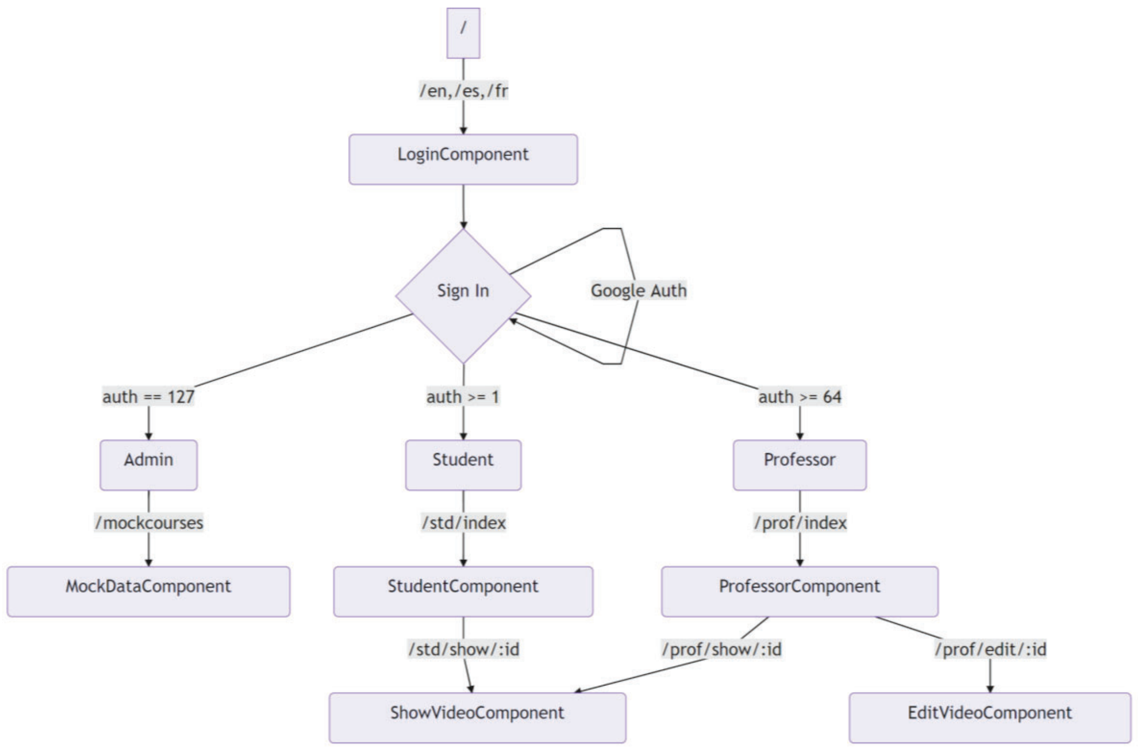


Figure 12 SPA Routes

It has been considered that this can generate access to resources without the need for permissions, but all this is controlled at both ends depending on the user who has logged in. The back-end will not return the data and the front-end will block the entry to that address with the resource to be mentioned below.

4.1.1.2 Globals

These elements make up the core of the application and allow greater granularity of the block that makes up the application by dividing the tasks to be performed. We are going to expose the global ones that direct the correct operation of the application besides offering a separation of the model within the model-view-controller pattern. It also covers the need to fix a model in Typescript.

On the one hand, we have the global variables of the system. They contain static variables that are used throughout the application. They allow changes to be made to the environment in which the system is used in a simple way.

This allows us to change the path in which the back-end is located or its version, by replacing a single line.

```
export class ModelsComponent {
  static version = '/api/'; //future
  static api = 'http://192.168.1.125:3000'; // s ToDo
  static staticsURL = 'assets/';
  static languages = [
    { code: 'en', label: 'English', img: 'english'},
    { code: 'es', label: 'Español', img: 'spanish'},
    { code: 'fr', label: 'Français', img: 'french'}
  ];

  constructor() { }

  logout() {
    sessionStorage.removeItem('token');
  }
}
```

Listing 4 Global variables

On the other hand, it also holds the classes that define the objects used to correctly relate the application and the definitions of the elements that make up these objects. It has been decided to use classes instead of interfaces so as not to have to initialize complex objects before associating them to the answers of the requests.

The application is formed by “users” that can be students or professors. This will be distinguished by their authentication level. However, as we are measuring 127 level of authentication, we can add extra checking levels. For example, this will be use while promoting a student to Teacher Assistant role. The application could add 3 point per correct transcription correction while it can reduce it as a mistake appears. By elevating or degrading the student authentication level, we will speed up corrections by adding power to good students or stop the irresponsible corrections over the transcripts.

Each of these “users” will have access to zero or more “courses”. Following to the next natural division that assets that each course will have zero or more videos. It represents the media content that is related to a transcript once the process has finished.

```
export class Error {
  resNum: number;
  resText: string;
  resBool: boolean;

  constructor() {
    this.resBool = false;
  }
}
```

```
export class Course {
  _id: string;
  name: string;
  number: number;
  professorID: string;
  professor: User;

  constructor() {
    this.professor = new User();
  }
}
```

```
export class User {
  _id: number;
  token: string;
  email: string;
  family_name: string;
  given_name: string;
  locale: string;
  picture: string;
  verified_email: boolean;
  coursesID: string[];
  courses: Course[];
  authlvl:number;


  constructor() {
    this.picture = "";
    this._id = 0;
    this.courses = [];
  }
}
```

```
export class Video {
  _id: string;
  name: string;
  url: string;
  duration: string;
  class: number;
  thumbnail: string;
  courseID: string;
  course: Course;

  constructor() {
    this.course = new Course();
  }
}
```

Listing 5 Global classes

4.1.2 View

The main advantage of Angular is that the view is always updated with the variables you have defined in the controller. For example, if you create a string variable and when you click on a button you decide to change its value, automatically the new value is reflected in the view, without you having to do anything.

This responsiveness work in two ways by using a virtual DOM, instead of updating the DOM when your application state changes, you simply create a virtual tree or Virtual Tree, which looks like the DOM state that you want. It allows that you don't have to worry about refreshing the view when you change a value or do an operation.

As angular make use of reusable elements called components, each view is composed of one to several components. We will take a look inside each of those child views. They have their own model-view-controller to extend the reusability concept.

4.1.2.1 Navbar

This is a child component that will receive two input parameters, the authentication level and the picture for the current user. The authentication level is needed return users to its main site, "/professor" or "/student", linked to the university logo.

```
<app-header [auth]="userInfo?.authlvl"
      [picture]="userInfo?.picture"></app-header>
```

As we can see, the parent component keeps the parameter about the information of the user optional while waiting for the requests.

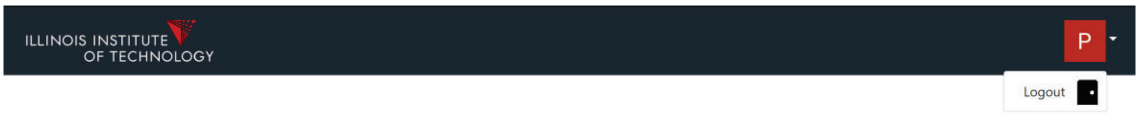


Figure 13 Navbar view

4.1.2.2 Footer

This component will present at the bottom of the site. It will provide information about usage of the application and provide links to guide sites.

It will not only require the parent component to set the authentication level of the current user to present the right information. But also, surround the other main content of the parent component inside the next DOM element.

```
<div
[ngClass]="(userInfo?.authlvl >= 64)?'bodyprofessor':'bodystudent'"
class="container-fluid">MAIN CONTENT</div>
```

This will trigger the next global styles:

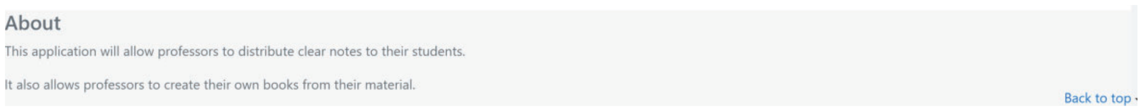
```
 .footer {
       position: fixed;
       left: 0;
       bottom: 0;
       width: 100%;
       background-color: #f5f5f5;
       max-width: unset;
 }

 .bodyprofessor {
       margin-bottom:120px;
 }

 .bodystudent {
       margin-bottom:160px;
 }
```

Listing 6 Footer fixed bottom

The current example of a professor footer is the next one:

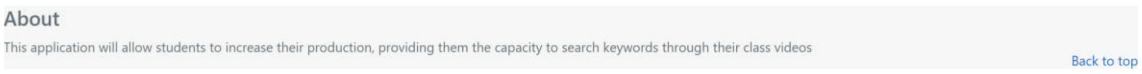

About

This application will allow professors to distribute clear notes to their students.

It also allows professors to create their own books from their material.

Figure 14 Professor footer

While the one for a student would be:



About

This application will allow students to increase their production, providing them the capacity to search keywords through their class videos

Back to top

Figure 15 Student footer

4.1.2.3 Log In

This component represents a complete view, it has capabilities to Log in and Sign up users into the application. This component queries the back end if the user has been already Sign up. If it is the user first time using the system, it also registers users with the Google technology, Oauth 2.0 to Sign up them and retrieve their basic profiling.

As this component was likely to connect with each university registration system, the implementation provides both fields to the backend. It currently stores the password entered on the first time Log in, generally called Sign up. More information on this subject will be given in the following Node development section.

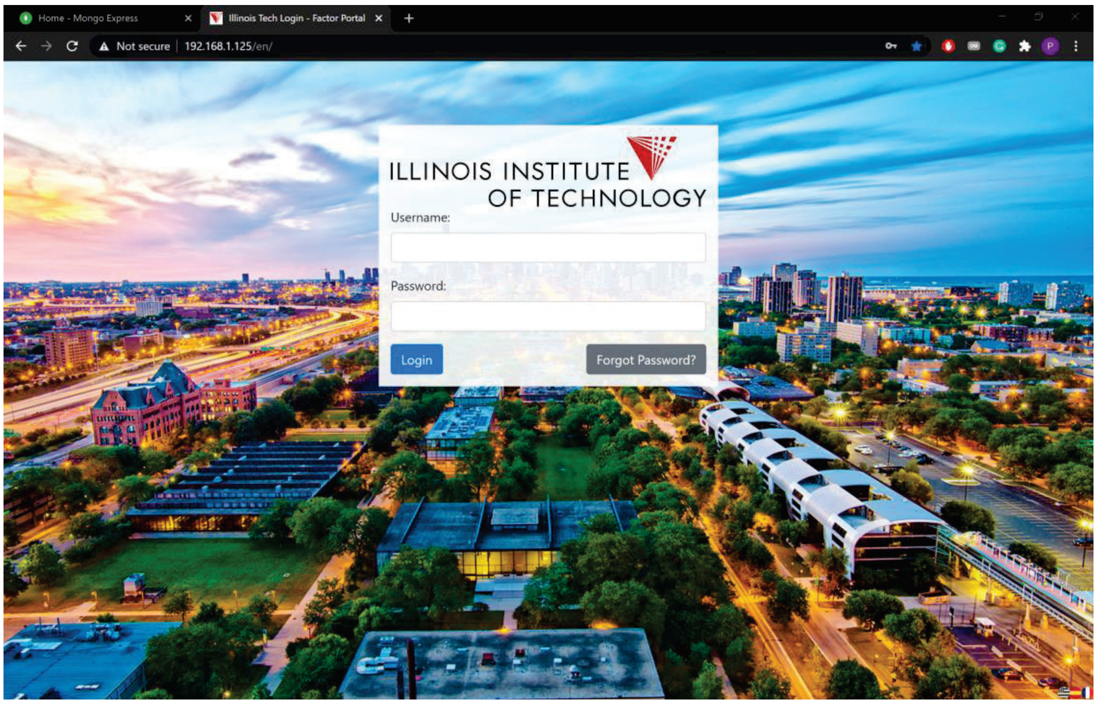

Home - Mongo Express
Illinois Tech Login - Factor Portal
Not secure | 192.168.1.125/en/
ILLINOIS INSTITUTE OF TECHNOLOGY
Username:
Password:
Login
Forgot Password?

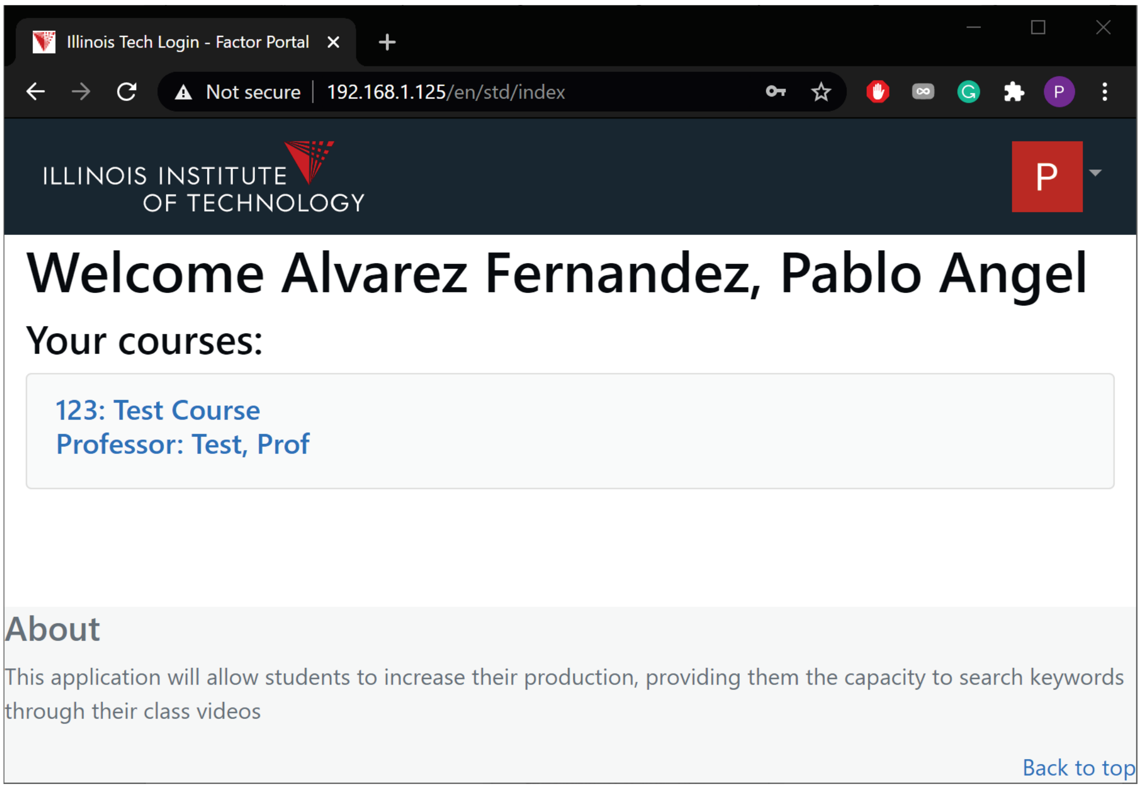
Figure 16 LogIn Test Suit

The second utility provided at the bottom right site of this view is the ability to change the site language. It will also regard on Nginx configurations reading the field "$http_accept_language", triggered when the user calls to the "/" route.

Figure 17 Hint languages

4.1.2.4 Student

Although this application serves multiple proposes, students represent the majority of users. They will see this parent component which represents the main professor site after log into our system. Here we can identify the navbar and the footer stated before. This view also contains a third component represented by a card accordion. It shows the courses to which a user is subscribed.


Illinois Tech Login - Factor Portal
Not secure
192.168.1.125/en/std/index
ILLINOIS INSTITUTE OF TECHNOLOGY
P

Welcome Alvarez Fernandez, Pablo Angel

Your courses:

123: Test Course
Professor: Test, Prof

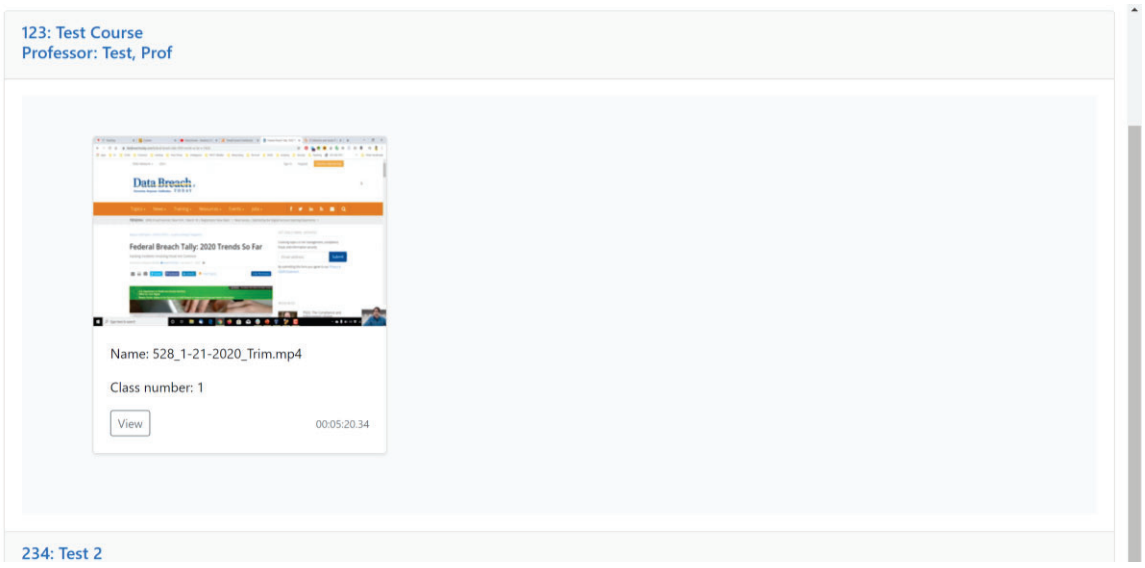
About

This application will allow students to increase their production, providing them the capacity to search keywords through their class videos

Back to top

Figure 18 Student home site

When a user clicks on any of the courses, the accordion will open and show the classes associated with that course.



123: Test Course
Professor: Test, Prof

Name: 528_1-21-2020_Trim.mp4

Class number: 1

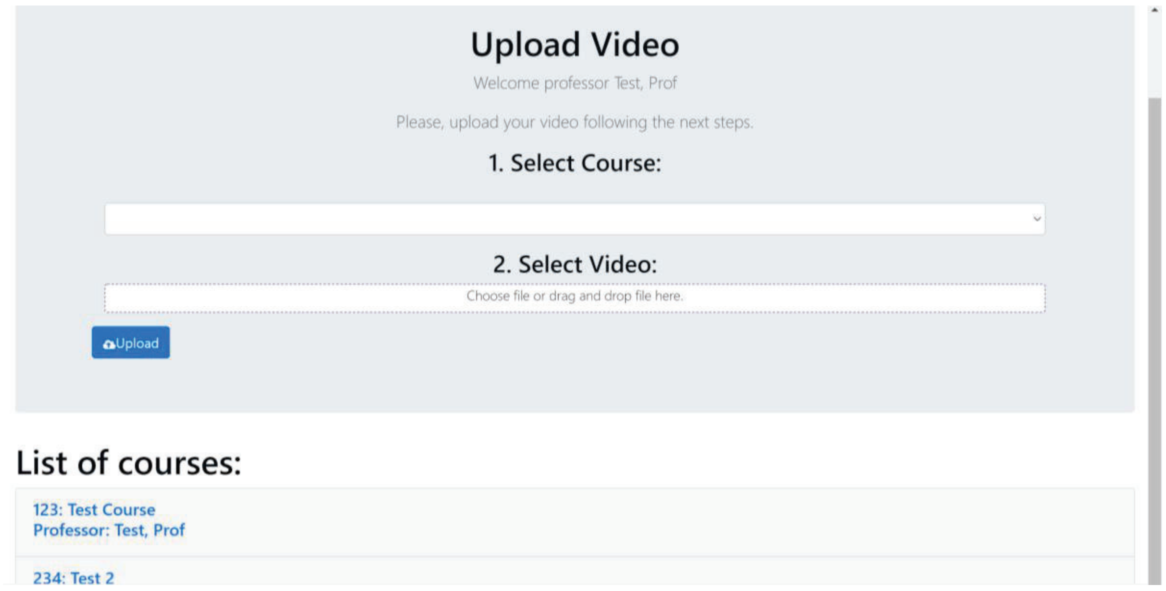
View 00:05:20.34

234: Test 2

Figure 19 Video accordion

4.1.2.5 Professor

Their initial page after logging in contains the same components, like navbar, footer and courses accordion, as the student site. However, this page has two main differences.


Upload Video

Welcome professor Test, Prof

Please, upload your video following the next steps.

1. Select Course:

2. Select Video:

Choose file or drag and drop file here.

Upload

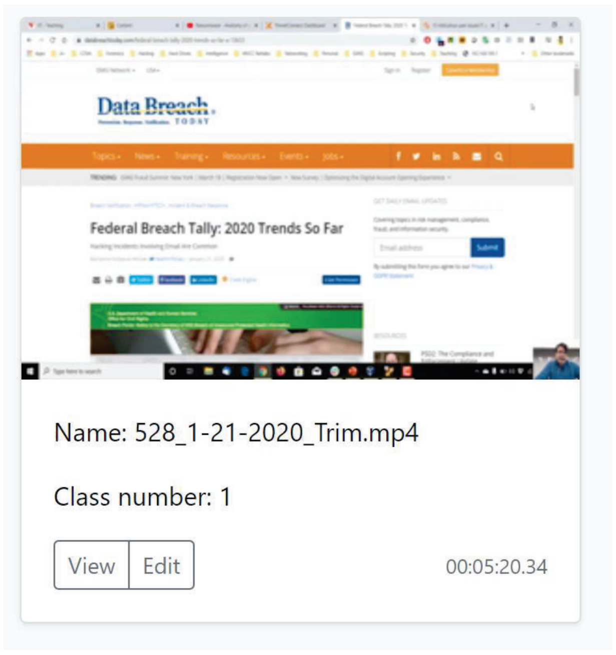
List of courses:

123: Test Course
Professor: Test, Prof

234: Test 2

Figure 20 Professor home site

First, the possibility to upload a course video to the system. And second, the ability to edit some of the parameters related to a video from their courses. Changing the video card to the next model.


Name: 528_1-21-2020_Trim.mp4

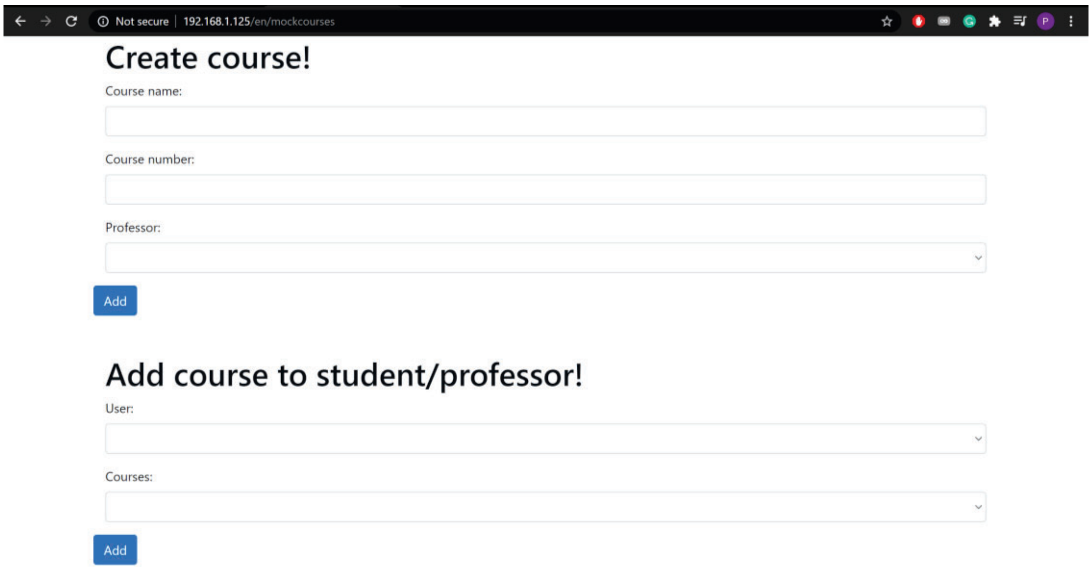
Class number: 1

View Edit 00:05:20.34

Figure 21 Video card professor

4.1.2.6 Admin

There will only be one user with administrator privileges. It will use the same logging method as the rest of the users. Although, this user can only access to the "/mockcourses" site. This site allows the administrator to create new courses and add professors and students to those courses. This information needs to be mocked at this moment. However, it will come from a certain university API which contains all this information.


Not secure | 192.168.1.125/en/mockcourses

Create course!

Course name:

Course number:

Professor:

Add

Add course to student/professor!

User:

Courses:

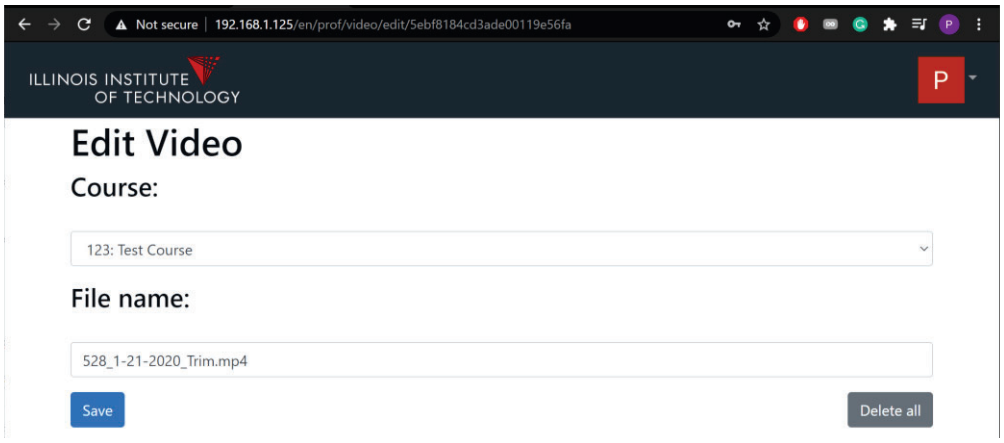
Add

Figure 22 Admin home site

In the current implementation the professor assigned to the course creation, gets registered automatically into that course. The second form gives the opportunity to add students to that courses and also to add extra professors to a course.

4.1.2.7 Edit Video

This view allows the professor to change the course or the name assigned to a particular video. It also allows professors to remove the video from the platform.


Not secure | 192.168.1.125/en/prof/video/edit/5ebf8184cd3ade00119e56fa

ILLINOIS INSTITUTE OF TECHNOLOGY

P

Edit Video

Course:

123: Test Course

File name:

528_1-21-2020_Trim.mp4

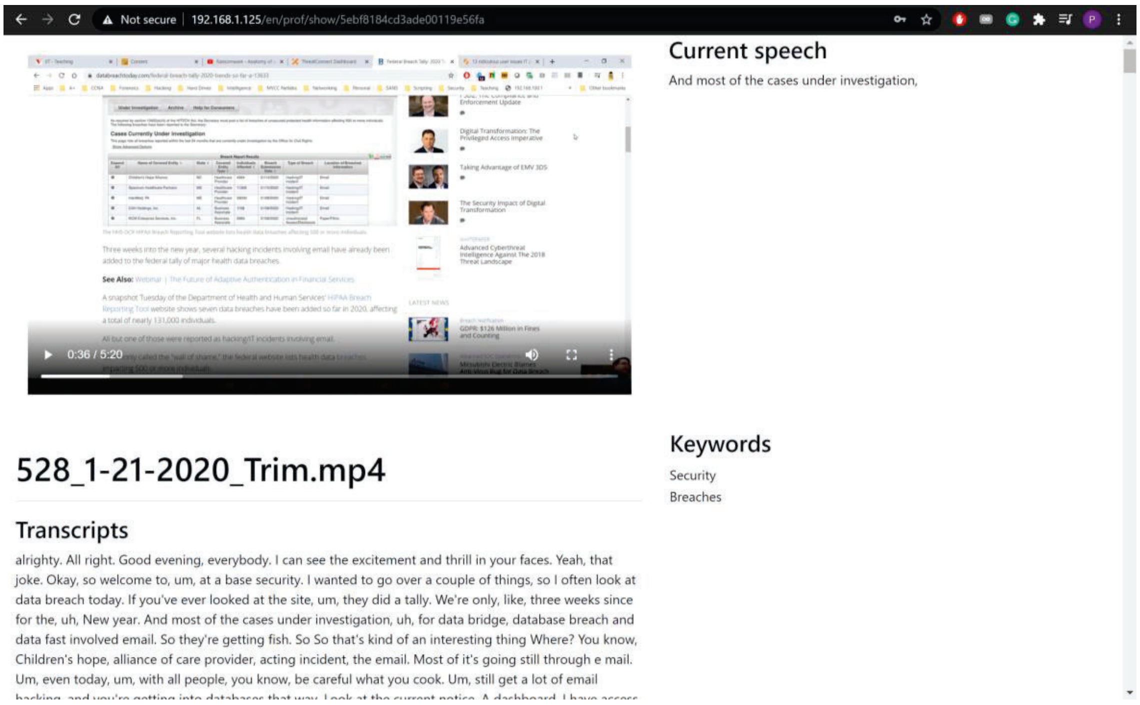
Save

Delete all

Figure 23 Edit video site

4.1.2.8 Show Video

Finally, this is the place where all the users will spend most of their time. It can be seen from the student and professor perspective with only one change. This is the possibility to directly edit the transcriptions. The actual implementation promotes students to TA, after three accepted as correct editions. Before this promotion, students will only be able to make suggestions that have to be accepted by a TA or a professor before seen the changes.


Not secure | 192.168.1.125/en/prof/show/5ebf8184cd3ade00119e56fa

Current speech

And most of the cases under investigation,

0:36 / 5:20

528_1-21-2020_Trim.mp4

Transcripts

alrighty. All right. Good evening, everybody. I can see the excitement and thrill in your faces. Yeah, that joke. Okay, so welcome to, um, at a base security. I wanted to go over a couple of things, so I often look at data breach today. If you've ever looked at the site, um, they did a tally. We're only, like, three weeks since for the, uh, New year. And most of the cases under investigation, uh, for data bridge, database breach and data fast involved email. So they're getting fish. So So that's kind of an interesting thing Where? You know, Children's hope, alliance of care provider, acting incident, the email. Most of it's going still through e mail. Um, even today, um, with all people, you know, be careful what you cook. Um, still get a lot of email hacking, and you're getting into databases that way. Look at the current notice. A dashboard. I have access

Keywords

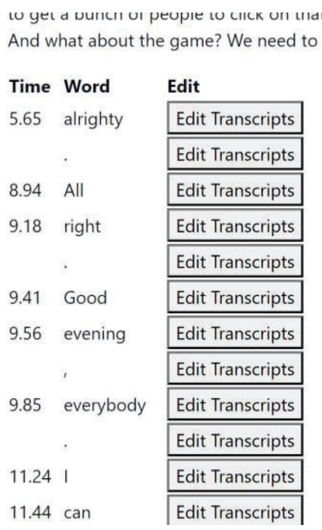
Security

Breaches

Figure 24 Show video view

This view contains four things at first eye view. The video, the current speech of the video at the current time between punctuation symbols, the complete transcripts and the keywords assigned to the video.

At the bottom of the page we will find the option to make suggestions or if the authentication level is enough, to directly correct the transcriptions.


to get a bunch of people to click on that. I don't know
And what about the game? We need to

Time	Word	Edit
5.65	alrighty	Edit Transcripts
	.	Edit Transcripts
8.94	All	Edit Transcripts
9.18	right	Edit Transcripts
	.	Edit Transcripts
9.41	Good	Edit Transcripts
9.56	evening	Edit Transcripts
	,	Edit Transcripts
9.85	everybody	Edit Transcripts
	.	Edit Transcripts
11.24	I	Edit Transcripts
11.44	can	Edit Transcripts

Figure 25 Edit transcripts

4.1.3 Controller

The controllers oversee and handle the view and creating all the logic of that component. They have all the variables and information that the view needs to be painted are created. It also takes care of the interactivity of the component, that is, specifying what is done when rendering the component, when removing the component from the view, or when clicking on an element of the HTML.

In the Angular vocabulary, there are other things apart from components like guards, services and pipelines that are strictly controllers. They aren't linked to a template view.

4.1.3.1 Authentication guard

Angular's route guards are interfaces which can tell the router if the current user should be allowed to a requested route or not. Its decision is defined by a true or false return value from a class which implements the given guard interface.

They give the possibility to redirect the user before he creates conflicts. By avoiding these shortcuts, unnecessary complexity will be removed from the code.

In the current case, we found that if a student changed the browser path after logging in with his account to a professor path, it could upload videos to his courses. Several solutions were proposed:

- The back-end would control what type of user was logging in and would cut off access to certain routes. This proposal is necessary, the problem is that it adds complexity to the back-end. But it does not solve anything in the front end as it could pass through the administration paths undetected until a blocked request is made for a user.
- Check the authorization level in the components of all paths. Something undesirable and rigid.
- Checking the authorization level on an isolated service that would decide whether to activate the route or perform other behavior. It is up to the router to guard.

The last solution was taken for its independence and flexibility. By the addition of router guards to the routes controlling the authentication token stored into the browser local storage, the system gains the desired security.

4.1.3.2 HTTP service

Most front-end applications need to communicate with a server over the HTTP protocol, in order to download or upload data and access other back-end services. By using this element, we avoid using the "HttpClient" dependency of Angular in the rest of the application and centralize the HTTP requests in a single file that presents the required back-end functionality.

This allows us:

- Perform quick adjustments in case of errors.
- Mask any request to the back-end by joining view and control.
- Easier migration.

This service is exported as an injectable element accessible from any part of the application by importing it as "providers: [HTTPService]". Most requests follow this style.

- return to send the observable to the component that made called the function.
- The method: GET, POST, PUT or DELETE.
- The server path attached to the address activated in the Api-REST of the back-end.
- To send the cookie we use the "withCredentials" parameter, set to true.

```
getUserAct(id:string): Observable<User> {
    return this.http.get<User>(
      ModelsComponent.api + ModelsComponent.version + 'users/' + id,
      {withCredentials: true});
  }
```

Listing 7 Example HTTP request

4.1.3.3 File service

This service provides several abstractions to process the upload of a video file. First it provides a general validator of the file type, in case that the format changes. Secondly, it offers another method to transform the form data into a real body and finally it binds the upload progress value to an Angular variable.

It has been designed following good practices to split these particular methods into a separate service, even though it is only used in the professor upload function. This allow us to encapsulate the code as one limited purpose, enhancing reutilization.

4.1.3.4 Video phrase pipeline

Pipelines are elements made to reduce the complexity in the templates making transformations on objects, variables or simply text.

They are declared through "@Pipe({ name: pipeName})" and arguments can be passed to them, tolerating some modifications in their normal operation.

In this case, a PipeTransform has been created to which is passed the second in which is the reproduction of the video. It shows the corresponding phrase between the previous, to the next punctuation symbol found in the transcripts.

4.2 Node

NodeJS is a JavaScript based environment that is placed in the server layer as a tool for building applications easily scalable.

It facilitates the performance of several tasks at the same time and allows a great deal of control from the server side. It also allows to fast handle high set of tasks developed in real time.

As a for the core development, we have made use of Express. It is a module that wraps the functionality of the Node.js http module to provide a simple to use interface. It also extends the functionality to make easer the handler of server routes, responses, cookies and statuses of the http requests. [17]

Another meaningful module is grpc. Node.js will use it from a client perspective. It allows us to connect the separated transcription microservice independent of the language used. It performs calls over the open source remote procedure call (RPC) framework that can run anywhere. It enables client and server applications to communicate transparently and makes it easier to build connected systems. [18]

4.2.1 Auth

This is a core component to separate the capabilities of each user and blocking illegal access. At login time a token will be exchanged with the web server and it will be checked each time the user asks for some data to the backend. This component also had to meet the requirement of not ask for any information from the user.

In order to achieve this capability, first we have centered on the no registration form needed. This make the user believe that he isn't leaving our organization and allow us to collect correct information.

While registering new users in a world-wide open platform, we had to take extra considerations like two-factor authentication or anti-spam account creation. However, in this development we will only accept accounts previously shipped by our own organizations, blocking all other accounts. This limitation also works in our favor while defining the division between students and professors as they have different domain names.

We had to create an assumption while the call to the university account service isn't fulfilled. It is to store the first time or sign up typed password as the fixed password for that email. Within the assumption of been creating a world-wide open platform, storing personal information always involves risks.

This risk comes greater while speaking about core information as password. Since a password don't need to be presented back to the user, we don't need a two-way function as encryption it is. That's why this solution is supported by the hash salt method.

First, hashing is meant to verify that a file or piece of data hasn't been altered. While it's technically possible to reverse-hash something, the computing power required makes it unfeasible. Hashing is one-way and makes every hash value unique. If two different files produce the same unique hash value this is called a collision and this force to change of algorithm to prevent attacks. That's what happened with SHA-1. [19]

Then, if two users use the same password, they will have the same hash. This isn't an algorithm hashing error, it's the way it works. To fix it we will add salt. Salting is a concept that typically pertains to password hashing. Essentially, it's a unique value that can be added to the end of the password to create a different hash value. This adds a layer of security to the hashing process, specifically against brute force attacks.

Following this method, we ensure that we don't store the real password, and any password will be repeated. However, since rainbow table attacks appeared, the next protection step is to create a different salt for each user and store it within its model.

This is how the hash and salt creation and validation look using crypto library and an extension of the SHA-2 algorithm.

```
userSchema.methods.setPassword = function(password){
    this.salt = crypto.randomBytes(16).toString('hex');
    this.hash  =  crypto.pbkdf2Sync(password,  this.salt,  1000,  64,
'sha512').toString('hex');
};

userSchema.methods.validPassword = function(password) {
    var  hash  =  crypto.pbkdf2Sync(password,  this.salt,  1000,  64,
'sha512').toString('hex');
    return this.hash === hash;
};
```

Listing 8 Mongoose functions for setting and validating a password

After managing the password, we need to retrieve other user data like first, last names or an account picture. This will be covered by the use of the Google API linked with the standard Oauth 2.0. Even though this authentication involves the client, we have encapsulated it in the back-end to make it exchangeable. For example, this could be achieved with the use of passport module, which provides connections with multiple Oauth API like Google, Facebook or GitHub. Since we will one use Google API, no extra complexity is required.

The registration process will be transparent to the user. The first time a user enters its email into our system, it will be detected as unregistered. This will result in the return of our own connection URL with Google Oauth API. The front-end will redirect the user to that URL where Google will ask the user to confirm the revelation of personal information to our suit. This confirmation will be limited to accounts within the organization limits.

After the confirmation has been successful, Google will redirect the user to a previous determined callback. This callback URL needs to be a public DNS or localhost and return a code within other parameters. Now the user will be again in our scope and with the use of that code, the new account will be processed directly querying Google from the back-end.

The query from Google will return the parameter "hd" which contain the domain name of the email account. It will be used to give the initial authentication level, 1 for students and 64 for professors, as students accounts always finish on "hawk.iit.edu" while professors account finish on "iit.edu".

Finally, the new user will be stored into our database and a token will be returned to the front-end to continue to the user site.

4.2.2 DeepSpeech integration

Besides this method hasn't been integrated in the test suit because of the criteria presented on the next points. It is integrated in the comparation tool as a callable microservice as a proof of concept. Extraction the transcription as a whole service allow us to scale it horizontally, increasing the number of containers or vertically, giving it more computational power, in case of usage peaks. All this without translating this impact to the hole system.

This tool has a grpc client in the Node.js side and a grpc server in the Python side. Even though Python is single threaded because of GIL (Global Interpreter Lock), thanks to this service we expand its capabilities creating and expandable number of workers and communicating different programming languages.

Both will communicate following this proto message structure. It configures a TranscribeService that receives the route of a file and returns the output transcripts.

```
syntax = "proto3";
package transcriber;

service TranscribeService {
    rpc getTranscription(fileRoute) returns (transcripts) {}
}

message fileRoute {
    string fileRoute = 1;// number order of attribute
}
message transcripts {
    string jsonTranscript = 1;
}
```

Listing 9 GRPC service proto description

4.2.3 STT Method Comparation

We have created meaningful set of tests, getting hand transcriptions of different academic scenarios to test the both private and open-source options and finally show a measure of each output accuracy with the application of string similarity algorithms as Levenshtein Distance, Cosine Similarity or Jaro Wrinker thresholds. [20] [21]

The data tested we used 7 five-minute length recordings of 3 types. Our first was a two-person technical conversation going over a class introduction. The next three samples were five-minute class lectures from a male voice and final three samples were class lecture samples using a female voice. The content of each recording is not the same, but we tested each sample in our test-suite on both the AWS Transcribe service and using DeepSpeech.

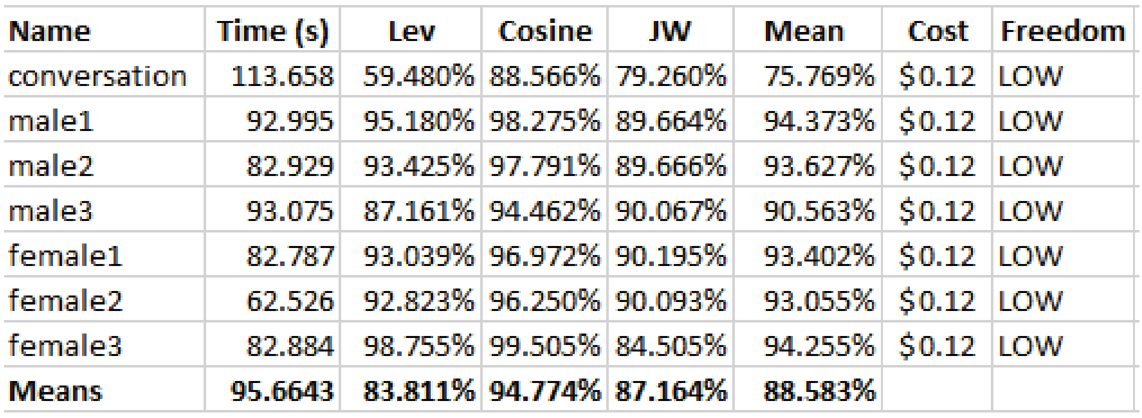
We felt these scenarios were a real test of an actual record with invariant noise as well as ambient sounds like sound from the back of the class were students may introduce noise, multiple speakers in a conversation. These class situations are likely to happen. Without forgetting to represent both equal number of male and female voices as the tone of voice changes and the transcription technology must take this into account. However, as we will see in the results, the outputs are mostly accurate.

4.2.4 Selection Criteria

Be aware that DeepSpeech technology require a CPU with support of AVX/AVX2 instruction, it shouldn't be a high entry barrier as AVX extensions were introduced on Intel CPUs in June 2013.

Despite the fact that cloud-based services have had a longer development term, we will first compare both selected methods to give a better view of their performance based on the most important measures: time and accuracy based on our sources. [22]

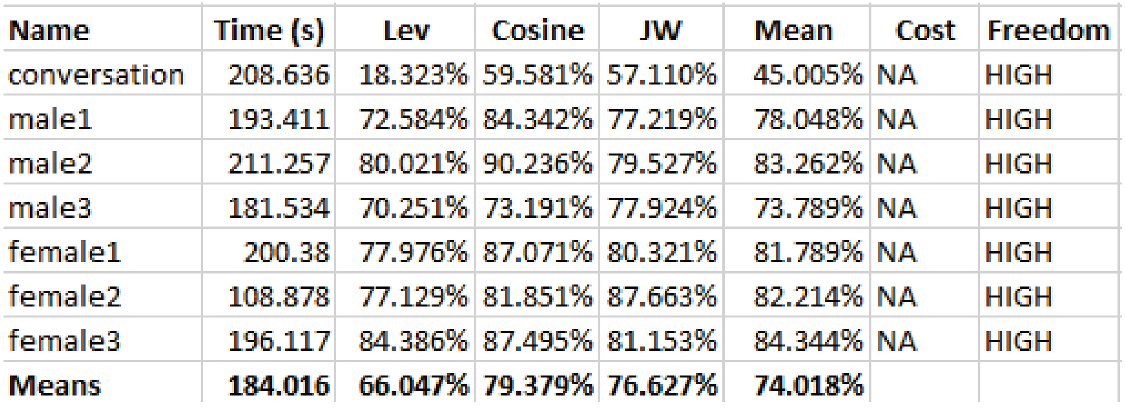
The AWS method here includes in the time category the latency for upload and retrieval of data. And lists the cost to render a 5-minute audio clip.



Name	Time (s)	Lev	Cosine	JW	Mean	Cost	Freedom
conversation	113.658	59.480%	88.566%	79.260%	75.769%	$0.12	LOW
male1	92.995	95.180%	98.275%	89.664%	94.373%	$0.12	LOW
male2	82.929	93.425%	97.791%	89.666%	93.627%	$0.12	LOW
male3	93.075	87.161%	94.462%	90.067%	90.563%	$0.12	LOW
female1	82.787	93.039%	96.972%	90.195%	93.402%	$0.12	LOW
female2	62.526	92.823%	96.250%	90.093%	93.055%	$0.12	LOW
female3	82.884	98.755%	99.505%	84.505%	94.255%	$0.12	LOW
Means	**95.6643**	**83.811%**	**94.774%**	**87.164%**	**88.583%**		

Figure 26 AWS measures

While, the DeepSpeech method has output the following performance measures:



Name	Time (s)	Lev	Cosine	JW	Mean	Cost	Freedom
conversation	208.636	18.323%	59.581%	57.110%	45.005%	NA	HIGH
male1	193.411	72.584%	84.342%	77.219%	78.048%	NA	HIGH
male2	211.257	80.021%	90.236%	79.527%	83.262%	NA	HIGH
male3	181.534	70.251%	73.191%	77.924%	73.789%	NA	HIGH
female1	200.38	77.976%	87.071%	80.321%	81.789%	NA	HIGH
female2	108.878	77.129%	81.851%	87.663%	82.214%	NA	HIGH
female3	196.117	84.386%	87.495%	81.153%	84.344%	NA	HIGH
Means	**184.016**	**66.047%**	**79.379%**	**76.627%**	**74.018%**		

Figure 27 Deepspeech measures

4.2.5 Results

We started on the presumption the cloud tools are more accurate. However, depending on the project and the desire to retain control of the transcripts, the cloud service might not be the best option. Also, in the case of a university, the production of media is very fast. This will be transformed into more than half of the monthly application budget spent on the cloud technology chosen. However, with the open source technologies, the own project would begin to feed back with its own videos improving the accuracy.

Comparing figure 3 and 4 and looking at the results we find some initial outliers. Female2 results are highly irregular. With the AWS test for the time category having a standard deviation of 15.45 and a mean of 96.66, we can discard this result as being greater than 1 standard deviation.

The same results happen in the DeepSpeech tests as there is a standard deviation of 35.31 with a mean 184.01. Female2 is outside of one standard deviation.

This result is due to a prevalence of white noise or silence in the recording, which reflected in the increased accuracy. Taking this into consideration our results are as follows:

- We see the time value of each test essentially doubled on DeepSpeech
- We see a marked difference in the results of the conversation sample between AWS Transcribe and DeepSpeech—113 vs 200 seconds due to multiple speakers
- We see similar accuracy percentages on the remaining six tests per speech library (internally consistent)
- The means are averages of all of the results per category and per test
- Most accurate was female2 using Cosine Similarity and Male1 Cosine Similarity for AWS Transcribe
- DeepSpeech most accurate was Female3 Cosine Similarity and Male2 Cosine Similarity
- The quickest render time was Female2 was most accurate (due to much whitespace) then followed by Female3 for AWS Transcribe
- The quickest render time for DeepSpeech was Female2 (white space again) Male3

With the current results, we can say that Speech-To-Text cloud services at this time are enough advanced to provide an adequate accuracy to the problem presented by this project. Along with the fact that we can combine the final output with small fixes made by professors, teachers assistants and even students to stand that mostly 100% of accuracy.

The AWS transcribe cost was $0.0004 per second and can be extrapolated out consistently. For comparison sake a final category entitled: Freedom, for Software Freedom was included in the decision matrix for those that value this aspect of technology.

4.3 MongoDB

As any application that needs data to work, we will need a database. The first concern is whether or not to choose a relational database. We can ask this question since our application doesn't support critical consistent decisions like medical treatment which should support ACID (atomicity, consistency, isolation, durability) principles. As an example, our application will create the video document before the transcriptions has been processed, which could take minutes, and the student will see a video without transcription at that time. The student can deal with the lack of transcription for some minutes.

There are a few main differences to be familiar with when deciding which database works best for your needs.

- Relational databases historically have worked well, when data structures were much simpler and more static. In the last decade, the non-relational offers a more flexible, scalable, cost-efficient, alternative.
- NoSQL databases feature dynamic schema and allow you to use what's known as "unstructured data." This means you can build your application without having to first define the schema. In a relational database, you are required to define your schema before adding data to the database. Changing the schema structure in a relational database can be extremely expensive
- Relational databases are table-based. NoSQL databases can be document based, graph databases, key-value pairs, or wide-column stores. Today, we know that data is much more complex.
- Relational databases are vertically scalable but typically expensive. Since they require a single server to host the entire database, in order to scale, you need to buy a bigger, more expensive server. Scaling a NoSQL database is much cheaper, compared to a relational database, because you can add capacity by scaling horizontally over cheap, commodity servers.
- NoSQL databases tend to be more a part of the open-source community. Relational databases are typically closed source with licensing fees baked into the use of their software. [23]

For the purpose of the test suit and looking at the statements defined before, we have determined the use of a non-relational database. The decision main drivers regard on this application as an open-source suit and also refer to the complexity of our data. As for the open-source licensing we need to be flexible and cost-efficient. For the complexity, even though we have decided to provide the transcription with an Amazon service, we leave the door open to change it to a different model as DeepSpeech has. Using a non-relational database would let our suit continue its work and additional enhances over DeepSpeech service could bring more changes to that model. Also, we can understand that complexity by look at the three different models active right now for storing AWS transcription service output.

```
const itemSchema = new mongoose.Schema({
    start_time: { type: String },
    end_time: { type: String },
    alternatives: [{ confidence: String, content: String}],
    type: { type: String }
},{collection: 'item'});
```

```
const resultSchema = new mongoose.Schema({
    transcripts: { type: String },
    item: [{type: mongoose.Schema.Types.ObjectId, ref: 'item'}]
},{collection: 'result'});

const transcriptSchema = new mongoose.Schema({
    jobName: { type: String },
    accountID: { type: Number },
    result: { type: mongoose.Schema.Types.ObjectId, ref: 'result'},
    videoID: { type: String }
},{collection: 'transcript'});
```

Listing 10 Nested schemas MongoDB

There are some additional features that reinforce our decision on MongoDB like the use of JSON objects throughout the application, the possibility to interact with the database without changing the programming language, the possibility to make updates faster or having a flexible data model that doesn't need to be immediately defined.

For development purposes, we have made use of the docker image called mongo-express. It provides a web interface over a mongo database or cluster. It has been used as an administrator view while the administrator user wasn't available. It allows us to perform all CRUD operations and see the current database schemas allowing us to mock the courses assigned to a user. Then, this feature was moved to and Angular view.

4.4 Testing

The main objective for this phase is to review whether the test suit developed, and specifically the code, meets the requirements that at the beginning of this project were proposed as an objective to achieve.

We have created the first level of software testing as it is unit test for both the front-end and the back-end mocking the relation between other components to ensure a correct deployment. unit test applies to individual units/ components of a software tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output. [24]

Something important while development following agile methodologies is to maintain a test structure that allows us to detect the failures introduced when implementing some new functionality or when refactoring the code. If good unit tests are written and if they are run every time any code is changed, we will be able to promptly catch any defects introduced due to the change. Another benefit is the generation of easier reusability code.

Between the tests, an object is always generated that simulates the behavior of the component, which is called MOCK so they must import and compile themselves the modules that the component or service they represent will need.

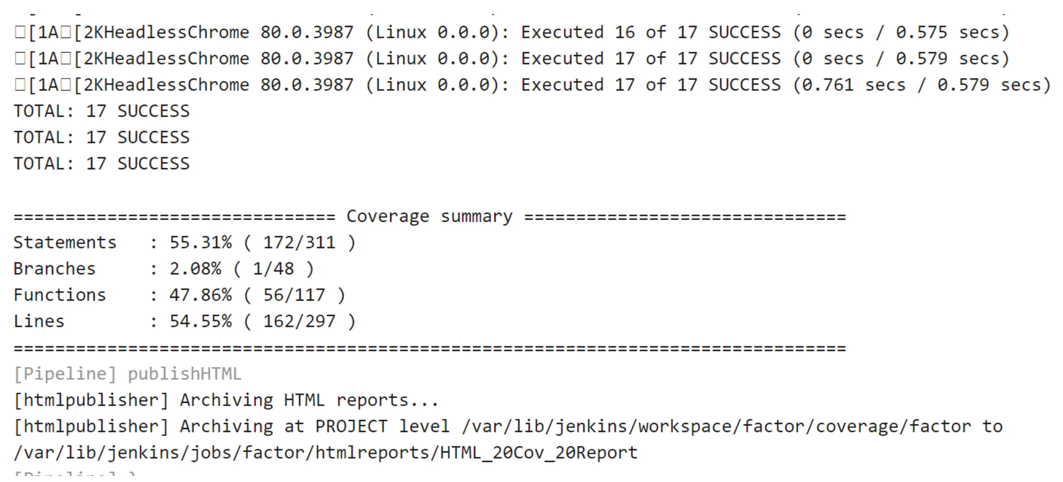
For the Angular front-end, the test has been written using Karma, a test suit created directly to suit Angular testing. Angular is a test-driven framework, they always provide initial test in a different file while creating any new component. Due to the execution of this test within the boundaries of a Jenkins pipeline, we had to change the initial GUI configuration for the puppeteer module. This module has in its packets, his own chromium distribution and it allows to run chrome headless and without a sandbox. Because a testing process cannot depend on having a view or Chrome installed in the server which runs Jenkins.

When performing these tests, we check that both the template and the controller work in coordinated and generate the view correctly, as it's expected by the basic test.

```
it('should be created', () => {
 expect(component).toBeTruthy();
});
```

Listing 11 Basic Angular component test

This are the actual number of test running. It covers more than the 50% of possible paths across the code.



```
□[1A□[2KHeadlessChrome 80.0.3987 (Linux 0.0.0): Executed 16 of 17 SUCCESS (0 secs / 0.575 secs)
□[1A□[2KHeadlessChrome 80.0.3987 (Linux 0.0.0): Executed 17 of 17 SUCCESS (0 secs / 0.579 secs)
□[1A□[2KHeadlessChrome 80.0.3987 (Linux 0.0.0): Executed 17 of 17 SUCCESS (0.761 secs / 0.579 secs)
TOTAL: 17 SUCCESS
TOTAL: 17 SUCCESS
TOTAL: 17 SUCCESS

=============================== Coverage summary ===============================
Statements   : 55.31% ( 172/311 )
Branches     : 2.08% ( 1/48 )
Functions    : 47.86% ( 56/117 )
Lines        : 54.55% ( 162/297 )
================================================================================
[Pipeline] publishHTML
[htmlpublisher] Archiving HTML reports...
[htmlpublisher] Archiving at PROJECT level /var/lib/jenkins/workspace/factor/coverage/factor to
/var/lib/jenkins/jobs/factor/htmlreports/HTML_20Cov_20Report
```

For the Node.js tier, the current dependencies are mocha and supertest modules that allow us to mock the mongo database for testing purposes.

For example:

```
describe('GET /', function() {
    it('respond with array', function(done) {
        request(app).get('/courses').end(function(err, res) {
            expect(res.statusCode).to.equal(200);
            expect(res.body).to.be.an('array');
            done();
        });
    });
});
```

Listing 12 Basic Node.js test

5 Reliable Delivery Process

This test suit is pretended to be distributed to the academic circle in order to wait for next contributions in order to achieve larger progress adding contributions to the overall deployment and it should be adapted to each university necessities. That is the reason why it has been planned to deploy a universal installation mechanism where the new client can select its configurations. Independently from the results that has been shown on the transcription selection process, a following study group could take more importance on the privacy section or could decide to train a new DeepSpeech model and see its results.

For any of these possible reasons. The ability to support the entire test suit over a fail-save process is considered crucial. This deployment should be aligned with the classic engineering cycle of design, construction, testing and validation.

5.1 Deployment

Containerized systems have been selected as they provide a way to build, test and deploy applications in multiple environments from a developer's local laptop to an on-premise data center and even the cloud, all producing the same result.

We will also take advantage of their benefits as its better support for microservices architectures, which can be more easily isolated, deployed, and scaled, or its DevOps support for continuous integration and deployment (CI/CD) like we will show later in the document.

As we are getting involved in container environments. We are likely to hear about two popular tools and platforms used to build and manage containers. These are Docker and Kubernetes.

Docker is a runtime environment used to create and build software inside containers. It packages up code and all its dependencies, so the application runs quickly and reliably from one computing environment to another. A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application. in the case of Docker containers, images become containers when they run on Docker Engine. [25]

The first step on the deployment of the test suit made use of three different containers, one for the Angular front-end service, one for the Node.js back-end and the last one for the Mongo database. All this was initially controlled with a docker tool called Compose.

This tool helps to define and run multi-container Docker applications. With Compose, you use a YAML file to configure your application's services. Then, with a single command, you create and start all the services from your configuration. [26]

It allows to see the test suit as a unique application which runs with the hit of only one terminal command, “docker-compose up”.

However, containerized applications can get complicated. This starts to get pictured when you want another container to manage and view the current database data. Then, you want to add some replication to the database tier, which involves some more containers, or you want to subdivide each tier creating a microservice structure. Finally, you get to the production deadline and depending on the quantity of the requests expected by the system, in our case the whole university, you realize that only one container for front-end and back-end tiers won't be enough. When in production, many applications might require hundreds to thousands of separate containers. Then when you realize the need of other tools that orchestrate or manage all that different containers. This is when Kubernetes takes in place.

5.2 Kubernetes

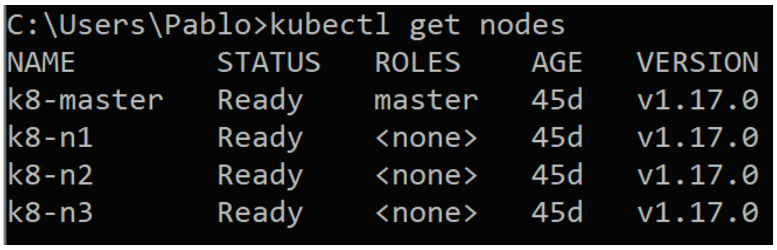
This is a container orchestrator that admits multiple container runtime environments, including Docker. According to its own website, Kubernetes (K8s) is an open-source system for automating deployment, scaling, and management of containerized applications. It handles areas within the underlying infrastructure resources for containerized applications such as the amount of compute, network, and storage resources required. Orchestration tools as K8s make it easier to automate and scale container-based workloads for live production environments. [27]

Kubernetes has a big number of features that make it the perfect platform to operate high amounts of containers. Some of them are:

- Service discovery so that each container can find and communicate with another.
- Horizontal scaling and load balancing to distribute the network traffic so that the deployment is stable.
- Storage orchestration local storages, public cloud providers, and more.
- Automated rollouts and rollbacks to create new containers for your deployment, remove existing containers and adopt all their resources to the new container.
- Automatic bin packing for the best use of CPU and RAM over a cluster of nodes.
- Self-healing (restarts crashed containers, reschedules pods to healthy hosts etc.)
- Secret and configuration management [28]

To apply all these abstractions to manage our container-based application, first we need a Kubernetes cluster. This cluster consists of one or more worker machines, in its notation called nodes. In our case, we have deployed a Kubernetes cluster with a master node, which only runs the control plane and route the requests to three slave nodes communicated over a Flannel network.

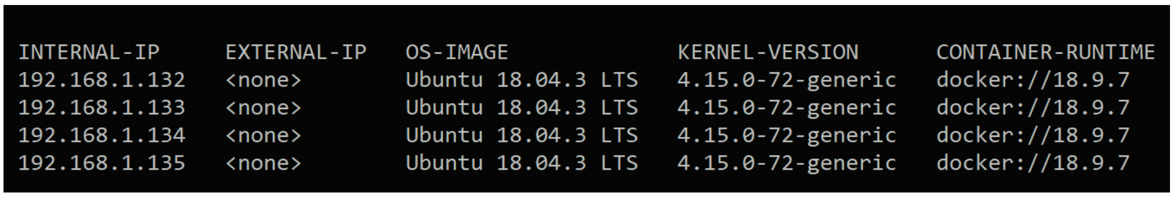
We have copied the kubeconfig file because it has the cluster credentials to the development computer, independent from been IOS, Windows or Linux. If we run in a console the next command "kubectl get nodes", we will get the actual status of the cluster.



```
C:\Users\Pablo>kubectl get nodes
NAME        STATUS   ROLES    AGE   VERSION
k8-master   Ready    master   45d   v1.17.0
k8-n1       Ready    <none>   45d   v1.17.0
k8-n2       Ready    <none>   45d   v1.17.0
k8-n3       Ready    <none>   45d   v1.17.0
```

Figure 28 Kubernetes cluster

We can add the parameter “-o wide” to get more information.



```
INTERNAL-IP     EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION      CONTAINER-RUNTIME
192.168.1.132   <none>        Ubuntu 18.04.3 LTS   4.15.0-72-generic   docker://18.9.7
192.168.1.133   <none>        Ubuntu 18.04.3 LTS   4.15.0-72-generic   docker://18.9.7
192.168.1.134   <none>        Ubuntu 18.04.3 LTS   4.15.0-72-generic   docker://18.9.7
192.168.1.135   <none>        Ubuntu 18.04.3 LTS   4.15.0-72-generic   docker://18.9.7
```

Figure 29 Kubernetes cluster description

As we said, the master node takes care of the services shown on the next image while the slave nodes run the container images. The control plane's components make global decisions about the cluster like scheduling or starting up a new pod when a deployment's replicas field is unsatisfied.

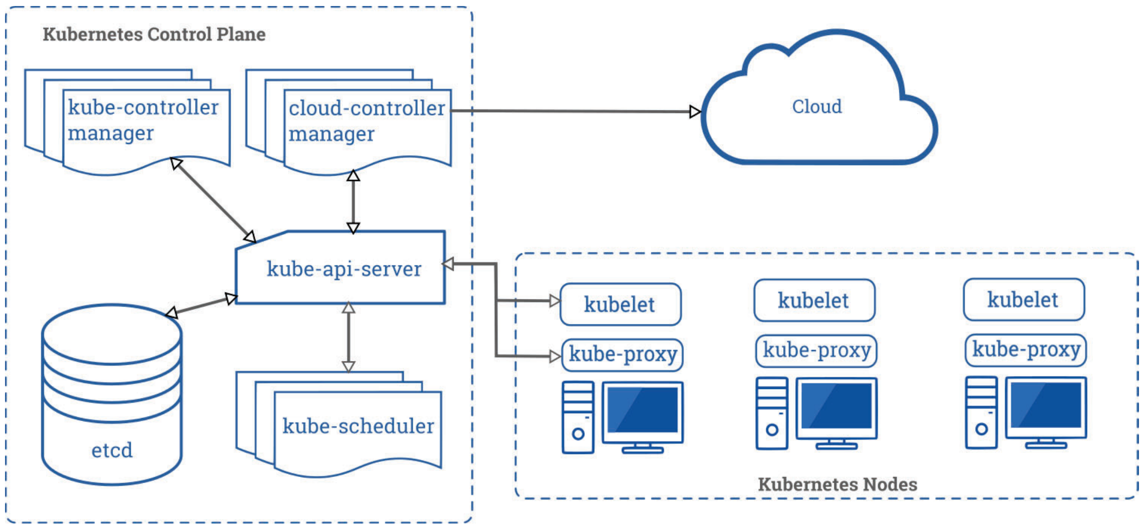
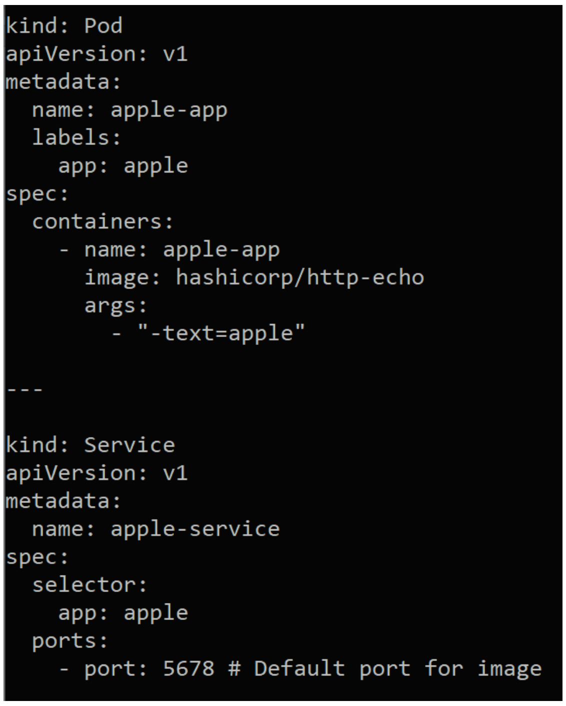


Figure 30 Kubernetes control plane. Source [28]

However, bare-metal environments lack on load balancers available on-demand requiring a slightly different setup to offer the same kind of access to external consumers. That’s why we required to set MetalLB, it is a software load balancer solution that provide a single external ip to the NGINX Ingress controller, allowing us to connect external clients with any application running inside the cluster. This IP relays on the kubernetes port type NodePort, which is usually ranged between ports 30000 and 32767. This was modified adding a list of allowed external IPs, the IP of all cluster nodes. Achieving the visualization of this internal applications over known ports as 80 and 443 from any of these IPs.

5.2.1 Infrastructure as code

To test the correct function of the cluster, we have created an example application. It consists of two containers, one is a nginx image with the index.html file showing "apple" and another showing "banana". On the Kubernetes side, we have a three yaml file which describes infrastructure as code. The first yaml describe the apple site. After running it, K8s will initialize the apple nginx container, in Kubernetes called Pod and a service that select this pod.



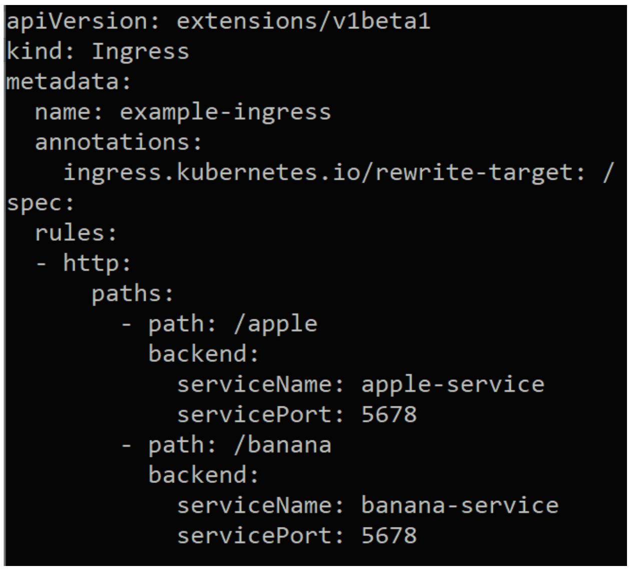
```
kind: Pod
apiVersion: v1
metadata:
  name: apple-app
  labels:
    app: apple
spec:
  containers:
    - name: apple-app
      image: hashicorp/http-echo
      args:
        - "-text=apple"

---

kind: Service
apiVersion: v1
metadata:
  name: apple-service
spec:
  selector:
    app: apple
  ports:
    - port: 5678 # Default port for image
```

Figure 31 Yaml pod and service Kubernetes

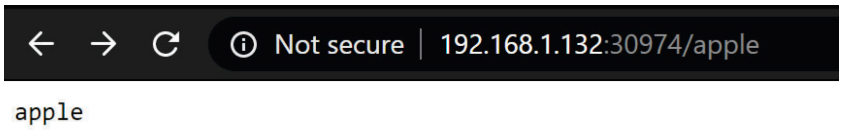
The second yaml will do the same for the banana site and the last one will describe an ingress. It will route the user request to the services depending on the requested URI.



```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: example-ingress
  annotations:
    ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - http:
      paths:
        - path: /apple
          backend:
            serviceName: apple-service
            servicePort: 5678
        - path: /banana
          backend:
            serviceName: banana-service
            servicePort: 5678
```

Figure 32 Yaml ingress kubernetes

Both Pods can now be accessed from the browser using the master node IP.

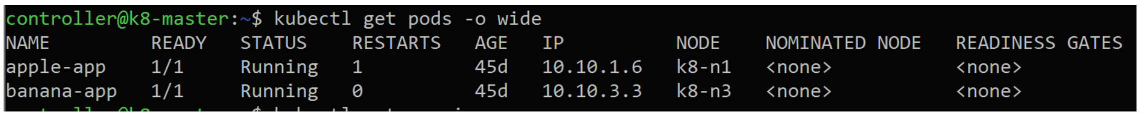


```
Not secure | 192.168.1.132:30974/apple

apple
```

Figure 33 Web access to ingress endpoint

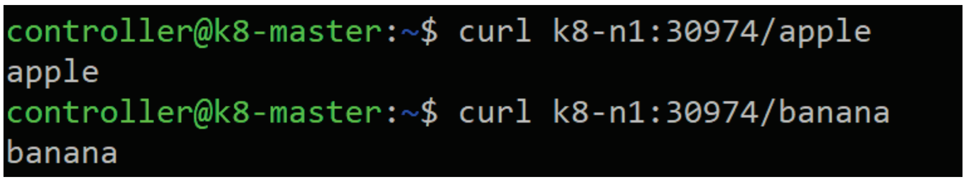
We can also see that the nodes run on the slave nodes running the following command.



```
controller@k8-master:~$ kubectl get pods -o wide
NAME          READY   STATUS    RESTARTS   AGE   IP           NODE    NOMINATED NODE   READINESS GATES
apple-app     1/1     Running   1          45d   10.10.1.6    k8-n1   <none>           <none>
banana-app    1/1     Running   0          45d   10.10.3.3    k8-n3   <none>           <none>
```

Figure 34 Pods description, running node detail

If we are inside the network, we can query a slave node directly. We will see the same response which was managed by Kubernetes before.



```
controller@k8-master:~$ curl k8-n1:30974/apple
apple
controller@k8-master:~$ curl k8-n1:30974/banana
banana
```

Figure 35 Console access to ingress endpoint

5.2.2 Final Resource Infrastructure

This are the objects infrastructure relevant to this test suit. It can be created with just one command and will maintain the whole environment.

Tier	**Pod manager**	**Number replicas**	**Persistent Volumes**	**Service**	**External Ingress**
Front-end	Deployment	3	No	Yes	Yes /*
Back-end	Deployment	2	1 shared	Yes	Yes /api(/|$)(.*)
Database	Statefulset	3	3 detached	Yes	No

Listing 13 Test Suit IaC Kubernetes

We have decided to use a stateful set for mounting the mongo replicaset because a StatefulSet maintains a sticky identity for each of their Pods while Deployments create random Pods.

The number of replicas can be increased at any time in case of necessity or it would be positive to configure an automatic horizontal scaler.

Finally, although the back-end handles requests on the "/api*" route our ingress controller rewrite all these routes erasing the "/api" and redirecting them to the port 3000.

5.3 Jenkins

Once the production cluster is prepared to receive the latest production version, we are able to automate the whole testing and delivery process automatically. This process will be managed by Jenkins leaving our automation instructions on a Jenkinsfile. The end purpose of adding all this complexity is to allow us to distribute our last reviewed versions to the public, by just using the command line “git push”.

According to its own documentation, “Jenkins is a self-contained, open source automation server which can be used to automate all sorts of tasks related to building, testing, and delivering or deploying software.”. [29]

We'll build a complete Continuous Integration/Continuous Delivery (CI/CD) pipeline with automated builds, deployments to a dev environment, testing of the image, and if everything follows the correct path, deliver the changes to the production Kubernetes cluster.

The first part, the CI is a developer practice that requires integrating the code changes continuously with the rest of the team. It works in three simple stages, push, test, and fix. As the integration test are performed automatically, once a developer gets a test failed warning, its first priority will be correcting its code. In case the purpose of the code has changed, test changes should be changed prior to code changes.

The second part, the CD is the process to update those changes to production. It is a critical environment which should be protected from errors as it will be the communication bridge with our clients. However, by introducing automation on this labor, we increase the sustainability of the process.

This automation involves on the coordination of several practices. In our case, we will use pipeline-as-code and infrastructure-as-code. Pipeline-as-code is stored on a Jenkinsfile at each service repository. It allows to manage and execute Jenkins jobs as code. Infrastructure-as-code use YAML files from that pipeline to update changes on the Kubernetes infrastructure. These last practice helps to dynamically provision and tears down environments. [30]

A complete pipeline in a complex project managed between different teams would look like this. It has a manual approval of the submitted changes due to the presence of a QA team. It could be interesting while introducing new testing patterns as blue-green deployments or canary strategies. Those strategies help to show the new changes to a reduced number of requests and see if the modification is accepted by that reduced size of clients before gradually replacing previous versions. [31]

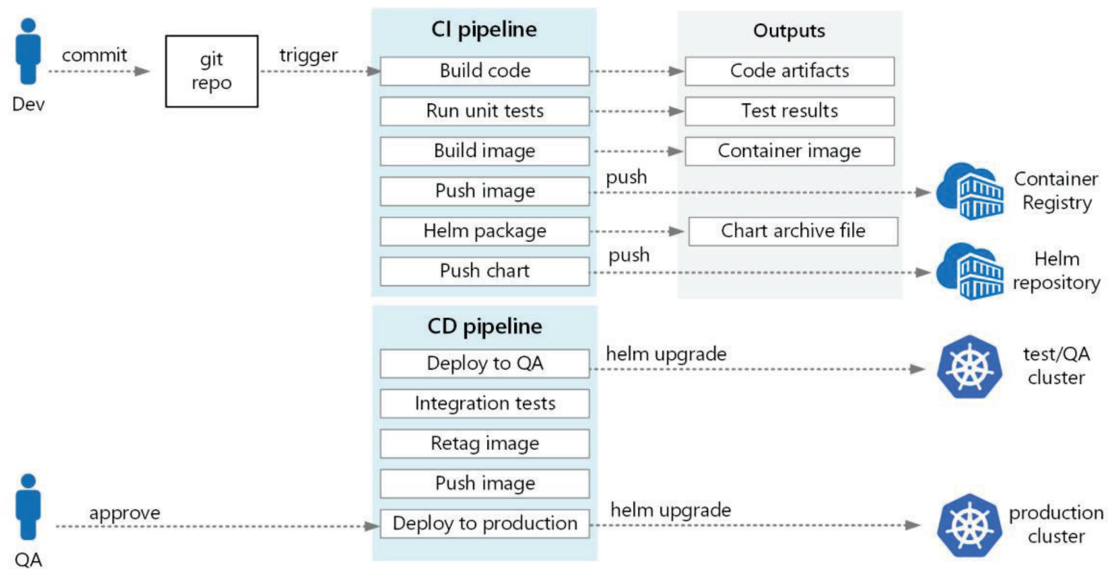


Figure 36 Complete CI/CD pipeline for microservices in azure. Source: [31]

In our case, due to time and employee restrictions, we had to change the assumption that there's only one developer and production changes will be automatically accepted if tests pass. Which end up on the creation of this pipeline.

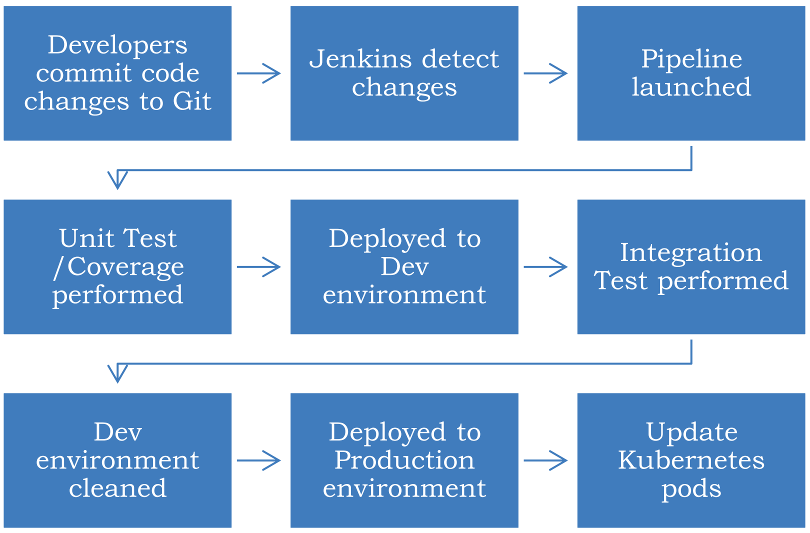

Developers commit code changes to Git
Jenkins detect changes
Pipeline launched
Unit Test /Coverage performed
Deployed to Dev environment
Integration Test performed
Dev environment cleaned
Deployed to Production environment
Update Kubernetes pods

Figure 37 CI/CD pipeline Test Suite

All these steps are described in a Jenkinsfile uploaded to the Github repository. This will be the pipeline-as-code side. Due to the fact that our Jenkins is only available from the internal university VPN, adding "triggers{ cron('H/15 H(9-16) * * 1-5') }" will detect changes on the Github repository each 15 min while in working hours. Allowing us to notice if something had stopped working without affecting the production environment.

The "Test/Coverage" stage allow us to run puppeteer as we had specified in the testing paragraph. This will output an HTML coverage report that will be published on Jenkins. Finally, the build command is executed and the distribution files are inserted into a Nginx image an push it to DockerHub making it globally available.

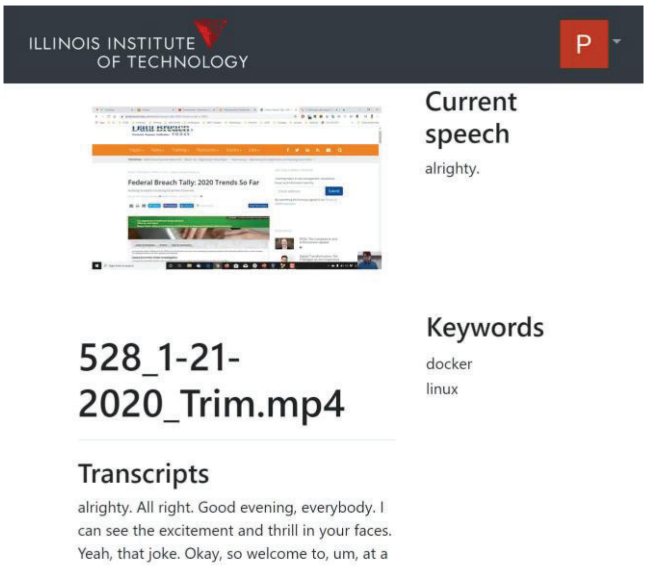
6 Conclusions

This case study provides a beginning to developing a test-suite and guide to compare Speech-to-Text libraries and their out of the box accuracy while integrating these libraries into a single test application.

After comparing both cloud an open-source methodology, we can state that the cloud option provides higher accuracy and lower processing time in its initial configurations. Nevertheless, there is not a clear answer for the best Speech-to-Text library.

It is cliché to say, “It depends,” but it does, depending on you and or your institutions values. If cost is a constraint that is one factor. If Software Freedom and the ability to control your own software is important, that is one factor. If you are willing to trade percentages of accuracy in order to reduce costs, that is one factor. Finally, if you have a large group of members or students willing to crowdsource and fix transcriptions then using an Opensource library such as DeepSpeech may be an appealing factor. It is important to test the results and compare the factors to make the most informed decision that is right for you. We believe our test suite is the first step in helping make that decision. We will allow next researchers to choose its transcription option on the environment settings.

The prototype is currently available for IIT students and regards on AWS transcribe to process the media files and Google Oauth 2.0 to limit the usage to these college accounts. It divides students, teachers and teacher assistant with a 2^7 number regarding to the authentication level. Its final propose is leading students to a view similar to one shown below.



ILLINOIS INSTITUTE OF TECHNOLOGY

P

Current speech

alrighty.

Keywords

docker

linux

528_1-21-2020_Trim.mp4

Transcripts

alrighty. All right. Good evening, everybody. I can see the excitement and thrill in your faces. Yeah, that joke. Okay, so welcome to, um, at a

Figure 38 Example view video transcriptions

The test suit value not only comes from its features. It also establishes a uniform deployment and delivery process to make it more stable and failure probe. We see that deploying a Kubernetes infrastructure is complex, but these are crucial tools to be prepared to the end-user delivery within a large project, which specs further improvements.

6.1 Future Improvements

The test suite is currently outfitted for only two Speech-To-Text frameworks. More research needs to be done to make a larger matrix comparison of the major services and Opensource libraries. In addition, options could be added to prevent of lockout non-freedom respecting software libraries. A larger shared corpus of sample videos would help to smooth out the initial proof-of-concept as well, by generating new and enhanced inference models.

In the application side, the promotion of a student account to teacher assistant feature hasn't been able to be provided. The authorization structure is set, on the back-end side it would only require the video site a way to handle the increase of authentication level. In the frontend side, it is required to allow student to propose modification and then, professors would accept them as a way to promote the account. While uploading a video, it could be also added a dictionary text box to support an enhance transcription parameters. Finally, the DeepSpeech WAD system provided by Mozilla, could be also enhanced to match the AWS transcribe output parameters, which has more details.

We invite collaboration and extension of this existing test suite.

7 Bibliography

[1] IBM, «IBM White Paper,» 8 2010. [En línea]. Available: ftp://public.dhe.ibm.com/ftp/lotusweb/IBM_White_Paper_-_Value_of__eLearning_-_2Q2010_-_FINAL.pdf.

[2] V. N. H. H. M. H. M. S. M. a. S. N. Sonal Shetty, «Content Based Audiobooks Indexing using Apache Hadoop Framework,» *Proceedings of the Third International Symposium on Women in Computing and Informatics - WCI '15,* 2015.

[3] L. Berke, «Displaying Confidence From Imperfect Automatic Speech Recognition for Captioning,» *SIGACCESS,* 2017.

[4] E. Vinson, «opaltranscriptionservices,» [En línea]. Available: https://www.opaltranscriptionservices.com/transcription-rates/. [Último acceso: 2 2020].

[5] «AWS Transcribe,» [En línea]. Available: https://aws.amazon.com/transcribe/pricing/. [Último acceso: 2020].

[6] B. Prajapati, «medium,» 20 4 2018. [En línea]. Available: https://medium.com/tribalscale/6-things-to-consider-when-building-large-scale-web-applications-237816d11b68.

[7] M. Richards, Software Architecture Patterns, O'Reailly Media.

[8] M. S. H. R. F. B. Douglas C. Schmidt, Pattern-Oriented Software Architecture, Wiley, 2000.

[9] thegradient, «Towards an imagenet moment for speech-to-text,» 28 3 2020. [En línea]. Available: https://thegradient.pub/towards-an-imagenet-moment-for-speech-to-text/.

[10] M. Loxton, «How to Use AI Transcription Services with MAXQDA,» *MAXQDA Research Blog,* vol. 4, nº 29, 2019/4.

[11] Mozilla, «Why Common Voice?,» Mozilla, 2020. [En línea]. Available: https://voice.mozilla.org/en/about.

[12] A. R. C. K. F. R. Shimaa Ahmed, «Prεεch: A System for Privacy-Preserving Speech Transcription,» 18 Feb 2020. [En línea]. Available: https://arxiv.org/pdf/1909.04198.pdf.

[13] mongodb, «Mean stack,» MongoDB, [En línea]. Available: https://www.mongodb.com/mean-stack.

[14] M. Sultan, «Angular and the Trending Frameworks of Mobile and Web-based Platform Technologies: A comparative analysis,» *Future Technologies Conference (FTC) ,* 2017.

[15] C. Cleary, «coreycleary,» 2018. [En línea]. Available: https://www.coreycleary.me/project-structure-for-an-express-rest-api-when-there-is-no-standard-way/. [Último acceso: 11 2019].

[16] J. Hibbard, J. Kolce, L. White y J. Wi, 9 Practical Node.js Project.

[17] B. D. C. D. Brad Dayley, Node.js, MongoDB and Angular Web Development, Pearson Education, 2018.

[18] grpc, «FAQ,» grpc, [En línea]. Available: https://grpc.io/faq/.

[19] P. Nohe, «thesslstore,» 18 12 2018. [En línea]. Available: https://www.thesslstore.com/blog/difference-encryption-hashing-salting/.

[20] T. L. a. E. Merlo, «An accurate estimation of the Levenshtein distance using metric trees and Manhattan distance,» IWSC, 2012. [En línea]. Available: https://dl-acm-org.ezproxy.gl.iit.edu/doi/pdf/10.5555/2664398.2664399.

[21] S. H. H. a. I. K. Matthew F. Dabkowski, «Improving record linkage for counter-threat finance intelligence with dynamic Jaro-Winkler thresholds,» *WSC,* 2019.

[22] Andler, «Predicate path expressions ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages,» *ACM Press, New York, NY, 226-236..*

[23] mongodb, «nosql vs relational databases,» mongodb, [En línea]. Available: https://www.mongodb.com/scale/nosql-vs-relational-databases.

[24] softwaretestingfundamentals, «unit-testing,» softwaretestingfundamentals, [En línea]. Available: http://softwaretestingfundamentals.com/unit-testing/.

[25] docker, «what container,» docker, [En línea]. Available: https://www.docker.com/resources/what-container.

[26] docker, «docker compose,» docker, [En línea]. Available: https://docs.docker.com/compose/.

[27] kubernetes, «Index,» kubernetes, [En línea]. Available: https://kubernetes.io/.

[28] kubernetes, «what is kubernetes,» kubernetes, [En línea]. Available: https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/.

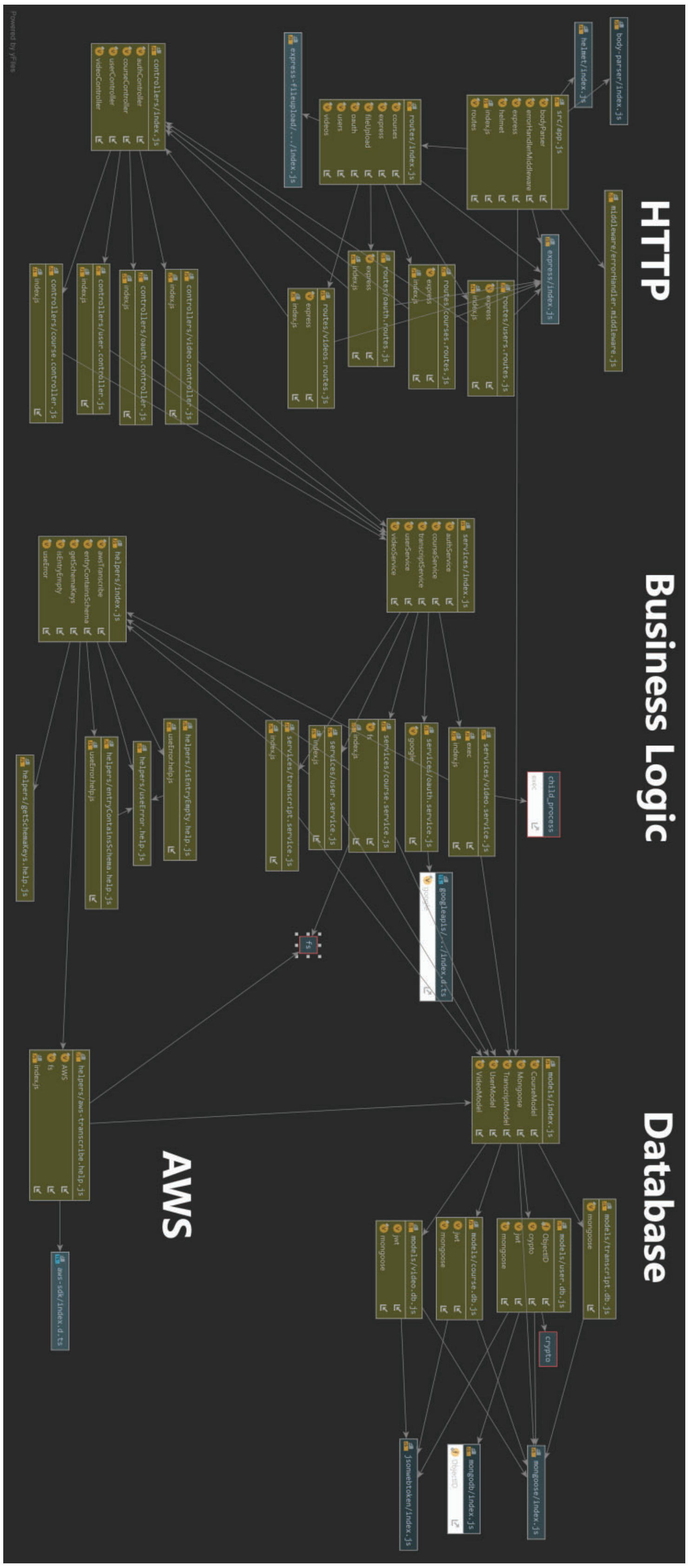
[29] jenkins, «jenkins,» [En línea]. Available: https://www.jenkins.io/doc/book/pipeline/syntax/.

[30] C. Melendez, «stackify,» 30 04 2019. [En línea]. Available: https://stackify.com/what-is-cicd-whats-important-and-how-to-get-it-right/.

[31] microsoft, «Building a CI/CD pipeline for microservices on Kubernetes,» microsoft, 04 11 2019. [En línea]. Available: https://docs.microsoft.com/en-us/azure/architecture/microservices/ci-cd-kubernetes.

8 Annex

8.1 Node code structure



HTTP
Business Logic
Database
AWS