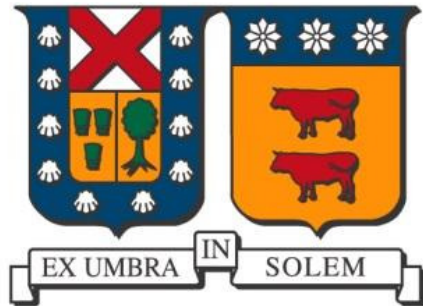


UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
VALPARAÍSO - CHILE



**“DISEÑO, DESARROLLO E IMPLEMENTACIÓN DE UN
SISTEMA DE MONITOREO DE ASIGNATURAS BASADO
EN PORTAFOLIOS DOCENTES”**

YEISON MARCELO FERNÁNDEZ JELVES

**MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN INFORMÁTICA**

Profesor Guía: Cecilia Reyes Covarrubias
Profesor Correferente: Luis Hevia Rodríguez

Agosto - 2023

DEDICATORIA

Para mi familia, pareja y amigos, quienes hicieron posible que llegase a esta instancia.

AGRADECIMIENTOS

En primera instancia me gustaría agradecer a mi familia por toda su paciencia y amor, quienes hicieron muchos sacrificios para que yo pudiese llegar a esta instancia.

Agradecer, también, a mi pareja, Susana, por toda su apoyo, tiempo, paciencia y comprensión.

Agradecer a todas las personas conocidas durante la carrera como Jordan, Allan, Juan Pablo León, José, JP, Felipe, Willy, Pablo entre tantos compañeros. Gracias por toda su ayuda y buena colaboración en toda nuestra vida universitaria.

Finalmente agradecer a todos los docentes que participaron en mi formación y agradecer en especial a Cecilia y Luis por su ayuda en el desarrollo de esta memoria.

RESUMEN

Cada año las universidades reciben una gran cantidad de estudiantes de nuevo ingreso, pero a medida que avanzan en la carrera van desertando por diferentes motivos. En el caso de Ingeniería Civil en Informática de la UTFSM, para el año 2023, el porcentaje de retención al segundo año es del 65%, lo que muestra que existe espacio para reducir la deserción del 35% que se ha tenido en promedio en los últimos 3 años. Es vital para los responsables de una carrera, conocer las causas de la deserción, lo cual se puede lograr monitoreando los contenidos impartidos en las asignaturas, el material usado en ellas y los avances pedagógicos logrados cada vez que se ha dictado, para así posteriormente ser evaluadas y generar acciones de mejoras.

Por lo tanto, en esta memoria se propondrá como solución una plataforma de monitoreo basado en portafolios docentes. Que ayude a la carrera a reducir la deserción o a aumentar la retención, a través de ajustes en la forma en que se dicten las asignaturas.

Palabras Clave: *AWS; MONITOREO; DEVOPS; MICROSERVICIOS*

ABSTRACT

Every year the universities receive many new students, but as they advance in the career they drop out for different reasons. In the case of Civil Engineering in Computer Science at UTFSM, by the year 2023, the retention rate for the second year is 65%, which shows that there is space to reduce the 35% dropout rate that has occurred on average in the last 3 years. It is vital for those responsible for a degree, to know the causes of desertion, which can be achieved by monitoring the contents taught in the courses, the material used in them, and the pedagogical advances achieved each time it has been dictated, in order to subsequently be evaluated and generate improvement actions.

Therefore, in this report a monitoring platform based on teaching portfolios will be proposed as a solution. That helps the career to reduce dropout or increase retention, through adjustments in the way in which the courses are taught.

Keywords: *AWS; MONITORING; MICROSERVICES; DEVOPS*

GLOSARIO

API (Application Programming Interface): corresponde a un conjunto de subrutinas, funciones y métodos que proporcionan una capa abstracta de comunicación entre componentes de software.

AWS (Amazon Web Services): corresponde al conjunto de servicios que ofrece *Amazon* para gestionar la infraestructura en la nube, ofreciendo despliegues de alta escalabilidad y confiabilidad.

CC (Casa Central): se refiere al campus Casa Central de la UTFSM.

CDN (Content Delivery Network): se refiere a un grupo de servidores distribuidos geográficamente, los cuales juntos proveen de contenido de Internet de manera rápida.

C/I (Continuous Integration): permite que desarrolladores incorporen cambios del código a un repositorio compartido, una vez incorporadas las modificaciones se validan con la compilación automática de la aplicación y la ejecución de distintas pruebas.

C/D (Continuous Deployment): corresponde a la automatización del lanzamiento de una aplicación a producción.

DNS (Domain Name System): sistema de nombres de dominios.

ECR (Elastic Container Registry): servicio de *AWS* que permite almacenar, compartir e implementar software de contenedores.

ECS (Elastic Container Service): servicio de *AWS* que permite la gestión de contenedores.

ELB (Elastic Load Balancing): servicio de *AWS* que permite distribuir el tráfico entrante entre varios destinos. Además, permite monitorear el estado de los destinos y realizar autoescalamiento automático para adaptarse a las cargas de trabajo.

GIT: herramienta para el control de versiones.

HTML (HyperText Markup Language): es un lenguaje descriptivo que especifica la estructura de las páginas web.

IAM (Identity and Access Management): servicio de *AWS* encargado de la gestión de usuarios y los permisos que estos tendrán.

IoT (Internet Of Things): se refiere a la red colectiva de dispositivos conectados y a la tecnología que facilita la comunicación entre los dispositivos y la nube, así como entre los propios dispositivos.

S3 (Simple Storage Service): servicio de *AWS* que permite almacenar objetos, archivos, etc.

UTFSM: Universidad Técnica Federico Santa María.

INDICE DE CONTENIDOS

RESUMEN 4

ABSTRACT 4

INDICE DE FIGURAS 9

INDICE DE TABLAS 12

INTRODUCCIÓN 13

CAPÍTULO 1: DEFINICIÓN DEL PROBLEMA..... 14

 1.1 CONTEXTO 14

 1.2 OBJETIVOS 17

 1.2.1 OBJETIVO GENERAL..... 17

 1.2.2 OBJETIVOS ESPECÍFICOS 17

CAPÍTULO 2: MARCO CONCEPTUAL 18

 2.1 SISTEMAS DE MONITOREO 18

 2.1.1 LA IMPORTANCIA DEL MONITOREO 18

 2.1.2 EL MONITOREO COMO UN SISTEMA..... 18

 2.2 PORTAFOLIOS DOCENTES Y LA EDUCACIÓN 20

 2.3 DESARROLLO DE SOFTWARE MODERNO..... 21

 2.3.1 DEVOPS 21

 2.3.2 MICROSERVICIOS..... 24

 2.3.3 AMAZON WEB SERVICES 25

 2.3.4 BASE DE DATOS NOSQL..... 27

 2.3.5 ANGULAR..... 28

 2.3.6 NODEJS..... 29

 2.3.7 NESTJS 29

 2.3.8 TESTING..... 29

CAPÍTULO 3: PROPUESTA DE SOLUCIÓN 31

 3.1 TOMA DE REQUERIMIENTOS..... 31

3.2	USUARIOS	32
3.3	HISTORIAS DE USUARIO	33
3.4	REFINAMIENTO	34
3.5	MODELO DE CLASES	37
3.6	HERRAMIENTAS DE DESARROLLO.....	38
3.7	DISEÑO DE LA ARQUITECTURA	39
3.8	PRESENTACIÓN DE LA PLATAFORMA.....	40
CAPÍTULO 4: VALIDACIÓN DE LA SOLUCIÓN		46
4.1	RESULTADOS PRUEBAS UNITARIAS.....	46
4.1	CUMPLIMIENTO DE LAS HISTORIAS DE USUARIO	47
4.1	IMPLEMENTACIÓN EN DEPARTAMENTO DE INFORMÁTICA	48
CAPÍTULO 5: CONCLUSIONES.....		50
5.1	CONCLUSIONES GENERALES	50
5.2	CUMPLIMIENTO DE OBJETIVOS	51
5.3	TRABAJO FUTURO	52
5.4	CONCLUSIÓN PERSONAL.....	53
ANEXOS		56
ANEXO A: IMPLEMENTACIÓN DE LA ARQUITECTURA CLOUD.....		56
CREAR USUARIO IAM		56
GENERAR CREDENCIALES <i>GIT</i>		57
CONSTRUCCIÓN DE LOS REPOSITORIOS		57
CARGAR CÓDIGO FUENTE		58
DOMINIO Y CERTIFICADOS		59
PREPARACIÓN PARA LEVANTAR FRONT-END.....		60
<i>ELASTIC</i> LOAD BALANCING		68
BACKEND		75
VARIABLES DE ENTORNO		76
ELASTIC CONTAINER REGISTRY.....		77

TASK DEFINITION	78
CLÚSTER	79
SUBDOMINIO	84
BASE DE DATOS	84
C/I y C/D	87
AWS CODE BUILD	88
AWS CODEPIPELINE	91
SEGURIDAD	93
ANEXO B: LEVANTAR PLATAFORMA USANDO DOCKER COMPOSE.....	97
ANEXO C: PRIMEROS PASOS PARA USAR LA PLATAFORMA	98

INDICE DE FIGURAS

Figura 1: Cantidad de matrículas nuevas en la UTFSM CC y Santiago. Fuente: Reportes USM (UTFSM, 2023)..... 14

Figura 2: Cantidad de matrículas nuevas de la carrera Ingeniería Civil en Informática de la UTFSM, durante 2022. Fuente: Reportes USM (UTFSM, 2023)..... 15

Figura 3: Porcentaje de retención al primer y segundo año de la carrera de Ingeniería Civil en Informática considerando Campus Casa Central y San Joaquín de la UTFSM. Fuente: Reportes USM (UTFSM, 2023)..... 16

Figura 4: Componentes de un sistema de monitoreo y cómo se vincula con el proyecto. Fuente: Diseño e implementación de un sistema de monitoreo, Terre des hommes (hommes, 2016). 19

Figura 5: Ejemplo de un pipeline. Fuente: *Amazon Web Services (AWS, ¿Qué es DevOps?, 2023).* 21

Figura 6: Ejemplo de una aplicación monolítica transformada usando microservicios. Fuente: *Amazon Web Services (AWS, Microservicios, 2023)*..... 24

Figura 7: Pirámide de Cohn, Fuente: Atlassian (Atlassian, 2023)..... 30

Figura 8: Modelo de dominio de la plataforma propuesta. Fuente: Elaboración propia..... 37

Figura 9: Diagrama de la arquitectura a montar. Fuente: Elaboración propia. 39

Figura 10: Pantalla de inicio de sesión de la plataforma: Fuente: Elaboración propia..... 41

Figura 11: Vista de inicio para un administrador: Fuente: Elaboración propia..... 41

Figura 12: Modal para crear una carrera. Fuente: Elaboración propia. 42

Figura 13: Vista de inicio del perfil de jefe de carrera. Fuente: Elaboración propia..... 42

Figura 14: Modal para crear un curso nuevo. Fuente: Elaboración propia. 43

Figura 15: Vista de gestión de aprendizajes críticos: Fuente: Elaboración propia..... 43

Figura 16: Vista dentro de un curso. Fuente: Elaboración propia..... 43

Figura 17: Vista de inicio del profesor. Fuente: Elaboración propia..... 44

Figura 18: Vista de edición de un portafolio. Fuente: Elaboración propia..... 44

Figura 19: Modal de edición de programa del curso. Fuente: Elaboración propia..... 45

Figura 20: Vista de carga de evaluaciones en un portafolio. Fuente: Elaboración propia. 45

Figura 21: Interfaz de servicio IAM. Fuente: AWS 56

Figura 22: Ventana de éxito al generar las credenciales GIT. Fuente: Elaboración propia..... 57

Figura 23: Creación del repositorio para el *frontend* de la plataforma. Fuente: Elaboración propia. 58

Figura 24: Formulario para agregar una zona alojada. Fuente: Elaboración propia..... 59

Figura 25: Configuración del certificado en *AWS Certificate Manager*. Fuente: Elaboración propia.
..... 60

Figura 26: Configuración del bucket inicial. Fuente: Elaboración propia. 61

Figura 27: Configuración de acceso del bucket. Fuente: Elaboración..... 61

Figura 28: Configuración final del bucket. Fuente: Elaboración propia..... 62

Figura 29: Configuración para alojar un sitio web estático en un *bucket*. Fuente: Elaboración propia.
..... 63

Figura 30: Configuración del apartado *Origin* de una distribución de Cloudfront. Fuente:
Elaboración propia. 64

Figura 31: Configuración del caché en una distribución. Fuente: Elaboración propia. 65

Figura 32: Configuración final de distribución de *Cloudfront*. Fuente: Elaboración propia. 66

Figura 33: Ejemplo de página de error 400 en una distribución. Fuente: Elaboración propia. 67

Figura 34: Lista de acceso permitido de una distribución de CloudFront. Fuente: Elaboración
propia. 67

Figura 35: Formulario para agregar un nuevo subdominio. Fuente: Elaboración propia..... 68

Figura 36: Configuración del grupo de seguridad. Fuente: Elaboración propia..... 69

Figura 37: Configuración de la parte uno del grupo de destino. Fuente: Elaboración propia. 70

Figura 38: Configuración de la parte dos del grupo de destino. Fuente: Elaboración propia. 71

Figura 39: : Configuración final del grupo de destino. Fuente: Elaboración propia..... 72

Figura 40: Tipos de balanceadores de carga de AWS. Fuente: AWS 73

Figura 41: Configuración básica del balanceador de carga. Fuente: AWS..... 73

Figura 42: Etapa de mapeo de red en un balanceador de carga. Fuente: AWS. 74

Figura 43: Configuración del grupo de seguridad. Fuente: AWS..... 74

Figura 44: Configuración final del balanceador de carga. Fuente: AWS. 75

Figura 45: Configuración para crear un nuevo repositorio privado. Fuente: Elaboración propia..... 77

Figura 46: Configuración inicial de una definición de tarea. Fuente: Elaboración propia. 78

Figura 47: Configuración del entorno de una definición de tarea. Fuente: Elaboración propia. 79

Figura 48: Creación de clúster. Fuente AWS..... 80

Figura 49: Configuración del entorno de un servicio. Fuente: AWS.	81
Figura 50: Configuración de red de un servicio. Fuente: AWS.	82
Figura 51: Configuración y vinculación del balanceador de carga al servicio. Fuente AWS.	83
Figura 52: Creación subdominio del <i>backend</i> . Fuente: Elaboración propia.....	84
Figura 53: Selección del tipo de clúster en MongoDB Atlas. Fuente: MongoDB Atlas.	85
Figura 54: Selección de <i>cluster tier</i> . Se debe elegir M0. Fuente: MongoDB Atlas.....	85
Figura 55: Creación de un usuario para acceder a la base de datos. Fuente: MongoDB Atlas.	86
Figura 56: Configuración de acceso de red a la base de datos. MongoDB Atlas.	86
Figura 57: Obtención de la URI para conectar la aplicación a la base de datos. Fuente: MongoDB Atlas.	87
Figura 58: Configuración básica del proyecto de compilación. Fuente: AWS.....	89
Figura 59: Configuración del apartado origen de un proyecto de compilación. Fuente: AWS.....	89
Figura 60: Especificación de compilación de un proyecto. Fuente: AWS.....	90
Figura 61: Configuración del entorno de un proyecto. Fuente: AWS.....	90
Figura 62: Etapa 1 de la creación de una pipeline. Fuente: AWS.....	91
Figura 63: Etapa dos de la creación de una pipeline. Fuente: AWS.....	92
Figura 64: Etapa 3 de la creación de una pipeline. Fuente: AWS.....	92
Figura 65: Etapa 4 de creación de una pipeline para <i>frontend</i> . Fuente: AWS.	93
Figura 66: Etapa 4 de creación de una pipeline para <i>backend</i> . Fuente: AWS.....	93
Figura 67: Creación de una pila en <i>Cloudformation</i> . Fuente: AWS.	94
Figura 68: Parámetros de la pila. Fuente: AWS.....	95
Figura 69: Vinculación de balanceador de carga en firewall. Fuente: AWS.	96
Figura 70: Ejemplo de archivo <i>docker-compose.yaml</i> . Fuente: Elaboración propia.	97
Figura 71: Comandos necesarios para agregar y vincular rol de administrador a un correo del departamento. Fuente: Elaboración propia.	98

INDICE DE TABLAS

Tabla 1: Requerimientos funcionales y no funcionales. Fuente: Departamento de Informática, elaboración propia. 32

Tabla 2: Historias de usuario a desarrollar. Fuente: Elaboración propia 34

Tabla 3: Historias de usuario nuevas a considerar. Fuente: Elaboración propia. 35

Tabla 4: Resumen *code coverage de frontend*. Fuente: Elaboración propia. 47

Tabla 5: Resumen *code coverage de backend*. Fuente: Elaboración propia..... 47

Tabla 6: Tabla de cumplimiento de historias de usuario. Fuente: Elaboración propia. 47

Tabla 7: Variables de entorno a generar. Fuente: Elaboración propia..... 76

INTRODUCCIÓN

Cada año ingresan muchos estudiantes a las universidades con el objetivo de poder terminar sus estudios y ser profesionales, pero no todos lo consiguen. Lamentablemente, en el proceso surgen ciertos factores que condicionan la permanencia de los alumnos en las carreras. Algunos de estos factores pueden ser problemas en el hogar, situación socioeconómica deficitaria, dificultades en el período de adaptación, depresión, crisis vocacional, carga académica desbalanceada, etc.

Según fuentes de admisión (UTFSM, 2023), la tasa de retención de estudiantes al segundo año de la carrera ronda el 65%. Puede parecer una cifra alta, pero este valor va cambiando a lo largo de los siguientes años, cifra que puede llegar a ser menor. Por lo tanto, en esta memoria se plantea que analizando la forma en que se imparten los contenidos y las herramientas que se utilizan para enseñar a los alumnos, además, de contar con un registro de las evaluaciones y trabajos realizados a lo largo de los semestres, es posible mejorar la retención.

La solución propuesta se describe en este documento a través de:

- Definición del problema: Se define la problemática que se plantea resolver.
- Marco Conceptual: Se abordan conceptos y herramientas necesarias para comprender lo desarrollado.
- Propuesta de solución: Se detalla el trabajo realizado, enfocándose en el diseño y presentación de la solución.
- Validación de la solución: Se da a conocer la evaluación de los usuarios principales de esta solución, el Departamento de Informática.
- Conclusiones: Se presentan las conclusiones del proyecto con los resultados obtenidos. Además, se dan a conocer propuestas de trabajos futuros a desarrollar en el tema abordado.

CAPÍTULO 1: DEFINICIÓN DEL PROBLEMA

1.1 CONTEXTO

Cada año, las universidades reciben una gran cantidad de nuevos alumnos, que tienen la esperanza de estudiar y titularse de la carrera escogida en el período real de esta y no alargarse más de lo necesario, pero la realidad es otra.

Según el portal de admisión de la UTFSM, en el campus Casa Central y Santiago, la cantidad de alumnos nuevos en el año 2022 fue de 2457 matrículas nuevas en modalidad diurna (ver Figura 1).

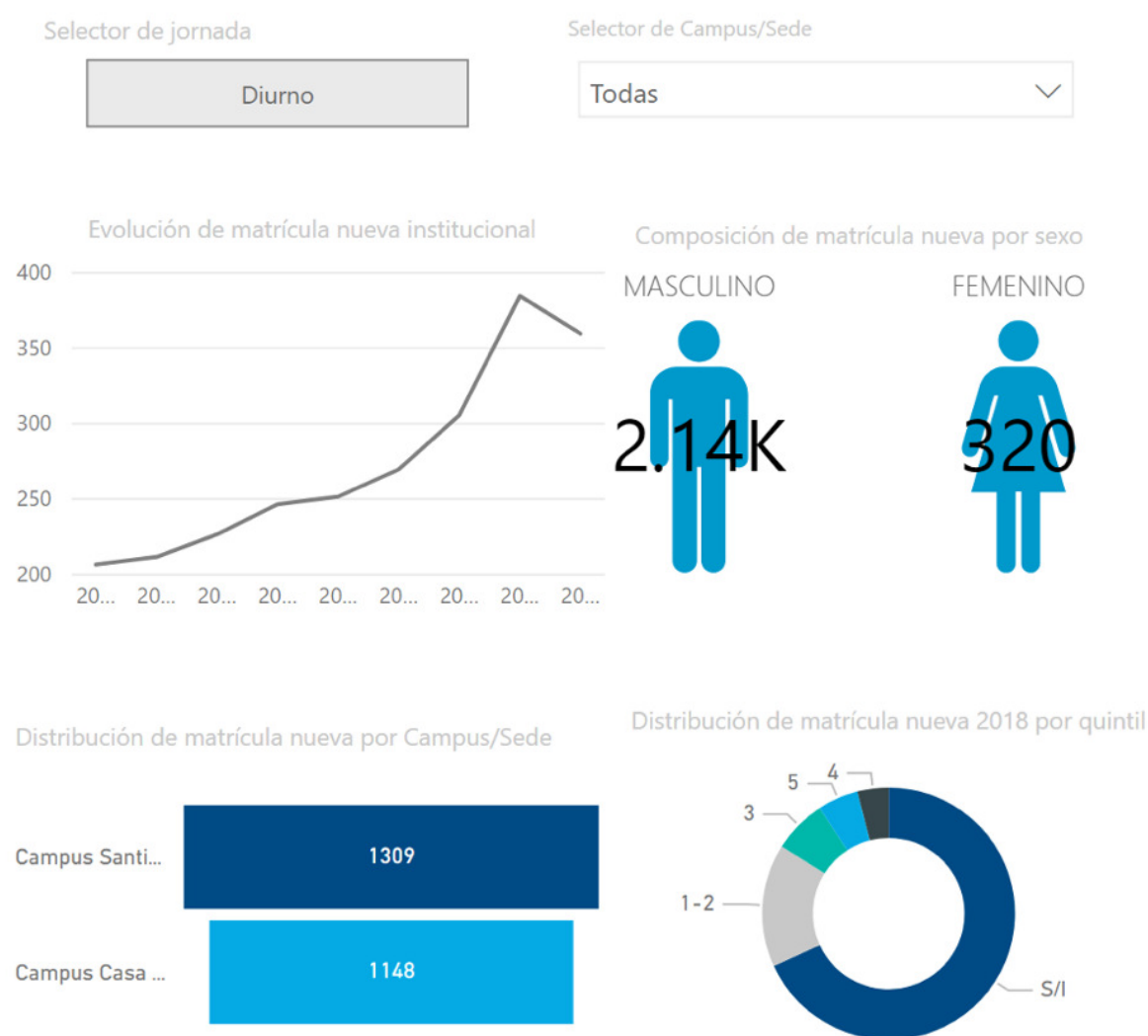


Figura 1: Cantidad de matrículas nuevas en la UTFSM CC y Santiago. Fuente: Reportes USM (UTFSM, 2023)

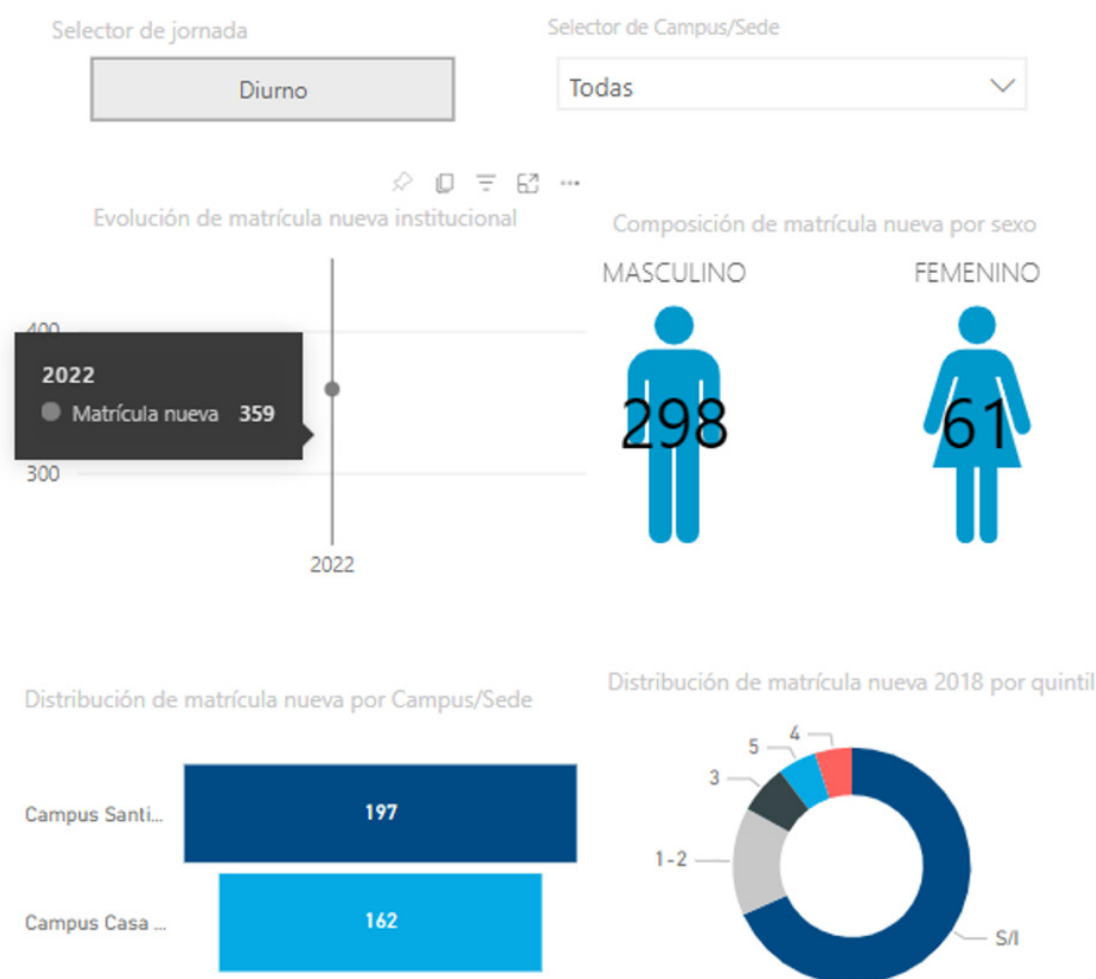


Figura 2: Cantidad de matrículas nuevas de la carrera Ingeniería Civil en Informática de la UTFSM, durante 2022. Fuente: Reportes USM (UTFSM, 2023)

En el caso de la carrera de Ingeniería Civil en Informática, la cantidad de nuevas matrículas en el año 2022 fue de 359 (ver Figura 2). Desde el 2014 ha ido en aumento la cantidad de nuevos estudiantes, pero ¿cuántos de ellos aún se mantienen en la carrera?

Según los datos (ver Figura 3), en el año 2017, la tasa de retención era cercana al 80% en el primer año de carrera y cercana al 65% en el segundo año de carrera y este valor suele ir variando a lo largo de los años, pero siempre manteniendo una brecha entre ellos. Por lo tanto, ¿existirá una forma de saber la razón de esta disminución? ¿Qué se puede hacer para mejorar este valor?

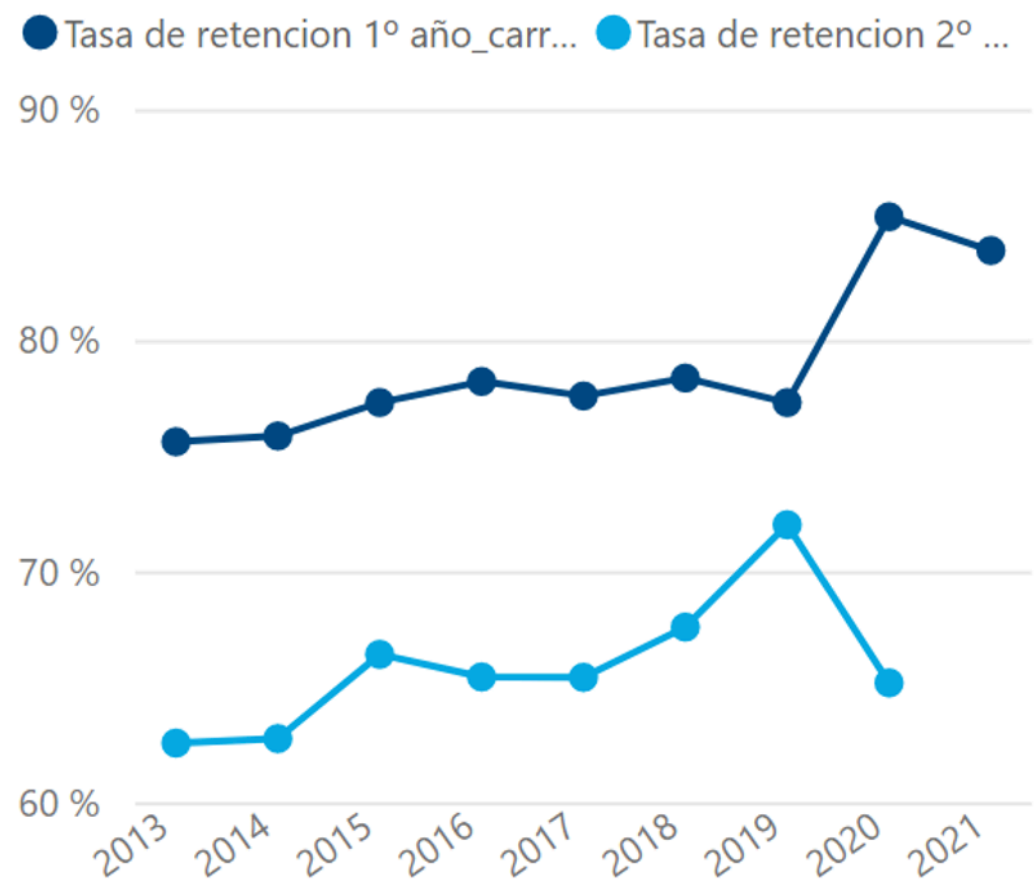


Figura 3: Porcentaje de retención al primer y segundo año de la carrera de Ingeniería Civil en Informática considerando Campus Casa Central y San Joaquín de la UTFSM.
Fuente: Reportes USM (UTFSM, 2023)

Las razones que llevan a la deserción pueden ser variadas, desde problemas familiares hasta la imposibilidad de poder seguir pagando la carrera. Pero sumado a lo anterior, también se considera la dificultad de los contenidos y el cómo se entregan estos, pues es sabido que algunas universidades son más exigentes que otras, pero no por eso son mejores.

La forma en que un profesor entrega los contenidos es de suma importancia. La diferencia entre un profesor que solo está al frente leyendo un par de diapositivas frente a uno que se esmera por hacer una clase divertida y práctica es mucha.

Dado el caso, es importante realizar un monitoreo sobre los contenidos que se presentan en la asignatura, las herramientas que se utilizan y la correspondiente planificación de cada semestre, pues así se podrá analizar la información obtenida y detectar las falencias que puedan existir, con el objetivo de mejorar la tasa de retención.

1.2 OBJETIVOS

1.2.1 OBJETIVO GENERAL

Ayudar a mejorar la tasa de retención en la carrera de Ingeniería Civil en Informática USM.

1.2.2 OBJETIVOS ESPECÍFICOS

- Diseñar, desarrollar y desplegar una plataforma de monitoreo de asignaturas basado en portafolios docentes.
- Utilizar las herramientas que ofrece AWS y MongoDB Atlas para poder construir sistemas informáticos de forma rápida, confiable y segura.
- Aplicar algunas de las prácticas DevOps para el desarrollo de la plataforma.

CAPÍTULO 2: MARCO CONCEPTUAL

2.1 SISTEMAS DE MONITOREO

2.1.1 LA IMPORTANCIA DEL MONITOREO

A medida que se avanza en el desarrollo de un proyecto, siempre surgen interrogantes como: ¿Vamos por la vía correcta?, ¿Por qué?, ¿Qué modificaciones se deben realizar para alcanzar los objetivos del proyecto?

En un proyecto pueden existir muchas personas o entes involucrados que tienen un determinado papel que deben cumplir para poder llevarlo a cabo de forma correcta, pero cada uno posee sus propios intereses y estrategias, por lo cual es importante conocer el cómo estas personas involucradas trabajan y qué pasos deben realizar para cumplir sus tareas.

El monitoreo es entonces un proceso esencial (Lauriac, 2016) para conducir un proyecto de manera a que este contribuya a un cambio efectivo para los beneficiarios. Sirve para apreciar el avance de un proyecto, para asegurarse de que este se sitúa sobre el buen camino para alcanzar los resultados esperados, o para observar y comprender las brechas, las dificultades o incluso las nuevas oportunidades. El monitoreo contribuye a decidir sobre los ajustes que son necesarios para alcanzar el objetivo del proyecto.

Para lograrlo, el monitoreo combina diferentes métodos que permiten definir la información que se requiere, recabar dicha información, analizarla y compartirla con los actores involucrados en el proyecto para luego tomar las decisiones correctas. Este proceso de trabajo sirve para conducir el proyecto, pero también nutre el aprendizaje de la organización, su rendición de cuentas e incluso su incidencia.

2.1.2 EL MONITOREO COMO UN SISTEMA

Dar el enfoque de sistema al monitoreo (Lauriac, 2016) permite destacar la combinación entre varios componentes, su interdependencia y sus interacciones con el proyecto. Esto queda reflejado en la Figura 4 y la posterior descripción de cada componente.

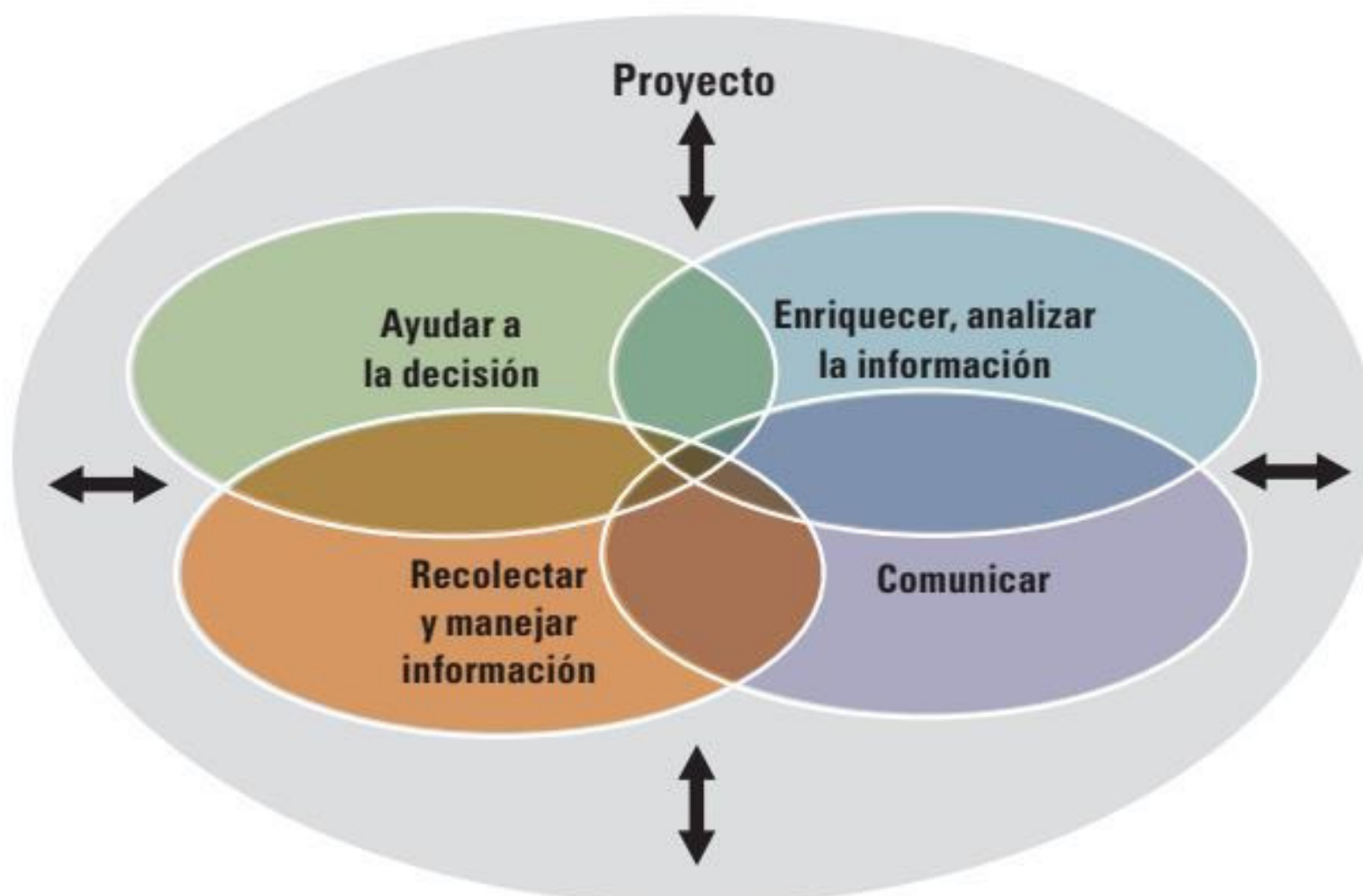


Figura 4: Componentes de un sistema de monitoreo y cómo se vincula con el proyecto.
Fuente: Diseño e implementación de un sistema de monitoreo, Terre des hommes (hommes, 2016).

- **Recolectar y manejar la información:** ya sea cualitativa, cuantitativa, formal o informal. La recolección de información a veces permite revisar el monitoreo bajo la perspectiva de los problemas que pueden encontrarse durante su obtención o interpretación, o del potencial no explotado que ella permitirá actualizar.
- **Enriquecer, analizar la información:** una vez reunida la información, se debe analizar con las personas involucradas en el proyecto. Esta reflexión crítica puede permitir enriquecer la información. Ella puede llevar directamente a una adaptación progresiva de la acción en sus puntos más débiles, pero a veces compartida y al refuerzo de las competencias de los actores.
- **Comunicar:** la comunicación de la información a aquellos que deben utilizarla puede también suscitar en sí una retroalimentación.
- **Ayudar a la decisión:** según sea el nivel al cual se apele, se manejarán tiempos diferentes y se requerirán responsabilidades y negociaciones diferentes.

2.2 PORTAFOLIOS DOCENTES Y LA EDUCACIÓN

Como tal, el nacimiento de los portafolios docentes surgieron gracias a Lee Shulman y sus colegas, donde en el 1985 junto al *Teacher Assessment Project (TAP)* de *Stanford* iniciaron un trabajo sobre la evaluación docente que los condujo al desarrollo y utilización de una carpeta o portafolio.

Durante la investigación, Shulman empezó a preocuparse sobre los conocimientos del profesor y los contenidos que enseñaba en sus clases, estas preocupaciones dieron lugar a una serie de interrogantes (Lyons, 1999):

- ¿De dónde provienen las explicaciones del docente?
- ¿Cómo decide el maestro qué enseñar, cómo representarlo, cómo interrogar a los estudiantes al respecto, y cómo tratar los problemas de la falta de comprensión?
- ¿Cuáles son las fuentes de sus conocimientos?
- ¿Cómo adquieren los nuevos conocimientos, cómo se recuerdan los viejos, y cómo se combinan unos y otros para formar una nueva base del saber?

Por lo tanto, un portafolio docente recoge un conjunto de características que permiten la resolución de las interrogantes anteriores. Según *The American Association for Higher Education (AAHE)*, las características más importantes de un portafolio docente son:

- Selectiva: recopilará los mejores ejemplos que pueden demostrar cómo se ha organizado el curso, la forma en la que se ha impartido la enseñanza, los resultados de la enseñanza y la evaluación, etc.
- Reflexiva: cada ejemplo debe contener una reflexión del profesor donde se explicará qué, cómo y por qué se hizo y los logros obtenidos. Deben considerarse cuáles han sido los cambios en los estudiantes al terminar el curso y qué ideas han surgido de la reflexión sobre esos ejemplo de enseñanza.
- Estructurada: los ejemplo responderán a unos cuantos criterios determinados o categorías que representan dimensiones de la calidad docente que se quiere demostrar.
- Actualizada: la información que se incluye en el portafolio docente corresponde a un periodo de tiempo determinado, por lo tanto, el mismo debe ser revisado constantemente.

El uso de los portafolios dentro de la educación pueden ser variados, pero los más destacados son lo que tienen una finalidad sumativa, evaluación, mejora de la enseñanza, valoración y promoción profesional.

2.3 DESARROLLO DE SOFTWARE MODERNO

2.3.1 DEVOPS

Las operaciones de desarrollo constituyen una combinación de filosofías culturales, prácticas y herramientas que incrementan la capacidad de una organización de proporcionar aplicaciones y servicios a gran velocidad: desarrollar y mejorar productos con mayor rapidez que las organizaciones que utilizan procesos tradicionales de desarrollo de software y administración de la infraestructura (AWS, ¿Qué es *DevOps*?, 2023). Esta velocidad permite a las organizaciones servir mejor a sus clientes y competir de forma más eficaz en el mercado. En la Figura 5 se ejemplifica su uso.

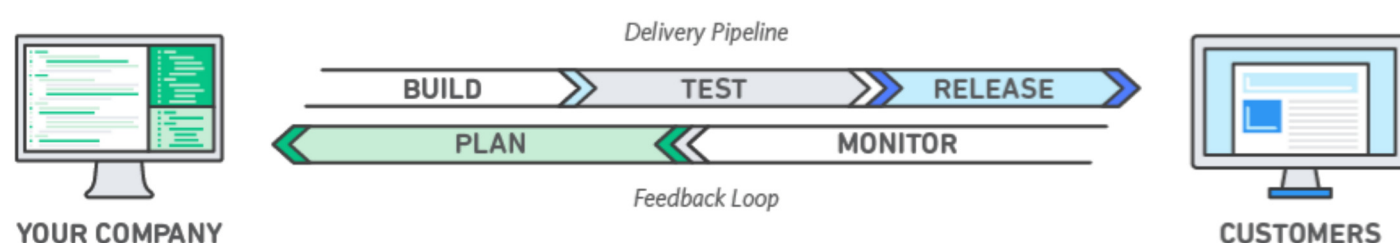


Figura 5: Ejemplo de un pipeline. Fuente: *Amazon Web Services (AWS, ¿Qué es DevOps?, 2023)*.

2.3.1.1 ¿EN QUÉ CONSISTE?

Bajo un modelo de DevOps, los equipos de desarrollo y operaciones ya no están “aislados”. A veces, los dos equipos se fusionan en uno solo, donde los ingenieros trabajan en todo el ciclo de vida de la aplicación, desde el desarrollo y las pruebas hasta la implementación y las operaciones, y desarrollan una variedad de habilidades no limitadas a una única función. En algunos modelos de *DevOps*, los equipos de control de calidad y de seguridad también se integran más con el desarrollo y las operaciones e intervienen durante todo el ciclo de vida de la aplicación. Cuando la seguridad es la prioridad de todos los miembros de *DevOps*, a veces se conoce como operaciones de seguridad de desarrollo (AWS, ¿Qué es *DevOps*?, 2023).

Los equipos utilizan prácticas para automatizar los procesos que anteriormente habían sido manuales y lentos. Utilizan una pila de tecnología y herramientas que los ayudan a operar y mejorar aplicaciones de forma rápida y confiable. Además, estas herramientas ayudan a los ingenieros a realizar de forma independiente tareas que normalmente hubieran requerido la ayuda de otros equipos (por ejemplo, implementar código o aprovisionar infraestructura), lo que incrementa todavía más la velocidad del equipo.

2.3.1.2 BENEFICIOS

- **Velocidad:** permite avanzar a gran velocidad e innovar con mayor rapidez, adaptarse mejor a los cambios en los mercados y adquirir mayor eficacia en la consecución de los resultados empresariales.
- **Entrega rápida:** permite incrementar la frecuencia y el ritmo de las versiones, a fin de innovar y mejorar el producto con mayor rapidez. Cuanto más rápido se publiquen nuevas características y se solucionen errores, más rápido se podrá responder a las necesidades de los clientes y desarrollar una ventaja competitiva. La integración y las entregas continuas son prácticas que automatizan el proceso de publicación de software, desde la creación hasta la implementación.
- **Confiabilidad:** permite garantizar la calidad de las actualizaciones de las aplicaciones y los cambios en la infraestructura a fin de poder realizar entregas más rápidas de forma confiable mientras ofrece una experiencia positiva a los usuarios finales.
- **Escalado:** permite operar y administrar la infraestructura y procesos de desarrollo a escala. La automatización y coherencia ayudan a administrar sistemas complejos o cambiantes de forma eficaz con menos riesgo.
- **Seguridad:** permite avanzar con rapidez mientras retiene el control y mantiene la conformidad.

2.3.1.3 IMPORTANCIA

El software y la Internet han transformado el mundo y sus industrias, desde las compras hasta el entretenimiento y la banca. El software ya no se limita a respaldar un negocio, sino que se convierte en componente integral de cada aspecto de este. Las compañías interactúan con sus clientes a través de software proporcionado como aplicaciones o servicios en línea en todo tipo de dispositivos. También utilizan software para incrementar la eficacia operativa al transformar cada sección de la cadena de valor, como la logística, la comunicación y las operaciones. Al igual que las compañías de productos físicos transformaron el modo en que diseñaban, fabricaban y entregaban productos gracias a la automatización industrial que se produjo en el siglo XX, las compañías de hoy en día deben transformar el modo en que crean y entregan software (AWS, ¿Qué es *DevOps*?, 2023).

2.3.1.4 BUENAS PRÁCTICAS

Existen un conjunto de buenas prácticas de DevOps (AWS, ¿Qué es *DevOps*?, 2023).

- **Integración Continua:** la integración continua es una práctica de desarrollo de software mediante la cual los desarrolladores combinan los cambios en el código en un repositorio central de forma periódica, tras lo cual se ejecutan versiones y pruebas automáticas. Los objetivos clave de la integración continua consisten en encontrar y arreglar errores con mayor rapidez, mejorar la calidad del software y reducir el tiempo que se tarda en validar y publicar nuevas actualizaciones de software.

- **Entrega Continua:** La entrega continua es una práctica de desarrollo de software mediante la cual se compilan, prueban y preparan automáticamente los cambios en el código y se entregan a la fase de producción. Amplía la integración continua al implementar todos los cambios en el código en un entorno de pruebas o de producción después de la fase de creación. Cuando la entrega continua se implementa de manera adecuada, los desarrolladores dispondrán siempre de un artefacto listo para su implementación que se ha sometido a un proceso de pruebas estandarizado.
- **Microservicios:** la arquitectura de microservicios es un enfoque de diseño que sirve para crear una sola aplicación como conjunto de servicios pequeños. Cada servicio se ejecuta en su propio proceso y se comunica con otros servicios mediante una interfaz bien definida utilizando un mecanismo ligero, normalmente una interfaz de programación de aplicaciones basada en *HTTP (API)*. Los microservicios se crean en torno a las capacidades empresariales. Cada servicio abarca un único propósito. Puede utilizar distintos marcos o lenguajes de programación para escribir microservicios e implementarlos independientemente, como servicio único, o como grupo de servicios.
- **Infraestructura como código:** la infraestructura como código es una práctica mediante la que se aprovisiona y administra infraestructura con técnicas de desarrollo de código y de software, como el control de versiones y la integración continua. El modelo orientado a la *API* de la nube permite a los desarrolladores y administradores de sistemas interactuar con la infraestructura mediante programación y a escala, en lugar de configurar y ajustar recursos manualmente. Así, los ingenieros pueden interactuar con la infraestructura con herramientas basadas en código y tratar la infraestructura de un modo parecido a como tratan el código de la aplicación. Como están definidos por el código, la infraestructura y los servidores se pueden implementar con rapidez con patrones estandarizados, actualizar con las últimas revisiones y versiones o duplicar de forma repetible.
- **Monitoreo y registro:** las organizaciones monitorean métricas y registros para ver cómo el desempeño de las aplicaciones y la infraestructura afecta a la experiencia que el usuario final tiene de su producto. Cuando recopilan, categorizan y analizan los datos y registros generados por las aplicaciones y la infraestructura, las organizaciones pueden entender cómo los cambios y actualizaciones afectan a los usuarios, esto les aporta información sobre la causa raíz de los problemas o cambios inesperados. El monitoreo activo se vuelve cada vez más importante, ya que los servicios deben estar disponibles las 24 horas del día, los 7 días de la semana, a medida que la frecuencia de actualizaciones de las aplicaciones e infraestructura incrementa. La creación de alertas y el análisis en tiempo real de los datos también ayuda a las organizaciones a monitorear sus servicios de forma proactiva.
- **Comunicación y colaboración:** el incremento en la comunicación y la colaboración en una organización es uno de los aspectos culturales clave de *DevOps*. El uso de las herramientas de *DevOps* y la automatización del proceso de entrega de software establece la colaboración al reunir físicamente los flujos de trabajo y las responsabilidades de los equipos de desarrollo y operaciones. Además, estos equipos establecen normas culturales sólidas que giran en torno a compartir información y facilitar la comunicación mediante el uso de aplicaciones de chat, sistemas de seguimiento de proyectos o problemas y *wikis*. De este modo, se acelera la comunicación entre los equipos de desarrollo y operaciones e

incluso con otros equipos, como marketing y ventas, lo que permite que todos los departamentos de la organización se alineen mejor con los objetivos y proyectos.

2.3.2 MICROSERVICIOS

Corresponde a un enfoque para el desarrollo de una aplicación única como un conjunto de pequeños servicios, cada uno ejecutándose en su propio proceso y mecanismos ligeros de comunicación, a menudo un recurso de una interfaz de programación de aplicaciones (*API*) sobre protocolo de transferencia de hipertexto (*HTTP*). Estos servicios están contruidos alrededor de las capacidades del negocio y con independencia de despliegue e implementación totalmente automatizada. Existe un mínimo de gestión centralizada de estos servicios, los que pueden estar escritos en lenguajes de programación diferentes y utilizar diferentes tecnologías de almacenamiento de datos. En la Figura 6 se ejemplifica su uso.

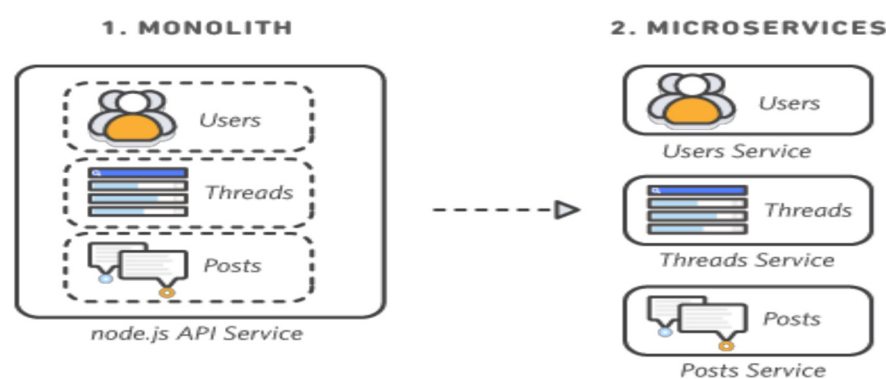


Figura 6: Ejemplo de una aplicación monolítica transformada usando microservicios.

Fuente: *Amazon Web Services (AWS, Microservicios, 2023)*.

2.3.2.1 CARACTERÍSTICAS

- **Autónomos:** cada servicio en una arquitectura de microservicios se puede desarrollar, implementar, operar y escalar sin afectar el funcionamiento de otros servicios. Los servicios no necesitan compartir ninguno de sus códigos o implementaciones con otros servicios. Cualquier comunicación entre componentes individuales ocurre a través de *APIs* bien definidas.
- **Especializados:** cada servicio está diseñado para un conjunto de capacidades y se enfoca en resolver un problema específico. Si los desarrolladores aportan más código a un servicio a lo largo del tiempo y el servicio se vuelve complejo, se puede dividir en servicios más pequeños.
- **Agilidad:** los microservicios fomentan una organización de equipos pequeños e independientes que se apropian de los servicios. Los equipos actúan en un contexto pequeño y bien comprendido, y están facultados para trabajar de forma más independiente y rápida. Esto acorta los tiempos del ciclo de desarrollo, lográndose beneficios significativos al aumentar el rendimiento de la organización.

- **Escalamiento flexible:** los microservicios permiten que cada servicio se escale de forma independiente para satisfacer la demanda de la característica de la aplicación que respalda. Esto permite a los equipos adecuarse a las necesidades de la infraestructura, medir con precisión el costo de una característica y mantener la disponibilidad si un servicio experimenta un aumento en la demanda.
- **Implementación sencilla:** los microservicios permiten la integración y las entregas continuas, lo que facilita probar nuevas ideas y revertirlas si algo no funciona. El bajo costo de los errores permite experimentar, facilita la actualización del código y acelera el tiempo de comercialización de las nuevas características.
- **Libertad tecnológica:** las arquitecturas de microservicios no siguen un enfoque de "diseño único". Los equipos tienen la libertad de elegir la mejor herramienta para resolver sus problemas específicos. Como consecuencia, los equipos que crean microservicios pueden elegir la mejor herramienta para cada trabajo.
- **Código reutilizable:** la división del software en módulos pequeños y bien definidos les permite a los equipos usar funciones para diferentes propósitos. Un servicio escrito para una determinada función se puede usar como un componente básico para otra característica. Esto permite que una aplicación arranque por sí sola, ya que los desarrolladores pueden crear nuevas capacidades sin tener que escribir código desde cero.
- **Resistencia:** La independencia del servicio aumenta la resistencia de una aplicación a los errores. En una arquitectura monolítica, un error en un solo componente puede provocar un error en toda la aplicación. Con los microservicios, si hay un error en todo el servicio, las aplicaciones lo manejan degradando la funcionalidad sin bloquear toda la aplicación.

2.3.3 *AMAZON WEB SERVICES*

Es la plataforma en la nube más adoptada y completa en el mundo, que ofrece más de 175 servicios integrales de centros de datos a nivel global. Millones de clientes, incluyendo las empresas emergentes que crecen más rápido, las compañías más grandes y los organismos gubernamentales líderes, están utilizando AWS para reducir los costos, aumentar su agilidad e innovar de forma más rápida (AWS, Informática en la nube con AWS, 2023).

2.3.3.1 **HERRAMIENTAS PARA DEVOPS**

AWS proporciona un conjunto de servicios flexibles diseñados para permitir a las compañías crear y proporcionar productos de forma más rápida y fiable que ayudan a los equipos a implementar e innovar con rapidez y fiabilidad. Estos servicios simplifican el aprovisionamiento y la administración de la infraestructura, la implementación del código de la aplicación, la automatización de procesos de publicación de software y la monitorización del desempeño de su aplicación e infraestructura.

Las operaciones de desarrollo constituyen una combinación de filosofías culturales, prácticas y herramientas que incrementan la capacidad de una organización de proporcionar aplicaciones y servicios a gran velocidad: desarrollar y mejorar productos con mayor rapidez que las organizaciones que utilizan procesos tradicionales de desarrollo de software y administración de la infraestructura. Esta velocidad permite a las organizaciones servir mejor a sus clientes y competir de forma más eficaz en el mercado.

A continuación, se describen algunas de sus herramientas.

- **AWS CodeCommit:** es un servicio de control de código fuente completamente administrado que facilita a las compañías el alojamiento de repositorios Git privados, seguros y con un alto nivel de escalabilidad. *CodeCommit*, que funciona perfectamente con las herramientas de *GIT* existentes, se puede utilizar para almacenar de forma segura cualquier elemento, ya sea código fuente o binario (AWS, *CodeCommit*, 2022).
- **AWS CodeBuild:** es un servicio de creación completamente administrado que compila código fuente, ejecuta pruebas y produce paquetes de software listos para su implementación. Con *CodeBuild*, no es necesario aprovisionar, administrar y escalar sus propios servidores de creación. *CodeBuild* se escala constantemente y procesa numerosas versiones a la vez, de manera que sus versiones no permanecen a la espera en una cola (AWS, *CodeBuild*, 2020).
- **AWS CodeDeploy:** automatiza las implementaciones de código en cualquier instancia, incluidas instancias de *Amazon EC2* y servidores *onpremise*. *AWS CodeDeploy* facilita el lanzamiento rápido de nuevas características, ayuda a evitar tiempos de inactividad durante la implementación de una aplicación y administra la compleja actualización de las aplicaciones (AWS, *CodeDeploy*, 2020).
- **AWS CodePipeline:** es un servicio de integración y entregas continuas para realizar actualizaciones de aplicaciones e infraestructura rápida y de confianza. *CodePipeline* compila, prueba e implementa el código cada vez que se produce un cambio en este, de acuerdo con los modelos de procesamiento de la publicación que se definan. Esto permite entregar características y actualizaciones de forma rápida y de confianza (AWS, 2022).
- **Amazon CloudWatch:** es un servicio de monitorización de los recursos de la nube de AWS y de las aplicaciones que se ejecutan en AWS. Se puede utilizar *Amazon CloudWatch* para recopilar y realizar el seguimiento de métricas y *logs*, establecer alarmas y reaccionar automáticamente a los cambios de los recursos AWS (AWS, 2022).
- **Amazon Route 53:** es un servicio de *DNS* (sistema de nombres de dominio) web escalable y de alta disponibilidad en la nube. Está diseñado para ofrecer a los desarrolladores y las empresas un método fiable y rentable para redirigir a los usuarios finales a las aplicaciones en Internet mediante la traducción de nombres legibles para las personas como *www.ejemplo.com* en direcciones IP numéricas como 192.0.2.1 que utilizan los equipos para conectarse entre ellos. *Amazon Route 53* también cumple con *IPv6* (AWS, 2021).

- **Amazon Simple Storage Service:** es un servicio de almacenamiento de objetos que ofrece escalabilidad, disponibilidad de datos, seguridad y rendimiento líderes en el sector. Esto significa que clientes de todos los tamaños y sectores pueden utilizarlo para almacenar y proteger cualquier cantidad de datos para diversos casos de uso, como sitios web, aplicaciones móviles, procesos de copia de seguridad y restauración, operaciones de archivado, aplicaciones empresariales, dispositivos *IoT* y análisis de *big data* (AWS, 2022).
- **API Gateway:** es un servicio completamente administrado que facilita a los desarrolladores la creación, la publicación, el mantenimiento, el monitoreo y la protección de *API* a cualquier escala. Está como "puerta delantera" para que las aplicaciones obtengan acceso a datos, lógica de negocio o funcionalidades desde sus servicios *backend*. *API Gateway* gestiona todas las tareas implicadas en la aceptación y el procesamiento de hasta cientos de miles de llamadas a *API* simultáneas, entre ellas, la administración del tráfico, el control de autorizaciones y acceso, el monitoreo y la administración de versiones de *API* (AWS, 2023).

2.3.4 BASE DE DATOS NOSQL

Las bases de datos *NoSQL* están diseñadas para modelos de datos específicos y tienen esquemas flexibles para crear aplicaciones modernas. Las bases de datos *NoSQL* son ampliamente reconocidas porque son fáciles de desarrollar, por su funcionalidad y el rendimiento a escala (AWS, ¿Qué es *NoSQL*?, 2022).

Las bases de datos *NoSQL* se adaptan perfectamente a muchas aplicaciones modernas, como dispositivos móviles, web y juegos, que requieren bases de datos flexibles, escalables, de alto rendimiento y altamente funcionales para proporcionar excelentes experiencias de usuario. Se caracterizan por su:

- **Flexibilidad:** las bases de datos *NoSQL* generalmente ofrecen esquemas flexibles que permiten un desarrollo más rápido e iterativo. El modelo de datos flexible hace que las bases de datos *NoSQL* sean ideales para datos semiestructurados y no estructurados.
- **Escalabilidad:** las bases de datos *NoSQL* generalmente están diseñadas para escalar usando clústeres distribuidos de hardware en lugar de escalar añadiendo servidores caros y sólidos. Algunos proveedores de la nube manejan estas operaciones en segundo plano, como un servicio completamente administrado.
- **Alto rendimiento:** la base de datos *NoSQL* está optimizada para modelos de datos específicos y patrones de acceso que permiten un mayor rendimiento que el intento de lograr una funcionalidad similar con bases de datos relacionales.
- **Alta funcionalidad:** las bases de datos *NoSQL* proporcionan *API* altamente funcionales y tipos de datos que están diseñados específicamente para cada uno de sus respectivos modelos de datos.

2.3.4.1 MONGODB

MongoDB es una base de datos de documentos que ofrece una gran escalabilidad y flexibilidad, y un modelo de consultas e indexación avanzado. Permite almacenar datos en documentos flexibles similares a *JSON*, por lo que los campos pueden variar entre documentos y la estructura de datos puede cambiarse con el tiempo (MongoDB, Inc., 2023).

2.3.4.2 MONGODB ATLAS

MongoDB Atlas es el servicio global de base de datos en la nube para aplicaciones modernas. MongoDB está implementado totalmente en *AWS*, *Azure* o *Google Cloud Platform*. La mejor automatización y prácticas comprobadas de su clase garantizan la disponibilidad, la escalabilidad y el cumplimiento de los estándares de seguridad y privacidad de datos más exigentes. Usar el robusto ecosistema de controladores, integraciones y herramientas de MongoDB permite construir más rápido y pasar menos tiempo administrando la base de datos.

2.3.5 ANGULAR

Es un *framework* para aplicaciones web desarrollado en *TypeScript*, mantenido por Google, que se utiliza para crear y mantener aplicaciones web de una sola página. Su objetivo es aumentar las aplicaciones basadas en navegador con capacidad de Modelo Vista Controlador (*MVC*), en un esfuerzo para hacer que el desarrollo y las pruebas sean más fáciles (Google, 2023).

La biblioteca lee el *HTML* que contiene atributos de las etiquetas personalizadas adicionales, entonces obedece a las directivas de los atributos personalizados, y une las piezas de entrada o salida de la página a un modelo representado por las variables estándar de *JavaScript*.

Angular se basa en clases tipo "Componentes", cuyas propiedades son las usadas para hacer el *binding* de los datos. En dichas clases tenemos propiedades (variables) y métodos (funciones a llamar).

Angular es la evolución de *AngularJS* aunque incompatible con su predecesor.

2.3.6 **NODEJS**

Es un entorno en tiempo de ejecución multiplataforma, para la capa del servidor (pero no limitándose a ello) basado en el motor V8 de Google. Fue creado con el enfoque de ser útil en la creación de programas de red altamente escalables, como, por ejemplo, servidores web.

2.3.7 **NESTJS**

Es un *framework* progresivo que utiliza *nodejs* para construir aplicaciones de servidor eficientes, confiables y escalables. Su filosofía es muy similar a lo que es *Angular* actualmente, facilitando su uso para desarrolladores que tienen experiencia con este.

2.3.8 **TESTING**

Cuando se construyen aplicaciones, un proceso importante corresponde a las pruebas de software o *testing*, las pruebas de software permiten evaluar y verificar que un producto o aplicación de software hace lo que se supone debe hacer. Entre los beneficios se encuentran:

- Prevención de errores
- Reducción de costos de desarrollo
- Mejora del rendimiento

Existen varios tipos de pruebas de software (IBM, 2023), entre ellas se pueden encontrar:

- Pruebas de aceptación: verifica si todo el sistema funciona según lo previsto.
- Pruebas de integración: verifica que los componentes o funciones del software operen juntos.
- Pruebas de unidad: valida que cada unidad de software funciona según lo esperado. Una unidad corresponde al componente de prueba más pequeño de una aplicación.
- Pruebas funcionales: verifica funciones mediante la emulación de escenarios de negocio, en función de los requisitos funcionales.
- Pruebas de rendimiento: se evalúa cómo funciona el software bajo diferentes cargas de trabajo. Las pruebas de carga, por ejemplo, se utilizan para evaluar el rendimiento en condiciones de carga reales.
- Pruebas de regresión: verifica si las nuevas características rompen o degradan la funcionalidad.
- Pruebas de estrés: prueba cuánta tensión puede soportar el sistema antes de que falle.

- Pruebas de usabilidad: valida qué tan bien un cliente puede usar un sistema o una aplicación para completar una tarea.

Comúnmente, existe una pirámide, llamada la Pirámide de Cohn (ver Figura 7). Esta pirámide suele tener la siguiente estructura:

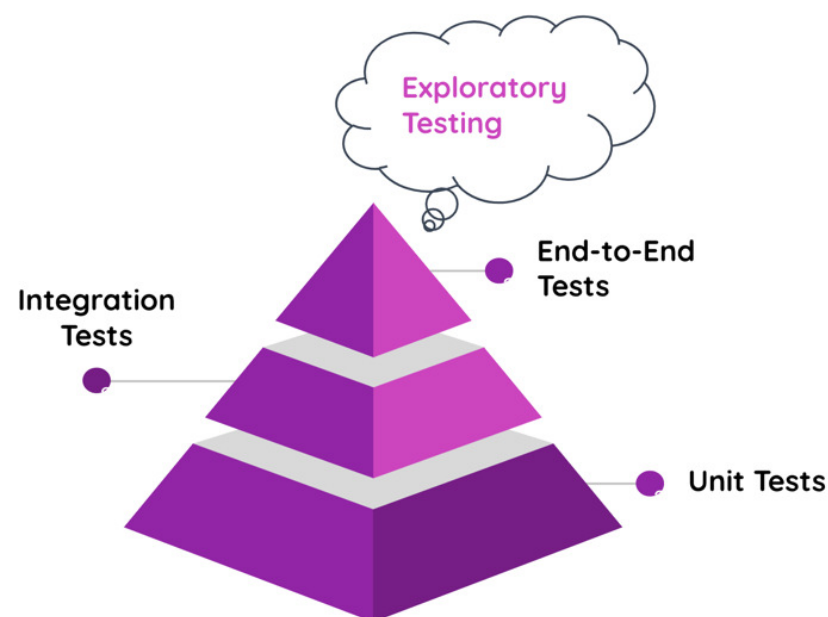


Figura 7: Pirámide de Cohn, Fuente: Atlassian (Atlassian, 2023)

Como base se tienen las pruebas unitarias, luego vienen las pruebas de integración y finalmente, las pruebas *e2e* y pruebas exploratorias. Mientras más abajo en la pirámide, más rápidos y fáciles son las pruebas y a medida que se sube, la complejidad y tiempo de dedicación aumenta.

Si nos enfocamos principalmente en las pruebas unitarias, éstas deben ser simples y rápidas y tener una cobertura de código razonable, un número bastante aceptado corresponde al 80% de cobertura de código, pero lo importante es no buscar el 100% de cobertura, sino darle prioridad a la calidad de estas pruebas.

CAPÍTULO 3: PROPUESTA DE SOLUCIÓN

A partir de la definición del problema y las diversas tecnologías planteadas anteriormente, como solución, se propone la construcción de una plataforma altamente escalable utilizando tecnologías de vanguardia, cuya función será almacenar portafolios docentes de las asignaturas de la carrera para luego facilitar a evaluadores internos y externos la inspección de estos datos. Diversos usuarios podrán interactuar con la plataforma, como el profesor o la secretaria de pregrado, quienes serán los encargados de poblar de datos. Importante destacar que se subirá información mixta, ya sean archivos o a través de completar formularios.

El objetivo de este trabajo es diseñar, desarrollar e implementar esta plataforma, de acuerdo con los requisitos estipulados por parte de la Subdirección de Pregrado del Departamento de Informática de la UTFSM.

A continuación, se pasará a explicar los pasos que se llevaron a cabo para diseñar y desarrollar el software.

3.1 TOMA DE REQUERIMIENTOS

Antes de comenzar a desarrollar, es importante determinar qué es lo que el cliente espera de la plataforma, qué restricciones desea imponer y cuáles serán los usuarios que la utilicen.

En una primera instancia, se tuvo una reunión para determinar los requerimientos necesarios a implementar, separándose entre funcionales y no funcionales.

Cuando se habla de un requerimiento funcional, se refiere a una descripción que define el comportamiento básico de un sistema, es decir, corresponden a lo que el sistema **debe** o **no debe** hacer ante *inputs* del usuario. Por otra parte, los requerimientos no funcionales corresponden a restricciones que se imponen a los requerimientos funcionales, es decir, especifican el **cómo** deben ser ejecutados o desarrollados.

A continuación, se detalla una tabla para presentar los requerimientos funcionales y no funcionales obtenidos de la reunión con los actores claves (ver Tabla 1).

Tabla 1: Requerimientos funcionales y no funcionales. Fuente: Departamento de Informática, elaboración propia.

Identificador	Requerimiento
PMD-R1	El sistema debe permitir generar y almacenar portafolios docentes.
PMD-R2	El sistema debe ser una plataforma web.
PMD-R3	Los portafolios deben contener a lo menos el programa oficial de la asignatura, programa del semestre, instrumentos de evaluación, material utilizado, herramientas de software utilizadas, una ficha de cierre de la asignatura, resultados de la encuesta docente, planillas de las notas del semestre.
PMD-R4	El sistema debe utilizar ingreso vía sistema de cuentas del Departamento de Informática.
PMD-R5	Evaluadores y jefe de carrera podrán revisar portafolios docentes generados.
PMD-R6	Los portafolios podrán ser llenados por los mismos profesores o por las secretarías.
PMD-R7	Los portafolios deben vincularse a aprendizajes críticos.
PMD-R8	Los instrumentos de evaluación deben contener el enunciado, copia de la mejor nota, copia de la peor nota y la nota promedio.
PMD-R9	El material utilizado pueden ser archivos de <i>Word</i> , <i>Excel</i> , <i>Power Point</i> , <i>PDF</i> o enlaces.
PMD-R10	La encuesta docente debe incluir comentarios por parte del profesor.
PMD-R11	La ficha de cierre de la asignatura debe incluir una reflexión por parte del profesor y un plan de mejora.

3.2 USUARIOS

A partir de los requerimientos, es posible inferir los distintos usuarios que interactuarán con la plataforma, los cuales son:

- Profesores: ellos serán los principales encargados de poblar los portafolios docentes.
- Secretaria de pregrado: podrá ayudar a los profesores a poblar los portafolios.
- Evaluadores: tendrán la facultad de revisar los portafolios docentes generados.
- Jefe de carrera: tendrán la facultad de revisar los portafolios docentes generados.

3.3 HISTORIAS DE USUARIO

A menudo se suelen confundir los requerimientos con las historias de usuarios, pero lo cierto es que no son lo mismo.

Dentro del desarrollo de software ágil es importante poner a las personas en primer lugar y las historias de usuario cumplen ese objetivo, al situar a los usuarios finales en el centro de la conversación.

Las historias utilizan un lenguaje no técnico para ofrecer contexto al equipo de desarrollo, esto permite al equipo saber el **por qué** se está desarrollando, **lo que se está** desarrollando y **qué** valor genera. En palabras simples, una historia corresponde a una explicación general e informal de una funcionalidad escrita desde la perspectiva de un usuario o cliente (Atlassian, Historias de usuario con ejemplos y plantilla, 2023), estas suelen estar estructuradas de la siguiente forma:

“Como [usuario], [quiero][para]”

Algunas de las ventajas que se obtienen con su uso son:

- **Las historias centran la atención en el usuario:** una lista de tareas pendientes mantiene al equipo centrado en tareas que deben completarse, pero un conjunto de historias lo mantiene centrado en solucionar problemas para usuarios reales.
- **Las historias permiten la colaboración:** con el objetivo definido, el equipo puede colaborar para decidir cómo ofrecer un mejor servicio al usuario y cumplir con dicho objetivo.
- **Las historias impulsan soluciones creativas:** las historias fomentan que el equipo piense de forma crítica y creativa sobre cómo lograr mejor un objetivo.
- **Las historias motivan:** con cada historia el equipo de desarrollo disfruta de un pequeño reto y una pequeña victoria, lo que aumenta la motivación.

Por lo anterior, se traspasaron los requerimientos a historias de usuarios, obteniéndose los siguientes resultados entregados en la Tabla 2.

Tabla 2: Historias de usuario a desarrollar. Fuente: Elaboración propia

Identificador	Descripción
PMD-1	Como profesor quiero poder generar portafolios docentes para almacenar mi trabajo semestral.
PMD-2	Como usuario quiero usar mi cuenta institucional para ingresar a la plataforma para que sea fácil y rápido el ingreso.
PMD-3	Como jefe de carrera deseo poder revisar los portafolios que se han generado para llevar un control de estos.
PMD-4	Como secretaria/o de pregrado quiero ayudar al profesor a construir los portafolios para no sobrecargar las labores del docente.
PMD-5	Como evaluador, externo al DI, quiero tener acceso a los portafolios para poder realizar evaluaciones
PMD-6	Como jefe de carrera quiero que los portafolios puedan vincularse a los aprendizajes críticos de la carrera para se puedan relacionar de manera más rápida al verlos.
PMD-7	Como jefe de carrera quiero que en las evaluaciones de los portafolios puedan incluirse las copias de la mejor y peor nota más el promedio para su evaluación posterior.
PMD-8	Como jefe de carrera quiero que los portafolios incluyan la encuesta docente del profesor de turno para su posterior evaluación.
PMD-9	Como jefe de carrera quiero que los portafolios incluyan los materiales utilizados en la asignatura para su posterior evaluación.

3.4 REFINAMIENTO

Luego de varias iteraciones, surgieron algunos problemas o inconvenientes que no se habían considerado al definir los requisitos. Estos inconvenientes son:

- ¿Cómo se puede gestionar las asignaturas vinculadas a los portafolios?
- ¿Qué ocurre con los aprendizajes críticos asociados a la carrera?
- Dada la restricción de utilizar el servicio de usuarios del Departamento de Informática, ¿cómo se gestionarían los evaluadores?

- ¿Cómo los evaluadores accederán a los portafolios?
- ¿Qué ocurre en el caso de tener varios paralelos en una asignatura?
- ¿Cómo diferenciar las funcionalidades entre los distintos usuarios?
- ¿Cómo sé qué tipo de usuario está utilizando la plataforma?
- ¿Cómo bloquear el acceso de un alumno?

Para resolver estos problemas se realizaron y agregaron las siguientes modificaciones y mejoras que permiten cubrir estas deficiencias dada las restricciones y dificultades en intentar integrar otros servicios de la universidad.

Para el problema de reconocer el tipo de usuario, diferenciar las funcionalidades entre usuarios y los aprendizajes críticos asociados a la carrera, se implementó un nuevo tipo de usuario y un sistema de roles. El nuevo usuario en cuestión fue el administrador y tendrá la facultad de gestionar los roles disponibles, vincularlos a cuentas existentes del departamento, gestionar las carreras disponibles y vincular a estas las cuentas de jefe de carrera y secretario(a) para que puedan acceder a esta. La vinculación se realizará por medio del correo del departamento.

Con esta adición, cada jefe de carrera tendrá la facultad de gestionar las asignaturas del departamento y los aprendizajes críticos asociados a la carrera.

Luego, con respecto al tema de los evaluadores y su acceso, en un principio se había considerado en cuentas de invitado para estos casos, pero dada la restricción del uso del servicio de usuarios del departamento, se optó por agregar las llamadas instancias de evaluación, estas son gestionadas por el jefe de carrera y cuando se encuentran activas, cualquier persona con el enlace correspondiente podrá entrar y visualizar los portafolios seleccionados y vinculados a la instancia.

Respecto al ingreso de alumnos, para poder identificarlos se analiza la lista a la cuales la cuenta pertenece, si pertenece a una lista de alumnos, su acceso será bloqueado, pero su código se encuentra desactivado, mientras se puede encontrar una mejor solución. Finalmente, el tema de los diversos paralelos se solucionó con la creación de múltiples instancias de generación de portafolios y asociarlos a un paralelo en específico.

Dado lo anterior, se agregaron las historias de usuario señaladas en Tabla 3.

Tabla 3: Historias de usuario nuevas a considerar. Fuente: Elaboración propia.

Identificador	Descripción
---------------	-------------

PMD-10	Como jefe de carrera quiero gestionar las carreras a las cuales estoy vinculado para poder realizar cambios y actualizaciones a estas.
PMD-11	Como jefe de carrera quiero gestionar los aprendizajes críticos de la carrera para poder mantenerlos actualizados o agregar nuevos.
PMD-12	Como jefe de carrera quiero gestionar las asignaturas de la carrera para poder permitir la revisión de portafolios de forma más simple a nivel de asignatura.
PMD-13	Como administrador quiero gestionar los roles de la plataforma para controlar el acceso de las funcionalidades permitidas a los usuarios.
PMD-14	Como administrador quiero gestionar las carreras que se utilicen en la plataforma para gestionar su participación.
PMD-15	Como jefe de carrera quiero gestionar instancias de evaluación para que evaluadores externos puedan entrar y revisar los portafolios seleccionados.
PMD-16	Como jefe de carrera quiero que las instancias de evaluación se puedan compartir por medio de un link para facilitar el acceso a evaluadores externos.

- *Course*: corresponde a un curso o asignatura. Un curso puede tener más de un profesor.
- *Evaluation Instance*: representa a instancias en la cual evaluadores externos puedan revisar portafolios de un período de tiempo específico.
- *Critical Learning*: *corresponden a* los aprendizajes críticos asociados a cada carrera.
- *Role*: representa los roles permitidos en la plataforma. Dado que el sistema del departamento no permite diferenciar las cuentas, se decidió utilizar una segmentación basada en roles. Estos roles pueden ser ADMIN, JEFE DE CARRERA, SECRETARIO(A), PROFESOR o ALUMNO. Los alumnos no tienen permitido el acceso, pero el bloqueo está desactivado.

Por otro lado, algunas de las entidades compuestas o secundarias son:

- *Evaluation*: representa una evaluación, puede certamen, control, trabajo, etc.
- *Learning Material*: corresponde a los materiales utilizados durante el semestre, estos pueden ser videos, enlaces, etc.
- *Survey*: corresponde a la encuesta de final de semestre.
- *Course Program*: corresponde al programa del curso, es decir, a la planificación esperada a cumplir durante el semestre, similar al syllabus.

3.6 HERRAMIENTAS DE DESARROLLO

Las herramientas utilizadas en el desarrollo de la plataforma y su despliegue en la nube fueron:

- *Angular 15*: framework para construir aplicaciones del lado del cliente.
- *NestJS 9*: framework para construir aplicaciones del lado del servidor.
- *MongoDB Atlas*: servicio que facilita la gestión de base datos basadas en *MongoDB*.
- *AWS CodeCommit*: servicio basado en *git*, utilizado para gestionar el control de versiones.
- *AWS CodeBuild*: servicio encargado de realizar la compilación del código.
- *AWS CodePipeline*: servicio encargado de gestionar las pipelines para la aplicación de C/I y C/D
- *AWS Route 53*: servicio encargado de gestionar dominios.
- *AWS Cloudfront*: servicio de red que permite entregar contenidos de forma segura con baja latencia y alta velocidad de transferencia.
- *AWS S3*: servicio para almacenar archivos estáticos u objetos.
- *AWS Certificate Manager*: servicio para gestionar los certificados SSL/TLS

- *AWS Elastic Load Balancing (ELB)*: servicio para gestionar el balanceador de carga.
- *AWS Elastic Container Repository (ECR)*: servicio para gestionar imágenes de *docker*.
- *AWS Elastic Container Service (ECS)*: servicio para gestionar aplicaciones que hacen uso de contenedores.
- *AWS Web Application Firewall (WAF)*: servicio que se encarga de proteger las aplicaciones web de los ataques más comunes.
- *AWS Virtual Private Cloud (VPC)*: servicio para gestionar redes virtuales aisladas.
- *AWS IAM*: servicio para gestionar usuarios y permisos en *AWS*.

3.7 DISEÑO DE LA ARQUITECTURA

La arquitectura a implementar se basa en microservicios y se desplegará en la nube de *AWS* y *MongoDB ATLAS*, esta última tiene la finalidad de administrar bases de datos *NoSQL* de manera simple y rápida, facilitando la escalabilidad e integración con otras herramientas.

En la Figura 9 se presenta un diagrama de la arquitectura indicando los elementos más importantes.

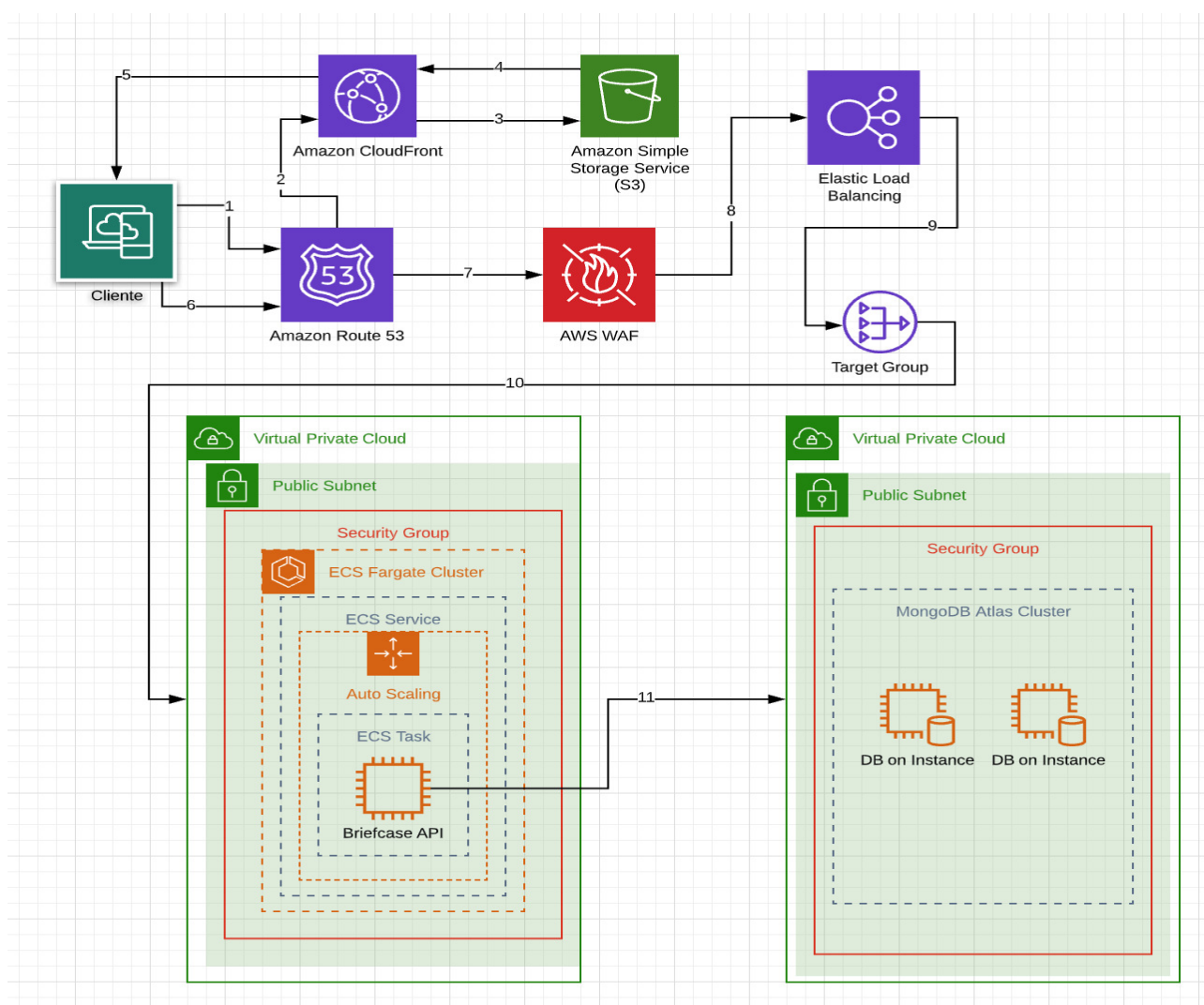


Figura 9: Diagrama de la arquitectura a montar. Fuente: Elaboración propia.

La plataforma consiste en dos aplicaciones, cada una con su repositorio correspondiente. El primer repositorio contiene el código fuente del cliente, el cual corresponde a una aplicación hecha en *Angular*, mientras que el segundo contiene el código fuente del *API* de portafolios, el cual corresponde a una aplicación de servidor hecha con *NestJS*.

Ambos repositorios se encuentran vinculados a dos *pipeline*, encargadas de detectar cambios en el código de la rama *main (master)*, y una vez detectados, comenzarán automáticamente los procesos para compilar el nuevo código, realizar los test unitarios y luego desplegar los cambios a los respectivos servicios. Los archivos compilados del código de la aplicación de *Angular* se almacenarán en un *bucket* de *S3*, este se encuentra configurado para actuar como un servidor de archivos estáticos y gracias a un *CDN* como *Cloudfront* permite a los usuarios obtener el código *JavaScript* a ejecutar en sus navegadores y poder correr la aplicación de *Angular*. Por otra parte, los archivos compilados del código de la *API* se guardarán como una imagen de *Docker* en los repositorios de *ECR*, para luego ser cargada esa imagen en el servicio de *ECS* encargado del mantenimiento de la *API*. Este servicio permite la gestión del autoescalamiento, ejecución de tareas, etc.

Volviendo a la Figura 9, *Route 53* corresponde al servicio de *DNS*, permitiendo la redirección de los usuarios a las aplicaciones. *WAF* corresponde al firewall, permite proteger aplicaciones web de ataques comunes. El *Elastic Load Balancing*, corresponde al balanceador de carga, el cual permite distribuir el tráfico de red para mejorar la escalabilidad y disponibilidad de las aplicaciones, este envía el tráfico a los *target groups* definidos, estos permiten agrupar, por ejemplo, un conjunto de direcciones *IP* como destino, las cuales pertenecen a la *VPC* donde está corriendo la instancia de la *API*.

3.8 PRESENTACIÓN DE LA PLATAFORMA

Con respecto a la plataforma desarrollada, esta tiene diversas vistas. A continuación, serán presentadas capturas de las principales vistas agrupadas por el rol utilizado.

Al entrar a la plataforma, lo primero en desplegarse será la pantalla de inicio de sesión, donde se deberá ingresar con la cuenta del Departamento de Informática (Figura 10).

Una vez dentro, dependiendo del rol seleccionado (un usuario puede tener varios roles), cambiará la pantalla de inicio.

El perfil de administrador (Figuras 11 y 12), tiene la capacidad de generar roles, vincular roles a correos en específico y gestionar las carreras (por ahora, solo para Ingeniería Civil en Informática) y vincular a las carreras las cuentas del jefe de carrera y secretaria (utilizando su correo del DI).



Figura 10: Pantalla de inicio de sesión de la plataforma: Fuente: Elaboración propia.

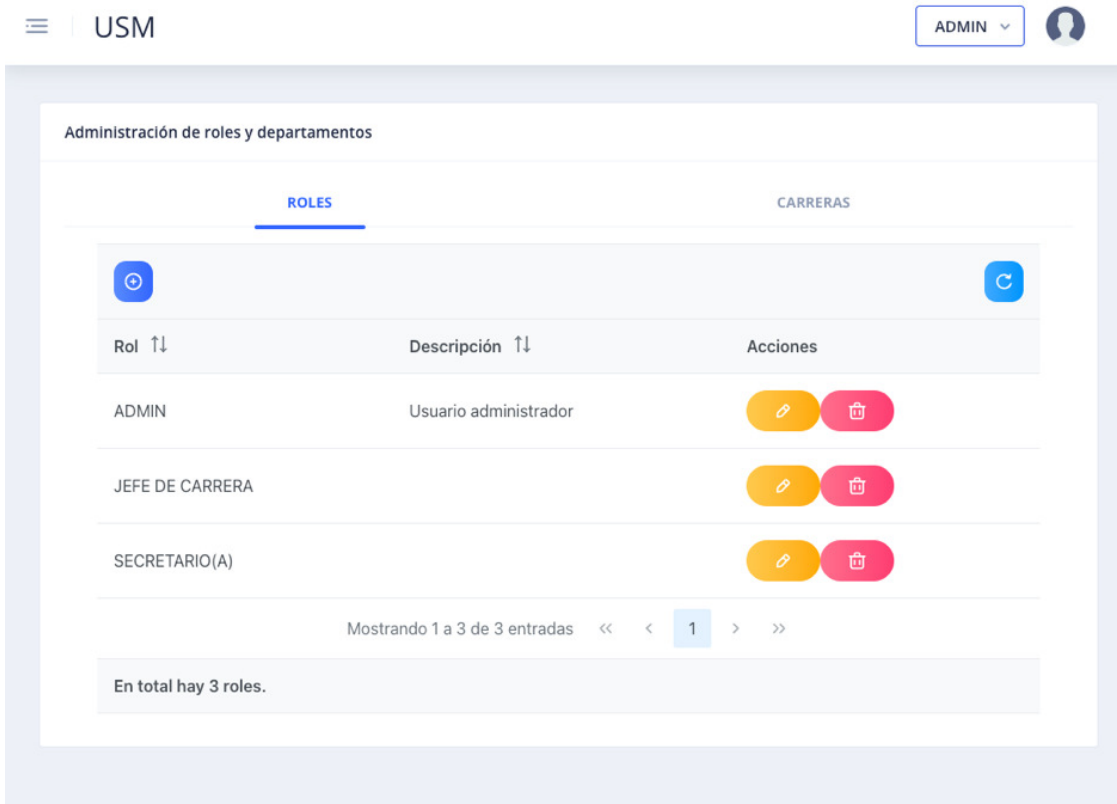


Figura 11: Vista de inicio para un administrador: Fuente: Elaboración propia.

Crear carrera

Nombre

Informática

11 / 80

Campus

Casa Central

Departamento

Informática

Programa

73

Descripción

0 / 150

Jefe de departamento

Nombre

Yelson

6 / 80

Apellido

Fernández

9 / 80

Correo del DI

yfernand@alumnos.inf.utfsm.cl

29 / 80

Secretario(a)

Nombre

Yelson

6 / 80

Apellido

Fernández

9 / 80

Correo del DI

yfernand@alumnos.inf.utfsm.cl

29 / 80

CREAR

Figura 12: Modal para crear una carrera. Fuente: Elaboración propia.

El perfil de jefe de carrera (Figuras 13, 14, 15 y 16), tiene la posibilidad de gestionar los cursos que pertenecen al departamento al cual se encuentra vinculado (puede tener más de un departamento), gestionar los aprendizajes críticos de la carrera correspondiente, visualizar los distintos portafolios que se han ido generando en las asignaturas, gestionar los profesores de cada asignatura y gestionar las instancias de evaluación para que personas ajenas a la universidad pueda revisar los portafolios sin tener que contar con una cuenta.

USM

JEFE DE CARRERA

Informática

Yelson Marcelo Fernandez Jelves

Carrera: Informática

CURSOS

APRENDIZAJES CRÍTICOS

INSTANCIAS DE EVALUACIÓN

Sigla	Nombre	Carrera	Campus	Acciones
IWI101	Programación	Informática	Casa Central	

Mostrando 1 a 1 de 1 entradas

En total hay 1 cursos.

Figura 13: Vista de inicio del perfil de jefe de carrera. Fuente: Elaboración propia.

DISEÑO, DESARROLLO E IMPLEMENTACIÓN DE UN SISTEMA DE MONITOREO DE ASIGNATURAS BASADO EN PORTAFOLIOS DOCENTE

Editar curso

Nombre

Programación

12 / 100

Sigla

IWI101

6 / 6

Carrera

Informática

Descripción

Asignatura que entrega una introducción a la programación

57 / 500

Créditos USM

3

Créditos SCT

3

Semestre

S1

Tipo de asignatura

Ciencias Básicas

AGREGAR CURSO

Figura 14: Modal para crear un curso nuevo. Fuente: Elaboración propia.

USM

JEFE DE CARRERAInformáticaYeison Marcelo Fernandez Jelves

Carrera: Informática

CURSOS

APRENDIZAJES CRÍTICOS

INSTANCIAS DE EVALUACIÓN

Aprendizaje	Descripción	Acciones
Aprendizaje crítico 1		

| Aprendizaje crítico 2 | | |

Mostrando 1 a 2 de 2 entradas

En total hay 2 aprendizajes críticos.

Figura 15: Vista de gestión de aprendizajes críticos: Fuente: Elaboración propia.

Curso: Programación

APRENDIZAJES CRÍTICOS

PROFESORES

PORTAFOLIOS

Nombre	Apellido	Email	Acciones
Yeison	Fernández	yfernand@alumnos.inf.utfsm.cl	

Mostrando 1 a 1 de 1 entradas

En total hay 1 profesores.

Figura 16: Vista dentro de un curso. Fuente: Elaboración propia.

Con respecto a los perfiles de secretaria (o) y profesor (Figuras 17, 18, 19 Y 20), las funcionalidades son similares, la diferencia principal es que la secretaria (o) puede trabajar en portafolios de múltiples cursos, mientras que el profesor solo puede manipular portafolios de los cursos a los cuales se encuentre vinculado. Por lo cual, comparten las mismas vistas.

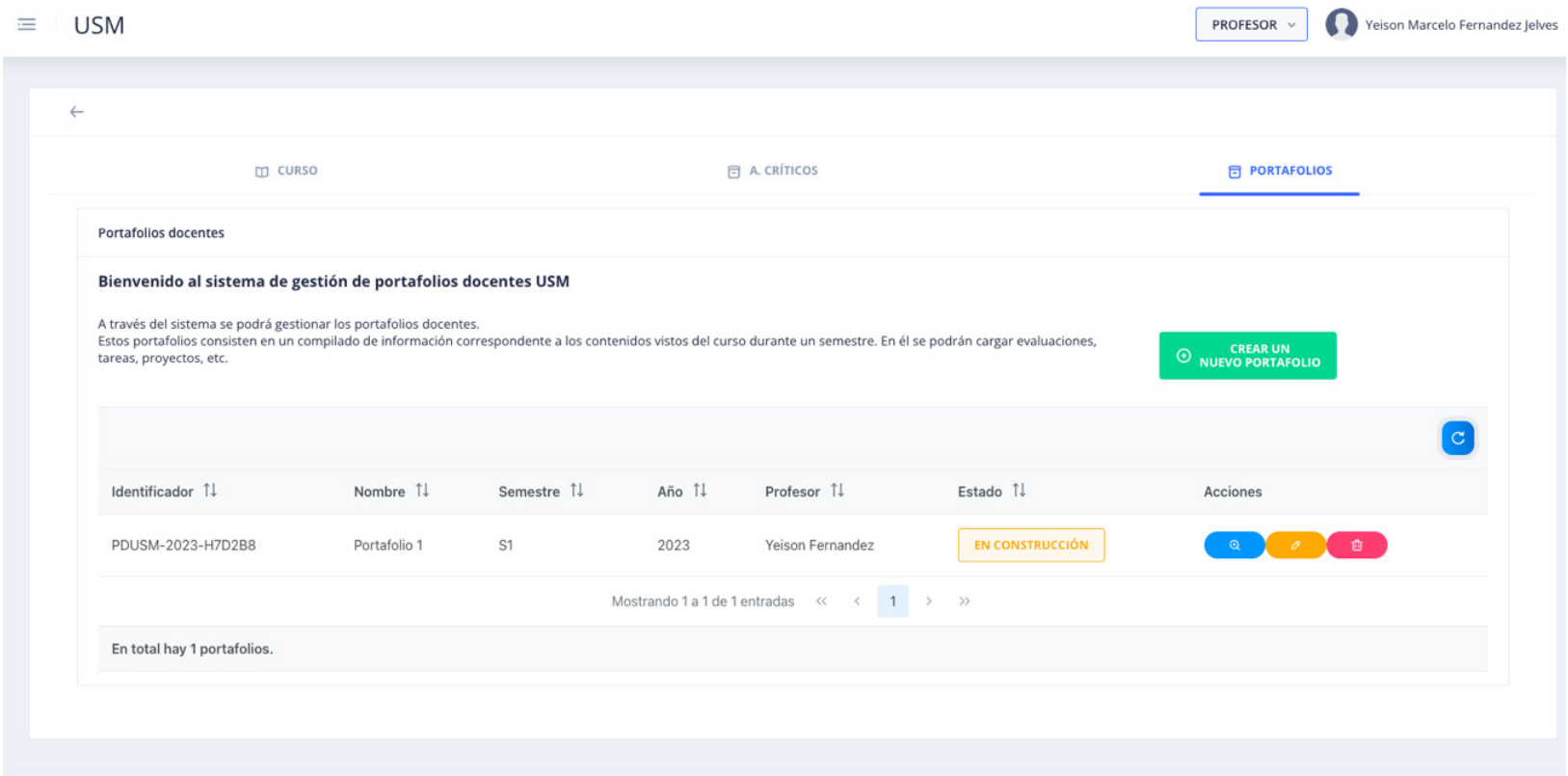
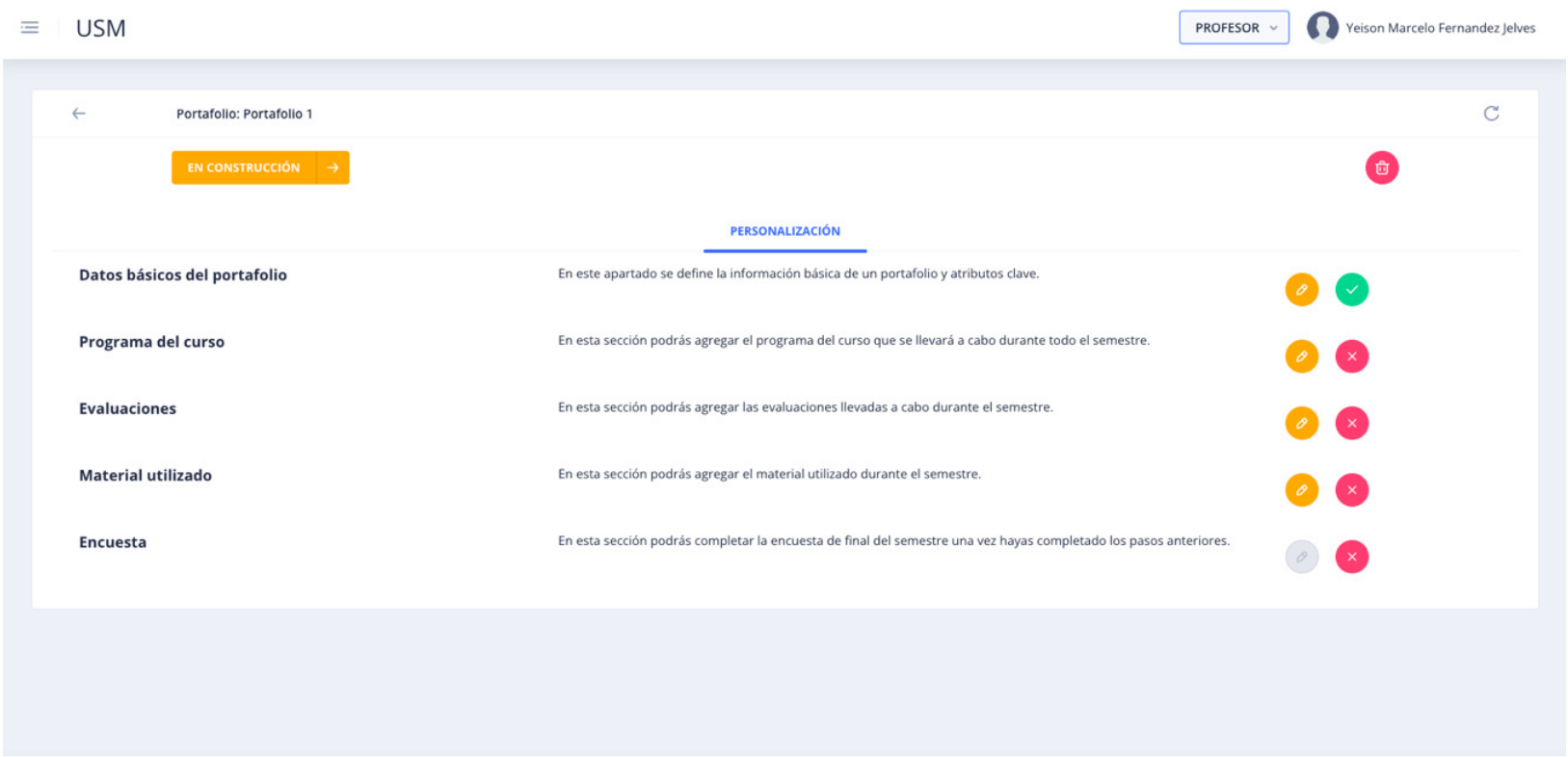


Figura 17: Vista de inicio del profesor. Fuente: Elaboración propia.



USM 2021

Figura 18: Vista de edición de un portafolio. Fuente: Elaboración propia.

DISEÑO, DESARROLLO E IMPLEMENTACIÓN DE UN SISTEMA DE MONITOREO DE ASIGNATURAS BASADO EN PORTAFOLIOS DOCENTE

Editar programa del curso

1

2

3

4

Datos básicos

P. de Competencias

R. A. a Obtener

Programa

Nombre de profesor

Yelson Fernandez

16 / 100

Correo de profesor

yfernand@alumnos.inf.utfsm.cl

29 / 100

Descripción general

Descripcion 1

13 / 400

Estrategia de aprendizaje

Estrategia 1

12 / 500

Resultados finales de aprendizaje

Resultados 1

12 / 300

GUARDAR

Figura 19: Modal de edición de programa del curso. Fuente: Elaboración propia.

Portafolio: Portafolio 1

Editar evaluaciones del curso

Certamen 1

Nombre

Certamen 1

10 / 100

Mejor nota

77

Peor nota

10

Nota promedio

44

Tipo

Certamen

Descripción

Evaluación correspondiente a unidades 1 y 2

43 / 300

Enunciados y resultados

Favor de subir los enunciados de la evaluación y las copias de la mejor y peor nota. Sólo se aceptarán en pdf.

Archivo cargado: trabajo02.zip

AGREGAR EVALUACIÓN

AGREGAR AL PORTAFOLIO

Figura 20: Vista de carga de evaluaciones en un portafolio. Fuente: Elaboración propia.

CAPÍTULO 4: VALIDACIÓN DE LA SOLUCIÓN

En este capítulo se presenta la validación de la solución enfrentada desde tres perspectivas. La primera corresponderá al apartado de *testing*, partiendo de la base del uso de *tests* unitarios y obtención, para su posterior análisis, de la métrica llamada cobertura de código o *code coverage*. El segundo proceso consistirá en la presentación de la solución a profesores del departamento para confirmar si se cumplieron las historias de usuarios planteadas anteriormente. Finalmente, se presentará la validación con el Encargado de Infraestructura del Departamento de Informática, de la arquitectura planteada, recepción de feedback y la posibilidad de implementación en el departamento.

4.1 RESULTADOS PRUEBAS UNITARIAS

La construcción de las pruebas fueron desarrolladas tanto en el *frontend* como en el *backend*.

En el caso del *frontend* se utilizó la *suite* integrada que ofrece *Angular* para el desarrollo y ejecución de estas, la cual corresponde a *Jasmine* y *Karma*. *Jasmine* permite la creación de *tests* unitarios, mientras que *Karma* es un ejecutor de tareas para los *tests*. Para el *backend*, se utilizó principalmente *Jest*, el cual corresponde a un *framework* de *testing* orientado a *JavaScript*, que funciona en proyectos que hacen uso de *TypeScript*, como *NestJS*.

Para cuantificar el resultado de las pruebas unitarias se utilizó la métrica llamada *code coverage*. Esta métrica permite saber que parte del código fue sometido a las pruebas y qué porcentaje no ha sido evaluado. Por lo general, se suelen utilizar los siguientes parámetros:

- Cobertura de funciones: cantidad de funciones definidas que se evalúan.
- Cobertura de ramas: cuántas ramas de las estructuras de control se han ejecutado.
- Cobertura de líneas: cuántas líneas de código se han evaluado.

La supervisión de las métricas de cobertura del código ayuda a garantizar que se mantiene un nivel suficiente de pruebas automatizadas. Si la cobertura del código comienza a disminuir, podría ser una señal de que no se está tratando las pruebas automatizadas como un elemento esencial de la escritura de nuevo código. Por lo general, un valor estándar a buscar corresponde a más del 80% de cobertura de código.

Utilizar esto no garantiza la calidad en lo desarrollado, pero es un punto de partida, ya que en conjunto con otras métricas, permite ir desarrollando pruebas más robustas y de mejor calidad.

Dicho lo anterior, luego del desarrollo y ejecución de estas pruebas unitarias, se obtuvo los porcentajes de *code coverage* de Tabla 4 para frontend y Tabla 5 para backend.

Tabla 4: Resumen *code coverage* de frontend. Fuente: Elaboración propia.

<i>Statements</i>	<i>Branches</i>	<i>Functions</i>	<i>Lines</i>
92,62%	76,18%	85,81%	93,25

Tabla 5: Resumen *code coverage* de backend. Fuente: Elaboración propia.

<i>Statements</i>	<i>Branches</i>	<i>Functions</i>	<i>Lines</i>
89,25%	85,64%	88,65%	91,32%

4.1 CUMPLIMIENTO DE LAS HISTORIAS DE USUARIO

Al finalizar el desarrollo de la plataforma, se realizó una presentación a profesores del departamento, mostrando el funcionamiento y las diversas funcionalidades implementadas. Los resultados obtenidos se presentarán en Tabla 6, donde para cada historia de usuario detallada en el apartado 3.3 y 3.4, se indica el nivel de cumplimiento de estas. Los posibles valores para el nivel de cumplimiento son: completada, parcialmente completada y no completada.

Tabla 6: Tabla de cumplimiento de historias de usuario. Fuente: Elaboración propia.

HISTORIA DE USUARIO	NIVEL CUMPLIMIENTO
PMD-1	COMPLETADA
PMD-2	COMPLETADA
PMD-3	COMPLETADA
PMD-4	COMPLETADA
PMD-5	COMPLETADA
PMD-6	COMPLETADA
PMD-7	COMPLETADA

PMD-8	COMPLETADA
PMD-9	COMPLETADA
PMD-10	COMPLETADA
PMD-11	COMPLETADA
PMD-12	COMPLETADA
PMD-13	COMPLETADA
PMD-14	COMPLETADA
PMD-15	COMPLETADA
PMD-16	COMPLETADA

A partir de la tabla anterior, se puede determinar que se cumplieron con todas las historias de usuarios planteadas en el apartado 3.3 y 3.4, por lo cual se lograron implementar las funcionalidades esperadas.

4.1 IMPLEMENTACIÓN EN DEPARTAMENTO DE INFORMÁTICA

En un principio se había conversado para que la plataforma pudiese implementarse y desplegarse en la nube de AWS, ya que se estaba explorando la posibilidad de hacer uso de esta en un futuro por parte del área de infraestructura del Departamento. Se trabajó con ese fin. Una vez completado el desarrollo, se implementó en una cuenta personal de AWS la arquitectura diseñada para ver su funcionamiento y posibles mejoras futuras.

En la etapa final de esta memoria, se tuvo que enfrentar la compleja situación que le tocó vivir al equipo de Infraestructura con el lamentable fallecimiento de su subdirector, y con los cambios en las prioridades que esto significó, entre ellos el postergar el plan de trabajar en la nube AWS.

Por lo que se realizó una presentación de lo desarrollado a quien quedó a cargo del área de Infraestructura, es decir, se mostró la aplicación web y la arquitectura en la nube considerada. Esto permitió concluir a modo de validación lo siguiente:

- Con respecto a la plataforma web no hubo grandes comentarios, pero se valoró que las herramientas usadas estuviesen en su última LTS, en este casi *Angular 15* y *NestJS 9*, y contar con los tests unitarios correspondientes.

- Con respecto a la arquitectura en la nube de *AWS*, se valoró la creación del manual de implementación en *AWS* (ver Anexo A) construido, pero el encargado planteó que se veía difícil en estos momentos su implementación en el departamento debido a los costos que esto conlleva y la administración correspondiente de este, por lo cual, fue solicitado la generación de un archivo *docker-compose* para que se pudiese levantar en una máquina virtual y poder ser implementado en un futuro. El archivo solicitado fue construido y el código se puede ver en Anexo B.

Por lo tanto, dada la complejidad de la plataforma y las dificultades que conlleva el uso de la nube de *AWS*, la implementación de esta se realizará a futuro por parte del departamento y se le hace entrega de los repositorios con el código fuente correspondiente, el manual de implementación tanto local como en la nube.

CAPÍTULO 5: CONCLUSIONES

5.1 CONCLUSIONES GENERALES

Las carreras universitarias o técnicas, pueden ser consideradas como sistemas cuyo propósito es preparar a las personas para enfrentar la etapa profesional o laboral. Para llevar a cabo esta labor, en las instituciones de educación superior se realizan procesos de forma interna que en conjunto se comunican y trabajan como un todo. Desde la labor de la gestión de una carrera hasta el desarrollo de las distintas asignaturas que la conforman, se logra que los alumnos vayan aprendiendo las competencias propias de su profesión y fortaleciendo sus habilidades personales, las que en conjunto permiten alcanzar el perfil de egreso declarado por la carrera, pero como todo sistema, pueden existir ciertas fallas o problemas a lo largo de la ejecución de los procesos. Entonces, es importante contar con herramientas para realizar un monitoreo de los procesos internos involucrados en la formación del estudiante, de tal manera de encontrar los puntos críticos que generan problemáticas y encontrar posibles soluciones. Por ejemplo, en este contexto uno de los principales problemas a lo largo de los años es la deserción y cómo afecta a la tasa de retención de la carrera de Ingeniería Civil Informática de la UTFSM. Según se investigó existen diversos factores que pueden llevar a la deserción, estos pueden ser tanto académicos como financieros o de carácter más personal, por lo cual, si se quiere enfrentar y atacar esta problemática, es importante elegir hacia donde se quiere enfocar la solución que se vaya a proponer.

Dado que es muy difícil abarcar y solventar todas las posibles problemáticas dentro de la solución, en este caso, se optó por un enfoque orientado a apoyar a quienes gestionan la carrera con una plataforma que a través de portafolios por asignatura, se mantenga un registro de lo que ha implicado desde el punto de vista docente cada una de las versiones en que se ha dictado, esperando con ella monitorizar las diversas asignaturas de la carrera al poder visualizar los contenidos y herramientas que se utilizan en el proceso de enseñanza-aprendizaje de los alumnos.

El monitoreo es un apartado importante en cada proceso, ya que permite poder estar pendiente de las falencias o problemas que puedan surgir y poder reaccionar a tiempo y dado el ambiente sobre el cual se estaba trabajando, era necesario darle un enfoque educacional. Es por esto que se utilizaron portafolios docentes.

El uso de portafolios docentes permite analizar qué conocimientos posee un profesor, cómo se imparten y qué herramientas pedagógicas se suelen utilizar. Esto permite realizar

comparaciones entre asignaturas con buenos porcentajes de aprobación versus otras que poseen porcentajes menores.

En consecuencia, en este proyecto la solución que se diseñó, desarrolló e intentó implementar corresponde a una plataforma de monitoreo basado en portafolios docentes.

Para el desarrollo de la solución se trató de utilizar varios conceptos y herramientas que se vieron a lo largo de la carrera, historias de usuarios, toma de requerimientos, modelos, etc. El desarrollar una solución de software es complejo y no es tan trivial como parece, los estándares siempre van cambiando y se tienen que considerar diversas aristas y, a veces, llevar lo modelado a la práctica puede ser más complejo de lo que se ve.

Inicialmente, se tenía pensado completar el desarrollo en un tiempo mucho menor, pero terminó siendo mayor a lo esperado y eso en el mundo del desarrollo es bastante común. Pero a pesar de todas las dificultades, se logró diseñar y construir una plataforma que permitiese cumplir en gran parte con los objetivos planteados al inicio.

Lamentablemente, para el apartado de la implementación en el Departamento de Informática, por motivos de fuerza mayor que sufrió el área a cargo, no pudo llevarse a cabo como se esperaba. Al ser una solución principalmente pensada en la nube, esto trae consigo costos, además de la dificultad de implementación para alguien u organización que no tiene experiencia directa con ella. Como aprendizaje, se puede rescatar que cuando se planifica un proyecto, es importante considerar que la planificación puede cambiar durante la marcha, ya que pueden ocurrir factores externos que afecten a las variables involucradas.

5.2 CUMPLIMIENTO DE OBJETIVOS

Para medir los resultados obtenidos, lo podemos hacer por medio del cumplimiento de cada uno de los objetivos propuestos, donde el cumplimiento de los objetivos específicos definen el cumplimiento del objetivo general.

- Diseñar, desarrollar y desplegar una plataforma de monitoreo de asignaturas basado en portafolios docentes.

Como se detalla principalmente en la sección 3 y en el Anexo A, se logró diseñar, desarrollar y desplegar correctamente una plataforma que permitiera la creación de portafolios docentes, aunque no se desplegó en el Departamento de Informática, sino que en una cuenta personal.

- Utilizar las herramientas que ofrece AWS y MongoDB Atlas para poder construir sistemas informáticos de forma rápida, confiable y segura.

Como se detalla en los puntos 3.6 y 3.7 y en el Anexo A, se logró utilizar las herramientas mencionadas para desarrollar e implementar, aunque no de forma oficial, la plataforma.

- Aplicar las prácticas *DevOps* para el desarrollo de la plataforma.

Como se detalla en los puntos 2.3.7, existen un conjunto de buenas prácticas para poder utilizar esta filosofía llamada *DevOps*. En concreto durante el diseño, desarrollo e implementación de la solución se lograron implementar con éxito la integración continua, despliegue continuo, uso de microservicios y el registro/monitoreo de la plataforma. Tanto en los puntos 3.6, 3.7 como en el Anexo A, queda evidenciado el uso de las herramientas necesarias para la aplicación de estas prácticas.

Una vez detallado el cumplimiento de los objetivos específicos, se puede inferir el cumplimiento del objetivo general: **Ayudar a mejorar la tasa de retención en la carrera de Ingeniería Civil en Informática USM.**

5.3 TRABAJO FUTURO

Para detallar esta sección, será separada en las acciones o trabajo a realizar en el corto, mediano y largo plazo.

Empezando en el corto plazo, es posible realizar mejoras en la plataforma cambiando el cómo se gestionan asignaturas con más de un paralelo, ya que actualmente cuenta con una solución bastante simple. Además, desarrollar un manual detallado del uso de esta más sesiones de inducción para que profesores y funcionarios del departamento estén capacitados y sepan hacer uso de esta.

En el mediano plazo se puede investigar en mejoras a la arquitectura *cloud* propuesta como el uso de un *API gateway* y funciones *Lambda* para la autorización de las conexiones, el levantar y gestionar la base datos *MongoDB* directamente en *AWS* y no utilizar un servicio externo como MongoDB Atlas. Además, integrar en la plataforma una funcionalidad para construir diversos gráficos tomando como datos la información cargada en los portafolios docentes. También, sumado al uso del *API gateway*, realizar modificaciones en el clúster para que sea privado y solo se permita el acceso por medio del *API gateway*.

Mejorar la integración de los roles y facilitar la identificación de las cuentas de alumno.

Finalmente, en el largo plazo realizar instancias de participación e inducción para el montaje y configuración de la plataforma en la nube de *AWS*, entendiendo que las condiciones futuras serán mejores a las actuales. También, ver la posibilidad de realizar una integración con los otros sistemas de la universidad para tener acceso, por ejemplo, a la encuesta docente de cada profesor, a las asignaturas del departamento, a las notas de los alumnos

de cada asignatura según semestre, etc. Actualmente, las asignaturas se gestionan de forma manual, por lo cual, una persona debe crearlas y vincularlas a la carrera y a su vez agregar a los profesores que las impartirán. Sería beneficioso contar con esa información de antemano y no ir replicándola en varios lugares. Adicionalmente, mejorar los tiempos de levantamiento de la infraestructura utilizando por ejemplo *Cloudformation*, para escribir la infraestructura como código.

5.4 CONCLUSIÓN PERSONAL

Durante todo el desarrollo de esta memoria, fueron surgiendo problemas o inconvenientes que fueron mermando o retrasando el avance de esta, desde la planificación (que al final no fue como se esperaba) hasta su finalización. Lo importante siempre es ir aprendiendo de los errores, surgieron muchos, pero cada uno de ellos fueron estudiados, entendidos, asimilados, enfrentados y superados. Siempre he considerado que si en un proyecto o el desarrollo de cualquier cosa, si no hay errores, es señal de peligro. Los errores son señales de que se va avanzando y alcanzando poco a poco los resultados esperados.

Por otra parte, cuando se construye un software, este se puede hacer de diversas formas, utilizando diversos lenguajes, tipos de arquitecturas, metodologías. Pero ¿cómo sabemos cuál es la correcta? Lo cierto es que cada herramienta o metodología se crea para solucionar una problemática o facilitar el uso de algo y para hacer esta elección dependerá del ambiente o cultura en la cual se esté trabajando. Por ejemplo, en una *startup* si es está a cargo de la construcción de un producto de software, dado que los recursos y el tiempo son valiosos, la cantidad y flexibilidad de posibles herramientas disponibles a usar es muy grande desde el lenguaje hasta servicios de terceros, en cambio, si se está en un banco, se le dará prioridad a tecnologías que cuenten con un gran respaldo y trayectoria, además de tener un enfoque en la seguridad y fiabilidad. Esto para alguien que recién está aprendiendo o partiendo en este mundo es muy difícil, pero con la experiencia y la capacidad de aprender distintos lenguajes o tecnologías, va ganando la habilidad de ver y enfrentar a problemas desde diversos puntos de vista, lo que mejora la elección de estas tecnologías.

REFERENCIAS BIBLIOGRÁFICAS

Atlassian. (20 de Junio de 2023). *Automatización de pruebas DevOps*. Obtenido de Cómo las pruebas automatizadas hacen posible DevOps:
<https://www.atlassian.com/es/devops/devops-tools/test-automation>

Atlassian. (2 de Mayo de 2023). *Historias de usuario con ejemplos y plantilla*. Obtenido de Historias de usuario con ejemplos y plantilla:
<https://www.atlassian.com/es/agile/project-management/user-stories>

AWS. (8 de Abril de 2023). *¿Qué es DevOps?* Obtenido de
<https://aws.amazon.com/es/devops/what-is-devops/>

AWS. (8 de Abril de 2023). *Microservicios*. Obtenido de
<https://aws.amazon.com/es/microservices/>

AWS. (8 de Abril de 2023). *Informática en la nube con AWS*. Obtenido de
<https://aws.amazon.com/es/what-is-aws/>

AWS. (7 de Abril de 2022). *CodeCommit*. Obtenido de
<https://aws.amazon.com/es/codecommit/>

AWS. (1 de Febrero de 2020). *CodeDeploy*. Obtenido de
<https://aws.amazon.com/es/codedeploy/>

AWS. (9 de Enero de 2022). *CodePipeline*. Obtenido de
<https://aws.amazon.com/es/codepipeline/>

AWS. (8 de Agosto de 2022). *Cloudwatch*. Obtenido de
<https://aws.amazon.com/es/cloudwatch/>

AWS. (21 de Septiembre de 2021). *Route53*. Obtenido de
<https://aws.amazon.com/es/route53/>

AWS. (5 de Diciembre de 2022). *S3*. Obtenido de <https://aws.amazon.com/es/s3/>

AWS. (5 de Enero de 2023). *API Gateway*. Obtenido de <https://aws.amazon.com/es/api-gateway/>

AWS. (2 de Marzo de 2020). *CodeBuild*. Obtenido de
<https://aws.amazon.com/es/codebuild/>

AWS. (2 de Junio de 2022). *¿Qué es NoSQL?* Obtenido de
<https://aws.amazon.com/es/nosql/>

Google. (2023, Enero 7). *Introduction to the Angular docs*. Retrieved from
<https://angular.io/docs>

hommes, T. d. (2016). *Diseño e implementación de un sistema de monitoreo*.

IBM. (22 de Junio de 2023). *¿Qué es la prueba de software?* Obtenido de
<https://www.ibm.com/es-es/topics/software-testing>

Lyons, N. (1999). *El uso de portafolios. Propuestas para un nuevo profesionalismo docente*.
Buenos Aires: Amorrortu.

Lauriac, N. (2016). *Diseño e implementación de un sistema de monitoreo*. Obtenido de
Terre des hommes - ayuda a la infancia:
https://www.tdh.ch/sites/default/files/161019_tdh_guidemonitoring_es_version_impression.pdf

MongoDB, Inc. (Marzo de 2023). *¿Qué es MongoDB?* Obtenido de
<https://www.mongodb.com/es/what-is-mongodb>

UTFSM. (4 de Abril de 2023). *Reportes USM*. Obtenido de Reportes USM:
<http://portalreportes.usm.cl/Reportes/MenuPrincipal.html>

ANEXOS

ANEXO A: IMPLEMENTACIÓN DE LA ARQUITECTURA CLOUD

A continuación, se detallan los pasos a realizar para montar la arquitectura en *AWS*.

CREAR USUARIO IAM

Es importante como primer paso crear usuarios con permisos específicos. En primera instancia para llevar a cabo este paso, se utilizará el servicio *IAM*. Este servicio proporciona un control de acceso detallado a todo *AWS*.

Para crear un usuario, es necesario ingresar a la consola de *AWS* y utilizar la cuenta raíz (la cuenta que se registró con el método de pago). Luego en el buscador, ingresar *IAM* y seleccionar la coincidencia. Una vez dentro, debería verse una interfaz similar a lo mostrado en Figura 21.

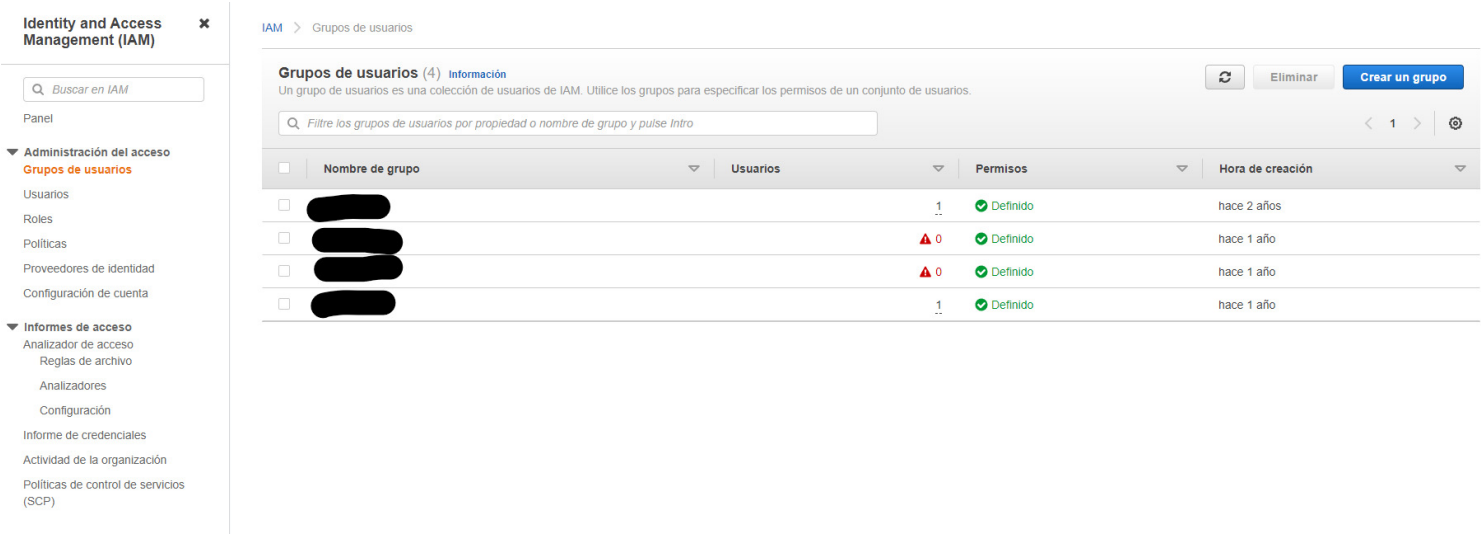


Figura 21: Interfaz de servicio IAM. Fuente: AWS

A continuación, seleccionar desde el panel lateral “Usuarios”, luego seleccionar “Agregar usuarios”.

Para crear un usuario consta de dos pasos, en la primera etapa debe agregar un nombre para el usuario y seleccionar el tipo de acceso. Para nuestros propósitos basta con que el usuario tenga clave de acceso. Luego en la siguiente etapa se deben asociar los permisos al usuario, ahí elegiremos “Asociar directamente las políticas existentes” y en el buscador tipear “*AmazonS3FullAccess*”.

Finalmente, del listado, se debe seleccionar la opción correspondiente y pasar a la siguiente etapa donde agregaremos una etiqueta que consiste en un par llave-valor, para identificar

fácilmente a quién corresponde el usuario. Luego pasamos a la etapa final donde revisaremos que todo esté en orden y creamos el usuario. Es importante descargar y guardar el archivo que contiene el identificador de la clave de acceso y la clave secreta.

GENERAR CREDENCIALES *GIT*

Para el control de versiones se utilizará el servicio de *AWS CodeCommit*, el cual trabaja con repositorios de *git* privados, por lo tanto, es importante generar credenciales para acceder a estos.

Una vez generado el usuario, se debe dirigir al apartado de usuarios y seleccionar el usuario recién creado. Luego elegir el apartado de credenciales de seguridad y desplazarse al apartado de “Credenciales de Git HTTPS para AWS CodeCommit”.

Finalmente se debe generar las credenciales apretando el botón. Una vez generadas se desplegará una ventana como la de la Figura 22.

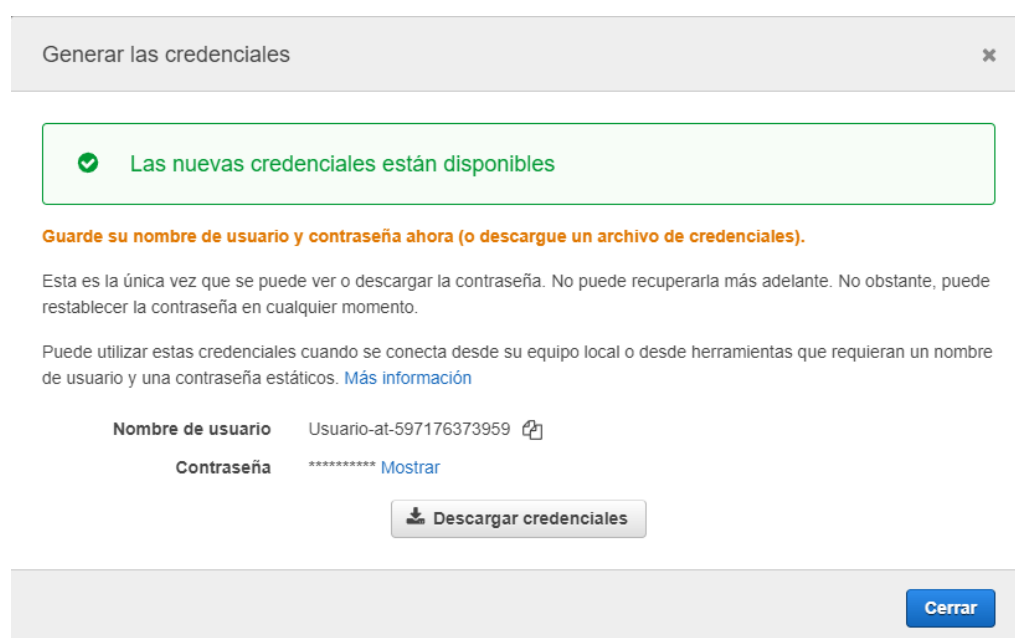


Figura 22: Ventana de éxito al generar las credenciales GIT. Fuente: Elaboración propia.

Es importante descargar las credenciales y mantenerlas seguras para su posterior uso.

CONSTRUCCIÓN DE LOS REPOSITORIOS

Como se mencionó anteriormente, se utilizará *AWS CodeCommit* para el control de versiones. En el buscador de la parte superior se debe escribir “CodeCommit” y luego

seleccione el servicio. Una vez dentro se debe apretar el botón de “Crear el repositorio” y después se deberá completar la información que se solicite (no es necesario habilitar *Amazon CodeGuru*), como se visualiza en ventana de la Figura 23.

Herramientas para desarrolladores > CodeCommit > Repositorios > Crear el repositorio

Crear el repositorio

Cree un repositorio seguro para almacenar y compartir el código. Comience escribiendo un nombre y una descripción de su repositorio. Los nombres de repositorio se incluyen en las URL de ese repositorio.

Configuración de repositorio

Nombre del repositorio

100 caracteres como máximo. Se aplican otros límites.

Descripción - *opcional*

1000 caracteres como máximo

Etiquetas

Agregar

☐ Habilitar Amazon CodeGuru Reviewer para Java y Python - *opcional*

Obtenga recomendaciones para mejorar la calidad de los códigos Java y Python de todas las solicitudes de extracción de este repositorio.

Se creará en IAM un rol vinculado al servicio en su nombre si no existe.

Cancelar **Crear**

Figura 23: Creación del repositorio para el *frontend* de la plataforma. Fuente: Elaboración propia.

Repetir los mismos pasos para crear un nuevo repositorio, pero para el *backend*, el cual consta de un servicio de portafolio, que funciona como una *API REST*.

CARGAR CÓDIGO FUENTE

A continuación, se deberá cargar el código fuente en los repositorios creados anteriormente.

Lo primero será clonar los repositorios vacíos, para clonarlos se deberá estar en la lista de repositorios de *CodeCommit*, luego seleccionar en la columna de “Clonar URL” el apartado “HTTPS”, obteniéndose una URL como la siguiente:

<https://git-codecommit.us-east-1.amazonaws.com/v1/repos/plataforma-monitoreo-docente>

Esta URL, antes de ser clonada, debe ser modificada a la siguiente:

```
https://<user>:<password>@git-codecommit.us-east-1.amazonaws.com/v1/repos/plataforma-monitoreo-docente
```

Donde deben ser remplazados el *user* y *password* por las obtenidas al generar las credenciales de *GIT anteriormente*.

DOMINIO Y CERTIFICADOS

Para la gestión de dominios se utilizará *AWS Route53*, es decir, *AWS* será el que gestione los dominios y subdominios de la plataforma.

Primero, nos dirigiremos a *AWS Route53* y en el panel lateral seleccionamos “Zonas hospedadas”, luego seleccionamos “Crear una zona alojada”, donde deberemos de completar el siguiente formulario que se visualiza en la Figura 24.

Crear una zona alojada

Configuración de zona alojada

Una zona alojada es un contenedor que incluye información sobre cómo usted desea dirigir el tráfico de un dominio (como example.com) y sus subdominios.

Nombre de dominio

Este es el nombre del dominio al que desea dirigir el tráfico.

example.com

Caracteres válidos: a-z, 0-9, ! * # \$ % & ' () * + , - / : ; < = > ? @ [\] ^ _ ` { | } . ~

Descripción - opcional

Este valor le permite distinguir las zonas alojadas que tienen el mismo nombre.

La zona alojada se utiliza para...

La descripción puede tener un máximo de 256 caracteres. 0/256

Tipo

El tipo indica si desea dirigir el tráfico en Internet o en una VPC de Amazon.

☒ Zona alojada pública

Una zona alojada pública determina cómo se dirige el tráfico en Internet.

☐ Zona alojada privada

Una zona alojada privada determina cómo se direcciona el tráfico dentro de una VPC de Amazon.

Etiquetas

Aplice etiquetas a las zonas alojadas para organizarlas e identificarlas.

No hay etiquetas asociadas al recurso.

Agregar una etiqueta

Puede agregar hasta 50 etiquetas más.

Cancelar

Crear una zona alojada

Figura 24: Formulario para agregar una zona alojada. Fuente: Elaboración propia.

Se debe agregar el nombre del dominio y configurarlo como “Zona alojada pública”. Una vez creada se podrán visualizar los servidores de nombres, los cuales serán necesarios para configurar la redirección del dominio. La redirección del dominio varía dependiendo del proveedor, por lo cual quedará a criterio del lector su configuración.

A continuación, teniendo configurado el dominio, hay que dirigirse a *AWS Certificate Manager*, servicio que permitirá la gestión e implementación de certificados SSL/TLS.

Dentro, en el panel lateral, se debe seleccionar “Enumerar certificados” y luego “Solicitar”, seleccionamos la opción del certificado público.

Finalmente, en el paso final se deberá de agregar los nombres de dominio a vincular y elegir el método de validación. Se debe seguir el ejemplo mostrado en la Figura 25.

AWS Certificate Manager > Certificados > Solicitar certificado > Solicitar certificado público

Solicitar certificado público

Nombres de dominio

Nombre de dominio completo [Información](#)

Eliminar

Eliminar

Eliminar

Agregar otro nombre a este certificado

Puede agregar nombres adicionales a este certificado. Por ejemplo, si solicita un certificado para "www.example.com", es posible que desee agregar el nombre "example.com" para que los clientes puedan acceder al sitio con cualquier nombre.

Seleccionar método de validación [Información](#)

Seleccione un método para validar la propiedad del dominio

☒ **Validación de DNS: recomendado**
Elija esta opción si está autorizado para modificar la configuración de DNS de los dominios de la solicitud de certificado.

☐ **Validación por email**
Elija esta opción si no tiene permiso o no puede obtenerlo para modificar la configuración de DNS de los dominios en la solicitud de certificado.

Etiquetas [Información](#)

Para administrar mejor sus certificados, puede asignar opcionalmente sus propios metadatos a cada recurso en forma de etiquetas.

Clave de etiqueta

Valor de etiqueta - opcional

Eliminar etiqueta

Agregar etiqueta

Puede agregar 49 más etiquetas.

Cancelar

Anterior

Solicitar

Figura 25: Configuración del certificado en *AWS Certificate Manager*. Fuente: Elaboración propia.

Se debe remplazar por el dominio en cuestión. Luego, se deberá esperar unas horas hasta que la validación del dominio se haya realizado correctamente. A continuación, se debe regresar a la lista de certificados, se elige el recientemente solicitado y una vez dentro agregaremos los registros a *Route53* seleccionando el botón correspondiente.

PREPARACIÓN PARA LEVANTAR FRONT-END

El siguiente paso será preparar la configuración necesaria para levantar lo que sería el *frontend*. Para lograr esto, es necesario utilizar el servicio de *AWS S3*, *AWS Cloudfront* y *AWS Route53*, ya que no se utilizará un servidor tradicional.

Lo primero que se debe de hacer es ir al servicio de S3. Ahí se debe de crear un bucket utilizando como nombre el dominio, como se puede ver en la Figura 26.

Crear bucket

Los buckets son contenedores de datos almacenados en S3.

Configuración general

Nombre del bucket

example.com

El nombre del bucket debe ser único y no debe contener espacios ni letras mayúsculas.

Región de AWS

EE. UU. Este (Norte de Virginia) us-east-1

Copiar la configuración del bucket existente: opcional

Solo se copia la configuración del bucket en los siguientes ajustes.

Elegir el bucket

Propiedad de objetos

Controle la propiedad de los objetos escritos en este bucket desde otras cuentas de AWS y concedidos mediante listas de control de acceso (ACL). La propiedad del objeto determina quién puede especificar el acceso a los objetos.

ACL deshabilitadas (recomendado)

Todos los objetos de este bucket son propiedad de esta cuenta. El acceso a este bucket y sus objetos se especifica solo mediante políticas.

ACL habilitadas

Los objetos de este bucket pueden ser propiedad de otras cuentas de AWS. El acceso a este bucket y sus objetos se puede especificar mediante ACL.

Propiedad del objeto

Aplicada al propietario del bucket

Figura 26: Configuración del bucket inicial. Fuente: Elaboración propia.

Continuando con la configuración del bucket, se debe dejar como disponible para público, siguiendo el ejemplo de la Figura 27.

Configuración de bloqueo de acceso público para este bucket

Se concede acceso público a los buckets y objetos a través de listas de control de acceso (ACL), políticas de bucket, políticas de puntos de acceso o todas las anteriores. A fin de garantizar que se bloquee el acceso público a todos sus buckets y objetos, active Bloquear todo el acceso público. Esta configuración se aplica exclusivamente a este bucket y a sus puntos de acceso. AWS recomienda activar Bloquear todo el acceso público, pero, antes de aplicar cualquiera de estos ajustes, asegúrese de que las aplicaciones funcionarán correctamente sin acceso público. Si necesita cierto nivel de acceso público a los buckets u objetos, puede personalizar la configuración individual a continuación para adaptarla a sus casos de uso de almacenamiento específicos.

☐

Bloquear todo el acceso público

Activar esta configuración equivale a activar las cuatro opciones que aparecen a continuación. Cada uno de los siguientes ajustes son independientes entre sí.

☒

Bloquear el acceso público a buckets y objetos concedido a través de nuevas listas de control de acceso (ACL)

S3 bloqueará los permisos de acceso público aplicados a objetos o buckets agregados recientemente, y evitará la creación de nuevas ACL de acceso público para buckets y objetos existentes. Esta configuración no cambia los permisos existentes que permiten acceso público a los recursos de S3 mediante ACL.

☒

Bloquear el acceso público a buckets y objetos concedido a través de cualquier lista de control de acceso (ACL)

S3 ignorará todas las ACL que conceden acceso público a buckets y objetos.

☒

Bloquear el acceso público a buckets y objetos concedido a través de políticas de bucket y puntos de acceso públicas nuevas

S3 bloqueará las nuevas políticas de buckets y puntos de acceso que concedan acceso público a buckets y objetos. Esta configuración no afecta a las políticas ya existentes que permiten acceso público a los recursos de S3.

☐

Bloquear el acceso público y entre cuentas a buckets y objetos concedido a través de cualquier política de bucket y puntos de acceso pública

S3 ignorará el acceso público y entre cuentas en el caso de buckets o puntos de acceso que tengan políticas que concedan acceso público a buckets y objetos.

Desactivar el bloqueo de todo acceso público puede provocar que este bucket y los objetos que contiene se vuelvan públicos

AWS recomienda que active la opción para bloquear todo el acceso público, a menos que se requiera acceso público para casos de uso específicos y verificados, como el alojamiento de sitios web estáticos.

☒

Reconozco que la configuración actual puede provocar que este bucket y los objetos que contiene se vuelvan públicos.

Figura 27: Configuración de acceso del bucket. Fuente: Elaboración.

Finalmente, el resto de la configuración debe ser como el ejemplo mostrado en Figura 28.

Página 61 de 98

DISEÑO, DESARROLLO E IMPLEMENTACIÓN DE UN SISTEMA DE MONITOREO DE ASIGNATURAS BASADO EN PORTAFOLIOS DOCENTE

Control de versiones de buckets

El control de versiones es una forma de mantener múltiples variantes de un objeto dentro del mismo bucket. Puede utilizar el control de versiones para conservar, recuperar y restaurar todas las versiones de los objetos almacenados en su bucket de Amazon S3. Con el control de versiones, puede recuperarse con facilidad de las acciones involuntarias de los usuarios y de los errores en las aplicaciones.
[Más información](#)

Control de versiones de buckets

☒ Desactivar
☐ Habilitar

Etiquetas (0) - opcional

Para realizar un seguimiento del costo del almacenamiento u otros criterios, etiquete el bucket.
[Más información](#)

No hay etiquetas asociadas a este bucket.

Agregar etiqueta

Cifrado predeterminado

Cifre automáticamente los nuevos objetos almacenados en este bucket.
[Más información](#)

Cifrado del lado del servidor

☐ Desactivar
☒ Habilitar

Tipo de clave de cifrado

Para cargar un objeto con una clave de cifrado proporcionada por el cliente (SSE-C), utilice la CLI de AWS, el SDK de AWS o la API REST de Amazon S3.

☒ Clave de Amazon S3 (SSE-S3)

Una clave de cifrado que Amazon S3 crea, administra y utiliza por usted.
[Más información](#)

☐ Clave de AWS Key Management Service (SSE-KMS)

Una clave de cifrado protegida por AWS Key Management Service (AWS KMS).
[Más información](#)

▼ Configuración avanzada

Bloqueo de objetos

Almacene objetos mediante un modelo de escritura única, lectura múltiple (WORM, write-once-read-many) para que lo ayude evitar que se eliminen o sobrescriban objetos durante un periodo de tiempo fijo o de manera indefinida.
[Más información](#)

☒ Desactivar
☐ Habilitar

Permitir permanentemente bloquear los objetos de este bucket. Después de la creación del bucket, se requiere una configuración adicional de bloqueo de objetos en los detalles del bucket para proteger sus objetos y que no se eliminen o sobrescriban.

☒ El bloqueo de objetos solo funciona en buckets con control de versiones. Al habilitar el bloqueo de objetos, se habilita automáticamente el control de versiones de buckets.

Figura 28: Configuración final del bucket. Fuente: Elaboración propia.

Una vez ya creado el bucket, se debe ingresar y dirigirse al apartado de “propiedades” y al final, en el apartado de “alojamiento de sitios web estáticos”, se deberá de editar la configuración siguiendo el ejemplo mostrado en Figura 29.

Editar alojamiento de sitios web estáticos [Info](#)

Alojamiento de sitios web estáticos

Utilice este bucket para alojar un sitio web o redirigir solicitudes. [Más información](#)

Alojamiento de sitios web estáticos

☐ Desactivar

☒ Habilitar

Tipo de alojamiento

☒ Alojar un sitio web estático

Utilice el punto de enlace del bucket como dirección web. [Más información](#)

☐ Redirigir las solicitudes de un objeto

Redirija las solicitudes a otro bucket o dominio. [Más información](#)

Para que sus clientes puedan obtener acceso al contenido en el punto de enlace del sitio web, debe hacer que todo el contenido sea legible públicamente. Para ello, puede editar la configuración Bloquear acceso público de S3 del bucket. Para obtener más información, consulte [Utilizar Bloquear acceso público de Amazon S3](#)

Documento de índice

Especifique la página predeterminada o de inicio del sitio web.

Documento de error - *opcional*

Esto se devuelve cuando se produce un error.

Reglas de redireccionamiento: *opcionales*

Redirija las reglas, escritas en JSON, para redirigir automáticamente las solicitudes de páginas web de contenido específico. [Más información](#)

1

Figura 29: Configuración para alojar un sitio web estático en un *bucket*. Fuente: Elaboración propia.

Ya completado este apartado, hay que dirigirse al servicio de *CloudFront*. Este servicio corresponde a un *CDN*, el cual permite la entrega de data de forma segura, como aplicaciones, videos de forma global.

Dentro de *Cloudfront*, específicamente en el listado de distribuciones, se debe crear una nueva.

La primera parte consta de la configuración de *Origin*, donde se deberá de seleccionar el bucket que se creó anteriormente, se debe guiar por el dominio. En la parte de *Origin Access Identity* se debe crear una nueva y asignarla, siguiendo el ejemplo mostrado en Figura 30.

Página 63 de 98

Origin

Origin domain

Choose an AWS origin, or enter your origin's domain name.

Q

X

Origin path - optional

Info

Enter a URL path to append to the origin domain name for origin requests.

Enter the origin path

Name

Enter a name for this origin.

S3 bucket access

Info

Use a CloudFront origin access identity (OAI) to access the S3 bucket.

Don't use OAI (bucket must allow public access)

Yes use OAI (bucket can restrict access to only CloudFront)

Origin access identity

Select an existing origin access identity (recommended) or create a new identity.

Create new OAI

Bucket policy

Update the S3 bucket policy to allow read access to the OAI.

No, I will update the bucket policy

Yes, update the bucket policy

Add custom header - optional

CloudFront includes this header in all requests that it sends to your origin.

Add header

Enable Origin Shield

Info

Origin Shield is an additional caching layer that can help reduce the load on your origin and help protect its availability.

No

Yes

Additional settings

Figura 30: Configuración del apartado *Origin* de una distribución de Cloudfront. Fuente: Elaboración propia.

En el siguiente apartado se configurará el tratamiento del caché. Se debe seguir el esquema mostrado en la Figura 31.

Default cache behavior

Path pattern [Info](#)

Default (*)

Compress objects automatically [Info](#)

No

Yes

Viewer

Viewer protocol policy

HTTP and HTTPS

Redirect HTTP to HTTPS

HTTPS only

Allowed HTTP methods

GET, HEAD

GET, HEAD, OPTIONS

GET, HEAD, OPTIONS, PUT, POST, PATCH, DELETE

Cache HTTP methods

GET and HEAD methods are cached by default.

OPTIONS

Restrict viewer access

If you restrict viewer access, viewers must use CloudFront signed URLs or signed cookies to access your content.

No

Yes

Cache key and origin requests

We recommend using a cache policy and origin request policy to control the cache key and origin requests.

Cache policy and origin request policy (recommended)

Legacy cache settings

Servicios

Buscar servicios, características, blogs, documentos y mucho más [Alt+S]

CachingDisabled

Policy with caching disabled

Create policy View policy

Origin request policy - optional

Choose an existing origin request policy or create a new one.

Select origin policy

Create policy

Response headers policy - optional

Choose an existing response headers policy or create a new one.

Select response headers

Create policy

Additional settings

Smooth streaming

Choose No if your origin is configured to use Microsoft IIS for Smooth Streaming.

No

Yes

Field-level encryption [Info](#)

Choose a field-level encryption configuration.

Select a field-level encryption profile

Enable real-time logs [Info](#)

No

Yes

Figura 31: Configuración del caché en una distribución. Fuente: Elaboración propia.

Continuando, el apartado de *Function associations* debe dejarse igual y finalmente en el apartado de *Settings* lo importante es elegir el certificado que se solicitó anteriormente y agregar un subdominio en el apartado de *Alternate domain name*, se deja un ejemplo en la Figura 32.

Página 65 de 98

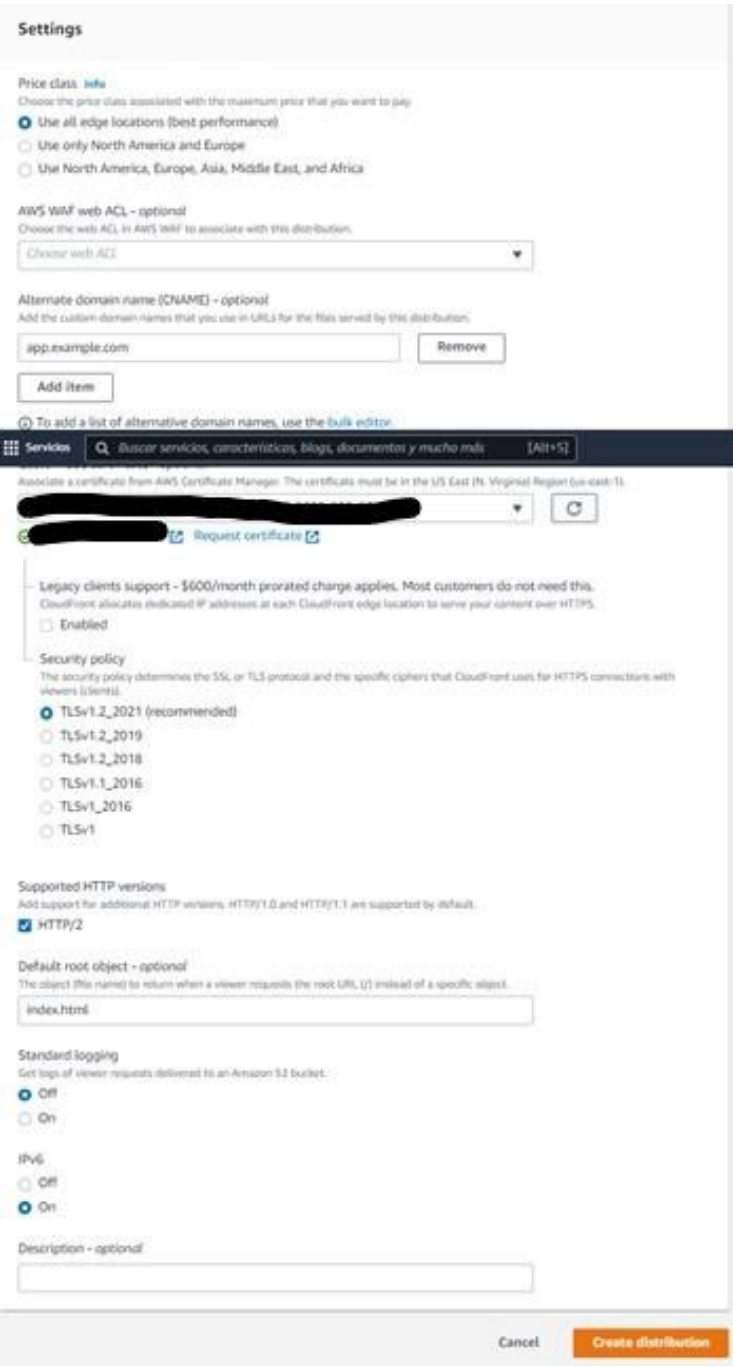


Figura 32: Configuración final de distribución de *Cloudfront*. Fuente: Elaboración propia.

Ya creado, es necesario realizar unas configuraciones extras. Dentro de la distribución hay que dirigirse al apartado de *Error pages* y agregar tres páginas de error correspondientes a los errores 400, 403 y 404. Es necesario basarse en el ejemplo mostrado en Figura 33.

The screenshot shows the 'Create custom error response' form in the AWS CloudFront console. The form is titled 'Create custom error response' and has a sub-header 'Error response Info'. It contains the following fields and options:

- HTTP error code:** A dropdown menu with '400: Bad Request' selected. Below it is a description: 'Customize the custom error response when the origin sends this error code.'
- Error caching minimum TTL:** A text input field with '10' entered. Below it is a description: 'Enter the error caching minimum time to live (TTL), in seconds.'
- Customize error response:** Radio buttons for 'No' and 'Yes'. 'Yes' is selected. Below it is a description: 'Send a custom error response instead of the error received from the origin.'
- Response page path:** A text input field with '/index.html' entered. Below it is a description: 'Enter the path to the custom error response page.'
- HTTP Response code:** A dropdown menu with '400: Bad Request' selected. Below it is a description: 'Choose the HTTP status code to return to the viewer. CloudFront can return a different status code to the viewer than what it received from the origin.'

At the bottom right, there are two buttons: 'Cancel' and 'Create custom error response'.

Figura 33: Ejemplo de página de error 400 en una distribución. Fuente: Elaboración propia.

Luego de crear las páginas de error, hay que ir al apartado de *Geographic restrictions*, se debe de editar y agregar una lista de acceso permitido y agregamos a Chile, como se muestra en la Figura 34.

The screenshot shows the 'Edit geographic restrictions' form in the AWS CloudFront console. The form is titled 'Edit geographic restrictions' and has a sub-header 'Settings Info'. It contains the following fields and options:

- Restriction type:** Radio buttons for 'No restrictions', 'Allow list', and 'Block list'. 'Allow list' is selected.
- Countries:** A dropdown menu with 'Select countries' selected. Below it is a list of countries, with 'Chile' highlighted.

At the bottom right, there are two buttons: 'Cancel' and 'Save changes'.

Figura 34: Lista de acceso permitido de una distribución de CloudFront. Fuente: Elaboración propia.

Finalmente, hay que dirigirse a *Route53* y agregar un nuevo subdominio a la zona alojada creada anteriormente, para que pueda ser vinculado a la distribución de CloudFront. Es importante copiar el nombre de dominio que aparece en el listado de distribuciones, pues será necesario para la vinculación. El formulario a completar al mostrado en Figura 35.

Registro de creación rápida [Info](#) [Cambiar al asistente](#)

▼ Registro 1 [Eliminar](#)

Nombre del registro [Info](#) [Redacted]

Caracteres válidos: a-z, 0-9, ! * # \$ % & ' () * + , - / : ; < = > ? @ [\] ^ _ ` { | } . ~

Tipo de registro [Info](#)

Dirigir el tráfico a [Info](#) ☒ Alias

Alias de la distribución de CloudFront

EE. UU. Este (Norte de Virginia)

El alias de una distribución de CloudFront y el alias de otro registro de la misma zona alojada son globales y solo están disponibles en EE. UU. Este (Norte de Virginia).

[Redacted] ×

Política de direccionamiento [Info](#)

Evaluar el estado del destino ☐ No

[Agregar otro registro](#)

[Cancelar](#) [Crear registros](#)

Figura 35: Formulario para agregar un nuevo subdominio. Fuente: Elaboración propia.

Es importante que el nombre del registro sea igual al subdominio configurado en la distribución de *CloudFront*. El nombre de dominio copiado anteriormente se debe poner en el apartado para dirigir el tráfico.

ELASTIC LOAD BALANCING

Antes de pasar a configurar lo que sería el *backend*, es importante configurar el balanceador de carga para poder utilizar el certificado *SSL/TLS* y el dominio.

Para realizar esta tarea es necesario ir al servicio de *EC2*. Ahí se deberá de configurar los grupos de seguridad, grupos de destino y el balanceador de carga.

En primera instancia, desde el menú lateral se debe ir al apartado de los grupos de seguridad, luego se debe crear uno nuevo, siguiendo el ejemplo mostrado en Figura 36.

Crear grupo de seguridad

Información

Un grupo de seguridad actúa como un firewall virtual para que la instancia controle el tráfico de entrada y salida. Para crear un nuevo grupo de seguridad, complete los campos siguientes.

Detalles básicos

Nombre del grupo de seguridad

Información

mi-grupo-de-seguridad

El nombre no se puede editar después de su creación.

Descripción

Información

Permite acceso a la plataforma de monitoreo docente UTFSM.

VPC

Información

Q vpc-d7a9d5ad

X

Reglas de entrada

Información

Tipo	Protocolo	Intervalo de puertos	Origen	Descripción: opcional
Información	Información	Información	Información	Información
HTTP	TCP	80	Anywhere-IPv4	
			Q 0.0.0.0/0	Eliminar
HTTPS	TCP	443	Anywhere-IPv4	
			Q 0.0.0.0/0	Eliminar
TCP personalizado	TCP	5000 - 8000	Anywhere-IPv4	
			Q 0.0.0.0/0	Eliminar
Agregar regla				

Reglas de salida

Información

Tipo	Protocolo	Intervalo de puertos	Destino	Descripción: opcional
Información	Información	Información	Información	Información
Todo el tráfico	Todo	Todo	Personalizada	
			Q 0.0.0.0/0	Eliminar
Agregar regla				

Figura 36: Configuración del grupo de seguridad. Fuente: Elaboración propia.

Por defecto, AWS crea una *VPC* pública en la cuenta, la cual es asignada por defecto en el formulario. Los puertos *TCP* puede ser establecido en rangos o indicando sólo un puerto, El puerto o rango de puertos *TCP* debe contener el puerto del servicio del backend que más tarde será configurado.

A continuación, se configura el grupo de destino. Ya que el servidor no tendrá una *IP* fija, sino que funcionará en un rango de *IPs*, es necesario crear un grupo el cual corresponderá a un rango de posibles *IPs* a partir de una *VPC*, que en este caso correspondería a la generada por defecto. Para hacer esto, es necesario ir al apartado de grupos de destino. Dentro, se deberá de crear un nuevo grupo destino, el cual consta de tres partes. Para la primera parte guiarse por el ejemplo mostrado en Figura 37.

Instances

- Supports load balancing to instances within a specific VPC.

IP addresses

- Supports load balancing to VPC and on-premises resources.
- Facilitates routing to multiple IP addresses and network interfaces on the same instance.
- Offers flexibility with microservice based architectures, simplifying inter-application communication.

Lambda function

- Facilitates routing to a single Lambda function.
- Accessible to Application Load Balancers only.

Application Load Balancer

- Offers the flexibility for a Network Load Balancer to accept and route TCP requests within a specific VPC.
- Facilitates using static IP addresses and PrivateLink with an Application Load Balancer.

Target group name

mi-grupo-destino

A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.

Protocol

Port

HTTP

:

80

IP address type

Only targets with the indicated IP address type can be included in this target group.

IPv4

IPv6

VPC

Select the VPC that hosts the load balancer. Only VPCs that support the IP address type selected above are available in this list. On the **Register targets** page, you can register IP addresses from this VPC, or from private IP addresses located outside of this load balancer's VPC (such as a peered VPC, EC2-Classic, or on-premises targets that are reachable over Direct Connect or VPN).

-

vpc-d7a9d5ad

IPv4: 172.31.0.0/16

Protocol version

HTTP1

HTTP2

gRPC

Send requests to targets using HTTP/1.1. Supported when the request protocol is HTTP/1.1 or HTTP/2.

Send requests to targets using HTTP/2. Supported when the request protocol is HTTP/2 or gRPC, but gRPC-specific features are not available.

Send requests to targets using gRPC. Supported when the request protocol is gRPC.

Figura 37: Configuración de la parte uno del grupo de destino. Fuente: Elaboración propia.

Para la segunda parte, configuración debiese ser como la mostrada en Figura 38.

Health checks

The associated load balancer periodically sends requests, per the settings below, to the registered targets to test their status.

Health check protocol

HTTP

Health check path

Use the default path of "/" to ping the root, or specify a custom path if preferred.

/

Up to 1024 characters allowed.

Advanced health check settings

Restore defaults

Port

The port the load balancer uses when performing health checks on targets. The default is the port on which each target receives traffic from the load balancer, but you can specify a different port.

Traffic port

Override

Healthy threshold

The number of consecutive health checks successes required before considering an unhealthy target healthy.

10

2-10

Unhealthy threshold

The number of consecutive health check failures required before considering a target unhealthy.

10

2-10

Timeout

The amount of time, in seconds, during which no response means a failed health check.

30

seconds

2-120

Interval

The approximate amount of time between health checks of an individual target

60

seconds

5-300

Success codes

The HTTP codes to use when checking for a successful response from a target. You can specify multiple values (for example, "200,202") or a range of values (for example, "200-299").

200

Tags - optional

Consider adding tags to your target group. Tags enable you to categorize your AWS resources so you can more easily manage them.

Cancel

Next

Figura 38: Configuración de la parte dos del grupo de destino. Fuente: Elaboración propia.

Para la parte tres, basta con dejarla como aparece, pero es importante que, si se tiene más de una VPC, que la seleccionada sea la misma que la del grupo de seguridad, que en este caso es la por defecto. Para guiarse se recomienda basarse en el ejemplo mostrado en Figura 39.

Página 71 de 98

Register targets

This is an optional step to create a target group. However, to ensure that your load balancer routes traffic to this target group you must register your targets.

IP addresses

Step 1: Choose a network

You can add IP addresses from the VPC selected for your target group or from outside the VPC. Note that you can assemble a mix of targets from multiple network sources by returning to this step and choosing another network.

Network

-

vpc-df-adf5cd

IPv4: 172.31.0.0/16

Step 2: Specify IPs and define ports

You can manually enter IP addresses from the selected network.

IPv4 address

172.31.0.

Remove

Add IPv4 address

You can add up to 4 more IP addresses.

Ports

Ports for routing to this target.

80

1-65535 (separate multiple ports with comma)

Include as pending below

Review targets

Step 3: Review IP targets to include in your group

Confirm the IP targets to include in your target group. Add more IP targets by repeating steps 1 and 2 on this page. You can also register additional targets after your target group is created.

Targets (0)

All

Filter resources by property or value

<

1

>

⚙

Remove IPv4 address

Health status

IP address

Port

Zone

No IP addresses included yet

Specify IP addresses above and add to list.

0 pending

Cancel

Previous

Create target group

Figura 39: : Configuración final del grupo de destino. Fuente: Elaboración propia.

A continuación, se procede a construir el balanceador de carga. En AWS existen 3 tipos de balanceadores de carga (ver Figura 40) y según el caso de uso se deberá de elegir el adecuado.

Página 72 de 98

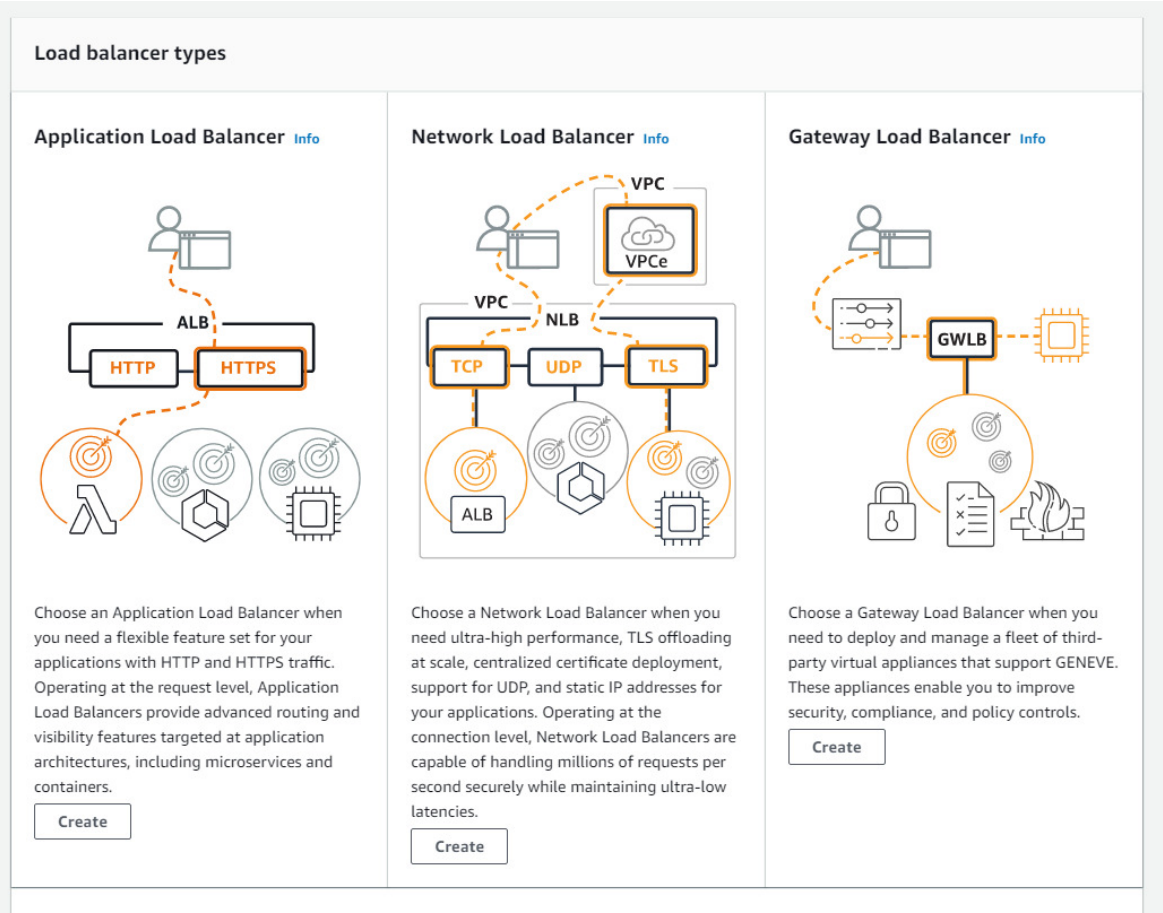


Figura 40: Tipos de balanceadores de carga de AWS. Fuente: AWS

Para configurar correctamente la arquitectura pensada, es necesario elegir el balanceador de carga de tipo aplicación. El primer apartado a completar corresponde a la información básica (ver Figura 41). En este es importante configurar el nombre, que el esquema corresponda a *Internet facing* y el tipo de *IP* sea *v4*.

The screenshot shows the 'Basic configuration' section of the AWS console. It includes the following fields and options:

- Load balancer name:** A text input field with a note: 'Name must be unique within your AWS account and cannot be changed after the load balancer is created.' Below the field, it states: 'A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.'
- Scheme:** A dropdown menu with 'Internet-facing' selected. A note says: 'Scheme cannot be changed after the load balancer is created.' Below the dropdown, it explains: 'An internet-facing load balancer routes requests from clients over the internet to targets. Requires a public subnet. [Learn more](#)'
- IP address type:** A dropdown menu with 'IPv4' selected. A note says: 'Select the type of IP addresses that your subnets use.' Below the dropdown, it explains: 'Recommended for internal load balancers.' Below this, there is an option for 'Dualstack' with the note: 'Includes IPv4 and IPv6 addresses.'

Figura 41: Configuración básica del balanceador de carga. Fuente: AWS.

El siguiente paso corresponde al mapeo de red. Es importante que la *VPC* a elegir sea la misma que la elegida en el grupo de destino. Además, ya que se eligió anteriormente un *ELB* de tipo aplicación, es necesario que se seleccione 2 o más *subnets*. La siguiente figura muestra un ejemplo.

Network mapping [Info](#)

The load balancer routes traffic to targets in the selected subnets, and in accordance with your IP address settings.

VPC [Info](#)

Select the virtual private cloud (VPC) for your targets. Only VPCs with an internet gateway are enabled for selection. The selected VPC cannot be changed after the load balancer is created. To confirm the VPC for your targets, view your [target groups](#).

-

vpc-d7a9d5ad

IPv4: 172.31.0.0/16

Mappings [Info](#)

Select at least two Availability Zones and one subnet per zone. The load balancer routes traffic to targets in these Availability Zones only. Availability Zones that are not supported by the load balancer or the VPC are not available for selection. Subnets cannot be removed after the load balancer is created, but additional subnets can be added.

☒ **us-east-1a**

Subnet

subnet-847f53d8

IPv4 settings

Assigned by AWS

☒ **us-east-1b**

Subnet

subnet-3cc9e95b

IPv4 settings

Assigned by AWS

Figura 42: Etapa de mapeo de red en un balanceador de carga. Fuente: AWS.

Luego es necesario configurar el grupo de seguridad y elegir el creado anteriormente, así como se visualiza en Figura 43.

Security groups [Info](#)

A security group is a set of firewall rules that control the traffic to your load balancer.

Security groups

Select security groups

Create new security group [Info](#)

default sg-8d33dbd2 X

VPC: vpc-d7a9d5ad

Figura 43: Configuración del grupo de seguridad. Fuente: AWS.

Finalmente, para poder concluir con la creación de un balanceador de carga, es necesario configurar los agentes de escuchas, ruteo y el certificado. Para poder usar *HTTPS* es necesario configurar el puerto 443 y elegir el grupo de destino correspondiente, el cual corresponderá al backend. Para guiarse se recomienda basarse en el ejemplo mostrado en Figura 44.

Listeners and routing

Info

A listener is a process that checks for connection requests, using the protocol and port you configure. Traffic received by the listener is then routed per your specification. You can specify multiple rules and multiple certificates per listener after the load balancer is created.

▼ Listener HTTPS:443

Remove

Protocol

Port

Default action

Info

HTTPS

:

443

Forward to

mi-grupo-destino

HTTP

1-65535

Target type: IP, IPv4

Create target group

Add listener

Secure listener settings

Info

These settings will apply to all of your secure listeners. Once created, you can manage these settings per listener if desired.

Security policy

Your load balancer uses a Secure Socket Layer (SSL) negotiation configuration, known as a security policy, to negotiate SSL connections with clients.

ELBSecurityPolicy-FS-1-2-Res-2020-10

Compare security policies

Default SSL certificate

The certificate used if a client connects without SNI protocol, or if there are no matching certificates. You can add more certificates after you create the load balancer.

From ACM

portafolio-inf-usm.me

Request new ACM certificate

Figura 44: Configuración final del balanceador de carga. Fuente: AWS.

Una vez completados los datos requeridos, se da comienzo a la configuración para levantar el backend de la plataforma. Para llevar esto a cabo, se utiliza *AWS ECS*. Este servicio, llamado *Elastic Container Service*, facilita la administración de contenedores de manera rápida y escalable.

BACKEND

Para levantar el backend es necesario utilizar *AWS ECS*. En este servicio se deberán de configurar tres apartados: *ECR*, *Task Definitions* y el clúster.

El primero corresponde a *Elastic Container Registry*, un servicio donde se pueden gestionar las imágenes de *Docker* que contienen el código del microservicio empaquetado para ser ejecutado como una tarea o aplicación.

Una *Task Definition* no es más que la configuración de lo que sería la tarea que correspondería al microservicio. En ella se configura los puertos a usar, los recursos a asignar, las variables de entorno, etc.

Finalmente, el clúster corresponde al lugar donde la aplicación se alojará, este se encuentra asociado a una *VPC*. Dentro se pueden configurar servicios, los cuales haciendo uso de las *Tasks Definitions*, son los encargados de gestionar el funcionamiento de los microservicios corriendo y de mantener siempre corriendo las aplicaciones. En ellos además se pueden configurar el auto escalamiento y la vinculación al balanceador de carga.

Página 75 de 98

VARIABLES DE ENTORNO

En primera instancia, es necesario configurar las variables de entorno. Estas son necesarias para poder correr el servicio del *backend* correctamente. Por lo tanto, se debe crear un archivo *.env* y dentro especificar las variables detalladas en Tabla 7.

Variable	Descripción
AWS_ACCESS_KEY_ID	Corresponde al identificador de acceso de la cuenta de <i>AWS</i> .
AWS_SECRET_ACCESS_KEY	Corresponde a la llave secreta de la cuenta de <i>AWS</i> .
BUCKET_S3	Nombre del bucket de <i>S3</i> donde se guardarán los archivos de los portafolios.
PORT	Puerto donde correrá la aplicación.
API_BRIEFCASE	<i>URL</i> del microservicio. Ej: https://a.example.com
NODE_ENV	Tipo de entorno: <i>development</i> o <i>production</i>
DB	Dirección de la base de datos de MongoDB. Suele ser una <i>URI</i> .
KEYCLOAK_AUDIENCE	<i>AUDIENCE</i> de <i>keycloak</i> , necesario para validar los <i>access tokens</i> .
KEYCLOAK_ISSUER	<i>ISSUER</i> de <i>keycloak</i> , necesario para validar los <i>access tokens</i> .
KEYCLOAK_JWKS_URI	<i>URI</i> donde se encuentra el conjunto de llaves públicas para verificar <i>JWT tokens</i> usando el algoritmo <i>RS256</i> o <i>RS512</i> .

Tabla 7: Variables de entorno a generar. Fuente: Elaboración propia.

Una vez generado este archivo de entorno, es necesario ir al servicio de *S3* y crear un nuevo *bucket*, este tiene que ser completamente privado. Una vez creado, se carga este archivo

de entorno. Finalmente, se deberá de copiar el *S3 ARN* que se generó del archivo para usarlo posteriormente.

ELASTIC CONTAINER REGISTRY

Luego se pasa a crear una imagen donde guardar los archivos generados al compilar la aplicación. Para llevar a cabo esto, hay que dirigirse al servicio de *AWS ECS* y en el menú lateral elegir *AWS ECR*. Dentro deberían estar la lista de repositorios privados y públicos. A continuación, se debe crear un repositorio privado. Dentro, se debe especificar el nombre y activar la encriptación *KMS* (ver Figura 45).

General settings

Visibility settings [Info](#)

Choose the visibility setting for the repository.

☒ Private

Access is managed by IAM and repository policy permissions.

☐ Public

Publicly visible and accessible for image pulls.

Repository name

Provide a concise name. A developer should be able to identify the repository contents by the name.

597176373959.dkr.ecr.us-east-1.amazonaws.com/

0 out of 256 characters maximum (2 minimum). The name must start with a letter and can only contain lowercase letters, numbers, hyphens, underscores, and forward slashes.

Tag immutability [Info](#)

Enable tag immutability to prevent image tags from being overwritten by subsequent image pushes using the same tag. Disable tag immutability to allow image tags to be overwritten.

☐ Disabled

 Once a repository is created, the visibility setting of the repository can't be changed.

Image scan settings

 **Deprecation warning**

ScanOnPush configuration at the repository level is deprecated in favor of registry level scan filters.

Scan on push

Enable scan on push to have each image automatically scanned after being pushed to a repository. If disabled, each image scan must be manually started to get scan results.

☐ Disabled

Encryption settings

KMS encryption

You can use AWS Key Management Service (KMS) to encrypt images stored in this repository. Instead of using the default encryption settings.

☒ Enabled

 The KMS encryption settings cannot be changed or disabled after the repository is created.

Figura 45: Configuración para crear un nuevo repositorio privado. Fuente: Elaboración propia.

TASK DEFINITION

Luego, es necesario configurar lo que sería la definición de la tarea. Usando el menú lateral, hay que dirigirse al listado de las definiciones de tareas, ahí se debe crear una nueva.

Como se ve en la Figura 46, inicialmente se debe completar el nombre de la tarea, el nombre del contenedor y la *URI* de la imagen del repositorio creado usando *AWS ECR* (para obtenerla basta con ir al listado y copiarlo de la columna correspondiente). Luego, se debe agregar el puerto, el cual debe ser igual al configurado por la aplicación (este será el configurado en la variable de entorno). Continuando, se debe agregar la ubicación del archivo de entorno, este corresponde al paso realizado anteriormente.

Configuración de definición de tareas

Familia de definición de tareas [Info](#)

Especifique un nombre de familia de definición de tarea único.

Se permiten hasta 255 letras (mayúsculas y minúsculas), números, guiones y guiones bajos.

Contenedor: 1 [Info](#)

Contenedor esencial Eliminar

Detalles del contenedor

Especifique un nombre, una imagen de contenedor y si el contenedor debe marcarse como esencial. Cada definición de tarea debe tener al menos un contenedor esencial.

Nombre

URI de imagen

Contenedor esencial

Sí

Mapeos de puertos [Info](#)

Agregue mapeos de puertos para permitir que el contenedor obtenga acceso a los puertos del anfitrión para enviar o recibir tráfico.

Puerto del contenedor

Protocolo

TCP

Eliminar

Agregar más mapeos de puertos

▼ Variables de entorno: opcional [Info](#)

Agregar individualmente

Agregue un par clave-valor para especificar una variable de entorno.

Agregar

Agregar desde archivo

Agregue variables de entorno de forma masiva al proporcionar un archivo de entorno alojado en Amazon S3.

Ubicación

Eliminar

Agregar

Puede agregar 9 archivos más de entorno.

Figura 46: Configuración inicial de una definición de tarea. Fuente: Elaboración propia.

Página 78 de 98

El siguiente paso será configurar el entorno a utilizar, entre las opciones de entorno la seleccionada será *Fargate*. Esto permite que sea autoadministrable y de rápida implementación. En cuanto a la potencia a necesitar dependerá del tamaño del servicio del *backend* y las exigencias que requiera. La Figura 47 muestra la configuración probada.

▼ **Entorno**
Especifique los requisitos de infraestructura para la definición de tarea.

Entorno de la aplicación [Info](#)
Especifique la infraestructura para la definición de tarea.

Agregar una opción ▼

AWS Fargate (sin servidor) X

Sistema operativo/arquitectura [Info](#)
Linux/X86_64 ▼

Tamaño de la tarea [Info](#)
Especifique la cantidad de CPU y memoria que reservar para su tarea.

CPU Memoria

.5 vCPU ▼ 1 GB ▼

► Tamaño del contenedor - opcional [Info](#)

▼ Roles de tarea, modo de red - condicional

Rol de tarea [Info](#)
Un rol de IAM de tarea permite a los contenedores de la tarea realizar solicitudes de API a los servicios de AWS. Puede crear un rol de IAM de tarea desde la [consola de IAM](#).

ecsTaskExecutionRole ▼

Modo de red [Info](#)
Modo de red que se va a utilizar para la tarea. De forma predeterminada, el modo de red awsvpc está seleccionado, lo que brinda a las tareas su propia interfaz de red elástica (ENI).

awsvpc ▼

Figura 47: Configuración del entorno de una definición de tarea. Fuente: Elaboración propia.

CLÚSTER

Finalmente, se da paso a crear el clúster donde el servicio del *backend* estará corriendo. Para llevar a cabo esto, hay que dirigirse al apartado de los clústeres, y se crea uno. Los datos que se deben rellenar son el nombre y la configuración de la *VPC*. Es importante que la *VPC* seleccionada sea la misma que la elegida en el balanceador de carga. El formulario que se debe completar es el mostrado en Figura 48.

Crear clúster

Un clúster de Amazon ECS es una agrupación lógica de tareas o servicios.

Configuración del clúster

Nombre del clúster

Nombre del clúster aquí. Por ejemplo, DevCluster.

Puede haber un máximo de 255 caracteres. Los caracteres válidos son letras (mayúsculas y minúsculas), números, guiones y guiones bajos.

▼ Redes Info

De forma predeterminada, las tareas y los servicios se ejecutan en las subredes predeterminadas de la VPC predeterminada. Para utilizar una VPC no predeterminada, especifique la VPC y las subredes.

VPC

Utilice una VPC con subredes públicas y privadas. De forma predeterminada, las VPC se crean para su cuenta de AWS. Para crear una nueva VPC, vaya a la [consola de VPC](#).

vpc-d7a9d5ad

valor predeterminado

Subredes - opcionales

Seleccione las subredes en las que se ejecutan las tareas. Le recomendamos que utilice tres subredes para producción.

Elegir subredes

subnet-94b340d9

us-east-1d

×

subnet-7e644a50

us-east-1c

×

subnet-c21166fc

us-east-1e

×

subnet-847f53d8

us-east-1a

×

subnet-3cc9e95b

us-east-1b

×

subnet-dab6acd5

us-east-1f

×

Figura 48: Creación de clúster. Fuente AWS.

Una vez creado, dentro del clúster se podrá administrar las tareas y servicios (ver Figura 49). Como se mencionó anteriormente, las tareas son las instancias de las aplicaciones que están corriendo, suelen ejecutar una cierta secuencia de pasos y luego apagarse, en cambio, un servicio permite gestionar varias tareas que deben estar corriendo siempre, permitiendo el auto-escalamiento y balanceo de carga, ya que se vincula a un balanceador de carga. A continuación, se debe crear un servicio. Inicialmente se debe configurar el entorno de este, el cual debe ser *Fargate*. Es recomendable que para el apartado de crear el servicio se utilice la consola antigua, pues a la fecha de la creación de esta memoria, aún tiene errores.

DISEÑO, DESARROLLO E IMPLEMENTACIÓN DE UN SISTEMA DE MONITOREO DE ASIGNATURAS BASADO EN PORTAFOLIOS DOCENTE

Configure service

A service lets you specify how many copies of your task definition to run and maintain in a cluster. You can optionally use an Elastic Load Balancing load balancer to distribute incoming traffic to containers in your service. Amazon ECS maintains that number of tasks and coordinates task scheduling with the load balancer. You can also optionally use Service Auto Scaling to adjust the number of tasks in your service.

Launch type

☒ FARGATE

☐ EC2

☐ EXTERNAL

Switch to capacity provider strategy

Operating system family

Linux

Task Definition

Family

briefcase-task

Enter a value

Revision

1 (latest)

Platform version

LATEST

Cluster

portfolio-docente-prod

Service name

service-briefcase

Service type*

REPLICA

Number of tasks

1

Minimum healthy percent

100

Maximum percent

200

Deployment circuit breaker

Enabled with rollback

Deployments

Choose a deployment option for the service.

Deployment type*

☒ Rolling update

☐ Blue/green deployment (powered by AWS CodeDeploy)

This sets AWS CodeDeploy as the deployment controller for the service. A CodeDeploy application and deployment group are created automatically with default settings for the service. To change to the rolling update deployment type after the service has been created, you must re-create the service and select the "rolling update" deployment type.

Task tagging configuration

☒ Enable ECS managed tags

Propagate tags from

Service

Tags

Key

Add key

Value

Add value

*Required

Cancel

Next step

Figura 49: Configuración del entorno de un servicio. Fuente: AWS.

Luego se debe realizar la configuración de red, donde se deberá seleccionar la misma VPC elegida en el balanceador de carga y las subredes correspondientes. Además, de elegir el grupo de seguridad creado anteriormente (ver Figura 50).

Página 81 de 98

Configure network

VPC and security groups

VPC and security groups are configurable when your task definition uses the awsvpc network mode.

Cluster VPC*

vpc-d7a9d5ad (172.31.0.0/16)

Subnets*

subnet-847f53d8
(172.31.32.0/20) - us-east-1a
assign ipv6 on creation: Disabled

subnet-3cc9e95b
(172.31.0.0/20) - us-east-1b
assign ipv6 on creation: Disabled

subnet-dab6acd5
(172.31.48.0/20) - us-east-1f
assign ipv6 on creation: Disabled

subnet-7e644a50
(172.31.80.0/20) - us-east-1c
assign ipv6 on creation: Disabled

subnet-94b340d9
(172.31.16.0/20) - us-east-1d
assign ipv6 on creation: Disabled

subnet-c21166fc
(172.31.64.0/20) - us-east-1e
assign ipv6 on creation: Disabled

Security groups*

servic-1828

Edit

Auto-assign public IP

ENABLED

Figura 50: Configuración de red de un servicio. Fuente: AWS.

Y luego se deberá vincular al balanceador de carga creado anteriormente, donde una vez vinculado, se deberá seleccionar el contenedor a vincular a este y luego seleccionar el grupo destino creado anteriormente. A continuación, se deja al lector un ejemplo en la Figura 51.

Health check grace period

If your service's tasks take a while to start and respond to ELB health checks, you can specify a health check grace period of up to 2,147,483,647 seconds during which the ECS service scheduler will ignore ELB health check status. This grace period can prevent the ECS service scheduler from marking tasks as unhealthy and stopping them before they have time to come up. This is only valid if your service is configured to use a load balancer.

Health check grace period

60

Load balancing

An Elastic Load Balancing load balancer distributes incoming traffic across the tasks running in your service. Choose an existing load balancer, or create a new one in the [Amazon EC2 console](#).

Load balancer type*

None

Your service will not use a load balancer.

Application Load Balancer

Allows containers to use dynamic host port mapping (multiple tasks allowed per container instance). Multiple services can use the same listener port on a single load balancer with rule-based routing and paths.

Network Load Balancer

A Network Load Balancer functions at the fourth layer of the Open Systems Interconnection (OSI) model. After the load balancer receives a request, it selects a target from the target group for the default rule using a flow hash routing algorithm.

Classic Load Balancer

Requires static host port mappings (only one task allowed per container instance); rule-based routing and paths are not supported.

Service IAM role

Task definitions that use the awsvpc network mode use the AWSServiceRoleForECS service-linked role, which is created for you automatically. [Learn more.](#)

Load balancer name

plataforma-docente-elb

Container to load balance

briefcase-task : 4800

Remove X

Production listener port*

443:HTTPS

Production listener protocol*

HTTPS

Target group name

briefcases-lq-2

Target group protocol

HTTP

Target type

ip

Path pattern

/

Evaluation order

default

Health check path

/

Additional health check options can be configured in the ELB console after you create your service.

Figura 51: Configuración y vinculación del balanceador de carga al servicio. Fuente AWS.

Página 83 de 98

El siguiente apartado correspondiente al auto-escalamiento se puede omitir ya que, para los efectos prácticos, no es relevante. Una vez completado los pasos anteriores se pasa a la creación y debería estar disponible en el listado.

SUBDOMINIO

Para completar la configuración del *backend*, es necesario crearle un subdominio en *Route43* para que pueda ser accedido por el cliente. Por tanto, hay que dirigirse al servicio mencionado y crear un nuevo registro en la zona alojada correspondiente. En él se debe seleccionar el balanceador de carga para que el tráfico sea redirigido a este (ver Figura 52).

Registro de creación rápida

Info

Cambiar al asistente

▼ Registro 1

Eliminar

Nombre del registro

Info

briefcase

.portafolio-inf-usm.me

Tipo de registro

Info

A: dirige el tráfico a una dirección IPv4 y a al...

Dirigir el tráfico a

Info

Alias

Alias del Classic Load Balancer y el de aplica...

EE. UU. Este (Norte de Virginia) [us-east-1]

Q dualstack.plataforma-docente-elb-16158 X

Caracteres válidos: a-z, 0-9, ! * # \$ % & ' () * + , - / : ; < = > ? @ [\] ^ _ ` { | } . ~

Política de direccionamiento

Info

Direccionamiento sencillo

Evaluar el estado del destino

Sí

Agregar otro registro

Cancelar

Crear registros

Figura 52: Creación subdominio del *backend*. Fuente: Elaboración propia.

Con todo lo realizado anteriormente, debería estar la arquitectura montada y lista para el montaje de la plataforma.

BASE DE DATOS

La base de datos a utilizar es *MongoDB*, gracias a su flexibilidad y alta escalabilidad. Para utilizarla se hace uso de un servicio externo llamado MongoDB Atlas. Por lo tanto, es necesario crearse una cuenta en este servicio. Una vez creada, lo primero a generar será un clúster. Para efectos prácticos se utilizará la versión gratuita. Es necesario que sea de tipo *shared* y el *cluster tier* debe ser *M0* (ver Figuras 53 y 54).

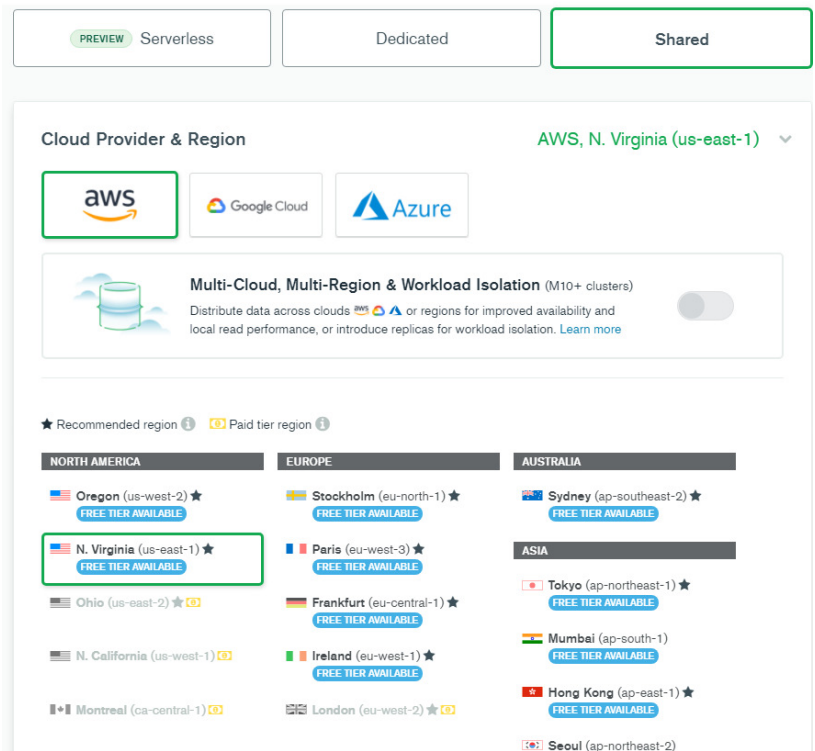


Figura 53: Selección del tipo de clúster en MongoDB Atlas. Fuente: MongoDB Atlas.

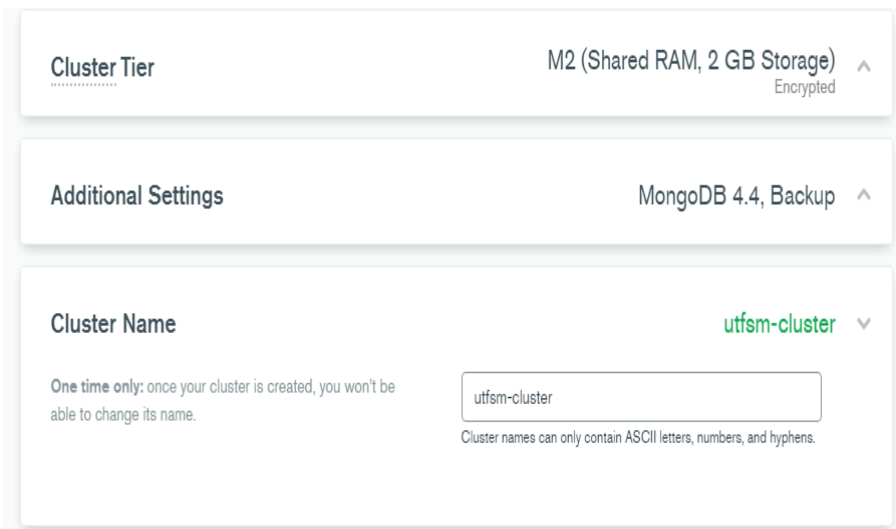


Figura 54: Selección de *cluster tier*. Se debe elegir M0. Fuente: MongoDB Atlas.

Una vez creado el clúster es necesario ir al apartado de seguridad y modificar los usuarios que tendrán acceso a la base de datos. Esto es necesario para que la aplicación pueda conectarse a la base de datos (ver Figura 55).

Add New Database User

Create a database user to grant an application or user, access to databases and collections in your clusters in this Atlas project. Granular access control can be configured with default privileges or custom roles. You can grant access to an Atlas project or organization using the corresponding [Access Manager](#)

Authentication Method

Password

Certificate

AWS IAM
(MongoDB 4.4 and up)

MongoDB uses [SCRAM](#) as its default authentication method.

Password Authentication

user

.....SHOW

Autogenerate Secure PasswordCopy

Database User Privileges

For advanced role based access control, you can apply your own custom roles or add specific privileges to this user. You can combine these role types with a built-in role. A user will gain access to all actions within the roles assigned to them, not just the actions those roles share in common. You must choose at least one role or privilege. [Learn more about roles.](#)

Built-in Role

1 SELECTED ^

Select one [built-in role](#) for this user.

Read and write to any database

Custom Roles

Select your [pre-defined custom role\(s\)](#). Create a custom role in the [Custom Roles](#) tab.

Specific Privileges

Select multiple privileges and what database and collection they are associated with. Leaving collection blank will grant this role for all collections in the database.

Restrict Access to Specific Clusters/Data Lakes

Enable to specify the resources this user can access. By default, all resources in this project are accessible.

OFF

Temporary User

This user is temporary and will be deleted after your specified duration of 6 hours, 1 day, or 1 week.

OFF

Figura 55: Creación de un usuario para acceder a la base de datos. Fuente: MongoDB Atlas.

Luego, es necesario ir al apartado de acceso de red y agregar la *IP 0.0.0.0/0* (ver Figura 56).

Atlas only allows client connections to a cluster from entries in the project's IP Access List. Each entry should either be a single IP address or a CIDR-notated range of addresses. [Learn more.](#)

ADD CURRENT IP ADDRESS

Access List Entry:

0.0.0.0/0

Comment:

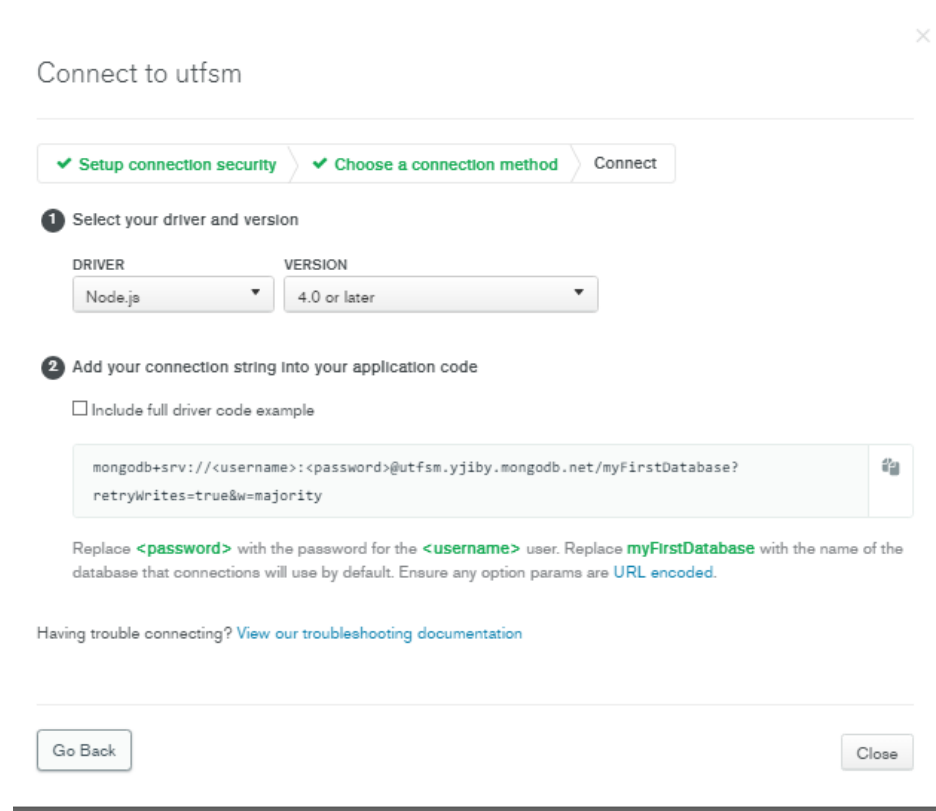
Optional comment describing this entry

Cancel

Confirm

Figura 56: Configuración de acceso de red a la base de datos. MongoDB Atlas.

Finalmente, en el apartado de *databases*, debiera aparecer el clúster creado. Para obtener la *URI* del clúster, la cual será la utilizada para conectarse desde la aplicación a las variables de entorno, se debe seleccionar el botón de conectar. Y luego elegir el método de conexión para la aplicación. En la Figura 57 se visualiza un ejemplo.



Connect to utfsm

✓ Setup connection security ✓ Choose a connection method Connect

1 Select your driver and version

DRIVER VERSION

Node.js 4.0 or later

2 Add your connection string into your application code

☐ Include full driver code example

```
mongodb+srv://<username>:<password>@utfsm.yjiby.mongodb.net/myFirstDatabase?
retryWrites=true&w=majority
```

Replace **<password>** with the password for the **<username>** user. Replace **myFirstDatabase** with the name of the database that connections will use by default. Ensure any option params are [URL encoded](#).

Having trouble connecting? [View our troubleshooting documentation](#)

Go Back Close

Figura 57: Obtención de la URI para conectar la aplicación a la base de datos. Fuente: MongoDB Atlas.

Una vez copiada la *URI*, es necesario remplazar el *username* y *password* por la del usuario creado anteriormente. Además, se debe remplazar *myFistDatabase* por el nombre de la base de datos que se desee. Completados los pasos, se podrá utilizar la *URI* en el archivo de la variable de entorno. Para generar la *URI* de la base de datos de *test*, basta con copiar la misma pero cambiando el nombre de la base de datos.

C/I y C/D

Hasta el momento se había logrado configurar exitosamente la arquitectura en *AWS*. Pero falta un apartado muy importante, como tratar con las actualizaciones de software y el despliegue de estas, pues cada vez que se realiza es necesario volver a levantar cada aplicación para que las actualizaciones tengan efecto. Para facilitar este trabajo, dentro de la cultura DevOps, existe lo que se conoce como integración continua (C/I) y despliegue continuo (C/D).

Gracias a la integración continua, los cambios son fusionados en la rama principal mucho más rápidos, detectándose y validándose para luego pasar a generar una compilación para ser posteriormente sometida a los *tests*. Una vez pasado los *tests* y gracias al despliegue continuo son publicados los cambios en producción para que los clientes puedan ver los cambios. Juntando todos estos pasos es lo que se conoce como *pipeline*.

Para poder llevar a cabo esto en *AWS*, se utilizarán los servicios de *AWS CodeCommit*, *AWS CodePipeline* y *AWS CodeBuild*.

AWS CodeCommit será el encargado de almacenar el código fuente de manera privada, su configuración ya se realizó anteriormente, por lo que no es necesario hacer nada.

AWS CodeBuild permitirá generar los archivos compilados, que luego serán utilizados para realizar los tests.

Finalmente, *AWS CodePipeline*, *permitirá construir una pipeline que será la encargada de implementar el flujo descrito anteriormente*. Desde la detección de los cambios del repositorio hasta la implementación de las actualizaciones en los servicios.

AWS CODE BUILD

Para comenzar a implementar una pipeline, primero es necesario generar dos proyectos de compilación. El primero corresponderá al código del cliente y el otro al código del backend.

Para generar un proyecto de compilación, se debe ir al servicio de CodeBuild y en el listado de proyectos de compilación crear uno. Para crear un proyecto se necesita configurar la información básica de este, el origen de donde se obtendrá el código fuente, el entorno donde se compilará el código y la especificación del archivo que contiene los comandos para realizar esta compilación. El resto puede quedar como viene por defecto. A continuación, en las Figuras 58, 59, 60 y 61, se puede ver cómo debería generarse el proyecto.

Configuración del proyecto

Nombre del proyecto

proyecto-angular-build

El nombre de un proyecto debe tener entre 2 y 255 caracteres. Puede incluir las letras de la A a la Z y de la a a la z, los números del 0 al 9 y los caracteres especiales - y _.

Descripción - *opcional*

Insignia de compilación - *opcional*

☐ Habilitar la insignia de compilación

Habilitar el límite de compilación simultánea - *opcional*

Limite el número de compilaciones simultáneas permitidas para este proyecto.

☐ Restringir el número de compilaciones simultáneas que este proyecto puede comenzar

► Configuración adicional

Etiquetas

Figura 58: Configuración básica del proyecto de compilación. Fuente: AWS.

Origen

Agregar el origen

Origen 1: principal

Proveedor del origen

AWS CodeCommit

Repositorio

Q

plataforma-monitoreo-docente

X

Tipo de referencia

Seleccione el tipo de referencia de la versión de origen que contiene el código fuente.

☒ Ramificación

☐ Etiqueta de Git

☐ ID de confirmación

Ramificación

Elija una ramificación que contenga el código para crear.

master

ID de confirmación - *opcional*

Elija una ID de la confirmación. Esto puede reducir la duración de la compilación.

Q

Versión de origen

Información

refs/heads/master

4c5708aa Merge branch 'hotfix/1.4.10'

► Configuración adicional

Profundidad del clon de Git, Submódulos de Git

Figura 59: Configuración del apartado origen de un proyecto de compilación. Fuente: AWS.

En el apartado de origen es importante que el proveedor sea *AWS CodeCommit* y la rama sea la *main* o *master*.

Especificación de compilación

Especificaciones de la compilación

☒ Utilizar un archivo de especificación de compilación
Almacenar los comandos de compilación en un archivo de especificación de compilación con formato YAML.

☐ Insertar comandos de compilación
Almacenar los comandos de compilación como la configuración del proyecto de compilación

Nombre de la especificación de compilación - *opcional*
De forma predeterminada, CodeBuild busca un archivo denominado buildspec.yml en el directorio raíz del código fuente. Si su archivo de especificación de compilación utiliza un nombre o una ubicación diferentes, escriba su ruta desde la raíz de origen aquí (por ejemplo, buildspec-two.yml o configuration/buildspec.yml).

Figura 60: Especificación de compilación de un proyecto. Fuente: AWS.

Entorno

Imagen del entorno

☒ Imagen administrada
Utilizar una imagen administrada por AWS CodeBuild

☐ Imagen personalizada
Especificar una imagen de Docker

Sistema operativo

Ubuntu

Los tiempos de ejecución del lenguaje de programación ya están incluidos en la imagen estándar de Ubuntu 18.04, que se recomienda para proyectos de CodeBuild nuevos creados en la consola. Consulte [Imágenes de Docker proporcionadas por CodeBuild para obtener más información](#).

Tiempo(s) de ejecución

Standard

Imagen

aws/codebuild/standard:5.0

Versión de la imagen

Utilizar siempre la imagen más reciente para esta versión del tiempo de ejecución

Tipo de entorno

Linux

Privilegiado

☐ Habilite esta marca si desea crear imágenes de Docker o desea que sus compilaciones obtengan privilegios elevados.

Rol de servicio

☒ Nuevo rol de servicio
Cree un rol de servicio en su cuenta.

☐ Rol de servicio existente
Elija un rol de servicio existente de su cuenta.

Nombre del rol

codebuild-proyecto-angular-build-service-role

Escriba el nombre del rol de servicio.

► Configuración adicional

Tiempo de espera, certificado, VPC, tipo de cómputo, variables de entorno, sistemas de archivos

Figura 61: Configuración del entorno de un proyecto. Fuente: AWS.

Finalmente, para crear un proyecto de compilación para el backend se debe seguir los mismos pasos, pero en el apartado del entorno, se debe habilitar el apartado “Privilegiado”, esto permitirá generar imágenes de *Docker*.

AWS CODEPIPELINE

Continuando con la configuración, ahora se pasa a construir las dos *pipelines* correspondientes al *frontend* y *backend*.

Hay que dirigirse al servicio mencionado y comenzar a crear una.

La creación de una pipeline consta de 4 etapas más una de revisión.

La primera etapa consiste completar con el nombre de la *pipeline* (ver Figura 62).

Configuración de la canalización

Nombre de la canalización

front-end-pipeline

No más de 100 caracteres

Rol de servicio

☒ Nuevo rol de servicio

☐ Rol de servicio existente

Cree un rol de servicio en su cuenta.

Elija un rol de servicio existente de su cuenta.

Nombre del rol

AWSCodePipelineServiceRole-us-east-1-front-end-pipeline

Escriba el nombre del rol de servicio.

☒ Permitir que AWS CodePipeline cree un rol de servicio para que pueda utilizarse con esta nueva canalización

▼ Configuración avanzada

Almacén de artefactos

☒ Ubicación predeterminada

☐ Ubicación personalizada

Utilice el almacén de artefactos predeterminado (Amazon S3 codepipeline-us-east-1-490896875879) designado en la misma región y cuenta que su canalización.

Elija una ubicación de S3 existente de su cuenta en la misma región y cuenta que su canalización.

Clave de cifrado

☒ Clave predeterminada administrada por AWS

☐ Clave administrada por el cliente

Utilice la clave maestra de cliente administrada por AWS para CodePipeline en su cuenta para cifrar los datos en el almacén de artefactos.

Para cifrar los datos en el almacén de artefactos bajo una clave administrada por el cliente de AWS KMS, especifique el ID de clave, el ARN de la clave o el ARN de alias.

Cancelar

Siguiente

Figura 62: Etapa 1 de la creación de una pipeline. Fuente: AWS.

La segunda etapa consiste en seleccionar el proveedor de origen, del cual se detectarán los cambios del código de la rama correspondiente. Dependiendo de si corresponde al *frontend* o *backend*, se deberá seleccionar el repositorio correspondiente de *AWS CodeCommit* (ver Figura 63).

Origen

Proveedor del origen
Aquí es donde almacenó sus artefactos de entrada para la canalización. Elija el proveedor y luego proporcione los detalles de conexión.

AWS CodeCommit

Nombre del repositorio
Elija un repositorio que ya haya creado donde se ha enviado el código fuente.

plataforma-monitoreo-docente

Nombre de la ramificación
Elegir una ramificación del repositorio

master

Cambiar las opciones de detección
Elija un modo de detección para iniciar la canalización de forma automática cuando se produce un cambio en el código fuente.

☒ Amazon CloudWatch Events (recomendado)
Utilizar Amazon CloudWatch Events para iniciar mi canalización de forma automática cuando se produzca un cambio

☐ AWS CodePipeline
Utilizar AWS CodePipeline para comprobar periódicamente si se han producido cambios

Formato de artefacto de salida
Elija el formato del artefacto de salida.

☒ CodePipeline predeterminado
AWS CodePipeline utiliza el formato .zip predeterminado para los artefactos de la canalización. No incluye metadatos de git sobre el repositorio.

☐ Clon completo
AWS CodePipeline transmite metadatos sobre el repositorio que permite que las acciones posteriores generen un clon de git completo. Solo se admite para acciones de AWS CodeBuild.

Cancelar Anterior Siguiente

Figura 63: Etapa dos de la creación de una pipeline. Fuente: AWS.

En la etapa 3 se deberá configurar el proveedor de compilación, el cual no es más que los proyectos de compilación creados anteriormente. Es importante seleccionar el proyecto de acuerdo a la pipeline correspondiente, ya sea si es para compilar código del *frontend* o *backend* (ver Figura 64).

Agregar la etapa de compilación Información

Compilación - opcional

Proveedor de compilación
Esta es la herramienta de su proyecto de compilación. Proporcione detalles del artefacto de compilación, como el sistema operativo, el archivo de especificación de compilación y los nombres de los archivos de salida.

AWS CodeBuild

Región
EE.UU. Este (Norte de Virginia)

Nombre del proyecto
Elija un proyecto de compilación que ya haya creado en la consola de AWS CodeBuild. O bien, cree un proyecto de compilación en la consola de AWS CodeBuild y luego vuelva a esta tarea.

angular-app

Crear el proyecto

Variables del entorno - opcional
Elija la clave, el valor y el tipo para las variables de entorno de CodeBuild. En el campo de valor, puede hacer referencia a las variables generadas por CodePipeline. Más información

Agregar la variable de entorno

Tipo de compilación

☒ Compilación única
Activa una compilación única.

☐ Compilación por lotes
Activa varias compilaciones como una sola ejecución.

Cancelar Anterior Omitir la etapa de compilación Siguiente

Figura 64: Etapa 3 de la creación de una pipeline. Fuente: AWS.

Finalmente, en la etapa 4 se deberá configurar donde se implementará el código compilado (ver Figuras 65 y 66). Es necesario seleccionar *AWS ECS* como proveedor de implementación si corresponde al código del backend, en caso contrario, es necesario elegir *AWS S3*, pues se debe recordar que el código del *frontend* se encuentra en un bucket de *S3*.

Agregar la etapa de implementación

Información

Implementar - *opcional*

Proveedor de implementación

Elija cómo implementar en las instancias. Seleccione el proveedor y luego proporcione los detalles de configuración para ese proveedor.

Amazon S3

Región

EE.UU. Este (Norte de Virginia)

Bucket

portfolio-inf-usm.me

Ruta de implementación - *opcional*

☒ Extraer el archivo antes de la implementación

El artefacto implementado se descomprimirá antes de la implementación.

► Configuración adicional

Cancelar

Anterior

Omitir la etapa de implementación

Siguiente

Figura 65: Etapa 4 de creación de una pipeline para *frontend*. Fuente: AWS.

Agregar la etapa de implementación

Información

Implementar - *opcional*

Proveedor de implementación

Elija cómo implementar en las instancias. Seleccione el proveedor y luego proporcione los detalles de configuración para ese proveedor.

Amazon ECS

Región

EE.UU. Este (Norte de Virginia)

Nombre del clúster

Elija un clúster que ya haya creado en la consola de Amazon ECS. O bien, cree un clúster en la consola de Amazon ECS y luego vuelva a esta tarea.

portfolio-docente-prod

Nombre del servicio

Elija un servicio que ya haya creado en la consola de Amazon ECS para su clúster. O bien, cree un nuevo servicio en la consola de Amazon ECS y luego vuelva a esta tarea.

briefcase-service

Archivo de definiciones de imágenes - *opcional*

Escriba el archivo JSON que describe el nombre del contenedor de su servicio, la imagen y la etiqueta.

taskdefinitions.json

Tiempo de espera de la implementación - *opcional*

Escriba el tiempo de espera en minutos para la acción de implementación.

Cancelar

Anterior

Omitir la etapa de implementación

Siguiente

Figura 66: Etapa 4 de creación de una pipeline para *backend*. Fuente: AWS.

SEGURIDAD

Un apartado importante es el de la seguridad. Existen diversas cosas a considerar para tener un mínimo de seguridad cuando se montan plataformas en la nube. AWS ofrece varias herramientas para poder implementar de manera rápida y simple recomendaciones que permiten proteger las aplicaciones. Para llevar esto a cabo es necesario crear un firewall que actúe en el balanceador de carga y realice un filtro de las peticiones. AWS ofrece una solución llamada *WAF Security Automations*, esta ofrece un conjunto de reglas para AWS WAF (*firewall* para aplicaciones web) que filtran los ataques más comunes.

En primera instancia hay que dirigirse al servicio de *AWS S3*, y crear un nuevo *bucket*, en este se almacenan los *logs* del *firewall*. Se recomienda que el *bucket* sea completamente privado y que el nombre haga referencia al *firewall*. Luego, para implementar la solución es necesario utilizar *AWS Cloudformation*, servicio que permite modelar y configurar recursos de *AWS* usando la infraestructura como código. Por lo tanto, hay que dirigirse al servicio y crear una nueva pila (ver Figura 67). Es importante utilizar la siguiente URL de *AWS S3*, en ella se encuentra el *template* con la configuración del *firewall*.

Crear pila

Requisito previo: preparar la plantilla

Preparar la plantilla
Cada pila se basa en una plantilla. Una plantilla es un archivo JSON o YAML que contiene información de configuración sobre los recursos de AWS que desea incluir en la pila.

☒ La plantilla está lista ☐ Utilizar una plantilla de ejemplo ☐ Crear plantilla en Designer

Especificar plantilla
Una plantilla es un archivo JSON o YAML que describe los recursos y las propiedades de la pila.

Origen de la plantilla
Al seleccionar una plantilla se genera una URL de Amazon S3 donde esta se almacenará.

☒ URL de Amazon S3 ☐ Cargar un archivo de plantilla

URL de Amazon S3

URL de plantilla de Amazon S3

URL de S3: Ver en Designer

Cancelar Siguiente

Figura 67: Creación de una pila en *Cloudformation*. Fuente: AWS.

Pasando al siguiente apartado, se debe configurar los parámetros de inicialización de la pila, es decir, las variables de entrada. Se recomienda la configuración mostrada en Figura 68.

Nombre de la pila

Nombre de la pila

AWSWAFSecurityAutomations

El nombre de la pila puede incluir letras (A-Z y a-z), números (0-9) y guiones (-).

Parámetros

Los parámetros se definen en la plantilla y le permiten introducir valores personalizados al crear o actualizar una pila.

Protection List

Activate AWS Managed Rules Protection

Choose yes to enable the AWS Managed Rules.

yes

Activate SQL Injection Protection

Choose yes to enable the component designed to block common SQL injection attacks.

yes

Activate Cross-site Scripting Protection

Choose yes to enable the component designed to block common XSS attacks.

yes

Activate HTTP Flood Protection

Choose yes to enable the component designed to block HTTP flood attacks.

yes - AWS WAF rate based rule

Activate Scanner & Probe Protection

Choose yes to enable the component designed to block scanners and probes.

yes - AWS Lambda log parser

Activate Reputation List Protection

Choose yes to block requests from IP addresses on third-party reputation lists (supported lists: spamhaus, torproject, and emergingthreats).

yes

Activate Bad Bot Protection

Choose yes to enable the component designed to block bad bots and content scrapers.

yes

Log Monitoring Settings

Endpoint Type

Select the type of resource being used.

ALB

Application Access Log Bucket Name

If you chose yes for the Activate Scanners & Probes Protection parameter, enter a name for the Amazon S3 bucket where you want to store access logs for your CloudFront distribution or Application Load Balancer. More about bucket name restriction here: <http://amzn.to/1p1YIU5>. If you chose to deactivate this protection, ignore this parameter.

nombre-bucket

Error Threshold

If you chose yes for the Activate Scanners & Probes Protection parameter, enter the maximum acceptable bad requests per minute per IP. If you chose to deactivate this protection protection, ignore this parameter.

50

Request Threshold

If you chose yes for the Activate HTTP Flood Protection parameter, enter the maximum acceptable requests per FIVE-minute period per IP address. Please note that AWS WAF rate based rule requires values greater than 100 (if you chose Lambda/Athena log parser options, you can use any value greater than zero). If you chose to deactivate this protection, ignore this parameter.

100

WAF Block Period

If you chose yes for the Activate Scanners & Probes Protection or HTTP Flood Lambda/Athena log parser parameters, enter the period (in minutes) to block applicable IP addresses. If you chose to deactivate log parsing, ignore this parameter.

240

Figura 68: Parámetros de la pila. Fuente: AWS

Es importante remplazar el *Application Access Log Bucket Name* con el nombre del *bucket* creado anteriormente para los *logs* del *firewall*. El resto se deja como viene por defecto. La creación dura unos minutos.

Página 95 de 98

Una vez creado, hay que dirigirse al servicio *WAF & Shield*. Dentro se debe seleccionar en el menú *Web ACLs* y debería aparecer en el listado la creada anteriormente. Se debe entrar a esta y en el apartado *Associated AWS Resources*, asociar el balanceador de carga creado en capítulos anteriores (ver Figura 69).

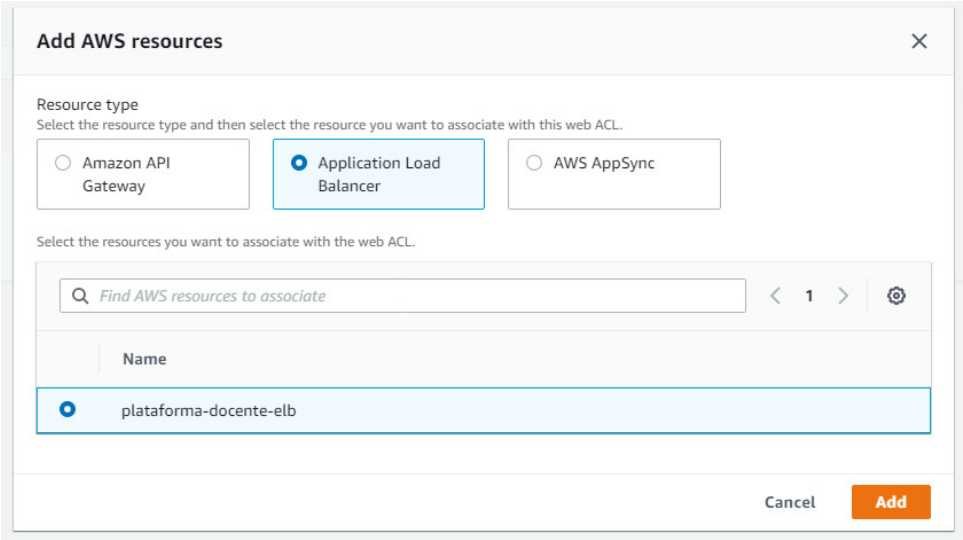


Figura 69: Vinculación de balanceador de carga en firewall. Fuente: AWS.

Con lo realizado anteriormente, se ha implementado el *firewall* de manera exitosa.

ANEXO B: LEVANTAR PLATAFORMA USANDO DOCKER COMPOSE

Si no se desea desplegar la plataforma en la nube, es posible levantarla y configurarla de forma rápida usando Docker. Es importante que previamente, dentro de un directorio se tengan los directorios correspondientes de cada proyecto (*frontend* y *backend*). Cada proyecto posee su propio *Dockerfile* con su configuración correspondiente y esta es usada desde un archivo central llamado *docker compose*. Este permite levantar múltiples contenedores sin necesidad de estar uno a uno levantándolos manualmente, además permite agregar políticas de reinicio en caso de fallas.

A continuación, se presentará un ejemplo del archivo *docker compose* que se puede utilizar para levantar la plataforma.

```
1  version: "3.9"
2  services:
3    plataforma_monitoreo_docente:
4      image: plataforma_monitoreo_docente_image
5      platform: linux/arm64/v8
6      restart: always
7      depends_on:
8        - briefcase_service
9      build:
10       context: ./plataforma-monitoreo-docente
11       dockerfile: ./Dockerfile
12      ports:
13        - "4500:80"
14      networks:
15        - pmd-network
16
17    briefcase_service:
18      image: briefcase_service_image
19      platform: linux/arm64/v8
20      restart: always
21      depends_on:
22        - mongodb
23      build:
24       context: ./api-briefcase
25       dockerfile: ./Dockerfile
26      environment:
27        AWS_ACCESS_KEY_ID: AKIAIYWCT4KLD5MS4F5HH
28        AWS_SECRET_ACCESS_KEY: /E9UXR9UG1yJrch3f+jXGixGdit0MUqoyna++BK
29        BUCKET_S3: utfsm-dev
30        PORT: 4800
31        API_BRIEFCASE: http://localhost:4800
32        NODE_ENV: local
33        DB: mongodb://mongodb:27017/pmd-db
34        KEYCLOAK_ISSUER: https://auth.inf.utfsm.cl/auth/realms/DI
35        KEYCLOAK_JWKS_URI: https://auth.inf.utfsm.cl/auth/realms/DI/protocol/openid-connect/certs
36        SWAGGER_USER: USM_USER
37        SWAGGER_PASS: Cd6pQU8YjqWnuj
38      ports:
39        - "4800:4800"
40      networks:
41        - pmd-network
42
43    mongodb:
44      image: mongo:latest
45      platform: linux/arm64/v8
46      restart: always
47      ports:
48        - '27017:27017'
49      volumes:
50        - dbdata6:/data/db
51      networks:
52        - pmd-network
53
54  volumes:
55    dbdata6:
56
57  networks:
58    pmd-network:
59    driver: bridge
60
```

Figura 70: Ejemplo de archivo *docker-compose.yaml*. Fuente: Elaboración propia.

Una vez listo el archivo, basta con ejecutar en la terminal “*docker compose up*” y se levantará la plataforma usando *Docker*.

ANEXO C: PRIMEROS PASOS PARA USAR LA PLATAFORMA

Una vez levantada la plataforma y configurada la base de datos, es necesario generar el rol de administrador y vincularlo a una cuenta en específico, para llevar a cabo esto, se puede utilizar las líneas de código señaladas en Figura 71. Es importante, ingresar a la base de datos vía terminal y utilizar *mongosh*.

```
1 use < dbname >;
2 db.createCollection( "Roles" );
3 // Roles: 'ADMIN' | 'JEFE DE CARRERA' | 'SECRETARIO(A)' | 'PROFESOR'
4 // users debe ser un array con correos del DI
5 db.Roles.insertOne( {role: "ADMIN", description: "Usuario administrador", users: ["correo@inf.utfsm.cl"]} )
```

Figura 71: Comandos necesarios para agregar y vincular rol de administrador a un correo del departamento. Fuente: Elaboración propia.

Para utilizar el correo que se estime necesario, basta con modificar en el comando, la propiedad *users* este es un *array* de correos a vincular al rol en cuestión. Es muy importante que estos correos sean del Departamento de Informática, dado que se tiene integración con el sistema de cuentas de este.