TecnoCampus
Escola Superior Politècnica

Centre adscrit a la
upf. Universitat Pompeu Fabra Barcelona

Grau en Enginyeria Informàtica de Gestió i Sistemes d'Informació

MONITORATGE INTERN DE LES PLATAFORMES DE L'EMPRESA ALPHANET SECURITY SYSTEMS: ALPHA DATA MANAGER I ALPHA MONITOR

Memòria

GERARD PIQUER ARREBOLA
TUTOR: JOSEP ROURE ALCOBÉ

03/06/2024

TecnoCampus
Mataró-Maresme

Abstract

In this final degree project, a set of tasks has been carried out with a single purpose, to monitor internally the two platforms of the company Alphanet Security Systems, which are AlphaDataManager and AlphaMonitor.

On the one hand, testing code has been developed with the purpose of testing that the code of each two platforms has stability and robustness important enough for developers to be certain that the code they have developed works correctly together with the already existing code.

On the other hand, monitoring has been designed and implemented to be able to track the entire state of the infrastructure of the company's two platforms in real-time with the aim of detecting potential operational errors at the slightest possible time and prevent possible incidents of the platforms that affect the user experience.

Throughout the course of this project, a study of the current system and the functioning of the entire infrastructure has been carried out, followed by an analysis of possible actions to achieve a stable and secure system in terms of errors. This project provides a solution to a need of the Alphanet company in terms of code quality and infrastructure state supervision in order to subsequently prevent possible operating errors of the company's platforms.

Resum

En aquest treball de fi de grau, s'ha realitzat un conjunt de tasques amb un únic objectiu, monitorar en l'àmbit intern les dues plataformes de l'empresa Alphanet Security Systems, les quals són AlphaDataManager i AlphaMonitor.

Per una banda, s'ha desenvolupat codi Testing amb l'objectiu de testejar que el codi de cadascuna de les dues plataformes té una estabilitat i robustesa prou important perquè els desenvolupadors tinguin una certesa del fet que el codi que han desenvolupat funciona correctament juntament amb el codi ja existent.

D'altra banda, s'ha dissenyat i implementat un monitoratge per poder fer un seguiment de tot l'estat de la infraestructura de les dues plataformes de l'empresa en temps real amb l'objectiu de detectar possibles errors de funcionament en el menor temps possible i prevenir possibles incidències de les plataformes que afectin l'experiència d'ús dels usuaris.

Durant el transcurs del treball, s'ha realitzat un estudi del sistema actual i del funcionament de tota la infraestructura i a continuació s'ha analitzat les possibles accions a realitzar per poder aconseguir un sistema estable i segur en termes d'errors. Aquest treball aporta una solució a una necessitat de l'empresa Alphanet en termes de qualitat de codi i supervisió de l'estat de la infraestructura per tal de prevenir a posteriori possibles errors de funcionament de les plataformes de l'empresa.

Resumen

En este trabajo de fin de grado se han realizado un conjunto de tareas con un único objetivo, monitorizar en el ámbito interno las dos plataformas de la empresa Alphanet Security Systems, que son AlphaDataManager y AlphaMonitor.

Por un lado, se ha desarrollado código Testing con el objetivo de testear que el código de cada una de las dos plataformas tiene una estabilidad y robustez lo suficientemente importante como para que los desarrolladores tengan una certeza de que el código que han desarrollado funciona correctamente junto con el código ya existente.

Por otra parte, se ha diseñado e implementado una monitorización para poder realizar un seguimiento de todo el estado de la infraestructura de las dos plataformas de la empresa en tiempo real con el objetivo de detectar posibles errores de funcionamiento en el menor tiempo posible y prevenir posibles incidencias de las plataformas que afecten a la experiencia de uso de los usuarios.

A lo largo del trabajo, se ha realizado un estudio del sistema actual y del funcionamiento de toda la infraestructura, seguido de un análisis de posibles acciones a realizar para lograr un sistema estable y seguro en términos de errores. Este trabajo proporciona una solución a una necesidad de la empresa Alphanet en términos de calidad de código y supervisión del estado de la infraestructura para prevenir a posteriori posibles errores de funcionamiento de las plataformas de la empresa.

Índex

Índex de figures III

Glossari de termes V

1 Introducció 7

2 Marc Teòric 9

2.1 Context 9

2.2 Alphanet Security Systems S.L 10

2.3 El Testing 11

2.3.1 Testing Unitari (Unit Testing) 11

2.3.2 Testing d'Integració (Integration Testing) 12

2.4 Monitoratge d'infraestructura software 13

2.5 Plataformes principals d'Alphanet 14

2.5.1 AlphaDataManager 14

2.5.2 AlphaMonitor 14

2.6 Arquitectura de software 15

2.6.1 AlphaDataManager 15

2.6.2 AlphaMonitor 16

2.7 Antecedents 19

3 Objectius i Abast 21

3.1 Objectius del client 21

3.2 Objectius del Testing 21

3.3 Objectius del monitoratge 22

3.4 Usuaris 22

3.5 Indicadors clau (KPIs) 23

3.5.1 Testing 23

3.5.2 Monitoratge 23

4 Anàlisi de Referents 25

4.1 Eines a valorar en el desenvolupament del Testing 25

4.2 Eines a valorar en el desenvolupament del Monitoratge 26

5 Metodologia 29

6 Definició de requeriments funcionals i tecnològics 31

6.1 Requeriments Funcionals 31
6.2 Requeriments Tecnològics 31
6.3 Requeriments de Testing 32
6.4 Requeriments de Monitoratge 32
7 Casos d'ús 33
8 Anàlisi i disseny 39
9 Desenvolupament 41
9.1 Entorns 41
9.1.1 Entorn de desenvolupament 41
9.1.2 Entorn de test 42
9.1.3 Entorn de producció 43
9.2 Testing 44
9.2.1 Disseny i modelat 44
9.2.2 Automatització dels tests 45
9.3 Monitoratge 46
9.3.1 Actuator 46
9.3.3 Grafana 48
9.3.4 Accions automatitzades 56
9.4 Notacions 57
9.4.1 Spring 57
9.4.2 Testing 58
9.4.3 Angular 59
9.5 Pantalles 60
10 Anàlisi de resultats 73
11 Conclusions 75
12 Possibles ampliacions 77
13 Bibliografia 79

Índex de figures

Figura 1. Representació gràfica Testing Unitari. Font: Martin Fowler, 5 maig 2014. 12
Figura 2. Esquema actual de l'arquitectura d'Alphanet. Font: Alphanet. 18
Figura 3. Cas d'ús executar i avaluar tests. Font: Elaboració pròpia 33
Figura 4. Cas d'ús visualitzar els gràfics del monitoratge. Font: Elaboració pròpia 34
Figura 5. Cas d'ús enviar notificació al Discord. Font: Elaboració pròpia 35
Figura 6. Cas d'ús enviar correu electrònic. Font: Elaboració pròpia 36
Figura 7. Cas d'ús executar accions automatitzades. Font: Elaboració pròpia 37
Figura 8. Diagrama de la relació Action - Service. Font: Elaboració pròpia 39
Figura 9. Diagrama de la relació Metric - Service. Font: Elaboració pròpia 40
Figura 10. Diagrama de l'entorn de desenvolupament d'ADM. Font: Elaboració pròpia 41
Figura 11. Diagrama de l'entorn de desenvolupament d'AM. Font: Elaboració pròpia. 42
Figura 12. Diagrama de l'entorn de test d'ADM. Font: Elaboració pròpia 42
Figura 13. Diagrama de l'entorn de test d'AM. Font: Elaboració pròpia 43
Figura 14. Diagrama de l'entorn de producció d'ADM. Font: Elaboració pròpia 43
Figura 15. Diagrama de l'entorn de producció d'AM. Font: Elaboració pròpia 44
Figura 16. Mètriques d'actuator. Font: Elaboració pròpia 46
Figura 17. Mètrica d'espai total del disc. Font: Elaboració pròpia 47
Figura 18. Mètrica del nombre de peticions HTTP al servidor. Font: Elaboració pròpia 47
Figura 19. Llistat d'Alert rules configurades. Font: Elaboració pròpia 51
Figura 20. Exemple d'Alert rule. Font: Elaboració pròpia 51
Figura 21. Llistat de Contact points definits a Grafana. Font: Elaboració pròpia 52
Figura 22. Configuració de la integració de Discord. Font: Elaboració pròpia 52
Figura 23. Configuració del template de la integració de Discord. Font: Elaboració pròpia 53
Figura 24. Exemple de notificació d'alerta al Discord. Font: Elaboració pròpia 53
Figura 25. Exemple de resolució d'alerta al Discord. Font: Elaboració pròpia 53
Figura 26. Configuració de la integració d'Email. Font: Elaboració pròpia 54
Figura 27. Configuració del template d'Email. Font: Elaboració pròpia 54
Figura 28. Exemple de correu electrònic d'una notificació d'alerta. Font: Elaboració pròpia 55
Figura 29. Exemple de correu electrònic de resolució d'alerta. Font: Elaboració pròpia 55
Figura 30. Exemple de Notification policy. Font: Elaboració pròpia 56
Figura 31. Notification policy per accions automatitzades. Font: Elaboració pròpia 56
Figura 32. Pantalla del monitoratge de la Ingesta d'ADM. Font: Elaboració pròpia 60
Figura 33. Pantalla completa del monitoratge de la Ingesta d'ADM. Font: Elaboració pròpia 60
Figura 34. Pantalla del monitoratge de l'Arquitectura d'ADM. Font: Elaboració pròpia 61
Figura 35. Pantalla completa del monitoratge de l'Arquitectura d'ADM. 61
Figura 36. Pantalla del monitoratge de la Ingesta d'AM. Font: Elaboració pròpia 61
Figura 37. Pantalla completa del monitoratge de la Ingesta d'AM. Font: Elaboració pròpia 62
Figura 38. Pantalla del monitoratge de l'Arquitectura d'AM. Font: Elaboració pròpia 62

Figura 39. Pantalla completa del monitoratge de l'Arquitectura d'AM. 62
Figura 40. Gràfic del nombre aproximat de missatges visibles a la SQS d'ADM. 63
Figura 41. Gràfic de flux de missatges en la Ingesta d'ADM. Font: Elaboració pròpia 63
Figura 42. Gràfic del percentatge d'utilització de CPU dels servidors d'ADM. 64
Figura 43. Gràfic del nombre de correus electrònics enviats. Font: Elaboració pròpia 64
Figura 44. Gràfic de la memòria utilitzada de la Redis. Font: Elaboració pròpia 65
Figura 45. Gràfic d'espai d'emmagatzematge disponible a la RDS d'ADM. 65
Figura 46. Gràfic del nombre de sessions actives a la RDS. Font: Elaboració pròpia 65
Figura 47. Gràfic del nombre d'invocacions de la funció de Lambda. Font: Elaboració pròpia 66
Figura 48. Gràfic de l'espai disponible de les instàncies a AWS. Font: Elaboració pròpia 66
Figura 49. Gràfic del percentatge d'ús de CPU de la RDS d'ADM. Font: Elaboració pròpia 67
Figura 50. Gràfic del nombre aproximat de missatges visibles a la SQS d'AM. 67
Figura 51. Gràfic del percentatge d'ús de CPU dels servidors d'AM. Font: Elaboració pròpia 68
Figura 52. Gràfic del nombre de missatges enviats a la SQS d'AM. Font: Elaboració pròpia 68
Figura 53. Gràfic d'espai d'emmagatzematge disponible a la RDS d'AM. 69
Figura 54. Gràfic del nombre d'invocacions de la Lambda d'AM. Font: Elaboració pròpia 69
Figura 55. Gràfic del percentatge d'ús de CPU de la RDS d'AM. Font: Elaboració pròpia 69
Figura 56. Pantalla d'accions automatitzades. Font: Elaboració pròpia 70
Figura 57. Modal per crear una nova acció. Font: Elaboració pròpia 70
Figura 58. Card amb informació d'un servei i les possibles accions. Font: Elaboració pròpia 70
Figura 59. Modal per editar mètriques d'un servei. Font: Elaboració pròpia 71
Figura 60. Pantalla de les mètriques d'un servei. Font: Elaboració pròpia 71
Figura 61. Gràfic de la memòria utilitzada per l'aplicació del servei. Font: Elaboració pròpia 71
Figura 62. Gràfic de la CPU utilitzada per l'aplicació del servei. Font: Elaboració pròpia 72
Figura 63. Gràfic del nombre de threads de l'aplicació del servei. Font: Elaboració pròpia 72
Figura 64. Gràfic del temps d'ús de la connexió de l'aplicació del servei. Font: Elaboració pròpia 72

Glossari de termes

ADM	AlphaDataManager
AM	AlphaMonitor
AWS	Amazon Web Services
SQS	Amazon Simple Queue Service
RDS	Amazon Relational Database Service
S3	Amazon Simple Storage Service
BBDD	Bases de Dades
CPU	Central Processing Unit
IDE	Integration Development Environment
EC2	Amazon Elastic Compute Cloud

1 Introducció

El projecte a desenvolupar és un projecte proposat per l'empresa Alphanet [1]. L'objectiu és testejar el codi desenvolupat de les dues plataformes de l'empresa, AlphaDataManager i AlphaMonitor, a més del Testing de la mateixa arquitectura que inclou la ingesta de dades de les càmeres, passant per la lambda i la cua d'AWS, per continuar amb el processament de les dades i posteriorment el seu emmagatzematge a la base de dades.

A més a més, el projecte també inclou monitorar tota la infraestructura que engloba les plataformes de l'empresa, incloent-hi la SQS, la lambda, la RDS i els serveis i servidors de producció, amb l'afegit de la creació d'un conjunt d'accions automatitzades que s'encarregaran de gestionar, tal com indica el seu nom, de forma automàtica l'estat dels serveis i servidors de les plataformes. Finalment, es pretén configurar un canal de difusió a través de la plataforma Discord, amb l'afegit de l'enviament de correus amb les corresponents notificacions per conèixer en tot moment l'estat del monitoratge en temps real.

En relació amb la part del Testing, el projecte se centra amb dos tipus de tests de software:

- Tests unitaris per testejar mètodes i funcions de forma individual.
- Tests d'integració per testejar mètodes i funcions que impliquen la interacció conjunta de més d'una interfície, servei, repositori, etc.

Dins d'aquesta part del Testing, el projecte també inclou l'automatització de tests a la plataforma GitLab la qual s'utilitza a l'empresa com a sistema de control de versions del codi de tots els projectes desenvolupats. L'automatització dels tests ofereix una major seguretat a l'hora de pujar codi a producció i també a l'hora d'ajuntar codi de diferents branques.

La motivació del projecte escollit prové, en major part, per les pràctiques en empresa realitzades justament a l'empresa Alphanet i la utilitat aportada pel projecte a l'hora de desenvolupar codi i monitorar en temps real el funcionament de les dues plataformes, per tal de saber a l'instant qualsevol error o mal funcionament de la infraestructura de les plataformes.

A més a més, l'interès de l'empresa per tenir una certesa tan acotada com sigui possible que tota la infraestructura en general funciona correctament també hi és present.

La solució tecnològica del projecte es troba definida per tots els tests desenvolupats tant a AlphaDataManager com a AlphaMonitor els quals s'utilitzen per tenir un software confiable, sense errors o amb molt pocs, amb l'afegit de tenir el software documentat pels mateixos tests desenvolupats i facilitant l'evolució del mateix software, ja que en el cas de modificar el codi, aleshores els tests ajuden a saber les modificacions realitzades.

A més a més, el monitoratge de la infraestructura que engloba les dues plataformes també es troba definit dins de la solució tecnològica a través de representacions visuals mitjançant gràfics del seu estat en temps real. Aquests gràfics ofereixen una informació molt rellevant per totes aquelles persones involucrades en vetllar pel correcte funcionament de tota la infraestructura de l'empresa i preveure possibles errors que puguin afectar posteriorment el funcionament de les dues plataformes, sigui AlphaDataManager o AlphaMonitor.

2 Marc Teòric

2.1 Context

Recentment, l'empresa Alphanet ha reescrit les seves dues plataformes i amb l'objectiu d'aconseguir un software més confiable i acotat en termes d'errors, ara és el moment oportú per desenvolupar els tests corresponents al codi de cadascuna de les plataformes.

A través del Testing, s'aconsegueix verificar i validar la qualitat d'una aplicació per assegurar el compliment dels requeriments en els quals es basa l'aplicació en qüestió. Aquesta utilitat que aporta el Testing és de vital importància, sobretot per a aquelles empreses que ofereixen els seus serveis a través de plataformes software.

Amb l'ajuda d'un monitoratge de la infraestructura que envolta tant ADM com AM, l'empresa pot controlar-ne l'estat en tot moment i adoptar les mesures necessàries per mantenir qualsevol part de la infraestructura en un bon estat.

En relació amb les tendències actuals, existeix la tendència d'utilitzar el Testing de forma més sovint i amb major freqüència, ja que permet obtenir un software sense errors o amb molt pocs, fet que permet obtenir un software més confiable. Dit això, el Testing té un paper fonamental a l'hora d'assegurar que el codi desenvolupat és correcte i robust i és per això que les empreses cada vegada posen més èmfasi en aquest aspecte, ja que el seu principal objectiu és satisfer els seus consumidors i oferir la màxima qualitat possible dels seus productes.

En relació amb el marc de negoci, apareixen les figures dels clients o usuaris finals. Aquests són els encarregats de definir l'èxit del producte en qüestió. Pel que fa al projecte, està enfocat a solucionar totes les necessitats d'aquests clients, els quals estaran presents en tot el desenvolupament del projecte.

A l'hora de definir els usuaris finals del producte, ens referim a totes aquelles persones encarregades de la supervisió del funcionament de la infraestructura de les plataformes de l'empresa. De forma indirecta, també podem incloure en aquest grup d'usuaris finals tant els clients que té l'empresa, ja que són els usuaris que utilitzen diàriament la plataforma ADM com també els tècnics de la mateixa empresa que utilitzen de forma diària la plataforma AM.

2.2 Alphanet Security Systems S.L.

L'empresa Alphanet Security Systems és una empresa catalana que compta amb més de quinze anys d'experiència i que proveeix als ajuntaments i a les policies locals amb productes tecnològics i serveis professionals amb l'objectiu de fer del municipi un lloc més eficient i segur pels ciutadans. Les solucions que aporta l'empresa estan implantades en diversos països, a més de 200 municipis i més de tres milions de ciutadans es troben protegits gràcies al seu servei.

La seva història s'inicia l'any 2007, on neix Alphanet Sistemes de Seguretat, una empresa integradora de solucions de videovigilància per al sector municipal. Al cap d'uns quants anys, concretament l'any 2018, Alphanet Seguretat es converteix en desenvolupador i fabricant de solucions pròpies. Tres anys més tard, Alphanet ja és molt més que una empresa de seguretat, proveeix solucions transversals amb dos denominadors comuns: avantguarda i intel·ligència.

Alphanet Security Systems es troba en diversos àmbits d'actuació, com són la seguretat ciutadana, la mobilitat, Smart Analytics i medi ambient. Cadascun d'ells aporta un valor significatiu i diferencial a l'empresa que fa que destaqui per davant d'altres empreses competidores en el sector.

A més a més, en relació amb el conjunt de solucions que aporta l'empresa als ajuntaments i a les policies locals, hi trobem la plataforma principal de l'empresa, ADM, la qual ofereix als clients tota la informació recopilada i estructurada dels sistemes de reconeixement de matrícules amb l'objectiu d'ajudar els mateixos clients a prendre les decisions més encertades possibles.

Actualment, l'empresa compta amb un total de 27 treballadors entre caps de departament, tècnics i enginyers. Aquesta xifra ha anat augmentat amb el pas dels anys, fet que implica que l'oficina de l'empresa també s'hagi anat ampliant.

2.3 El Testing

A l'hora de desenvolupar software, per norma general solen aparèixer un munt d'errors que s'han de solucionar perquè el software funcioni correctament. Si aquests errors no se solucionen en el moment adequat, totes les parts involucrades en el procés de desenvolupament es poden veure afectades i en conseqüència el resultat final no és l'esperat. És per això, que existeix el Testing de Sofware, també conegut en anglès com a "Software Testing", el qual aporta una garantia molt important a les empreses a l'hora de desenvolupar el seu propi software.

A través del Testing, el que es pretén és detectar possibles errors en el software perquè puguin ser corregits amb l'objectiu d'aconseguir un software lliure d'errors. Tot i això, amb la utilització del Testing no es pot assegurar que el software funcioni sota totes les condicions possibles, però sí que es pot assegurar que funcioni sota unes condicions en concret.

Una altra característica del Testing és la seva utilitat en el moment de fer manteniment de software, és a dir, a l'hora de fer refactoring del codi ja existent, el Testing ajuda a millorar el que ja està fet i proporciona certa seguretat a l'hora d'aplicar noves funcionalitats al codi i que aquestes noves funcionalitats no afectin els tests ja existents.

Dintre del món del Testing, existeix un ventall molt ampli de tipus de tests i cadascun d'ells aporta una funcionalitat diferent que el fa destacar respecte als altres segons la situació en la qual s'utilitzi. En aquest projecte es fa ús de dos tipus de tests els quals s'introdueixen en els següents apartats.

2.3.1 Testing Unitari (Unit Testing)

El Testing Unitari, tal com indica el seu nom, fa referència a "unitats" de codi aïllades respectivament que són utilitzades en tests en els quals hi ha unes dades d'entrada concretes i on s'expecten uns resultats esperats. Segons el document [2], *unit tests invoke one or more methods from a class to produce observable results that are verified automatically.* El que es pretén aconseguir amb els tests unitaris és analitzar detalladament tots els mètodes d'una classe o de diverses classes del codi amb l'objectiu de comprovar que no hi ha errors i així tenir la certesa que el mètode testejat funciona correctament.

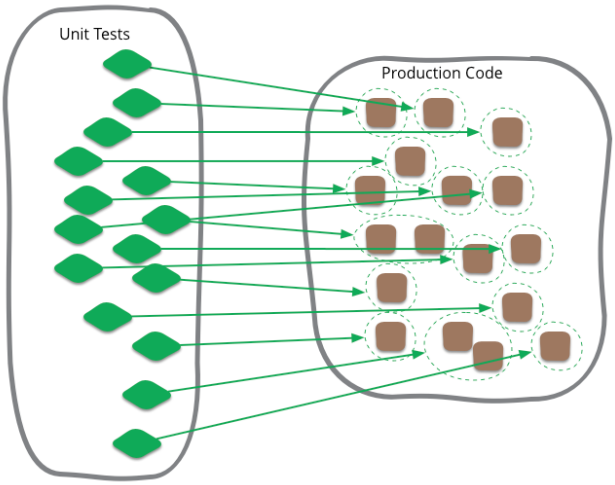

Figura 1. Representació gràfica Testing Unitari. Font: Martin Fowler, 5 maig 2014

Tal com s'exposa en el document [3], sovint el Testing Unitari és desenvolupat pels programadors i a l'inici de tot el procés de desenvolupament, abans d'integrar i testejar el codi com a un únic sistema.

Els objectius principals d'aquest tipus de Testing són:

- Aïllar una secció de codi en concret perquè pugui ser testejada.
- Verificar la correctesa del codi.
- Testejar cada mètode i funció del codi.
- Arreglar bugs a l'inici del desenvolupament per estalviar costos.

2.3.2 Testing d'Integració (Integration Testing)

Segons Martin Fowler i la seva entrada d'opinió [4], els tests d'integració determinen si unitats independents del software treballen correctament quan aquestes estan connectades conjuntament.

Principalment, el Testing d'Integració es basa a verificar el comportament de mètodes, interfícies i components quan aquests interactuen de forma conjunta dins del software. En certs moments, aquest tipus de Testing es pot utilitzar amb el Testing Unitari, ja que, en primer lloc, es pot testejar un mòdul del software en concret i després es pot testejar aquest mòdul amb d'altres a partir del Testing d'Integració.

Amb aquest tipus de Testing, s'observa el temps d'interacció entre diversos components d'un mateix software i també els problemes que hi pot haver quan aquests components interactuen entre si.

2.4 Monitoratge d'infraestructura software

Un altre dels temes en els quals s'emmarca el projecte és el monitoratge de la infraestructura que engloba les dues plataformes de l'empresa Alphanet, ja sigui la SQS, la lambda, la RDS o els diferents serveis i servidors de l'entorn de producció. Qualsevol d'aquestes parts que conformen la infraestructura són importants i utilitzades amb la finalitat que les dues plataformes funcionin correctament en tot moment.

Actualment, existeixen infinitat d'empreses les quals disposen d'un sistema de monitoratge que els permet saber amb exactitud l'estat real i en directe de la seva pròpia infraestructura.

El monitoratge és una part molt important dins de les empreses tecnològiques i de qualsevol empresa en general, ja que aporta una informació addicional a l'hora de gestionar la infraestructura de l'empresa, a més d'una seguretat pels mateixos tècnics i enginyers encarregats del manteniment de la infraestructura.

2.5 Plataformes principals d'Alphanet

2.5.1 AlphaDataManager

AlphaDataManager [5] és una plataforma de gestió i explotació de dades al núvol que transforma la informació que capten els sistemes de reconeixement de matrícules en informació amb l'objectiu d'ajudar a prendre les decisions més encertades. La plataforma està dissenyada per ajudar als municipis i a les ciutats d'arreu del territori a ser més intel·ligents, sostenibles i segures; amb l'afegit d'utilitzar al màxim el valor que tenen les dades de trànsit.

Es troba en tres àrees d'actuació:

- **Seguretat ciutadana:** col·laboració policial, investigacions policials i anàlisi per prevenció.
- **Mobilitat:** Zones de Baixes Emissions (ZBE), controls semafòrics, gestió de pàrquing públic i control de zones de càrrega i descàrrega.
- **Smart Analytics:** mobilitat i trànsit, turisme, medi ambient i economia.

2.5.2 AlphaMonitor

AlphaMonitor és una plataforma al núvol d'ús intern de l'empresa que s'utilitza per monitorar tot el funcionament dels sistemes de reconeixement de matrícules. La plataforma ajuda als tècnics a l'hora de muntar dispositius, detectar possibles irregularitats o mancances i a tenir tota la informació estructurada i ben organitzada de les actuacions dutes a terme.

Els usos principals de la plataforma són:

- **Monitoratge:** els tècnics saben en temps real l'estat dels dispositius instal·lats per tot el territori.
- **Manteniment:** en tot moment es té un control de l'inventari dels dispositius.
- **Prevenció:** s'aconsegueix predir les possibles futures avaries.

2.6 Arquitectura de software

2.6.1 AlphaDataManager

La plataforma AlphaDataManager utilitza les següents tecnologies pel seu correcte funcionament:

- **AWS** [6]
  - Lambda, SQS, RDS, S3.
- **OpenSearch** [7]
  - Motor de cerca i analítica de codi lliure.
  - Permet realitzar i combinar diversos tipus de cerques: estructurades, no estructurades, geogràfiques, etc.
  - Ofereix escalabilitat i eficiència a l'hora de gestionar una gran quantitat de registres.
  - S'utilitza per emmagatzemar tot el conjunt de dades dels vehicles detectats per les càmeres.
- **BBDD Relacional**
  - PostgreSQL 10.
- **Backend**
  - Desenvolupat amb Java 17 i implementat amb Spring.
- **Frontend**
  - Desenvolupat amb Angular 13.
- **Processor**
  - Processa les dades provinents de SQS i les guarda a Elasticsearch, a S3 en cas d'imatges i a la BBDD.
  - Desenvolupat amb Java 21 i implementat amb Spring.
- **Redis** [8]
  - Base de dades que emmagatzema les dades duplicades.
  - El Processor és l'encarregat d'enviar-li les dades duplicades.

La plataforma està configurada perquè es pugui treballar en diferents entorns a l'hora de desenvolupar noves funcionalitats, testejar-les i fer demostracions als clients. Aquests entorns són:

- **Dev:** entorn de desenvolupament en el qual es desenvolupa el nou codi i les noves funcionalitats de la plataforma.
- **Test:** entorn de testing en el qual es prova el codi desenvolupat amb unes interfícies i dades molt similars a les de producció.
- **Demo:** entorn de demostració el qual s'utilitza per mostrar als clients i grups d'interès el funcionament de la plataforma com si fos a producció.
- **Producció:** entorn de producció en el qual hi ha les dades en temps real que provenen dels lectors i on els clients interactuen amb la plataforma.

2.6.2 AlphaMonitor

La plataforma AlphaMonitor utilitza les següents tecnologies pel seu correcte funcionament:

- **AWS**
  - Lambda, SQS, RDS, S3.
- **OpenSearch**
- **BBDD Relacional**
  - PostgreSQL 10.
- **Backend**
  - Desenvolupat amb Java 17 i implementat amb Spring.
- **Frontend**
  - Desenvolupat amb Angular 13.
- **Consumer**
  - Consumeix les dades provinents de SQS i les guarda a OpenSearch i a la Redis en cas que siguin duplicades.
- **Connector**
  - Connecta les dades provinents d'ADM Backend i les dades provinents de SQS.
- **Redis**
  - Base de dades que emmagatzema les dades vàlides de la ingesta.

- **Prometheus** [9]
    - Sistema de monitoratge de codi lliure.
    - Permet obtenir i modelar mètriques per la seva corresponent visualització.
    - S'utilitza per recollir mètriques provinents d'AM Backend.
- **Grafana** [10]
    - Sistema de visualització i analítiques de software de codi lliure.
    - Permet aconseguir, visualitzar, alertar i explorar mètriques de qualsevol font de dades.
    - S'utilitza per representar les mètriques de les plataformes ADM i AM i les mètriques provinents d'AWS, la Redis i Prometheus.

La plataforma també està configurada perquè es pugui treballar en diferents entorns a l'hora de desenvolupar noves funcionalitats, testejar-les i fer demostracions als clients. Aquests entorns són:

- **Dev:** entorn de desenvolupament en el qual es desenvolupa el nou codi i les noves funcionalitats de la plataforma.
- **Test:** entorn de testing en el qual es prova el codi desenvolupat amb unes interfícies i dades molt similars a les de producció.
- **Producció:** entorn de producció en el qual hi ha les dades en temps real que provenen dels lectors ALPR Helix, passant pel Town Agent, i ALPR estàndard i on els tècnics interactuen amb la plataforma.

Esquema general de les plataformes AlphaDataManager i AlphaMonitor:

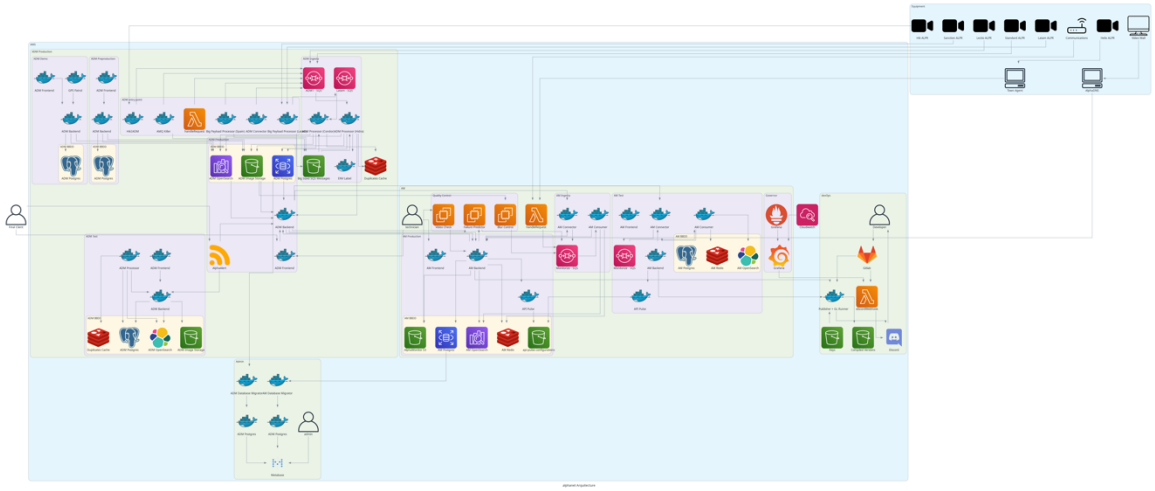

Figura 2. Esquema actual de l'arquitectura d'Alphanet. Font: Alphanet

2.7 Antecedents

En els últims anys, l'empresa Alphanet ha vist com les seves dues plataformes s'han anat quedant enrere en l'àmbit tecnològic i no acabaven de satisfer del tot les necessitats dels seus clients. És per això que recentment s'ha pujat una nova versió juntament amb un canvi d'imatge de cadascuna de les plataformes.

Després d'aquesta actualització de les dues plataformes, apareix nou codi que s'ha de testejar per evitar que a l'hora de pujar aquest codi a producció, sorgeixin errors inesperats que dificultin el procés de pujada i l'actualització pertinent a la plataforma en qüestió.

En relació amb el monitoratge, actualment la plataforma AM està dotada d'un monitoratge intern de totes les càmeres de les quals disposa l'empresa i que estan repartides arreu del territori. La informació recopilada d'aquest monitoratge es representa en forma de gràfiques dins de la plataforma, fet que permet als tècnics gestionar en tot moment l'estat de qualsevol de les càmeres i realitzar les accions pertinents en cas que sigui necessari.

En alguns moments, sí que és cert que durant l'aparició d'algun error inesperat sigui en l'àmbit de codi o de la mateixa infraestructura de les plataformes, als desenvolupadors i encarregats de solucionar aquests errors se'ls hi és difícil detectar-ne l'origen, saber d'on prové, com s'ha originat i perquè, i és per això que en certa manera el projecte se centra a aportar una eina en forma de solució amb la finalitat que aporti informació als encarregats de solucionar els errors per tal que sigui molt més ràpid i àgil la detecció i solució d'aquests errors.

La informació prové del monitoratge de tot el procés d'ingesta de dades de les càmeres de les quals disposa Alphanet, passant per la lambda i la cua d'AWS, per continuar amb el processament de les dades i posteriorment el seu emmagatzematge a la base de dades. En definitiva, en el projecte es pretén analitzar i realitzar un monitoratge del flux de dades entre els diferents components que componen la infraestructura de les plataformes ADM i AM.

A més a més, actualment no existeix cap sistema de notificacions com a tal que permeti saber a l'instant el que està fallant dins de les plataformes o a qualsevol punt de la seva infraestructura i, per tant, el que es pretén amb el projecte és aportar en certa manera un canal de difusió a través de la plataforma Discord, amb l'afegit de l'enviament de les notificacions per correu electrònic. La notificació es crea un cop el sistema de monitoratge detecta algun error en el flux de dades i s'envia un missatge tant al canal de difusió com al correu.

Finalment, el fet de tenir un conjunt d'accions automatitzades que permeten en qualsevol moment fer un 'Start', 'Stop' o 'Restart' de qualsevol dels serveis o servidors de l'entorn de producció involucrats en el funcionament de les plataformes de l'empresa aporta una seguretat als desenvolupadors pel fet que en el moment que un servei o servidor no funciona correctament, automàticament s'executa l'acció determinada per evitar que es col·lapsi el sistema i permetent així que les plataformes continuïn funcionant de forma correcta.

3 Objectius i Abast

3.1 Objectius del client

- Conèixer en tot moment l'estat en el qual es troben els servidors i serveis de producció de les plataformes ADM i AM.
- Reduir el temps de solució d'errors dels servidors i serveis de producció de les plataformes ADM i AM.
- Millorar l'eficiència i eficàcia de detecció d'errors.
- Tenir la seguretat que el software de les plataformes és lliure d'errors.
- Conèixer a l'instant la detecció d'un error durant el procés d'ingesta de dades.
- Veure de forma visual l'estat del monitoratge dels servidors i serveis de producció de les plataformes ADM i AM.

3.2 Objectius del Testing

- Millorar la qualitat i robustesa del software de les plataformes de l'empresa.
- Permetre mantenir un software lliure d'errors i documentat d'acord amb els tests.
- Assegurar que el codi desenvolupat per cada plataforma és compatible amb el codi ja existent.
- Identificar errors de codi i bugs abans de pujar una nova versió de qualsevol de les dues plataformes.
- Comprovar que cada funció del codi compleix amb els requisits especificats.

3.3 Objectius del monitoratge

- Analitzar l'estat de la SQS tant d'ADM com d'AM.
- Monitorar l'estat de la lambda utilitzada per la ingesta de dades.
- Comprovar l'estat de la RDS tant d'ADM com d'AM.
- Vigilar l'espai disponible dels serveis i servidors vinculats tant a ADM com a AM.
- Controlar la utilització de la CPU de les instàncies a AWS.
- Inspeccionar l'ús de memòria de la Redis.
- Proporcionar gràfiques per analitzar l'estat global de la infraestructura d'ADM i d'AM.
- Generar notificacions en cas de detectar errors durant el monitoratge.

3.4 Usuaris

- Desenvolupadors del software de les plataformes ADM i AM.
- Personal encarregat de supervisar el monitoratge de les plataformes ADM i AM.
- Personal encarregat de testejar el software de les plataformes ADM i AM.

3.5 Indicadors clau (KPIs)

3.5.1 Testing

- Percentatge de detecció d'errors: mesurat com el nombre d'errors detectats durant l'execució dels tests dividit pel nombre total de tests executats.
- Cobertura de tests (Coverage): mesurat com el percentatge de codi de la plataforma que ha sigut executat pels tests.
- Temps d'execució de tests: mesurat com el temps total necessari per executar tots els tests.
- Percentatge de tests positius/negatius: mesurat com el percentatge de tests que s'executen amb un resultat positiu dividit pels tests que s'executen amb un resultat negatiu.
- Cobertura de requisits: mesurat com el percentatge de requisits que tenen casos de prova associats i amb un resultat positiu.

3.5.2 Monitoratge

- Nombre aproximat de missatges visibles a SQS: mesurat com el nombre aproximat de missatges acumulats a SQS en temps real.
- Nombre de missatges enviats des dels lectors de matrícules cap a la SQS.
- Nombre de missatges enviats des de la SQS cap al Processor.
- Nombre de missatges buits rebuts durant el procés d'ingesta de dades.
- Percentatge d'utilització de CPU de les instàncies dels serveis i servidors de producció ubicats a AWS en temps real.
- Nombre de correus electrònics enviats als clients d'ADM en temps real.
- Memòria utilitzada de la Redis.
- Espai d'emmagatzematge disponible de la RDS.
- Nombre de sessions actives a la RDS.
- Nombre d'invocacions de la funció de lambda d'AWS durant el procés d'ingesta de dades.
- Espai d'emmagatzematge disponible de les instàncies dels serveis i servidors de producció ubicats a AWS en temps real.

- Percentatge d'utilització de CPU de la RDS ubicada a AWS en temps real.
- Nombre de missatges enviats a SQS: mesurat com el nombre de missatges enviats a SQS en temps real.

4 Anàlisi de Referents

4.1 Eines a valorar en el desenvolupament del Testing

GitLab Pipelines [11]**:** són els components de més alt nivell de la integració, entrega i desplegament contínua. Permeten automatitzar el procés de construcció, prova i desplegament de software. Estan formats per Jobs els quals defineixen què s'ha de fer i Stages els quals defineixen quan s'executen els Jobs. Els Jobs són executats per Runners els quals es configuren dins de GitLab. En el cas que tots els Jobs s'executin correctament, aleshores el Pipeline es mou al següent Stage. En cas contrari, no s'executa el següent Stage i la Pipeline acaba la seva execució.

- **Avantatges:**
  - Com que el repositori on es penja el codi desenvolupat és GitLab i els Pipelines estan integrats amb aquest repositori, els Pipelines es poden configurar i executar fàcilment des del mateix entorn en el qual es desenvolupa el codi.
  - Permet automatitzar tasques repetitives com la compilació, les proves unitàries, les proves d'integració amb el propòsit d'estalviar temps i reduir errors humans.
  - Permet executar múltiples Jobs d'un mateix Stage en paral·lel.
- **Desavantatges:**
  - L'execució de Pipelines depèn de la infraestructura proporcionada per GitLab.
  - Corba d'aprenentatge en configurar i optimitzar els Pipelines.

JUnit [12]**:** és un framework de testing unitari per Java. S'utilitza principalment per escriure i executar proves automatitzades en components individuals de codi, com per exemple mètodes, classes o conjunts de classes.

- **Avantatges:**
  - Fàcil d'usar a l'hora de codificar tests unitaris de codi.
  - Integrat amb la majoria de IDE per Java, com Eclipse o IntelliJ IDEA.
  - Ofereix la possibilitat de repetir les proves, és a dir, es poden executar tantes vegades com sigui necessari.
- **Desavantatges:**
  - Complex en cas de voler desenvolupar proves unitàries complexes.

Mockito [13]**:** és una llibreria de tests de Java que s'utilitza per crear objectes simulats anomenats "mocks" en tests unitaris. Aquests objectes simulats s'usen per simular el comportament d'objectes reals dins del sistema durant els tests.

- **Avantatges:**
  - Proporciona flexibilitat a l'hora de crear "mocks" d'interfícies, classes abstractes i classes concretes.
  - Aporta una sintaxi clara i llegible, fet que permet que els tests siguin més comprensibles i fàcils de mantenir.
- **Desavantatges:**
  - No pot arribar a simular tots els possibles escenaris.
  - Si es fan servir "mocks" en excés, pot donar una falsa sensació de seguretat en els tests.

4.2 Eines a valorar en el desenvolupament del Monitoratge

Prometheus: és un sistema de monitoratge i alerta de codi obert dissenyat per monitorar sistemes i serveis de forma escalable i confiable. Està enfocat en la recopilació i emmagatzematge de mètriques en temps real, anàlisi de tendències i visualització de dades.

- **Avantatges:**
  - Utilitza un model de dades multidimensional, el qual proporciona una gran flexibilitat a l'hora d'organitzar i consultar dades.
  - Consta d'una arquitectura escalable que permet treballar amb grans volums de mètriques.
  - Ofereix la possibilitat d'integrar-se fàcilment amb eines de visualització de dades.
- **Desavantatges:**
  - Corba d'aprenentatge gran a l'hora d'utilitzar l'eina.

Grafana: és una eina de visualització de dades de codi obert que s'utilitza per crear panells de control i gràfics interactius per monitorar i analitzar dades.

- **Avantatges:**
  - S'integra fàcilment amb un ventall molt ampli de fonts de dades, com per exemple Prometheus.
  - Interfície intuïtiva i fàcil d'usar que permet crear panells de control i gràfics interactius amb facilitat.
  - Gran varietat de visualitzacions i personalització.
- **Desavantatges:**
  - Pot arribar a consumir una quantitat significativa de recursos del sistema.

5 Metodologia

En primer lloc, es realitza una etapa d'investigació, recopilació i cerca d'informació sobre el desenvolupament de Testing Unitari i Testing d'Integració amb l'afegit del monitoratge de tota la infraestructura d'una empresa tecnològica. A més a més, es fa una recerca d'informació sobre el desenvolupament d'un canal de difusió a la plataforma Discord, l'enviament de correus electrònics un cop es generen alertes a Grafana i la forma de gestionar un conjunt d'accions automatitzades per tal que un cop es genera una alerta provinent del Grafana, automàticament s'executi l'acció o les accions en qüestió.

Una vegada s'ha recopilat tota la informació, es duu a terme una etapa d'anàlisi i síntesi per avaluar i analitzar la informació recopilada.

- **Anàlisi de contingut:** s'apliquen tècniques d'anàlisi de contingut per identificar els temes més importants i tendències en la informació recopilada.
- **Anàlisi estadístic:** s'utilitzen eines estadístiques per analitzar les dades recopilades, com per exemple estadístiques referents al monitoratge d'infraestructura d'empreses tecnològiques com poden ser serveis, servidors o estadístiques referents a l'ús de tests per obtenir un software més confiable i documentat.
- **Anàlisi de casos:** s'estudien casos específics d'ús de tests en empreses de software i empreses que utilitzen el monitoratge com a eina per analitzar l'estat de la seva infraestructura.

Una vegada s'ha dut a terme l'anàlisi i la síntesi, es duu a terme el disseny i la planificació. En aquesta etapa, s'usen les dades recollides en l'etapa d'anàlisi per dissenyar un pla d'acció amb la finalitat d'assolir els objectius establerts, repartits en tasques i per ordre de prioritat. Dins d'aquesta etapa s'inclou la definició d'objectius tant a curt com a llarg termini.

Un cop acabada la part de disseny i planificació, es fan servir eines de gestió com "Jira Software" amb l'ajut d'Alphanet, per organitzar les tasques a fer, saber l'estat en el qual es troben i poder fer un seguiment diari de la seva evolució.

A continuació, es duu a terme la fase de desenvolupament i implementació. Durant tota aquesta fase, es fa servir l'eina de control i emmagatzematge de codi anomenat GitLab. Dins d'aquesta fase, s'inclou la part de Testing, la part de monitoratge, la part de les notificacions i la part de les accions automatitzades.

En primer lloc, pel que fa a la part de Testing, es codifiquen totes els tests unitaris i d'integració tant de la plataforma ADM com AM.

En segon lloc, pel que fa a la part de monitoratge de tota la infraestructura que engloba les plataformes ADM i AM, es desenvolupa el codi per aconseguir monitorar correctament la infraestructura i s'integren aquelles eines que ajuden a monitorar d'una forma més clara i senzilla, com són Grafana i Prometheus.

A continuació, en relació amb la part de les notificacions, es desenvolupa el codi per poder enviar notificacions a la plataforma Discord a través de canals de difusió i l'enviament de correus quan es generin alertes a Grafana.

Finalment, en relació amb la part de les accions automatitzades, es desenvolupa el codi tant al Backend com al Frontend de la plataforma AM per gestionar el conjunt d'accions i utilitzar-les quan faci falta.

Després d'acabar la fase de desenvolupament i implementació, s'entra a la fase de seguiment i avaluació. En aquesta fase es fa un seguiment constant del monitoratge de tota la infraestructura que engloba les plataformes per validar el seu correcte funcionament, a més d'avaluar els tests desenvolupats en la fase anterior, l'estat del canal de difusió, l'enviament de correus i la correcta utilització de les accions automatitzades.

6 Definició de requeriments funcionals i tecnològics

6.1 Requeriments Funcionals

- **Cas 1:** Executar un conjunt de tests i avaluar-ne el resultat.
- **Cas 2:** Mostrar i visualitzar gràfics del monitoratge de la infraestructura de les plataformes i extreure'n informació rellevant.
- **Cas 3:** Enviar una notificació al canal de difusió de Discord per informar de la detecció d'un error durant el monitoratge.
- **Cas 4:** Enviar un correu electrònic a les adreces de correu pertinents per informar de la detecció d'un error durant el monitoratge.
- **Cas 5:** Executar una o diverses accions automatitzades un cop s'ha generat una alerta des de Grafana.

6.2 Requeriments Tecnològics

- Dotar al monitoratge dels suficients recursos per suportar increments sobtats de dades durant la seva ingesta.
- Mantenir els tests unitaris i d'integració actualitzats i alineats amb el software de les plataformes.
- Utilitzar tecnologies i eines conegudes dins de l'empresa per garantir un correcte funcionament del sistema.
- Capacitar als equips per poder realitzar el monitoratge correctament de tota la infraestructura que engloba les dues plataformes de l'empresa.
- Integrar el sistema de monitoratge amb la infraestructura actual de l'empresa.

6.3 Requeriments de Testing

- Dur a terme tests per verificar la robustesa i estabilitat del software desenvolupat juntament amb el software ja existent.
- Assegurar que totes les funcionalitats especificades en els requeriments de les plataformes de l'empresa tenen tests associats.
- Verificar que els tests unitaris desenvolupats es relacionen amb cada mòdul i component de forma individual tant per ADM com per AM.
- Comprovar que els tests d'integració desenvolupats es relacionen amb diferents mòduls de cada plataforma i funcionen correctament.

6.4 Requeriments de Monitoratge

- Monitorar la ingesta de dades d'ADM. Això inclou la SQS, les instàncies d'AWS relacionades amb aquest procés, els correus electrònics enviats als clients d'ADM, la Redis, la RDS i la lambda.
- Monitorar l'arquitectura d'ADM. Això inclou la RDS i les instàncies d'AWS relacionades amb l'arquitectura de la plataforma com són els serveis i servidors.
- Monitorar la ingesta de dades d'AM. Això inclou la SQS, les instàncies d'AWS relacionades amb aquest procés, la RDS i la lambda.
- Monitorar l'arquitectura d'AM. Això inclou únicament la RDS.

7 Casos d'ús

Cas 1 - Requisit Funcional: Executar un conjunt de tests i avaluar-ne el resultat.

Actor/s: Desenvolupador encarregat d'escriure codi nou o modificar parts de codi ja existent.

Precondicions: Els tests referents al codi ja existent estan desenvolupats.

Flux normal d'execució:

1. El desenvolupador escriu el codi nou o modifica les parts pertinents del codi ja existent en relació amb la seva tasca assignada.
2. El desenvolupador escriu els tests relacionats amb el codi.
3. El desenvolupador executa el conjunt de tests desenvolupats i avalua els resultats obtinguts per prendre posteriorment les decisions pertinents.

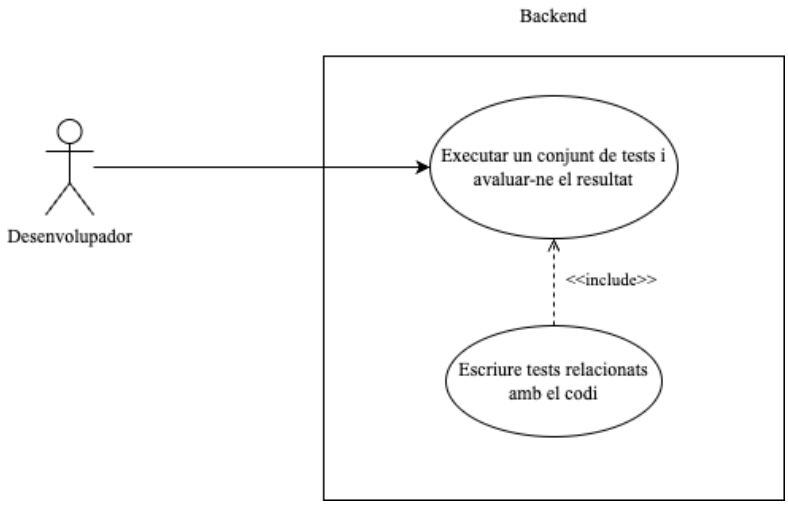

Figura 3. Cas d'ús executar i avaluar tests. Font: Elaboració pròpia

Cas 2 - Requisit Funcional: Mostrar i visualitzar gràfics del monitoratge de la infraestructura de les plataformes i extreure'n informació rellevant.

Actor/s: Usuari de la plataforma.

Precondicions: L'usuari ha accedit a la plataforma amb les seves credencials.

Flux normal d'execució:

1. L'usuari accedeix a l'apartat 'Governor' i espera a la càrrega dels gràfics.
2. La plataforma obté els gràfics i la informació de cadascun d'ells de Grafana i carrega els gràfics.
3. L'usuari visualitza els gràfics i obté la informació en temps real del funcionament de la infraestructura.

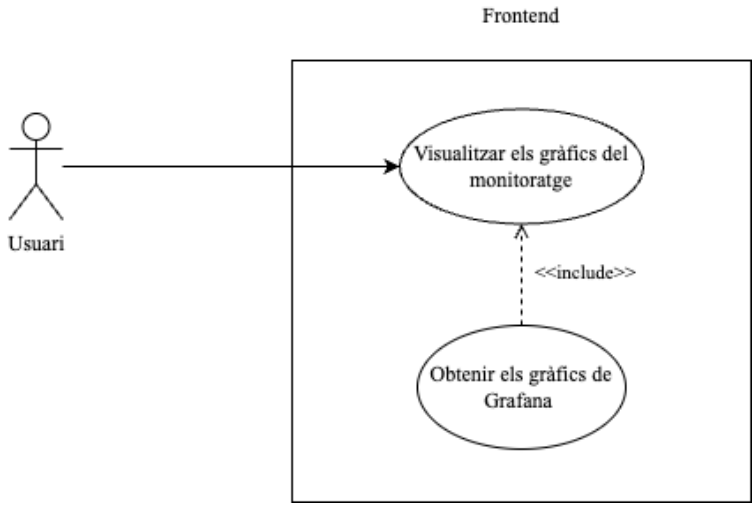

Figura 4. Cas d'ús visualitzar els gràfics del monitoratge. Font: Elaboració pròpia

Cas 3 - Requisit Funcional: Enviar una notificació al canal de difusió de Discord per informar de la detecció d'un error durant el monitoratge.

Actor/s: (Execució automàtica de Grafana en generar una alerta), Administrador del sistema.

Precondicions: El monitoratge de Grafana està actiu, la integració de Discord amb Grafana està configurada i l'administrador del sistema té accés a Discord.

Flux normal d'execució:

1. Grafana genera una alerta a causa d'un error durant el monitoratge de les plataformes.
2. Grafana envia la notificació corresponent a l'alerta generada al canal de difusió de Discord.
3. L'administrador rep la notificació de l'alerta al Discord i actua en conseqüència per a solucionar el problema.

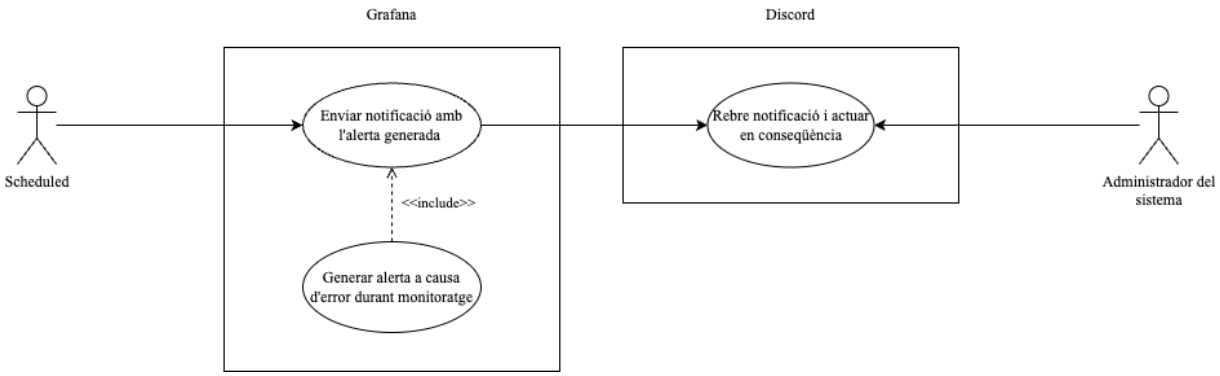

Figura 5. Cas d'ús enviar notificació al Discord. Font: Elaboració pròpia

Cas 4 - Requisit Funcional: Enviar un correu electrònic a les adreces de correu pertinents per informar de la detecció d'un error durant el monitoratge.

Actor/s: (Execució automàtica de Grafana en generar una alerta), Administrador del sistema.

Precondicions: El monitoratge de Grafana està actiu, la integració d'Email amb Grafana està configurada i l'administrador del sistema té accés al correu electrònic.

Flux normal d'execució:

1. Grafana genera una alerta a causa d'un error durant el monitoratge de les plataformes.
2. Grafana envia la notificació corresponent a l'alerta generada al correu electrònic configurat durant la integració.
3. L'administrador rep el correu electrònic i actua en conseqüència per a solucionar el problema.

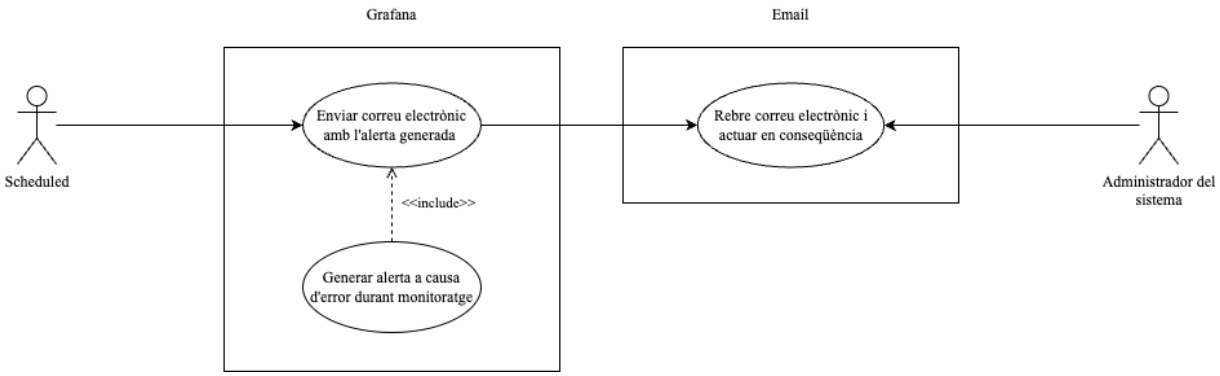

Figura 6. Cas d'ús enviar correu electrònic. Font: Elaboració pròpia

Cas 5 - Requisit Funcional: Executar una o diverses accions automatitzades un cop s'ha generat una alerta des de Grafana.

Actor/s: (Execució automàtica de Grafana en generar una alerta), Sistema (Backend)

Precondicions: El monitoratge de Grafana està actiu, la integració del Webhook el qual està enllaçat amb l'endpoint del Backend està configurada i el sistema es troba en execució.

Flux normal d'execució:

1. Grafana genera una alerta a causa d'un error durant el monitoratge de les plataformes.
2. Grafana envia la notificació corresponent a l'alerta generada a l'endpoint del Backend.
3. El sistema passa per tot el fil d'execució i executa les accions automatitzades pertinents.

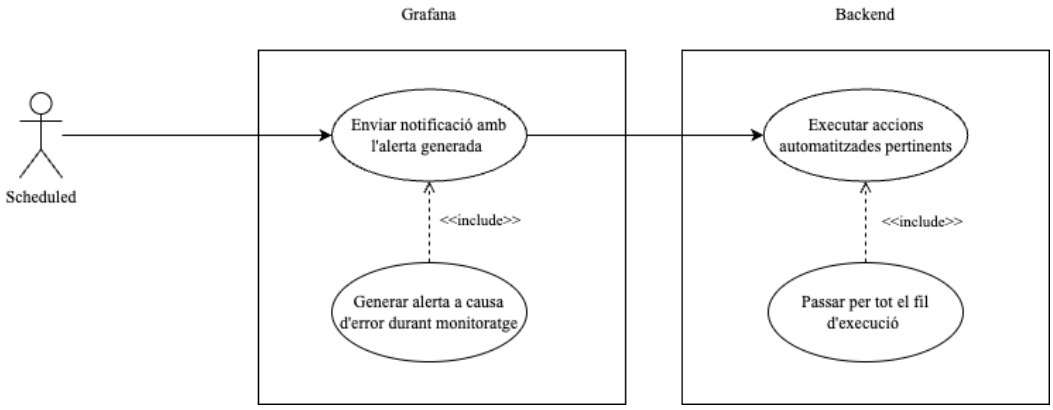

Figura 7. Cas d'ús executar accions automatitzades. Font: Elaboració pròpia

8 Anàlisi i disseny

Per la realització del projecte, s'ha plantejat la creació de dues noves taules anomenades 'Action' i 'Metric' a la base de dades de la plataforma AM per tal d'assolir els objectius plantejats en aquest treball. Després de dur a terme una primera anàlisi de la base de dades existent tant de la plataforma ADM com de la plataforma AM, s'ha arribat a la conclusió de la necessitat de crear aquestes dues taules per tenir a l'abast un llistat amb totes les accions automatitzades i la informació referent als gràfics de Grafana relacionats amb el monitoratge dels serveis de la infraestructura de les plataformes respectivament.

Per una banda, la taula 'Action', la qual fa referència al conjunt d'accions automatitzades que s'executen quan es generen certes alertes des de Grafana. Aquestes accions automatitzades s'emmagatzemen en una sola taula per tenir una informació clara i ben organitzada sobre el seu estat i l'ús que se'n fa.

Tal com es pot observar a la figura 8, s'utilitza una id composta, amb els atributs key_id i service_id; aquest últim fa referència a l'atribut id de la taula 'Service'. Pel que fa a l'atribut key_id, fa referència a l'etiqueta de l'alerta generada per Grafana i que defineix a quina part de la infraestructura es refereix l'acció en qüestió, per exemple, ADM-SQS o AM-RDS. La relació entre les taules 'Action' i 'Service' és la següent: una acció té associat un únic servei, però un servei pot tenir associades més d'una acció.

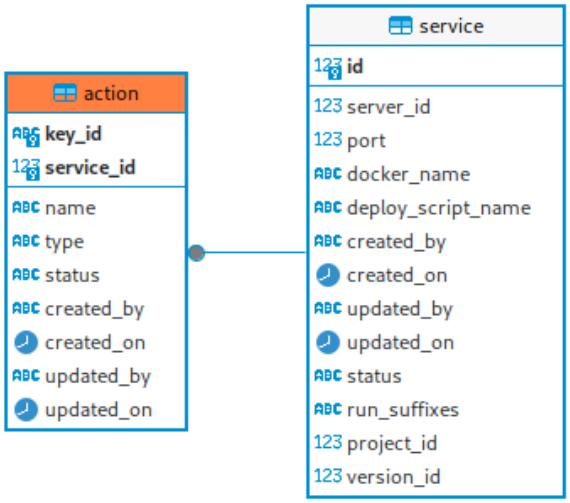

Figura 8. Diagrama de la relació Action - Service. Font: Elaboració pròpia

D'altra banda, la taula 'Metric', la qual es refereix al conjunt de mètriques (gràfics de monitoratge) que té associades un servei. A través d'aquesta taula, es pot saber amb exactitud quantes mètriques té cada servei i on es poden trobar els gràfics relacionats amb aquestes mètriques.

Tal com es pot observar a la figura 9, s'utilitza una id que es genera automàticament a través d'una seqüència, a més dels atributs service_id i grafana_url. El primer fa referència a l'atribut id de la taula 'Service', mentre que el segon fa referència a l'URL de Grafana del gràfic. La relació entre les taules 'Metric' i 'Service' és la següent: una mètrica té associada un únic servei, però un servei pot tenir associades més d'una mètrica.

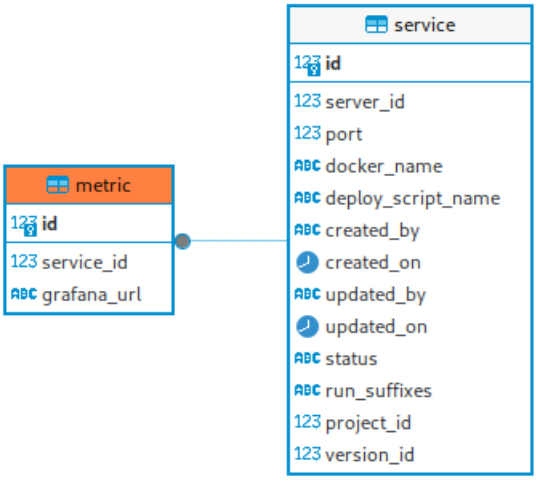

Figura 9. Diagrama de la relació Metric - Service. Font: Elaboració pròpia

9 Desenvolupament

9.1 Entorns

9.1.1 Entorn de desenvolupament

Aquest entorn sempre és l'inici del desenvolupament de totes les funcionalitats de les plataformes de l'empresa i consisteix en un entorn en local on els serveis i les eines a utilitzar per al correcte desenvolupament de les funcionalitats es troben a la mateixa màquina on es desenvolupa el codi, és a dir, en el mateix ordinador en local. Una de les eines que es troba en local és la base de dades PostgreSQL, la qual ofereix la possibilitat d'inserir qualsevol dada sense tenir cap afectació en el funcionament real de les plataformes, ja que tal com s'ha esmentat anteriorment, tots els serveis i eines que s'usen en aquest entorn es troben en la mateixa màquina on es desenvolupa el codi.

A l'entorn també s'hi pot trobar el Backend i el Frontend de les plataformes, els quals es fan servir per desenvolupar una primera idea de la funcionalitat i visualitzar-la a la mateixa plataforma respectivament.

En relació amb la plataforma ADM, també s'hi troba l'ADM Processor, la Redis per guardar els duplicats, l'ADM OpenSearch i Amazon S3 per guardar les imatges capturades pels lectors. A l'entorn de desenvolupament, es fa ús de la versió de testing tant per ADM OpenSearch com per Amazon S3.

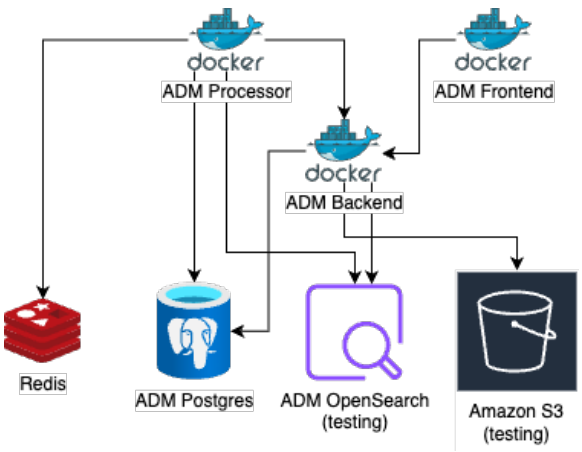

Figura 10. Diagrama de l'entorn de desenvolupament d'ADM. Font: Elaboració pròpia

En relació amb la plataforma AM, hi podem trobar a part d'AM Backend i AM Frontend, el Publisher amb el Gitlab Runner els quals s'utilitzen per pujar versions de codi i automatitzar els processos. Finalment, també hi ha AM Redis, AM Postgres i AM OpenSearch que en aquest entorn s'usen amb una versió en local.

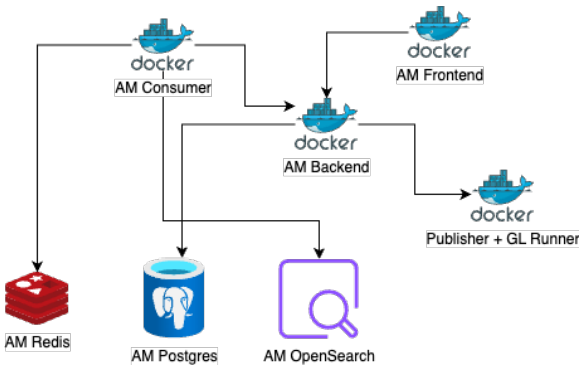

Figura 11. Diagrama de l'entorn de desenvolupament d'AM. Font: Elaboració pròpia

9.1.2 Entorn de test

Aquest entorn s'utilitza per provar el comportament de les noves funcionalitats desenvolupades amb dades més properes a les reals per tenir una idea de com es veuen implementades al Frontend un cop es puja el codi a l'entorn de producció. Els serveis ja no es troben en local, sinó que es troben en màquines a AWS per facilitar el seu desplegament i el seu ús. Tot i això, a la màquina en local sí que hi ha la Postgres de test.

En relació amb la plataforma ADM, les màquines d'ADM Processor, ADM Frontend i ADM Backend es troben a AWS en comptes d'estar en local. La versió d'ADM OpenSearch i Amazon S3 és la mateixa que en local, ja que es fa ús de la de testing.

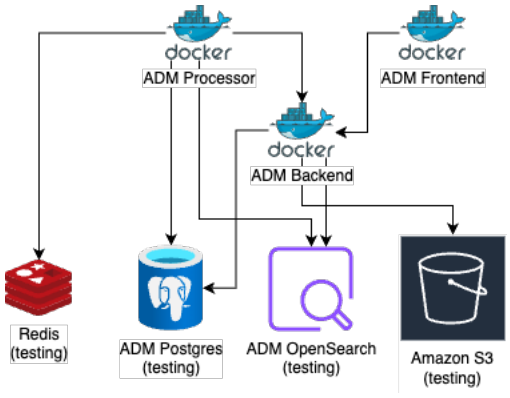

Figura 12. Diagrama de l'entorn de test d'ADM. Font: Elaboració pròpia

En relació amb la plataforma AM, les màquines AM Consumer, AM Frontend, AM Backend i el Publisher amb GitLab Runner es troben a AWS. Pel que fa a AM OpenSearch i a la Redis, s'utilitzen les versions de testing corresponents.

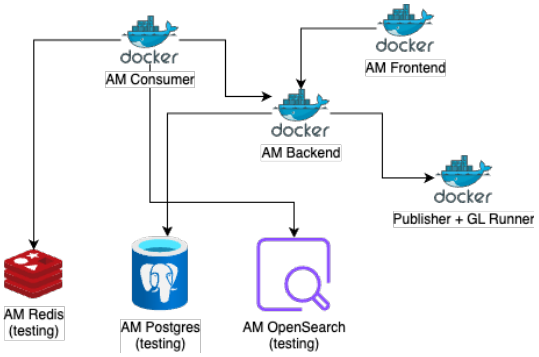

Figura 13. Diagrama de l'entorn de test d'AM. Font: Elaboració pròpia

9.1.3 Entorn de producció

Aquest entorn és el final del desenvolupament i la implementació de funcionalitats de les plataformes ADM i AM. Un cop el codi es troba a l'entorn de producció, ja està operatiu i en funcionament per tots els clients d'ADM i tots els tècnics d'AM. Per això, és imprescindible tenir un pas previ entre el desenvolupament inicial de les funcionalitats i la implementació final, ja que s'assegura una compatibilitat amb el codi ja existent de les plataformes.

En relació amb la plataforma ADM, les màquines d'ADM Processor, ADM Frontend i ADM Backend es troben a AWS, però no són les mateixes que les que s'utilitzen a l'entorn de test, sinó que es troben en altres servidors diferents. Pel que fa al ADM OpenSearch, ADM Postgres i la Redis, tenen les seves pròpies versions de producció i, per tant, és on es troben les dades reals que provenen dels lectors de matrícules.

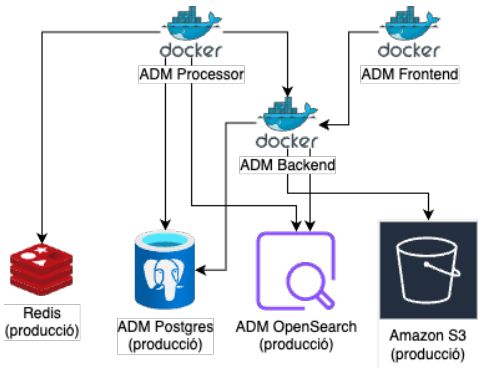

Figura 14. Diagrama de l'entorn de producció d'ADM. Font: Elaboració pròpia

En relació amb la plataforma AM, les màquines AM Consumer, AM Frontend, AM Backend i el Publisher amb GitLab Runner es troben a AWS, però no són les mateixes que les que s'utilitzen a l'entorn de test, sinó que es troben en altres servidors diferents. Pel que fa a AM OpenSearch, AM Postgress i la Redis, s'utilitzen les versions de producció corresponents i, per tant, és on es troben les dades reals que provenen dels lectors de matrícules.

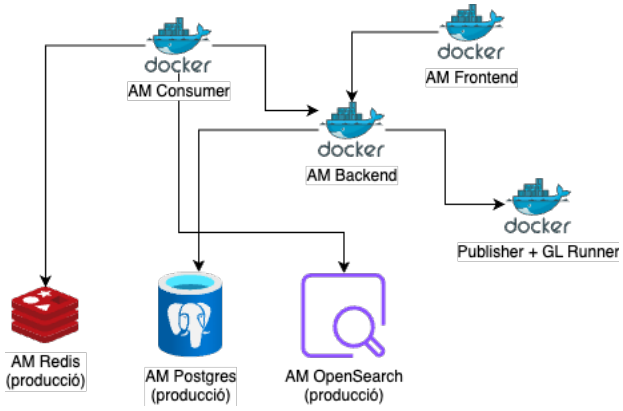

Figura 15. Diagrama de l'entorn de producció d'AM. Font: Elaboració pròpia

9.2 Testing

9.2.1 Disseny i modelat

En relació amb el disseny i modelat dels tests, en primer lloc, es crea un "package" amb nom test dins de la mateixa estructura del codi. A continuació, es creen tres "package":

- **Config:** en aquest apartat es configuren els "Bean" que s'utilitzen durant l'execució dels tests. A més a més, també es desenvolupa codi amb funcionalitats que s'usen durant els tests. Finalment, es defineixen les interfícies i classes que es fan servir més durant els tests amb l'objectiu de tenir un codi més ben estructurat.
- **Support:** en aquest apartat es configura la "annotation" que s'usa durant tots els tests i es creen les classes que s'utilitzen als tests com a "mocks" de les entitats del model.
- **Test:** en aquest apartat és on es desenvolupen tots els tests. S'estructuren a partir de "packages" en funció de la interfície o classe a la qual fan referència. Hi ha una classe que s'anomena "BaseTest" en la qual es defineixen tots els mètodes que es fan ús a posteriori en els tests a més interfícies i atributs.

En el desenvolupament de qualsevol test, se segueix el patró Arrange, Act, Assert (AAA), el qual es basa en tres simples passes que defineixen qualsevol test. La secció Arrange inicialitza els objectes i defineix el valor de les dades que s'utilitzen en el test. La secció Act invoca al mètode en qüestió amb els paràmetres corresponents. Finalment, la secció Assert comprova el valor esperat amb el valor obtingut de l'execució del test.

Els projectes on s'han desenvolupat tests són ADM Backend, AM Backend i ADM Processor. A cada projecte se segueixen les passes descrites anteriorment per dur a terme el testing, amb l'afegit que es crea un fitxer de configuració anomenat "application-test.yml" en el qual es defineixen la base de dades a fer servir, l'OpenSearch, la Redis, etc.

9.2.2 Automatització dels tests

Des del GitLab, s'ofereix la possibilitat d'automatitzar els tests desenvolupats en el codi de cada projecte a través de Runners i Pipelines. En primer lloc, es crea un Runner el qual s'ubica en la màquina del Publisher a AWS. En segon lloc, es crea un fitxer anomenat ".gitlab-ci.yml" dins del projecte el qual es vol automatitzar els tests i es defineixen "stages", variables, "jobs", "artifacts", "rules" i notificacions que es dirigeixen als canals de difusió. Amb l'ajut d'aquest fitxer, quan un codi es puja al repositori de GitLab o quan se sol·licita un "merge request", s'executa aquest fitxer que a la vegada construeix els tests definits al codi i posteriorment els executa. Finalment, al GitLab s'observa un "report" amb el resum dels tests executats i els resultats obtinguts.

9.3 Monitoratge

9.3.1 Actuator

Al capdavall, Actuator [14] aporta un conjunt de característiques molt útils durant el desenvolupament i el posterior ús d'una aplicació software. Alguns dels trets claus que proporciona aquesta llibreria són la supervisió d'aplicacions, la recopilació de mètriques i el coneixement de l'estat de la base de dades en temps real. Amb l'exposició d'informació operativa referent a l'aplicació en execució a través dels endpoints HTTP que proporciona Actuator, l'usuari pot analitzar informació referent a la salut de l'aplicació, les mètriques que proporciona la mateixa llibreria, el nombre de sessions actives, les propietats actuals de l'entorn, etc.

Amb la utilització d'Actuator a AM Backend s'aconsegueix obtenir mètriques del rendiment del servei, ja sigui la memòria utilitzada, el nombre de threads, el nombre de peticions, etc. S'implementa la llibreria a l'arxiu "build.gradle" d'AM Backend i a través de l'endpoint /metrics es recopila el conjunt de mètriques. Un exemple del conjunt de mètriques obtingut a través de Postman és:


```
{
    "names": [
        "application.ready.time",
        "application.started.time",
        "disk.free",
        "disk.total",
        "executor.active",
        "executor.completed",
        "executor.pool.core",
        "executor.pool.max",
        "executor.pool.size",
        "executor.queue.remaining",
        "executor.queued",
        "hikaricp.connections",
        "hikaricp.connections.acquire",
        "hikaricp.connections.active",
        "hikaricp.connections.creation",
        "hikaricp.connections.idle",
        "hikaricp.connections.max",
        "hikaricp.connections.min",
        "hikaricp.connections.pending",
        "hikaricp.connections.timeout",
        "hikaricp.connections.usage",
        "http.server.requests",
        "jdbc.connections.max",
        "jdbc.connections.min",
        "jvm.buffer.count",
        "jvm.buffer.memory.used",
        "jvm.buffer.total.capacity",
        "jvm.classes.loaded",
        "jvm.classes.unloaded",
        "jvm.gc.live.data.size",
        "jvm.gc.max.data.size",
        "jvm.gc.memory.allocated",
        "jvm.gc.memory.promoted",
        "jvm.gc.overhead",
        "jvm.gc.pause",
```


Figura 16. Mètriques d'actuator. Font: Elaboració pròpia

Per obtenir el valor d'una mètrica en concret, l'URL és /metrics/<nom_de_la_mètrica>. Per exemple, quan es vol obtenir l'espai total del disc, l'URL és /metrics/disk.total i llavors el resultat que s'obté amb el Postman és:


```
{
    "name": "disk.total",
    "description": "Total space for path",
    "baseUnit": "bytes",
    "measurements": [
        {
            "statistic": "VALUE",
            "value": 2.41285562368E11
        }
    ],
```


Figura 17. Mètrica d'espai total del disc. Font: Elaboració pròpia

Un altre exemple és quan es vol obtenir les peticions al servidor, llavors la mètrica que s'utilitza és "http.server.requests". El resultat que s'obté amb el Postman és:


```
{
    "name": "http.server.requests",
    "description": null,
    "baseUnit": "seconds",
    "measurements": [
        {
            "statistic": "COUNT",
            "value": 6.0
        },
        {
            "statistic": "TOTAL_TIME",
            "value": 0.253167029
        },
        {
            "statistic": "MAX",
            "value": 0.0
        }
    ],
    "availableTags": [
        {
            "tag": "exception",
            "values": [
                "None"
            ]
        },
        {
            "tag": "method",
            "values": [
                "GET"
            ]
        },
        {
            "tag": "uri",
            "values": [
                "/actuator",
                "/actuator/metrics/{requiredMetricName}",
                "/actuator/metrics"
            ]
        },
        {
            "tag": "outcome",
            "values": [
                "SUCCESS"
            ]
        },
        {
            "tag": "status",
            "values": [
                "200"
            ]
        }
    ]
}
```


Figura 18. Mètrica del nombre de peticions HTTP al servidor. Font: Elaboració pròpia

L'ús principal d'Actuator en aquest projecte és personalitzar una mètrica per poder monitorar-la a posteriori. Concretament, la mètrica correspon a l'espai d'emmagatzematge disponible dels servidors. L'ús de MeterRegistry i el mètode gauge(), permeten personalitzar la mètrica per poder-la registrar a Prometheus i des d'allà, Grafana pot obtenir els valors de la mètrica i representar-la en un gràfic per poder dur a terme el monitoratge.

9.3.2 Prometheus

Per la utilització de Prometheus, s'ha instal·lat el servei en un docker a AWS i s'ha configurat un fitxer de configuració "prometheus.yml" amb els següents paràmetres:

- **Job_name:** es configura el nom del procés d'obtenció de la mètrica.
- **Metrics_path:** és l'URL d'on s'obté la mètrica la qual s'ha configurat prèviament en el Backend.
- **Scrape_interval:** és l'interval de temps des que s'aconsegueix el valor de la mètrica fins que es torna a obtenir.
- **Targets:** es configura l'adreça IP amb el port corresponent del Backend.
- **Authorization:** es configura el tipus d'autentificació i les credencials per poder aconseguir les mètriques amb certa seguretat.

Un cop s'ha configurat Prometheus, es fa restart del docker per aplicar correctament els canvis i a continuació es comencen a obtenir els resultats de la mètrica. Aquests resultats es guarden a Prometheus perquè després Grafana els pot recopilar i representar en un gràfic.

9.3.3 Grafana

Per utilitzar Grafana, s'ha instal·lat el servei en un docker a AWS i pel seu correcte funcionament, s'ha configurat diversos paràmetres en el fitxer de configuració "grafana.ini", com són:

- **Allow_embedding:** es configura amb el valor True per poder incrustar els gràfics al Frontend de l'aplicació.
- **Anonymous_enabled:** es configura amb el valor True per poder visualitzar correctament els gràfics al Frontend de l'aplicació.

- **Anonymous_org_name:** es configura el nom de l'organització per poder accedir als gràfics.
- **Anonymous_org_role:** es configura el rol de l'organització per poder accedir als gràfics.
- **Protocol:** es configura el tipus de protocol web, ja sigui HTTP, HTTPS, etc. En aquest cas, es configura amb el valor HTTPS per tenir un lloc web segur i, per tant, es necessita un fitxer i unes claus de seguretat que es troben als paràmetres 'cert_file' i 'cert_key' respectivament.
- **Root_url:** es configura l'adreça URL per la qual es podrà accedir als gràfics de Grafana. En aquest cas, l'URL és https://grafana.alphadatamanager.com:3000/
- **Smtp:** es configura tota la informació necessària per poder enviar correus electrònics des de Grafana, com per exemple l'adreça d'origen, el host, la contrasenya del correu o l'usuari que envia el correu.

En relació amb els dashboards, s'ha creat quatre dashboards que són els següents:

- **Ingesta ADM:** inclou gràfics relacionats amb SQS, EC2, Redis, RDS, Lambda i Emails.
- **Ingesta AM:** inclou gràfics relacionats amb SQS, EC2 i RDS.
- **Arquitectura ADM:** inclou gràfics relacionats amb EC2 i RDS.
- **Arquitectura AM:** inclou gràfics relacionats amb RDS.

A més a més, en relació amb la visualització de les mètriques de cada servei, s'ha creat un cinquè dashboard (Actuator Dashboard) amb el qual s'observa el percentatge de memòria utilitzada, l'ús de la CPU, el nombre de threads o el temps d'ús de connexió.

Pel que fa a l'apartat de connexions, s'ha configurat un conjunt de "Data sources" des dels quals s'obté les mètriques que es representen en els dashboards mencionats anteriorment. A cada "Data source" s'ha configurat un conjunt de paràmetres per poder tenir accés a les mètriques. Alguns d'aquests paràmetres són credencials, regions, adreces IP, etc. Els "Data sources" configurats són:

- **PostgreSQL d'ADM i AM**
- **CloudWatch d'Amazon**
- **OpenSearch**
- **Prometheus**

- **Redis**

En relació amb la configuració d'alertes perquè quan un valor d'un gràfic superi un llindar establert salti l'alerta en qüestió, s'ha definit un conjunt de "Alert rules" les quals es van avaluant en un període de temps establert prèviament en la configuració. Les alertes tenen tres estats possibles:

- **Normal:** el gràfic que té associada aquesta "Alert rule" està en bon estat, és a dir, el que s'està monitorant en el gràfic en qüestió està funcionant correctament.
- **Pending:** fa referència al fet que s'ha superat el llindar definit prèviament i, per tant, el que s'està monitorant en el gràfic en qüestió pot no estar funcionant correctament.
- **Firing:** fa referència al fet que el gràfic que s'està monitorant encara no ha recuperat l'estat òptim i, en conseqüència, s'ha d'enviar una alerta als canals de difusió avisant que el que s'està monitorant no està funcionant correctament.

El pas entre Normal i Pending es realitza quan s'ha superat el llindar establert en una avaluació de l'"Alert rule", mentre que el pas de Pending a Firing es realitza quan s'ha tornat a avaluar l'"Alert rule" i el que s'està monitorant encarà no ha recuperat el seu valor òptim.

Finalment, es pot tornar a l'estat Normal des de Firing quan el valor que retorna la mètrica del gràfic en qüestió es troba per sota del llindar establert. Aleshores, s'envia una notificació al Discord i un correu electrònic avisant que s'ha recuperat l'estat del monitoratge.

En funció del que s'està monitorant, l'alerta s'ubica en una carpeta o en una altra i el seu temps d'avaluació canvia, ja que per exemple el monitoratge del nombre de missatges acumulats a la cua (Number of messages visible), interessa que es monitori més sovint que en comptes de la utilització de la CPU de la RDS, per tant, la primera mètrica té un interval de temps d'avaluació menor que la segona.

Dins de les carpetes, hi pot haver més d'una alerta, ja que per exemple es pot avaluar el percentatge d'ús de la CPU de la RDS tant de la plataforma ADM com de la plataforma AM. En aquests casos, les dues alertes tenen el mateix interval de temps d'avaluació definit prèviament per la carpeta.

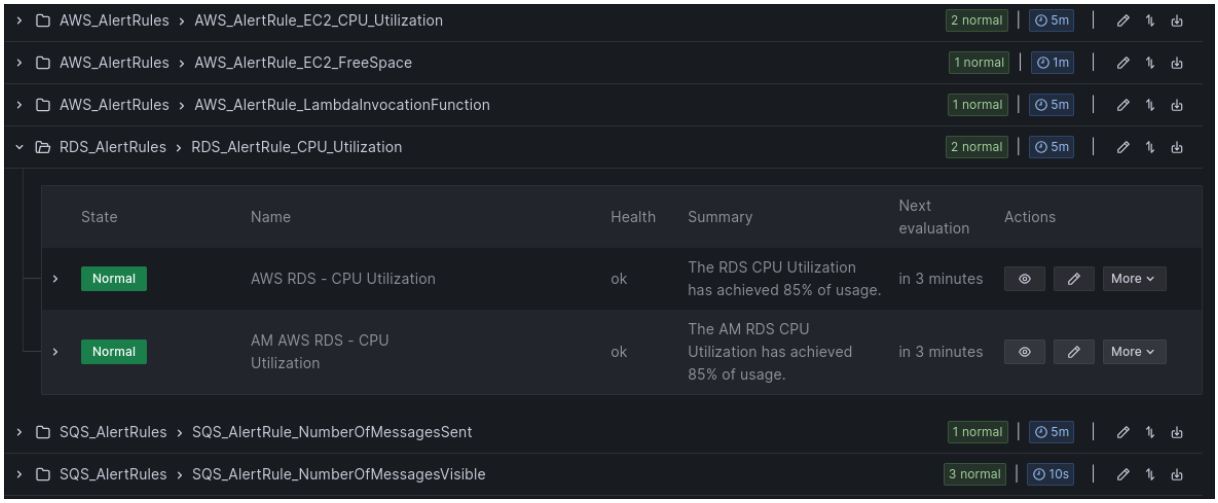

Figura 19. Llistat d'Alert rules configurades. Font: Elaboració pròpia

A la figura 20 es pot observar un exemple d'una "Alert rule", concretament s'està monitorant el nombre de missatges enviats dels lectors cap a la SQS. L'estat és Normal i el llindar està definit en 50000 missatges, és a dir, si el nombre de missatges enviats supera aquest llindar, l'estat passa a Pending.

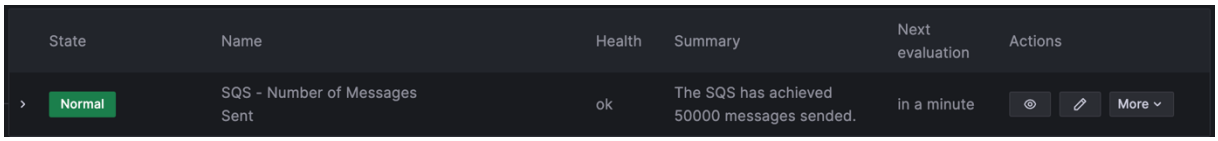

Figura 20. Exemple d'Alert rule. Font: Elaboració pròpia

Un cop s'ha configurat les alertes, després es configuren els "Contact points" que són els canals on s'enviaran les notificacions d'alertes provinents de Grafana. En aquest cas, s'ha configurat dos "Contact points" que són 'Automated Actions - Grafana' i 'Notification contact points'.

El primer s'utilitza per l'execució d'accions automatitzades a través d'un Webhook que apunta a un endpoint configurat a AM Backend, tot i que també s'envien les notificacions d'alerta pertinents al Discord i correu electrònic.

El segon s'utilitza per enviar les notificacions de les alertes generades del monitoratge de la infraestructura de les plataformes ADM i AM a la plataforma Discord i també per enviar correus electrònics amb la informació referent a l'alerta generada.

En resum, l'única diferència entre els dos "Contact points" configurats és que un s'utilitza per enviar una petició HTTP a AM Backend per executar accions automatitzades i l'altre no, però tots dos envien notificacions al Discord i correus electrònics.

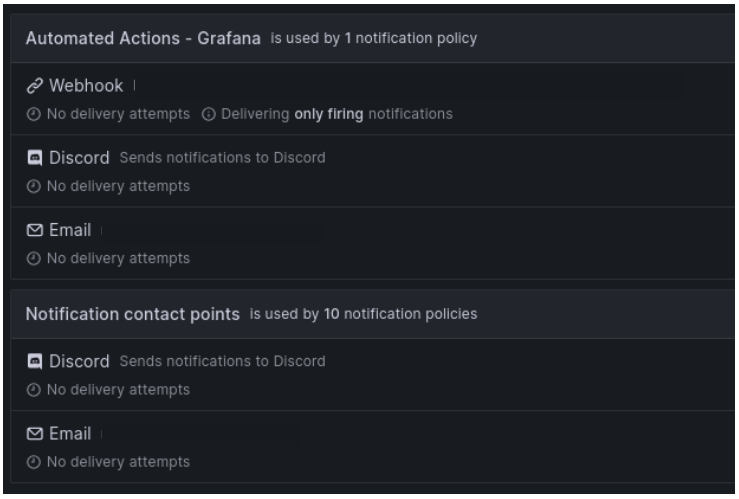

Figura 21. Llistat de Contact points definits a Grafana. Font: Elaboració pròpia

A la figura 22 es pot observar la configuració de la integració de Discord i les etiquetes "template" que fan referència a una personalització del missatge que s'envia a la notificació d'alerta.

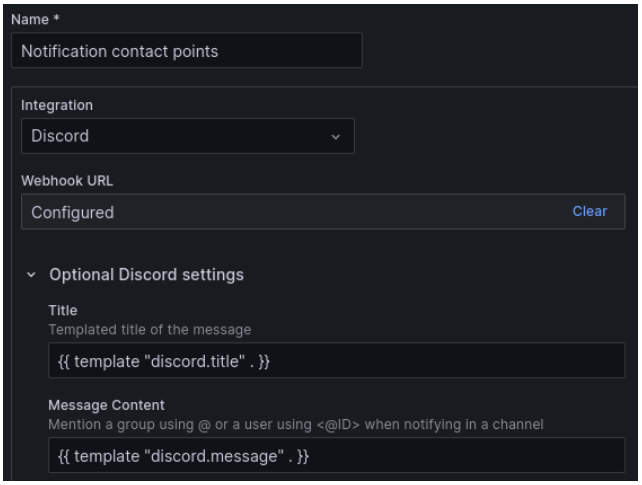

Figura 22. Configuració de la integració de Discord. Font: Elaboració pròpia

```
{{ define "discord.title" }}
{{ if eq .Status "firing" }}
[ALERT] {{ .Labels.alertname }}
{{ else }}
[RESOLVED] {{ .Labels.alertname }}
{{ end }}
{{ end }}

{{ define "discord.message" }}
{{ if eq .Status "firing" }}
STATUS: ALERT
{{ else }}
STATUS: OK
{{ end }}
Resumen: {{ index .CommonAnnotations "summary" }}
{{ end }}
```

Figura 23. Configuració del template de la integració de Discord. Font: Elaboració pròpia

Un cop configurat el "template", en cas que arribi una notificació d'alerta al Discord, el missatge d'aquesta notificació tindrà l'estructura definida prèviament al "template".

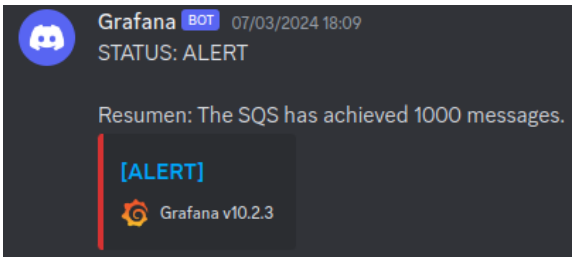

Figura 24. Exemple de notificació d'alerta al Discord. Font: Elaboració pròpia

Un cop s'ha resolt l'error pel qual s'ha generat l'alerta i el valor de la mètrica torna a ser òptim, s'envia una notificació informant de la resolució de l'alerta.

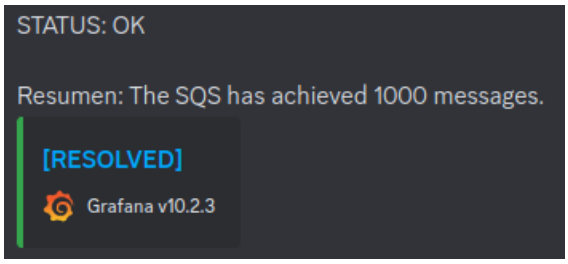

Figura 25. Exemple de resolució d'alerta al Discord. Font: Elaboració pròpia

En relació amb l'enviament de correus electrònics, també es configura la integració a Grafana i s'utilitza un "template" personalitzat tal com es pot observar a la figura 26.

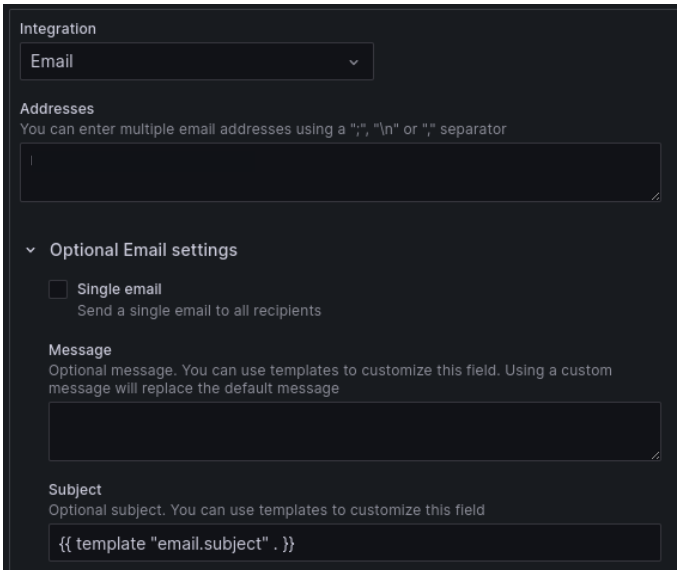

Figura 26. Configuració de la integració d'Email. Font: Elaboració pròpia

A la figura 27 es pot observar la configuració del template de la integració per enviar correus electrònics un cop es genera una alerta a Grafana.

```
{{ define "email.subject" }}
{{ if eq .Status "firing" }}
[ALERT]: {{ index .CommonLabels.Values 0 }}
{{ else }}
[RESOLVED] {{ index .CommonLabels.Values 0 }}
{{ end }}
{{ end }}
```

Figura 27. Configuració del template d'Email. Font: Elaboració pròpia

Finalment, en cas que arribi una notificació d'alerta mitjançant un correu electrònic, l'estructura del correu serà la definida prèviament al template. A la figura 28, es pot observar un exemple de correu electrònic amb un summary, uns valors que fan referència al valor actual de la mètrica i el seu estat (0 = Normal, 1 = Firing) respectivament. A més a més, un conjunt d'etiquetes com el nom de l'alerta, el nom de la cua, la carpeta de Grafana i els missatges a la SQS.

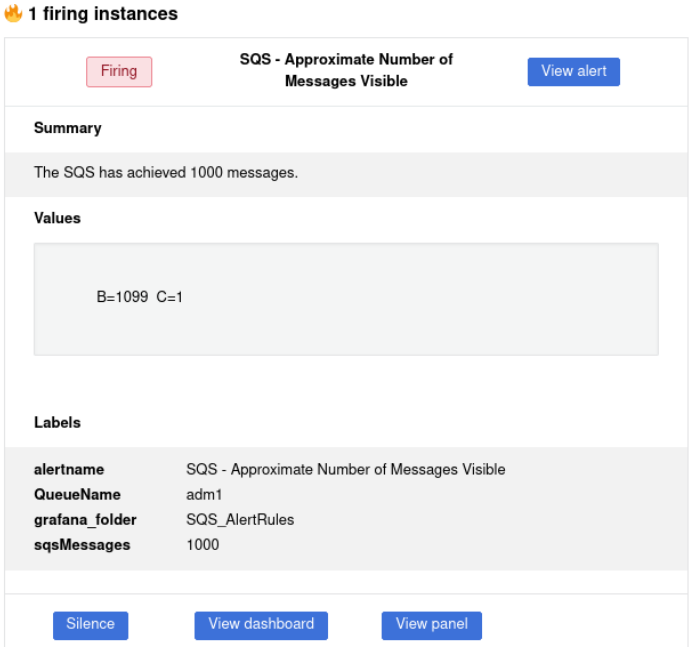

Figura 28. Exemple de correu electrònic d'una notificació d'alerta. Font: Elaboració pròpia

Un cop s'ha resolt l'error pel qual s'ha generat l'alerta i el valor de la mètrica torna a ser òptim, s'envia un correu electrònic informant de la resolució de l'alerta.

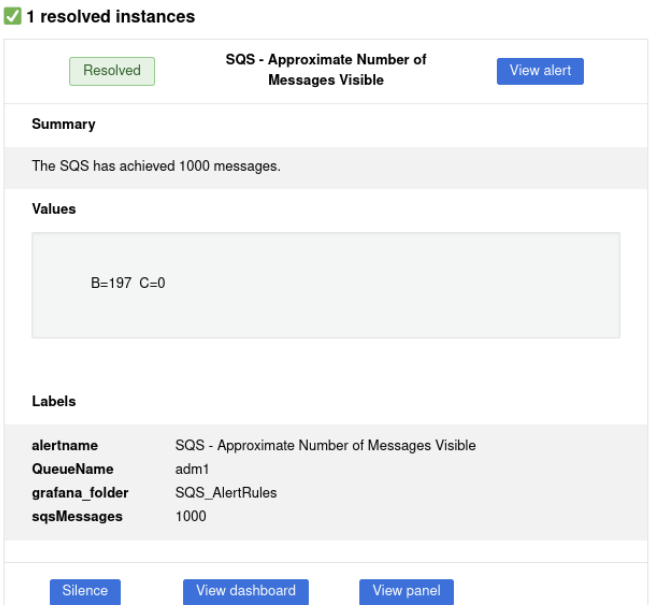

Figura 29. Exemple de correu electrònic de resolució d'alerta. Font: Elaboració pròpia

En últim lloc, existeixen un conjunt de "Notification policies" les quals s'utilitzen per determinar com s'han d'encaminar les alertes cap als "Contact points" definits prèviament. En aquestes "Notification policies" es configuren les etiquetes que han de coincidir amb les de les alertes generades. El sistema mira si coincideixen i aleshores s'encamina l'alerta cap al "Contact point" configurat per aquella política en concret. En el cas que les etiquetes de l'alerta no coincideixin amb cap política, l'alerta utilitza la política per defecte.

A la figura 30, es pot observar un exemple de "Notification policy" per la qual en el cas que les etiquetes de l'alerta generada coincideixin amb les etiquetes definides en aquesta política, aleshores l'alerta s'encaminarà cap al 'Notification contact points'.

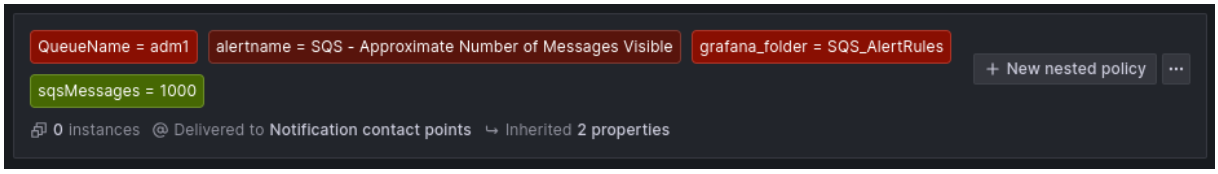

Figura 30. Exemple de Notification policy. Font: Elaboració pròpia

En relació amb el "Contact point" utilitzat per executar accions automatitzades, només té una política associada, ja que actualment només està previst utilitzar aquesta opció per un cas específic, la SQS d'Andorra.

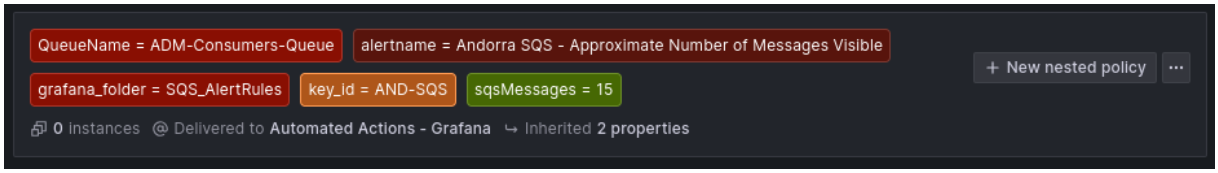

Figura 31. Notification policy per accions automatitzades. Font: Elaboració pròpia

9.3.4 Accions automatitzades

Per la part de les accions automatitzades, s'ha desenvolupat codi tant pel que fa a Backend com pel que fa a Frontend. En relació amb el Backend, s'ha creat un controller amb els endpoints necessaris, ja sigui per processar el missatge provinent de Grafana en cas de generar-se una alerta com també per crear, editar, obtenir o eliminar una acció determinada.

A continuació, s'ha utilitzat un service per implementar cadascun dels mètodes necessaris per al correcte funcionament del flux d'execució i un repositori per guardar tota la informació a la taula 'Action' de la BBDD.

A més a més, pel cas del processament del missatge provinent de Grafana, s'utilitza un altre servei amb el qual s'apliquen les accions pertinents ('Start', 'Stop', 'Restart') a un objecte Servei en concret, i s'ha creat un atribut per habilitar o deshabilitar l'execució de certs mètodes en funció de l'entorn en el qual s'estigui desenvolupant o executant l'aplicació. Aquesta configuració s'ha ubicat als diversos fitxers .yml de l'aplicació.

Un cop desenvolupat el codi al Backend, s'han dut a terme les proves pertinents a través del Postman i posteriorment s'ha desenvolupat el codi al Frontend. En aquest punt, s'han creat dos components, un per visualitzar el llistat amb totes les accions presents a la BBDD i poder cercar-les ràpidament i un altre per crear o editar una acció. A més a més, s'ha creat un servei per poder enviar les peticions realitzades per l'usuari al Frontend cap al Backend i un model amb el qual poder visualitzar la informació de l'acció correctament.

9.4 Notacions

9.4.1 Spring

@PostConstruct: Aquesta anotació s'utilitza en Spring per executar un mètode després de la injecció de dependències per inicialitzar qualsevol instància que estigui dins del mètode.

@Value: Aquesta anotació s'utilitza en Spring per indicar un valor d'una expressió per defecte d'un atribut o un paràmetre d'un mètode o constructor.

@RestController: Aquesta notació s'utilitza en Spring per indicar que la classe s'ha de tractar com un controlador, per exemple, un controlador web. Normalment se sol utilitzar amb la notació següent.

@RequestMapping: Aquesta notació s'utilitza en Spring per mapejar les peticions web cap als mètodes de les classes que esperen rebre peticions HTTP. Normalment se sol utilitzar amb la notació anterior.

@Service: Aquesta notació s'utilitza en Spring per indicar que la classe és un Servei.

@Transactional: Aquesta notació s'utilitza en Spring per descriure un atribut transaccional en un mètode individual o en una classe. Quan s'utilitza en l'àmbit de classe, s'aplica per defecte a tots els mètodes de la classe si no s'indica el contrari.

@Repository: Aquesta notació s'utilitza en Spring per indicar que la classe és un Repositori.

@Entity: Aquesta notació s'utilitza en Spring per especificar que la classe és una Entitat.

@EmbeddedId: Aquesta notació s'utilitza en Spring per especificar que l'atribut o propietat és una clau primària composta de la classe.

@Embeddable: Aquesta notació s'utilitza en Spring per especificar que la classe forma una clau primària composta i tots els seus atributs es mapejen a la taula de l'entitat a la base de dades.

@ManyToOne: Aquesta notació s'utilitza en Spring per especificar una única associació amb una altra entitat que té una relació de molts a un.

@JoinColumn: Aquesta notació s'utilitza en Spring per especificar una columna d'una altra entitat perquè s'ajunti amb l'atribut que té la notació.

9.4.2 Testing

@Autowired: Aquesta anotació s'utilitza en Testing per marcar un constructor, camp, mètode setter o mètode de configuració com a autowired per facilitat la injecció de dependències de Spring.

@MockBean: Aquesta anotació s'utilitza en Testing per afegir mocks a l'ApplicationContext de Spring.

@SpyBean: Aquesta anotació s'utilitza en Testing per introduir espies de Mockito a l'ApplicationContext de Spring.

@TestConfiguration: Aquesta anotació s'utilitza en Testing per definir beans addicionals o personalitzacions pels tests.

@Bean: Aquesta anotació s'utilitza per indicar que un mètode produeix un bean que serà utilitzat pel contenidor de Spring.

@Target: Aquesta anotació s'utilitza en Testing per indicar els contexts en els quals una anotació és aplicable.

@SpringBootTest: Aquesta anotació s'utilitza en Testing per especificar la classe que executarà els tests basats en Spring Boot.

@AutoConfigureMockMvc: Aquesta anotació s'utilitza en Testing per habilitar i configurar l'autoconfiguració de MockMvc.

@ActiveProfiles: Aquesta anotació s'utilitza en Testing per declarar quin perfil s'ha d'utilitzar quan es carrega un ApplicationContext per les classes de test.

@DisplayName: Aquesta anotació s'utilitza en Testing per declarar un nom personalitzat per la classe de test o el mètode de test en qüestió.

@Test: Aquesta anotació s'utilitza en Testing per assenyalar que el mètode amb aquesta anotació és un mètode de test.

@Captor: Aquesta anotació s'utilitza en Testing per capturar una classe o interfície en concret.

9.4.3 Angular

@Component: Aquesta anotació s'utilitza en Angular per marcar una classe com un component d'Angular i proporciona configuració de metadada que determina com s'ha de processar, instanciar i utilitzar el component en temps d'execució. Els components són els blocs de construcció UI més bàsics d'una aplicació Angular.

@Inject: Aquesta anotació s'utilitza en Angular per especificar un proveïdor personalitzat de la dependència en un constructor d'una classe.

@Input: Aquesta anotació s'utilitza en Angular per marcar un atribut de classe com una propietat d'entrada i permet que el valor de l'atribut es pugui passar des d'un component pare a un component fill.

Interpolació: És un mecanisme d'Angular que permet substituir una expressió per un valor de tipus string al codi d'HTML. Per aplicar aquest mecanisme s'utilitzen claus de doble corxetes {{ }}.

Injecció de dependències: Angular utilitza aquest concepte per gestionar les dependències entre els diversos components, serveis i classes de l'aplicació.

9.5 Pantalles

Pantalla del monitoratge de la Ingesta d'ADM que serveix per tenir una idea clara de l'estat dels diferents serveis involucrats en la ingesta de dades de la plataforma ADM gràcies als diversos gràfics representats a la pantalla.

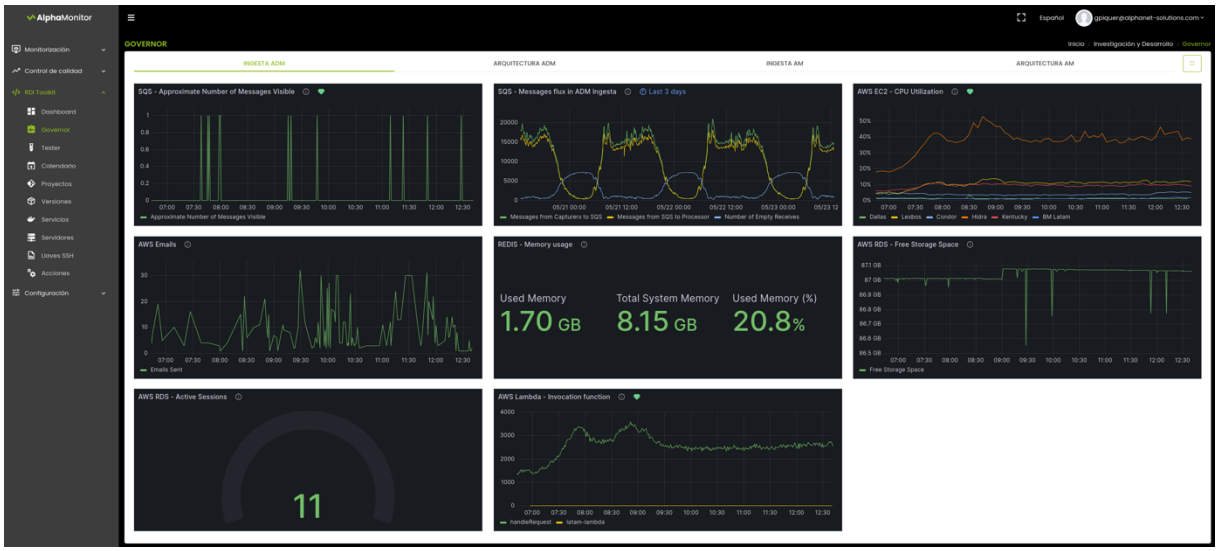

Figura 32. Pantalla del monitoratge de la Ingesta d'ADM. Font: Elaboració pròpia

Pantalla completa del monitoratge de la Ingesta d'ADM. Aquesta versió de pantalla és la que s'utilitza en els monitors de les oficines de l'empresa quan es vol obtenir una visió actual de l'estat dels serveis involucrats en la ingesta de dades de la plataforma ADM.

Figura 33. Pantalla completa del monitoratge de la Ingesta d'ADM. Font: Elaboració pròpia

Pantalla del monitoratge de l'Arquitectura d'ADM que serveix per tenir una idea clara de l'estat de l'arquitectura de la plataforma ADM.

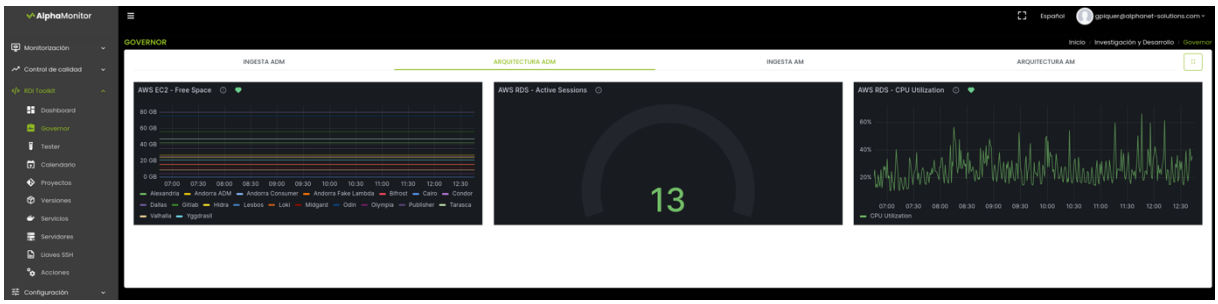

Figura 34. Pantalla del monitoratge de l'Arquitectura d'ADM. Font: Elaboració pròpia

Pantalla completa del monitoratge de l'Arquitectura d'ADM. Aquesta versió de pantalla és la que s'utilitza en els monitors de les oficines de l'empresa quan es vol obtenir una visió actual de l'estat de l'arquitectura de la plataforma ADM.

Figura 35. Pantalla completa del monitoratge de l'Arquitectura d'ADM.

Font: Elaboració pròpia

Pantalla del monitoratge de la Ingesta d'AM que serveix per tenir una idea clara de l'estat dels diferents serveis involucrats en la ingesta de dades de la plataforma AM gràcies als diversos gràfics representats a la pantalla.

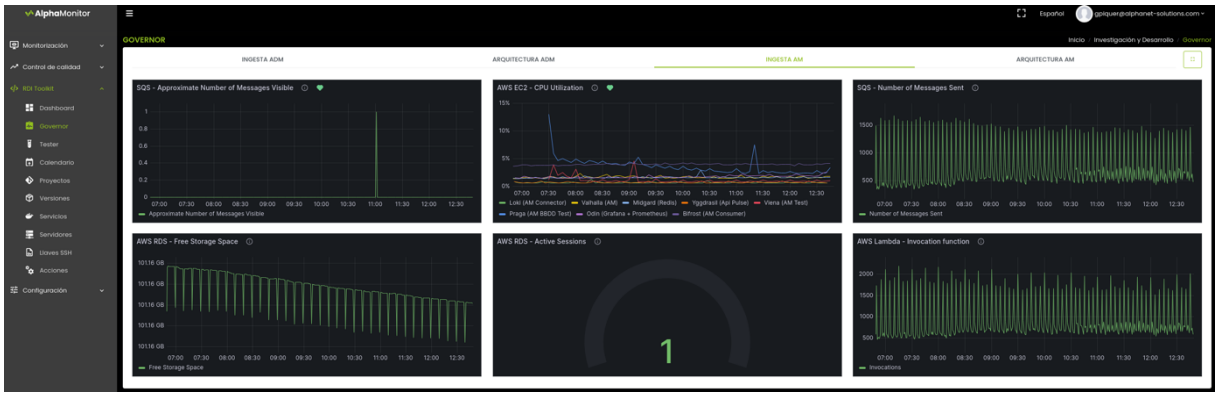

Figura 36. Pantalla del monitoratge de la Ingesta d'AM. Font: Elaboració pròpia

Pantalla completa del monitoratge de la Ingesta d'AM. Aquesta versió de pantalla és la que s'utilitza en els monitors de les oficines de l'empresa quan es vol obtenir una visió actual de l'estat dels serveis involucrats en la ingesta de dades de la plataforma AM.

Figura 37. Pantalla completa del monitoratge de la Ingesta d'AM. Font: Elaboració pròpia

Pantalla del monitoratge de l'Arquitectura d'AM que serveix per tenir una idea clara de l'estat de l'arquitectura de la plataforma AM.

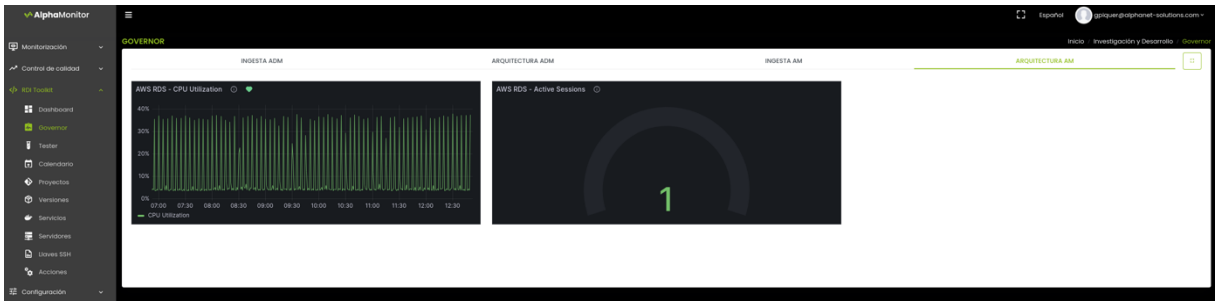

Figura 38. Pantalla del monitoratge de l'Arquitectura d'AM. Font: Elaboració pròpia

Pantalla completa del monitoratge de l'Arquitectura d'AM. Aquesta versió de pantalla és la que s'utilitza en els monitors de les oficines de l'empresa quan es vol obtenir una visió actual de l'estat de l'arquitectura de la plataforma AM.

Figura 39. Pantalla completa del monitoratge de l'Arquitectura d'AM.

Font: Elaboració pròpia

En relació amb els gràfics utilitzats pel monitoratge de les plataformes ADM i AM i que es poden visualitzar a través de les pantalles mostrades anteriorment:

El gràfic del nombre aproximat de missatges visibles a SQS indica el nombre de missatges que es troben a la cua (SQS) d'ADM. El gràfic mostra un interval de temps de sis hores i el llindar de missatges perquè no salti l'alerta és de mil missatges acumulats.

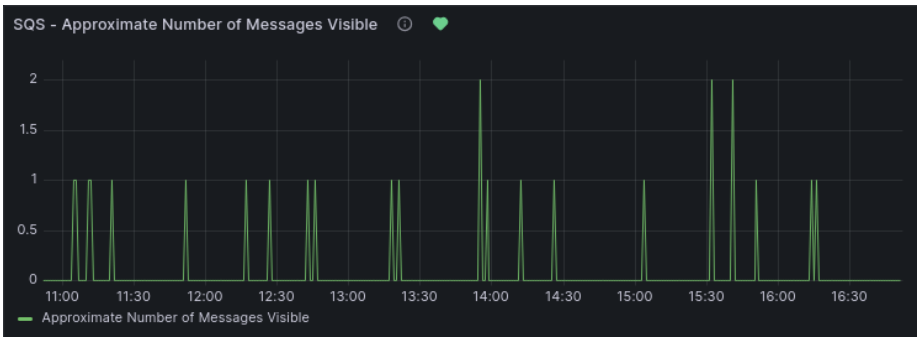

Figura 40. Gràfic del nombre aproximat de missatges visibles a la SQS d'ADM.

Font: Elaboració pròpia

El gràfic del flux de missatges en la ingesta d'ADM mostra en un interval dels últims tres dies:

- **Color verd:** nombre de missatges que rep la SQS provinents dels lectors de matrícules.
- **Color groc:** nombre de missatges que envia la SQS cap a ADM Processor.
- **Color blau:** nombre de missatges buits que rep la SQS.

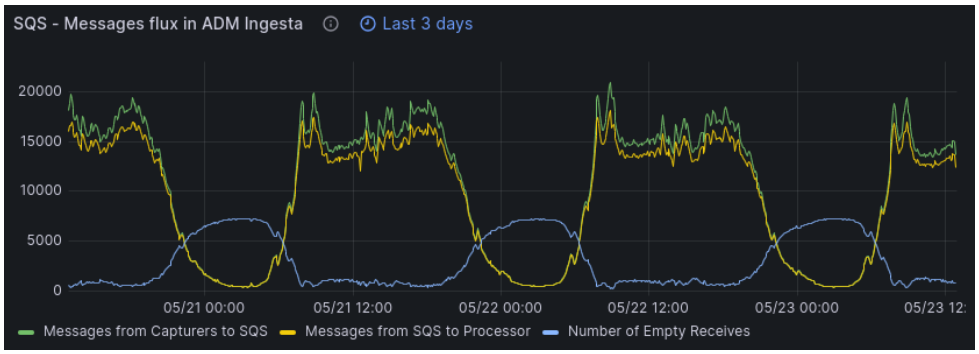

Figura 41. Gràfic de flux de missatges en la Ingesta d'ADM. Font: Elaboració pròpia

El gràfic del percentatge d'utilització de CPU dels servidors d'ADM indica el percentatge d'ús de CPU de cadascun dels servidors que estan relacionats amb ADM, ja siguin de l'entorn de test, de l'entorn de producció, de llatinoamèrica, etc. El gràfic mostra el percentatge d'ús de CPU en un interval de temps de sis hores i es genera una alerta si el percentatge és superior al 85%.

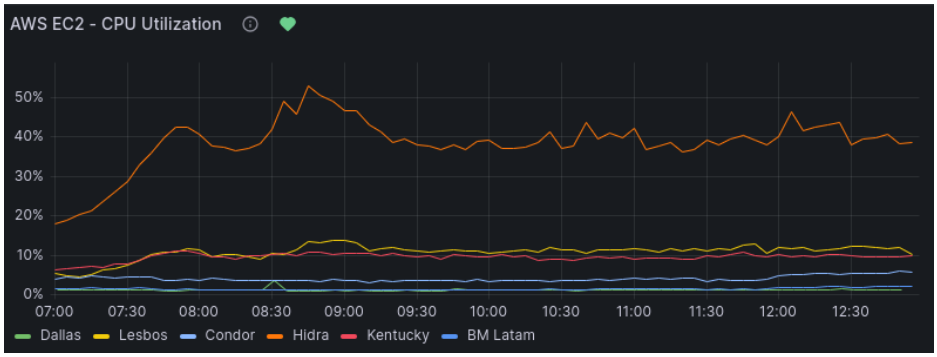

Figura 42. Gràfic del percentatge d'utilització de CPU dels servidors d'ADM.

Font: Elaboració pròpia

El gràfic del nombre de correus electrònics enviats indica el nombre de correus enviats als clients d'ADM en relació amb alertes, sancions o altres gestions referents a la plataforma. El gràfic mostra el nombre de correus electrònics enviats en un interval de temps de sis hores.

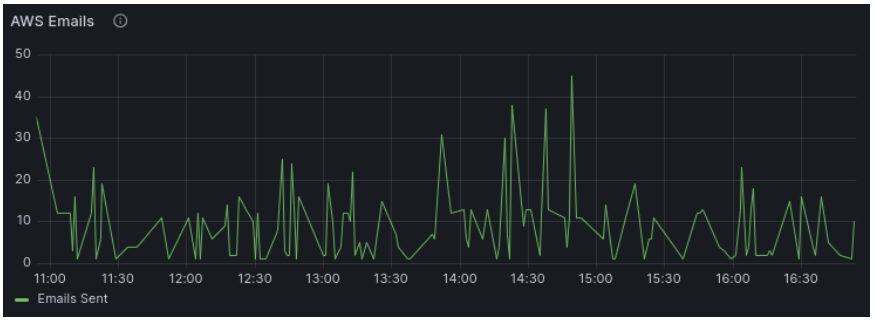

Figura 43. Gràfic del nombre de correus electrònics enviats. Font: Elaboració pròpia

El gràfic de la memòria utilitzada de la Redis mostra la memòria utilitzada en megabytes, la memòria total del sistema en gigabytes i el percentatge de memòria utilitzada respecte a la memòria total del sistema.

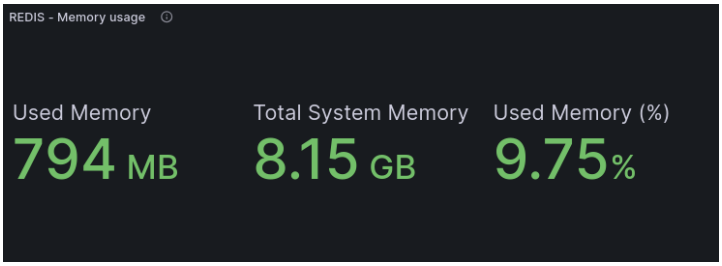

Figura 44. Gràfic de la memòria utilitzada de la Redis. Font: Elaboració pròpia

El gràfic d'espai d'emmagatzematge disponible a la RDS indica el nombre de gigabytes que queden d'emmagatzematge disponible a la RDS d'ADM. El gràfic mostra l'espai d'emmagatzematge disponible en un interval de temps de sis hores.

Figura 45. Gràfic d'espai d'emmagatzematge disponible a la RDS d'ADM.

Font: Elaboració pròpia

El gràfic del nombre de sessions actives a la RDS indica el nombre de sessions actives que hi ha en aquell precís instant a la RDS. Normalment, el valor ha d'estar entre una i dues sessions actives per considerar-se adequat, tot i que a vegades pot aparèixer un valor superior i continuar sent òptim.

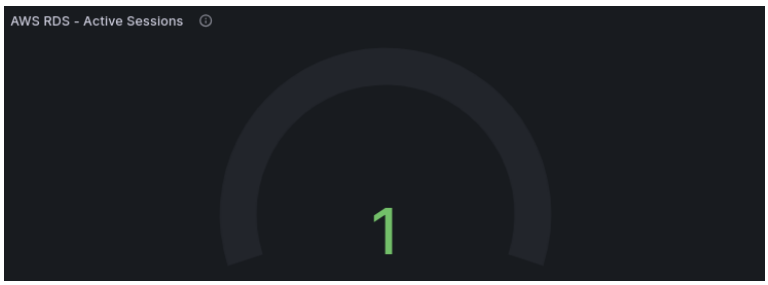

Figura 46. Gràfic del nombre de sessions actives a la RDS. Font: Elaboració pròpia

El gràfic del nombre d'invocacions de la funció de Lambda indica el nombre de vegades que es crida a la funció de Lambda en aquell precís instant. Per una banda, de color groc hi ha la Lambda de llatinoamèrica mentre que de color verd hi ha la Lambda d'ADM. El gràfic mostra el nombre d'invocacions en un interval de temps de sis hores i es genera una alerta si el valor de la Lamda de llatinoamèrica és superior a mil o si el valor de la Lambda d'ADM és superior a 5000.

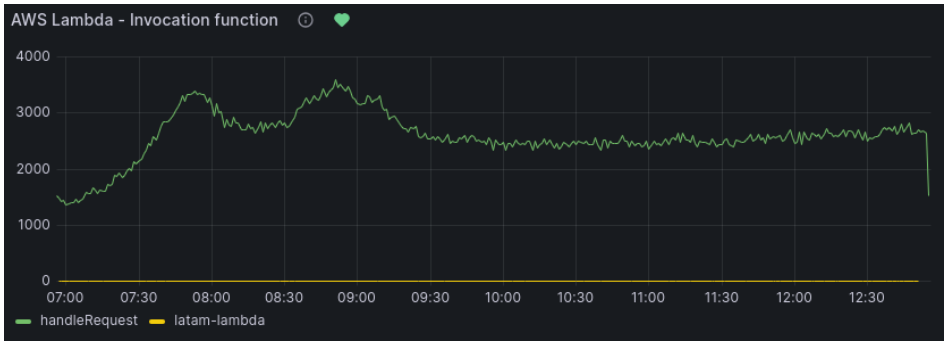

Figura 47. Gràfic del nombre d'invocacions de la funció de Lambda. Font: Elaboració pròpia

El gràfic de l'espai disponible de les instàncies a AWS mostra l'espai disponible en gigabytes de totes les instàncies que es troben a l'arquitectura de les plataformes. El gràfic mostra l'espai disponible en un interval de temps de sis hores i es genera una alerta si l'espai disponible d'una instància és inferior a tres gigabytes.

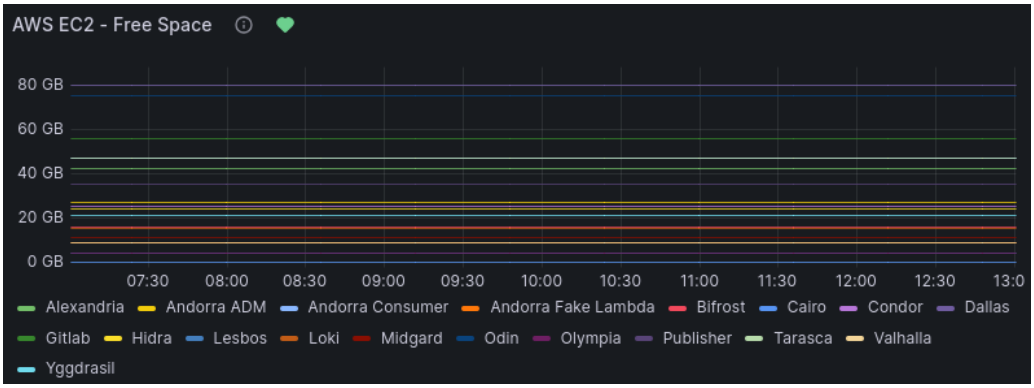

Figura 48. Gràfic de l'espai disponible de les instàncies a AWS. Font: Elaboració pròpia

El gràfic del percentatge d'utilització de CPU de RDS mostra el percentatge d'ús de la CPU de la RDS d'ADM. El gràfic mostra el percentatge en un interval de temps de sis hores i es genera una alerta si el percentatge és superior al 85%.

Figura 49. Gràfic del percentatge d'ús de CPU de la RDS d'ADM. Font: Elaboració pròpia

El gràfic del nombre aproximat de missatges visibles a SQS indica el nombre de missatges que es troben a la cua (SQS) d'AM. El gràfic mostra un interval de temps de sis hores i el llindar de missatges perquè no salti l'alerta és de mil missatges acumulats.

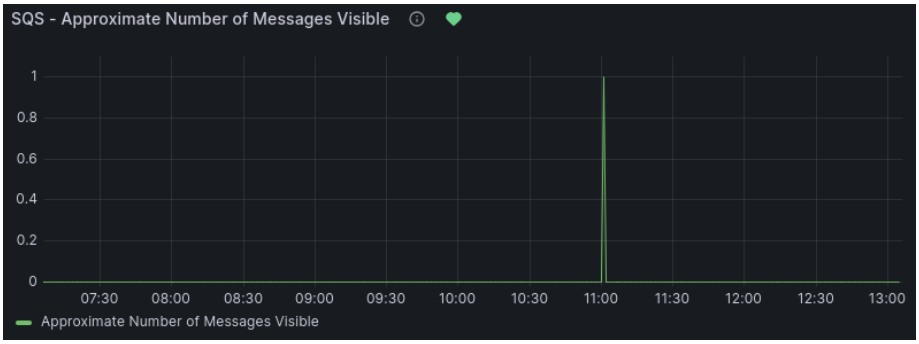

Figura 50. Gràfic del nombre aproximat de missatges visibles a la SQS d'AM.

Font: Elaboració pròpia

El gràfic del percentatge d'ús de CPU dels servidors d'AM mostra el percentatge de CPU que utilitza cadascun dels servidors que estan relacionats amb AM, com el Connector, el Publisher, el Consumer, l'entorn de test, l'entorn de producció, etc. El gràfic mostra el percentatge en un interval de sis hores i es genera una alerta si el percentatge supera el 85%.

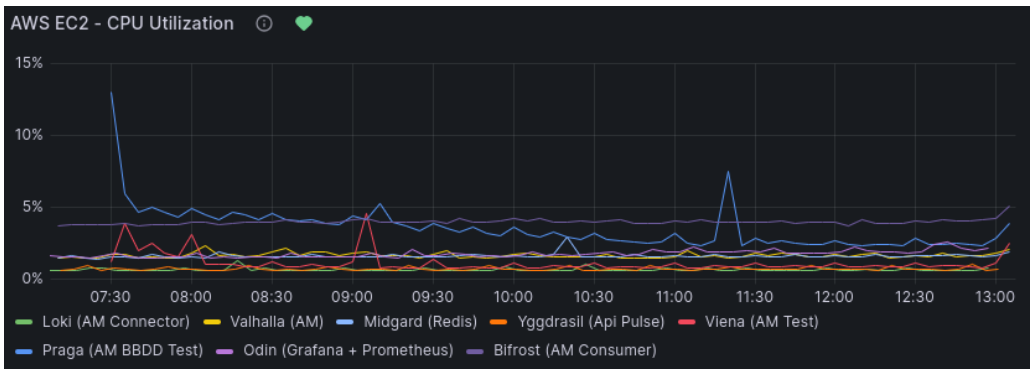

Figura 51. Gràfic del percentatge d'ús de CPU dels servidors d'AM. Font: Elaboració pròpia

El gràfic del nombre de missatges enviats a la SQS mostra el nombre de missatges que envien els lectors de matrícules cap a la SQS d'AM. El gràfic mostra el nombre de missatges en un interval de temps de sis hores.

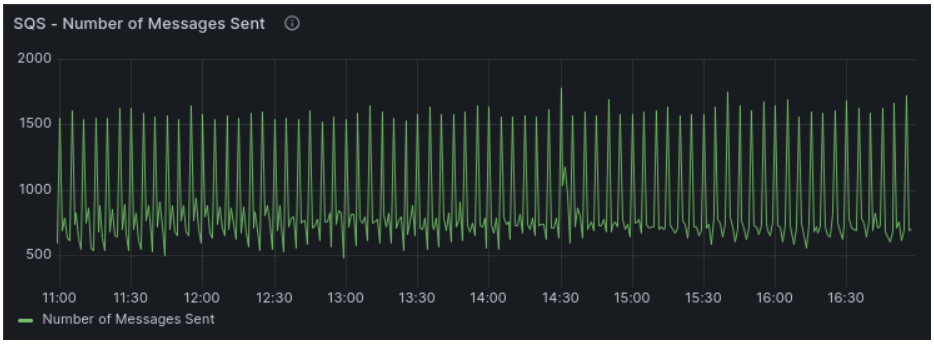

Figura 52. Gràfic del nombre de missatges enviats a la SQS d'AM. Font: Elaboració pròpia

El gràfic d'espai d'emmagatzematge disponible a la RDS indica el nombre de gigabytes que queden d'emmagatzematge disponible a la RDS d'AM. El gràfic mostra l'espai d'emmagatzematge disponible en un interval de temps de sis hores.

Figura 53. Gràfic d'espai d'emmagatzematge disponible a la RDS d'AM.

Font: Elaboració pròpia

El gràfic del nombre d'invocacions de la funció de Lambda indica el nombre de vegades que es crida a la funció de Lambda en aquell precís instant. El gràfic mostra el nombre d'invocacions de la Lambda d'AM en un interval de temps de sis hores.

Figura 54. Gràfic del nombre d'invocacions de la Lambda d'AM. Font: Elaboració pròpia

El gràfic del percentatge d'utilització de CPU de RDS mostra el percentatge d'ús de la CPU de la RDS d'AM. El gràfic mostra el percentatge en un interval de temps de sis hores i es genera una alerta si el percentatge és superior al 85%.

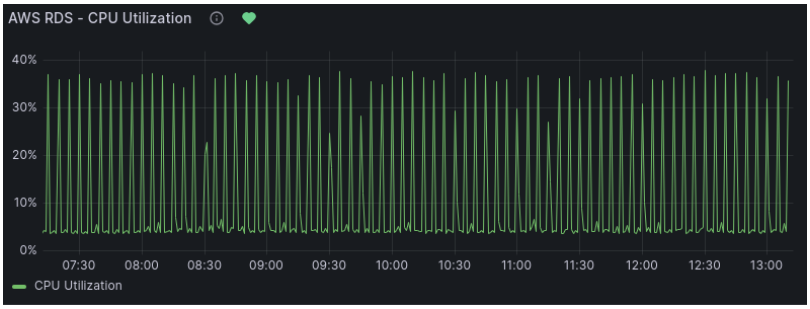

Figura 55. Gràfic del percentatge d'ús de CPU de la RDS d'AM. Font: Elaboració pròpia

En relació amb les accions automatitzades, s'ha creat una pantalla per visualitzar un llistat de les accions. En aquesta pantalla es poden utilitzar un conjunt de filtres (key, servei, tipus, nom i estat) per fer una cerca més acurada.

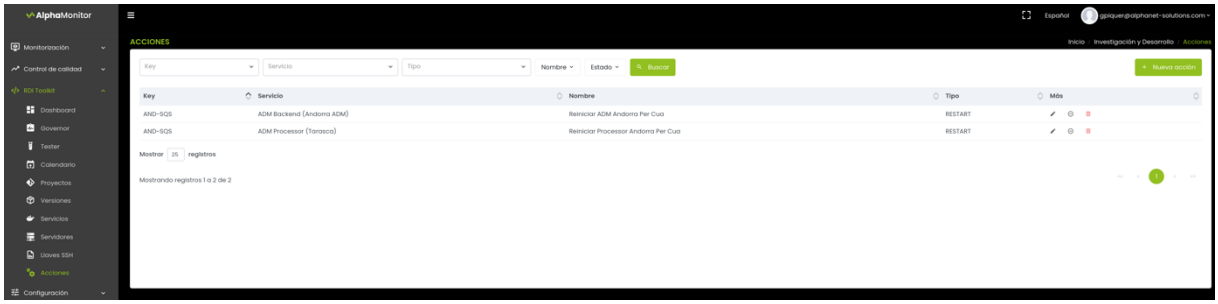

Figura 56. Pantalla d'accions automatitzades. Font: Elaboració pròpia

També s'ha creat un modal per poder crear una nova acció.

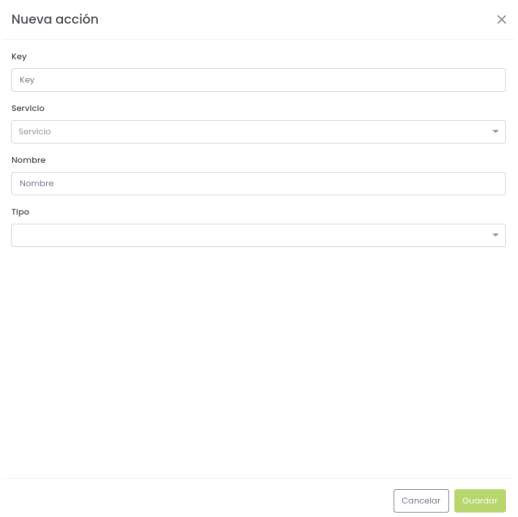

Figura 57. Modal per crear una nova acció. Font: Elaboració pròpia

Pel que fa a les mètriques de cada servei, també s'ha creat una pantalla on es pot visualitzar un conjunt de serveis amb la informació de cadascun d'ells i veure i editar les seves mètriques (els dos botons superiors a la dreta respectivament).

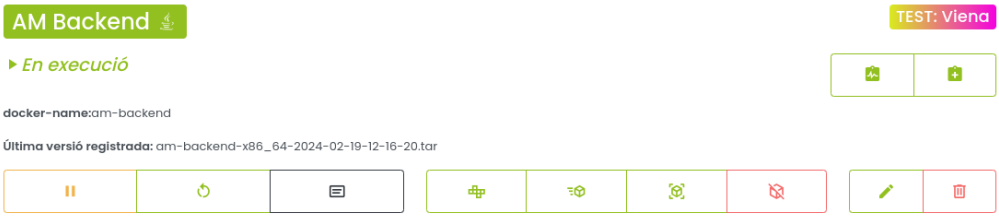

Figura 58. Card amb informació d'un servei i les possibles accions. Font: Elaboració pròpia

Per editar les mètriques d'un servei, s'ha creat un modal per poder dur a terme aquesta acció, sigui afegir una nova mètrica o eliminar-ne una ja existent.

Figura 59. Modal per editar mètriques d'un servei. Font: Elaboració pròpia

A l'hora de visualitzar les mètriques d'un servei, s'ha creat una pantalla amb els gràfics corresponents a cadascuna de les mètriques que té un servei.

Figura 60. Pantalla de les mètriques d'un servei. Font: Elaboració pròpia

El gràfic de la memòria utilitzada per l'aplicació en Java Spring Boot del servei mostra el percentatge de memòria utilitzada per l'aplicació del servei.

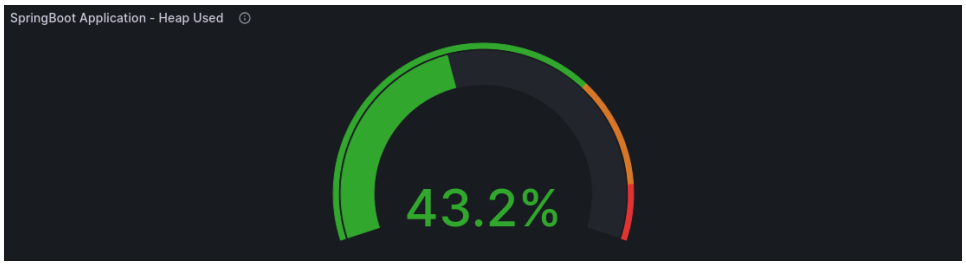

Figura 61. Gràfic de la memòria utilitzada per l'aplicació del servei. Font: Elaboració pròpia

El gràfic de la CPU utilitzada per l'aplicació en Java Spring Boot del servei mostra de color verd la CPU utilitzada pel sistema i de color groc la CPU utilitzada pels processos de l'aplicació.

Figura 62. Gràfic de la CPU utilitzada per l'aplicació del servei. Font: Elaboració pròpia

El gràfic del nombre de threads de l'aplicació en Java Spring Boot del servei mostra de color verd els threads en mode "Daemon", de color groc els threads actius i de color blau el pic en el nombre de threads.

Figura 63. Gràfic del nombre de threads de l'aplicació del servei. Font: Elaboració pròpia

El gràfic del temps d'ús de la connexió de l'aplicació en Java Spring Boot del servei mostra els temps d'ús de l'aplicació en mil·lisegons.

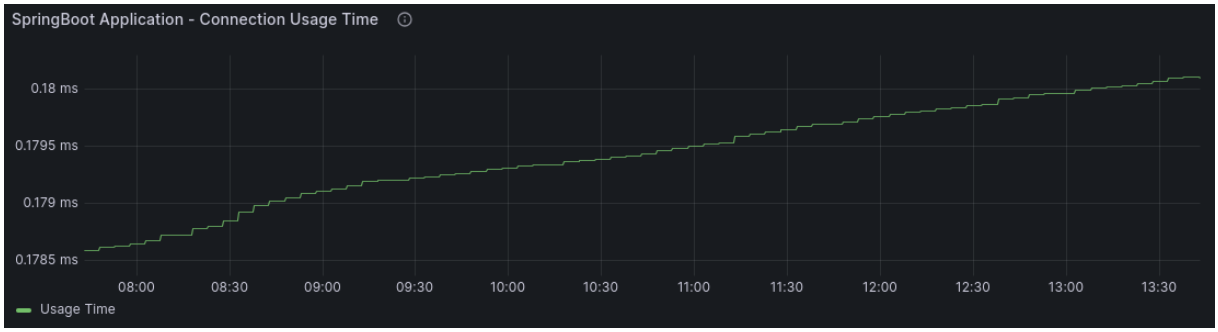

Figura 64. Gràfic del temps d'ús de la connexió de l'aplicació del servei. Font: Elaboració pròpia

10 Anàlisi de resultats

Després d'implementar tot el desenvolupament del projecte a l'entorn de producció, el personal encarregat de supervisar el monitoratge de la infraestructura de les plataformes ha fet ús de les noves pantalles i funcionalitats d'AM aportant un feedback molt positiu i una satisfacció molt gran a l'hora d'utilitzar aquests nous canvis a la plataforma.

Els resultats obtinguts per part del personal són els següents:

- Les pantalles referents al monitoratge de la Ingesta i Arquitectura tant d'ADM com d'AM són molt visuals i aporten molta informació.
- S'expressa una bona satisfacció en termes d'experiència d'ús de les noves pantalles i funcionalitats d'AM.
- L'execució de tests i la posterior obtenció de resultats positius aporta una seguretat important a l'hora de desenvolupar codi de les plataformes.
- La recepció de notificacions d'alertes tant a la plataforma Discord com a través de correus electrònics és de molta utilitat i un gran avenç en la prevenció d'errors de la infraestructura de les plataformes de l'empresa.
- L'ús d'accions automatitzades a causa d'un error en la infraestructura de les plataformes contribueix a mantenir en un estat òptim la mateixa infraestructura i alliberar al personal de certes tasques.

Pel que fa a l'ús actual dels nous canvis introduïts a la plataforma AM per part del personal, s'ha pogut constatar l'aportació per part d'aquest projecte de noves utilitats al personal de l'empresa i els seus conseqüents beneficis, com un monitoratge acurat de la infraestructura de les plataformes i un codi amb més qualitat i més ben documentat.

11 Conclusions

Un cop completat el treball amb tot el desenvolupament i una posterior implementació del monitoratge intern de les plataformes de l'empresa Alphanet Security Systems, es pot concloure que s'ha aconseguit assolir els objectius i requeriments inicials establerts a l'inici del projecte, aportant eines pel monitoratge intern de les plataformes, tals com la visualització de gràfics representant l'estat en temps real de la infraestructura i un sistema de notificacions eficaç en cas de generacions d'alertes durant el monitoratge.

D'altra banda, el desenvolupament d'un testing capaç d'assegurar una bona qualitat de codi i robustesa del mateix també és visible dins del marc de realització d'aquest treball aportant seguretat als desenvolupadors de les plataformes en termes de validesa del software.

Les noves pantalles i funcionalitats desenvolupades en aquest treball estan totalment operatives i el personal encarregat de la supervisió del monitoratge intern de les plataformes ja les està utilitzant. A més a més, els usuaris finals estan molt satisfets de les noves funcionalitats i solucions aportades en aquest treball.

La realització del treball dins d'una empresa també ha aportat aprenentatge en l'àmbit tecnològic i professional a l'hora de desenvolupar una solució tecnològica per la mateixa empresa, ja que l'ús de noves eines, tals com GitLab i Grafana i metodologies de treball com el desenvolupament per sprints, les reunions d'equip i un calendari concret de pujades de codi a l'entorn de producció han sigut dues característiques molt presents durant el transcurs del treball.

En general, aquest treball de final de grau representa l'esforç i la dedicació durant mesos per aportar una solució a una necessitat de l'empresa Alphanet Security Systems i l'ús de tot el coneixement adquirit durant els diversos cursos i etapes de la carrera universitària.

12 Possibles ampliacions

Com a possibles propostes de millora o ampliacions referents a aquest treball, es podrien esmentar les següents:

- **Augment del nombre de tests desenvolupats:** tot i l'esforç per desenvolupar el nombre més gran de tests en aquest treball, encara queda força feina a fer i moltes línies de codi a testejar tant de la plataforma ADM com de la plataforma AM.

- **Major nombre de mètriques i gràfics:** ampliar el nombre de mètriques i gràfics a l'apartat de monitoratge per aconseguir més informació sobre l'estat de la infraestructura de les plataformes

- **Implementar noves integracions a Grafana:** utilitzar noves integracions a Grafana a part de les ja existents augmentant així el nombre de canals de difusió a través de noves plataformes.

- **Configurar més alertes durant el monitoratge:** configurar més alertes a Grafana a partir de nous gràfics per tenir un major control sobre l'estat de la infraestructura de les plataformes.

- **Ampliar l'apartat d'accions automatitzades:** en l'àmbit de les accions automatitzades, es pot utilitzar per a molts serveis i es poden ampliar les funcionalitats d'aquest apartat de la plataforma AM, ja sigui pel que fa a la creació de noves accions o desenvolupant nou codi al Backend per optimitzar el seu ús.

13 Bibliografia

[1] Alphanet. [en línia] [consulta: 2024-02-26]. Disponible a: https://alphanet.cat/

[2] Michael Olan, “Unit Testing: Test Early, Test Often”, Computer Science and Information Systems - Richard Stockton College, Pomona, NJ, USA, Dec. 2003.

[3] GeeksForGeeks. “Unit Testing - Software Testing”. GeeksForGeeks. [en línia] [consulta: 2023-12-22]. Disponible a: https://www.geeksforgeeks.org/unit-testing-software-testing/

[4] Martin Fowler. “IntegrationTest”. martinFowler.com. [en línia] [consulta: 2023-12-22]. Disponible a: https://martinfowler.com/bliki/IntegrationTest.html

[5] AlphaDataManager, La plataforma líder en dades de trànsit. [en línia] [consulta: 2024-02-26]. Disponible a: https://alphanet.cat/solucions/alpha-data-manager/

[6] AWS, Amazon Web Services. [en línia] [consulta: 2024-02-29]. Disponible a: https://aws.amazon.com/es/

[7] OpenSearch. Documentació oficial. [en línia] [consulta: 2024-02-29]. Disponible a: https://opensearch.org/docs/latest/about/

[8] Redis. Documentació oficial. [en línia] [consulta: 2024-02-29]. Disponible a: https://redis.io/docs/

[9] Prometheus. Documentació oficial. [en línia] [consulta: 2024-01-16]. Disponible a: https://prometheus.io/docs/introduction/overview/

[10] Grafana. Documentació oficial. [en línia] [consulta: 2024-01-16]. Disponible a: https://grafana.com/docs/grafana/latest/introduction/

[11] GitLab Pipelines. Documentació oficial. [en línia] [consulta: 2023-12-11]. Disponible a: https://docs.gitlab.com/ee/ci/pipelines/index.html

[12] JUnit. Documentació oficial. [en línia] [consulta: 2023-12-11]. Disponible a: https://junit.org/junit5/docs/current/user-guide/

[13] Mockito, Tasty mocking framework for unit tests in Java. [en línia] [consulta: 2023-12-12]. Disponible a: https://site.mockito.org/

[14] Actuator, Spring Boot Actuator. Baeldung. [en línia] [consulta: 2024-05-22]. Disponible a: https://www.baeldung.com/spring-boot-actuators