

UNIVERSIDAD DE OVIEDO



Universidad de Oviedo



Escuela de
Ingeniería
Informática
Universidad de Oviedo

ESCUELA DE INGENIERÍA INFORMÁTICA

TRABAJO FIN DE GRADO

Apolo: Sistema inteligente de monitorización y detección de ataques
basados en inteligencia artificial

AUTOR: Sergio Arroni del Riego

DIRECTORES: Dr. Vicente García Díaz &
Dr. Antonio Payá González

Declaración de Originalidad

Como autor de este proyecto, Sergio Arroni del Riego, alumno del grado de Ingeniería Informática del Software en la Universidad de Oviedo con DNI 58431204A, declaro que el Trabajo Fin de Grado “*Apolo: Sistema inteligente de monitorización y detección de ataques basados en inteligencia artificial*” es original y totalmente de mi autoría y que se han citado debidamente todas las fuentes bibliográficas utilizadas.

Gijón, a 11 de junio de 2023.

Agradecimientos

Quisiera agradecer enormemente a mis padres por toda la confianza brindada durante toda mi vida, siempre he sentido vuestro apoyo incondicional propio de los mejores padres que se puede pedir. Gracias, madre por aguantarme en todos los enfados y frustraciones que me ha dado esta carrera. Hablando de familia, no puedo no agradecer a Nuri y a Darío, gracias por todos los buenos momentos vividos.

Especial agradecimientos a las dos mujeres de mi vida. A mi hermana Laura, sin ti seguramente no hubiera conseguido todo esto, siempre has sido mi camino a seguir en la vida, creo que nadie hace mejor de hermana mayor que tú. Y a mi pareja Lucía, al final del día eres la que más me aguanta las quejas de las asignaturas y entregas, gracias por la confianza ciega hasta en los momentos más complicados, fuiste mi luz en la oscuridad. Os quiero.

Agradecer a mi tutor Vicente una de las mejores decisiones de mi vida universitaria fue unirme a la beca, siempre he podido contar contigo en los momentos de dudas.

Una de las personas a las que más tengo que agradecer es a mi cotutor y amigo Antonio, siempre has sido un referente profesional, un espejo al que mirarme, sin ti nada hubiera sido igual, agradezco a la vida por haberme cruzado contigo y poder aprender de ti.

No hubiera sido lo mismo sin mis amigos de la carrera Hugo, Diego y Miguel, gracias por aguantarme durante las clases, nos volveremos a cruzar, de eso estoy seguro. También a mis compañeros de beca Nuria, Yeray y los profesores Oscar, Dani, Edward, Cristian y Alberto Gómez ha sido un auténtico placer compartir laboratorio y proyectos con vosotros, sobre todo a Yeray juntos hemos aprendido y reído a partes iguales, gracias por todo. Y a mis amigos de toda la vida que siempre me han ayudado a desconectar y a divertirme, Balbuena, Víctor, Marcos y especialmente Alberto, mi Séneca, sé que siempre puedo contar contigo.

Por último, a todos esos profesores que me han enseñado todo durante la carrera, especial mención a todos los que se les nota que aman su trabajo y lo trasmiten en cada clase.

Seguramente me esté dejando gente en el tintero, gracias a todos los que se han cruzado por mi vida y me han ayudado a mejorar como persona.

Resumen

Actualmente todo el mundo quiere implementar las inteligencias artificiales en sus empresas, sistemas o proyectos. El gran problema viene cuando la mayoría de los creadores, usuarios y clientes desconocen el potencial riesgo de los ciberataques a este tipo de tecnologías, y más concretamente los ciberataques de *Adversarial Machine Learning (AML)*, ataques especializados en inteligencias artificiales.

Una solución popularmente utilizada para la defensa de sistemas frente a los ciberataques tradicionales, son los *Sistemas de Detención de Intrusos (IDS)*, los cuales han sido utilizados durante un amplio espacio de tiempo, obteniendo muy buenos resultados frente a ciberataques clásicos, pero se han visto claramente superados frente a los ataques de *AML*.

En este proyecto proponemos **Apolo**, un modelo que realiza las funciones de *IDS* tradicional, pero con el añadido de ser capaz de detectar ataques de *AML* y defenderse de estos obteniendo buenos resultados en comparación con los modelos de *Machine Learning (ML)* tradicionales.

Apolo cuenta con tres capas, las cuales actúan en conjunto para clasificar si una petición es benigna o maligna. Estas capas utilizan diferentes técnicas con el objetivo de dificultar a un atacante replicar nuestro modelo, como la *clusterización* de peticiones red, la utilización de múltiples modelos y la implementación del algoritmo *Multi-Armed Bandits (MAB)*.

Además, hemos implementado **Apolo** en un entorno real, mediante la creación de una arquitectura de un sistema de gestión y monitorización de anomalías en peticiones web. Dicho sistema cuenta con interfaces para gestionar las peticiones y las posibles anomalías que pueda detectar **Apolo**, en caso de que se detecte una anomalía, el sistema mandará un mensaje de alerta al administrador para que este actúe con la mayor brevedad posible.

En el contexto del **TFG**, este proyecto tiene una base amplia de investigación, ya que, contamos con dos artículos de esta temática enviados a diferentes revistas de prestigio, “**Apollon**” el cual es la creación y explicación del modelo **Apolo** utilizado en este proyecto y “**Harpe**” que trata sobre la creación de un nuevo ataque de evasión (*AML*), el cual hemos utilizado, en una versión simplificada, para probar nuestro modelo **Apolo**.

Palabras Clave

Inteligencia Artificial. Aprendizaje Automático. Aprendizaje Profundo. Aprendizaje Automático Adversarial. Sistema de Detección de Intrusos. Ciberseguridad. MAB.

Abstract

Nowadays, everyone wants to implement artificial intelligence in their companies, systems, or projects. The big problem comes when most creators, users and customers are unaware of the potential risk of cyber-attacks on this type of technology, and more specifically *Adversarial Machine Learning (AML)* cyber-attacks, specialised in artificial intelligences.

A popular solution for defending systems against traditional cyber-attacks is *Intrusion Detection Systems (IDS)*, which have been used for a long time with very good results against more traditional cyber-attacks, but have been clearly surpassed by *AML* attacks.

In this project we propose **Apollo**, a model that performs the functions of traditional *IDS*, but with the addition of being able to detect and defend against *AML* attacks, obtaining good results compared to conventional *Machine Learning (ML)* models.

Apollo has three layers, which act together to classify whether a request is malicious or not. These layers use different techniques in order to make it difficult for an attacker to replicate our model, such as the clustering of network requests, the use of multiple models and the implementation of the *Multi-Armed Bandits (MAB)* algorithm.

In addition, we have implemented **Apollo** in a real environment, by creating an architecture for a web request anomaly management and monitoring system. This system has interfaces to manage the requests and the possible anomalies that **Apollo** can detect. If an anomaly is detected, the system will alert the administrator so that they can act as soon as possible.

In the context of this bachelor's thesis, this project has a broad research base, since we have sent two articles on this subject to different prestigious journals, "**Apollon**" which is the creation and explanation of the **Apollo** model used in this project and "**Harpe**" which deals with the creation of a new evasion attack (*AML*), which we have used, in a simplified version, to test our **Apollo** model.

Keywords

Artificial Intelligence. Machine Learning. Deep Learning. Adversarial Machine Learning. Intrusion Detection System. Cybersecurity. MAB.

Índice General

CAPÍTULO 1. DEFINICIÓN DEL PROYECTO	33
1.1. INTRODUCCIÓN	34
1.2. MOTIVACIÓN DEL PROYECTO.....	36
1.3. DESCRIPCIÓN Y ALCANCE DEL PROYECTO.....	38
1.4. OBJETIVOS DEL PROYECTO	42
1.5. RESUMEN DE TODOS LOS ASPECTOS.....	43
CAPÍTULO 2. ESTADO ACTUAL DE LOS CONOCIMIENTOS CIENTÍFICO-TÉCNICOS	45
2.1. INFRAESTRUCTURA DE RED SEGURA (SNI)	46
2.2. SISTEMAS DE DETECCIÓN DE INTRUSOS	48
2.2.1. <i>Principales Datasets Utilizados en Sistemas de Detección de Intrusos</i>	49
2.2.1.1. CIC-IDS2017	49
2.2.1.2. CSE-CIC-IDS2018	49
2.2.1.3. CIC-DDoS2019	51
2.2.1.4. Datasets Descartados	52
2.2.2. <i>Clasificadores Utilizados en los IDS</i>	53
2.3. ADVERSARIAL MACHINE LEARNING (AML)	55
2.3.1. <i>Ataques en Función del Conocimiento del Modelo</i>	55
2.3.1.1.1. Ataques de Caja Blanca	55
2.3.1.1.2. Ejemplos de Ataques de Caja Blanca	56
2.3.1.2. Ataques de caja negra/gris	56
2.3.1.2.1. Redes Generativas Adversariales (GAN)	56
2.3.1.2.2. Ejemplos de Ataques de Caja Negra/Gris	57
2.3.2. <i>Ataques en Función de su Taxonomía</i>	57
2.3.2.1. Ataques de Evasión	58
2.3.2.2. Ataques de Inferencia.....	60
2.3.2.3. Ataques de Envenenamiento	61
2.3.2.4. Ataque de Extracción.....	62
2.4. MULTI-ARMED BANDITS (MAB).....	64
CAPÍTULO 3. APOLÓ	67
3.1. MODELOS	70
3.1.1. <i>CIC-FlowMeter</i>	70
3.1.2. <i>Preprocesado</i>	71
3.1.3. <i>Clasificadores</i>	72
3.1.3.1. Arboles de Decisión.....	73
3.1.3.2. Random Forest	74
3.1.3.3. Naive Bayes.....	74
3.1.3.4. MultiLayer Perceptron	75
3.1.3.5. Regresión Logística	76
3.1.3.6. SVC	77
3.1.3.7. K-Nearest Neighbors	77
3.1.4. <i>Ejemplo</i>	78
3.2. MAB	80
3.2.1. <i>Ejemplo</i>	82
3.3. CLUSTERIZACIÓN	84

3.3.1. Ejemplo.....	85
CAPÍTULO 4. PLANIFICACIÓN Y PRESUPUESTO DEL PROYECTO (INICIAL)	87
4.1. PLANIFICACIÓN INICIAL	88
4.2. PRESUPUESTO INICIAL.....	94
4.2.1. Definición de la Empresa	94
4.2.1.1. Personal	94
4.2.1.1.1. Productividad del Personal.....	95
4.2.1.2. Costes Indirectos.....	95
4.2.1.3. Amortizaciones	96
4.2.1.4. Precio Por Hora del Personal	97
4.2.1.5. Total	98
4.2.1.5.1. Total Costes Indirectos.....	98
4.2.1.5.2. Total Anual	98
4.2.2. Costes Directos	99
4.2.2.1. Partida 1	99
4.2.2.2. Partida 2	100
4.2.2.3. Partida 3	103
4.2.3. Resúmenes	106
4.2.3.1. Costes (Empresa)	106
4.2.3.2. Costes (Cliente)	107
CAPÍTULO 5. ANÁLISIS	109
5.1. REQUISITOS DEL SISTEMA	110
5.1.1. Interfaces Externas	110
5.1.1.1. Interfaz de Usuario	110
5.1.1.2. Interfaces de Hardware	110
5.1.1.3. Interfaces de Software	111
5.1.1.4. Interfaces de Comunicaciones	111
5.1.2. Requisitos Funcionales	111
5.1.3. Requisitos de rendimiento	114
5.1.4. Lógicos de BD.....	114
5.1.5. Restricciones de Diseño	114
5.1.6. Atributos del Sistema.....	114
5.1.6.1. Seguridad	114
5.1.6.2. Mantenimiento.....	115
5.1.7. Otros Requisitos	115
5.2. IDENTIFICACIÓN DE ACTORES DEL SISTEMA	116
5.3. IDENTIFICACIÓN DE LOS COMPONENTES DE LA ARQUITECTURA DEL SISTEMA	117
5.3.1. Descripción de los Componentes	117
5.3.1.1. Firewall y Proxy Reverso.....	117
5.3.1.2. Aplicación de Ejemplo	118
5.3.1.3. Gestión de Peticiones.....	118
5.3.1.4. Clasificación	119
5.3.1.4.1. Sistema de Colas	119
5.3.1.4.2. Apolo.....	120
5.3.1.4.3. Gestión de Anomalías.....	121
5.4. DIAGRAMA DE CONTEXTO	123
5.4.1. Caso de Uso: Identificarse UIX Monitorización	123
5.4.2. Caso de Uso: Identificarse UIX Anomalías.....	124

Apolo: Sistema inteligente de monitorización y detección de ataques basados en inteligencia artificial

5.4.3. Caso de Uso: Simular Peticiones	124
5.4.4. Caso de Uso: Tratar las Peticiones.....	125
5.4.5. Caso de Uso: Guardar Peticiones	125
5.4.6. Caso de Uso: Monitorizar Peticiones	125
5.4.7. Caso de Uso: Clasificar Peticiones	126
5.4.8. Caso de Uso: Guardar Clasificaciones	126
5.4.9. Caso de Uso: Enviar Notificaciones.....	127
5.4.10. Caso de Uso: Mostrar Clasificaciones	127
5.5. ANÁLISIS DE CASOS DE USO Y ESCENARIOS.....	128
5.5.1. Diagramas de Casos de Uso	128
5.5.1.1. Caso de Uso Expandido: Tratamiento de las Peticiones	128
5.5.1.2. Caso de Uso Expandido: Clasificación	129
5.5.2. Escenarios.....	129
5.5.2.1. Escenario: Tratamiento de las Peticiones	129
5.5.2.1.1. Caso de Uso Expandido: Simular Peticiones	129
5.5.2.1.2. Caso de Uso Expandido: Dar Formato Inicial	130
5.5.2.1.3. Caso de Uso Expandido: Recibir Peticiones	130
5.5.2.1.4. Caso de Uso Expandido: Tratar y Distribuir	130
5.5.2.1.5. Caso de Uso Expandido: Indexar.....	131
5.5.2.2. Escenario: Clasificación	131
5.5.2.2.1. Caso de Uso Expandido: Elector de Módulos por Clúster.....	131
5.5.2.2.2. Caso de Uso Expandido: Predicción de los modelos	132
5.5.2.2.3. Caso de Uso Expandido: Elección de los Resultados de los Modelos	132
5.5.2.2.4. Caso de Uso Expandido: Guardar Clasificaciones	132
5.5.2.2.5. Caso de Uso Expandido: Monitorizar Clasificaciones	133
5.5.2.2.6. Caso de Uso Expandido: Gestión del Dashboard	133
CAPÍTULO 6. DEFINICIÓN DEL SISTEMA	135
6.1. MICROSERVICIOS	136
6.2. DEFINICIÓN DE LOS COMPONENTES DE LA ARQUITECTURA	140
6.2.1. Firewall y Proxy Reverso	140
6.2.2. Aplicación de Ejemplo.....	142
6.2.3. Gestión de Peticiones	142
6.2.4. Clasificación	145
6.2.4.1. Sistema de Colas	146
6.2.4.2. Gestión de Anomalías	148
6.3. DESCRIPCIÓN DE LA COMUNICACIÓN ENTRE COMPONENTES DE LA ARQUITECTURA	150
6.4. ESTÁNDARES Y NORMAS SEGUIDOS	151
6.4.1. Estándares	151
6.4.1.1. IEE 830-1998	151
6.4.2. Normas.....	151
6.4.2.1. UNE 157801	151
6.4.2.2. UNE 157001	152
6.5. LENGUAJES, ENTORNOS Y HERRAMIENTAS.....	153
6.5.1. Lenguajes	153
6.5.1.1. Python	153
6.5.1.2. Shell Script	155
6.5.1.3. YAML	156
6.5.2. Entornos	158
6.5.2.1. Visual Studio Code.....	158
6.5.2.2. Jupyter Notebook	159

6.5.3. Herramientas	161
6.5.3.1. Git	161
6.5.3.2. Docker Hub.....	162
6.5.3.3. Draw.io	164
6.5.3.4. PlantUML.....	165
CAPÍTULO 7. EVALUACIÓN DE ALTERNATIVAS	167
7.1. ALTERNATIVAS A APOLO	168
7.1.1. Forecasting	168
7.1.2. Reentrenar con AML.....	169
7.2. ALTERNATIVAS A LOS COMPONENTES DE LA ARQUITECTURA.....	170
7.2.1. Firewall y Proxy Reverso	170
7.2.1.1. Apache HTTP Server	170
7.2.1.1.1. Ventajas.....	170
7.2.1.1.2. Desventajas.....	171
7.2.1.1.3. Conclusión.....	172
7.2.1.2. Caddy.....	172
7.2.1.2.1. Ventajas.....	173
7.2.1.2.2. Desventajas.....	174
7.2.1.2.3. Conclusión.....	174
7.2.2. Gestión de peticiones.....	175
7.2.2.1. Fluentd, Grafana y Prometheus.....	175
7.2.2.1.1. Ventajas.....	177
7.2.2.1.2. Inconvenientes	177
7.2.2.1.3. Conclusión.....	177
7.2.2.2. Graylog	178
7.2.2.2.1. Ventajas.....	178
7.2.2.2.2. Desventajas.....	179
7.2.2.2.3. Conclusión.....	179
7.2.3. Clasificación	180
7.2.3.1. Sistema de colas	180
7.2.3.1.1. Hazelcast.....	180
7.2.3.1.1.1. Ventajas	181
7.2.3.1.1.2. Desventajas	181
7.2.3.1.1.3. Conclusión	182
7.2.3.1.2. Azure Cosmos DB	182
7.2.3.1.2.1. Ventajas	183
7.2.3.1.2.2. Desventajas	183
7.2.3.1.2.3. Conclusión	184
7.2.3.2. Gestión de Anomalías	184
7.2.3.2.1. Prometheus	184
7.2.3.2.1.1. Ventajas	185
7.2.3.2.1.2. Desventajas	185
7.2.3.2.1.3. Conclusión	186
7.2.3.2.2. TimescaleDB	186
7.2.3.2.2.1. Ventajas	187
7.2.3.2.2.2. Desventajas	187
7.2.3.2.2.3. Conclusión	188
7.3. ALTERNATIVAS GENERALES	189
7.3.1. Microservicios	189
7.3.1.1. Podman	189

Apolo: Sistema inteligente de monitorización y detección de ataques basados en inteligencia artificial

7.3.1.1.1. Ventajas.....	189
7.3.1.1.2. Desventajas.....	190
7.3.1.1.3. Conclusión.....	191
7.3.2. Entornos	191
7.3.2.1. PyCharm.....	191
7.3.2.1.1. Ventajas.....	192
7.3.2.1.2. Desventajas.....	192
7.3.2.1.3. Conclusión.....	193
7.3.3. Lenguajes.....	193
7.3.3.1. R.....	193
7.3.3.1.1. Ventajas.....	194
7.3.3.1.2. Desventajas.....	194
7.3.3.1.3. Conclusión.....	194
7.3.3.2. Julia	195
7.3.3.2.1. Ventajas.....	195
7.3.3.2.2. Desventajas.....	196
7.3.3.2.3. Conclusión.....	196
CAPÍTULO 8. DISEÑO DEL SISTEMA.....	197
8.1. DIAGRAMAS DE PAQUETES.....	198
8.1.1. Paquete Apolo	200
8.1.1.1. Layers	200
8.1.1.2. Paquetes Preprocesing, Model Train y Model Predict.....	201
8.1.2. Paquetes Utils, Services y Storage	202
8.2. DIAGRAMAS DE DESPLIEGUE	204
8.3. DIAGRAMA DE CLASES.....	205
8.3.1. Clases del Sistema.....	205
8.3.1.1. Paquete Apolo.....	205
8.3.1.1.1. Layers	205
8.3.1.1.1.1. Models	205
8.3.1.1.1.2. Clustering.....	210
8.3.1.1.1.3. MAB.....	211
8.3.1.1.2. Paquetes Preprocesing, Model Train y Model Predict	213
8.3.1.1.2.1. Preprocesing.....	213
8.3.1.1.2.2. Model Train.....	220
8.3.1.1.2.3. Model Predict	221
8.3.1.2. Paquetes Utils, Services y Storage	222
8.3.1.2.1. Utils	222
8.3.1.2.2. Services	224
8.3.1.2.3. Storage	226
8.3.2. Patrones de Diseño	227
8.3.2.1. Clase Model y sus Hijas	228
8.3.2.2. Clase ClearData y sus Hijas	228
8.3.3. Diagrama de Clases	229
8.4. DISEÑO DEL SISTEMA DE COLAS.....	231
8.4.1. Cola FIFO.....	231
8.4.2. Gestión de Peticiones mediante Clave-Valor.....	231
8.5. DISEÑO DE LA BASE DATOS DE SERIES TEMPORALES	232
8.5.1. Buckets.....	232
8.5.2. Measurements.....	233
CAPÍTULO 9. EVALUACIÓN DEL SISTEMA.....	235

9.1. HARDWARE UTILIZADO.....	236
9.2. METODOLOGÍA.....	237
9.2.1. Precisión / Exactitud	237
9.2.2. Tasa de Detección.....	238
9.2.3. Curva ROC	239
9.2.4. Valor F1	240
9.3. VALIDACIÓN	241
9.3.1. Validación Cruzada	241
9.4. EXPERIMENTACIÓN DE APOLO	243
9.4.1. Primer Escenario	243
9.4.1.1. Definición del Escenario.....	243
9.4.1.2. Resultados Obtenidos	244
9.4.2. Segundo Escenario.....	247
9.4.2.1. Definición del Escenario.....	248
9.4.2.2. Ataques Utilizados.....	248
9.4.2.3. Resultados Obtenidos	249
9.4.2.3.1. Ataque de Optimización de Orden Cero (ZOO).....	249
9.4.2.3.2. Ataque HopSkipJump (HSJA)	251
9.4.2.3.3. Red Adversaria Generativa de Wasserstein (W-GAN)	253
9.4.2.3.4. Conclusión.....	255
9.4.3. Tercer Escenario.....	255
9.4.3.1. Definición del Escenario.....	255
9.4.3.2. Resultados Obtenidos	256
CAPÍTULO 10. MANUALES DEL SISTEMA.....	258
10.1. MANUAL DE INSTALACIÓN Y EJECUCIÓN	259
10.1.1. Instalación.....	259
10.1.1.1. Instalación de Docker	259
10.1.1.1.1. Linux	259
10.1.1.1.2. MacOS	260
10.1.1.1.3. Windows 10 Pro/Enterprise.....	260
10.1.1.2. Instalación de Docker Compose	260
10.1.1.2.1. Linux / Ubuntu / macOS	260
10.1.1.2.2. Windows 10 Pro/Enterprise.....	261
10.1.1.3. Verificación	261
10.1.2. Ejecución del Sistema	261
10.1.3. Detención del Sistema	262
10.2. MANUAL DE USUARIO	264
10.2.1. Kibana.....	264
10.2.2. InfluxDB	272
10.3. MANUAL DEL PROGRAMADOR	279
10.3.1.1. Añadir un nuevo modelo de aprendizaje automático	279
10.3.1.2. Añadir una nueva clase de preprocesamiento de datos	280
CAPÍTULO 11. CONCLUSIONES Y FUTURO TRABAJO	283
11.1. CONCLUSIONES	284
11.2. FUTURO TRABAJO	285
CAPÍTULO 12. PLANIFICACIÓN Y PRESUPUESTO DEL PROYECTO (FINAL)	287
12.1. PLANIFICACIÓN FINAL	288

Apolo: Sistema inteligente de monitorización y detección de ataques basados en inteligencia artificial

12.1.1. Planificación	288
12.1.2. Organigrama	302
12.2. PRESUPUESTO FINAL.....	304
12.2.1. Definición de la Empresa.....	304
12.2.1.1. Personal	304
12.2.1.1.1. Productividad del Personal.....	306
12.2.1.2. Costes Indirectos.....	308
12.2.1.3. Amortizaciones.....	308
12.2.1.4. Precio Por Hora del Personal.....	309
12.2.1.5. Total	310
12.2.1.5.1. Total Costes Indirectos	310
12.2.1.5.2. Total Anual	311
12.2.2. Costes Directos	312
12.2.2.1. Partida 1	312
12.2.2.2. Partida 2	313
12.2.2.3. Partida 3.....	314
12.2.2.4. Partida 4.....	316
12.2.2.5. Partida 5.....	317
12.2.2.6. Partida 6.....	319
12.2.2.7. Partida 7.....	321
12.2.2.8. Partida 8.....	321
12.2.2.9. Partida 9.....	322
12.2.2.10. Partida 10.....	324
12.2.2.11. Partida 11.....	325
12.2.2.12. Partida 12.....	326
12.2.2.13. Partida 13.....	327
12.2.2.14. Partida 14.....	328
12.2.3. Resúmenes	328
12.2.3.1. Costes (Empresa).....	328
12.2.3.2. Costes (Cliente)	329
CAPÍTULO 13. REFERENCIAS BIBLIOGRÁFICAS	331
APÉNDICES	339
A.1. ARTÍCULOS.....	340
A.1.1. Harpe.....	340
A.1.1.1. Abstract	340
A.1.1.2. Revista	341
A.1.1.3. Estado actual	342
A.1.2. Apollon	342
A.1.2.1. Abstract	342
A.1.2.2. Revista	342
A.1.2.3. Estado actual	343
A.2. ANÁLISIS Y GESTIÓN DE LOS RIESGOS.....	344
A.2.1. Metodología.....	344
A.2.1.1. Conceptos Generales	344
A.2.1.2. Categorización.....	344
A.2.1.3. Estimación de la Probabilidad e Impacto.....	345
A.2.1.4. Estrategias para la Gestión de Riesgos.....	347
A.2.2. Análisis de Riesgos	348
A.2.2.1. Ciberataque al sistema	348
A.2.2.2. Rechazo de los Artículos Realizados	350
A.2.2.3. Retraso en la Entrega	351

A.2.2.4. Cálculo Erróneo del Presupuesto	352
A.2.2.5. Falta de Madurez del Personal	353
A.2.2.6. Aumento de los Costes Eléctricos.....	354
A.2.2.7. Problemas con la Aceptación del Cliente	355
A.2.2.8. Problemas con la Integración del Sistema	356
A.2.2.9. Problemas con la Escalabilidad.....	357
A.2.2.10. Aparición de Competidores	358
A.2.3. Gestión de Riesgos	360
A.2.3.1. Ciberataque al sistema	360
A.2.3.2. Rechazo de los Artículos Realizados	361
A.2.3.3. Retraso en la Entrega	361
A.2.3.4. Cálculo Erróneo del Presupuesto	362
A.2.3.5. Falta de Madurez del Personal	362
A.2.3.6. Aumento de los Costes Eléctricos.....	363
A.2.3.7. Problemas con la Aceptación del Cliente	363
A.2.3.8. Problemas con la Integración del Sistema	364
A.2.3.9. Problemas cos la Escalabilidad.....	365
A.2.3.10. Aparición de Competidores	366
A.3. GLOSARIO Y DICCIONARIO DE DATOS	367
A.4. CONTENIDO ENTREGADO EN EL ARCHIVO ADJUNTO.....	370
A.4.1. Project.....	370
A.4.2. Excel.....	370
A.4.3. Código.....	370
A.5. ÍNDICE ALFABÉTICO	372
A.6. CÓDIGO FUENTE	374

Índice de Figuras

Ilustración 1. Representación abstracta de la arquitectura del sistema.	40
Ilustración 2. Infraestructura de red segura (SNI)	41
Ilustración 3. Ataque de caja negra y de caja blanca, extraído del artículo de He Y. et al. [6] ...	55
Ilustración 4. Adversarial Machine Learning attacks por Nicolae M. et al. [53].....	58
Ilustración 5. Adversarial Machine Learning attacks por He Y. et al. [6]	58
Ilustración 6. Ataque de evasión, extraído del artículo de Qiu S. et al. [5]	59
Ilustración 7. Ataque de evasión, extraído del artículo de Qiu S. et al. [5].....	60
Ilustración 8. Ataque de inversión, extraído del artículo de He Y. et al. [6]	61
Ilustración 9. Ataque de envenenamiento, extraído del artículo de He Y. et al. [6]	62
Ilustración 10. Ataque de extracción, extraído del artículo de He Y. et al. [6]	63
Ilustración 11. Ejemplo del proceso de entrenamiento de Apolo.	68
Ilustración 12. Ejemplo del proceso de selección de los clasificadores de Apolo.	68
Ilustración 13. El funcionamiento de Apolo.	69
Ilustración 14. Apolo: Capa Modelos	79
Ilustración 15. Algoritmo: Apolo Thompson Sampling	81
Ilustración 16. Apolo: El algoritmo Multi-Armed Bandits con varios clasificadores.....	82
Ilustración 17. Apolo: Capa MAB	83
Ilustración 18. Apolo: Capa Clúster.....	85
Ilustración 19. Planificación de las tareas de reuniones semanales.....	90
Ilustración 20. Planificación de las tareas de desarrollo.	91
Ilustración 21. Planificación de las tareas de documentación.	91
Ilustración 22. Diagrama de Gantt de las reuniones semanales.	92
Ilustración 23. Diagrama de Gantt del desarrollo.	92
Ilustración 24. Diagrama de Gantt de la documentación.	93
Ilustración 25. Identificación de la Arquitectura del Sistema.	117
Ilustración 26. Diagrama de Contexto.	123
Ilustración 27. Caso de Uso Expandido 1.	128
Ilustración 28. Caso de Uso Expandido 2.	129
Ilustración 29. Logo de Docker.	139
Ilustración 30. Logo de Docker Compose.....	139
Ilustración 31. Arquitectura del Sistema.	140
Ilustración 32. Logo de NGINX.	142
Ilustración 33. Logo de Stack ELK incluyendo Beats.	145
Ilustración 34. Logo de Redis.....	147
Ilustración 35. Logo de InfluxData.	149
Ilustración 36. Logo de InfluxDB.	149
Ilustración 37. Logo de Python.	154
Ilustración 38. Logo de Shell Script.	156
Ilustración 39. Logo de YAML.	157
Ilustración 40. Logo de Visual Studio Code.	159
Ilustración 41. Logo de Jupyter Notebook.	161
Ilustración 42. Logo de Git.	162
Ilustración 43. Logo de Docker Hub.	163
Ilustración 44. Logo de Draw.io.....	165

Ilustración 45. Logo de PlantUML	166
Ilustración 46. Logo de Apache HTPP Server.....	170
Ilustración 47. Logo de Caddy.....	173
Ilustración 48. Logo de Fluentd.....	175
Ilustración 49. Logo de Grafana.	176
Ilustración 50. Logo de Prometheus.	176
Ilustración 51. Logo de Graylog.	178
Ilustración 52. Logo de Hazelcast.....	180
Ilustración 53. Logo de Azure Cosmos DB.	182
Ilustración 54. Logo de Prometheus.	184
Ilustración 55. Logo de Timescale.	186
Ilustración 56. Logo de Podman.	189
Ilustración 57. Logo de PyCharm.	191
Ilustración 58. Logo de R.	193
Ilustración 59. Logo de Julia.	195
Ilustración 60. Diagrama Paquetes: Global	199
Ilustración 61. Diagrama Paquetes: Layers	201
Ilustración 62. Diagrama Paquetes: Preprocesing, model train y model predict	202
Ilustración 63. Diagrama Paquetes: Paquetes independientes al paquete "apolo".	203
Ilustración 64. Diagrama de despliegue.	204
Ilustración 65. Diagrama de Despliegue: Red Virtual.....	204
Ilustración 66. Diseño de Clases: Model.	207
Ilustración 67. Diseño de Clases: SVC.	208
Ilustración 68. Diseño de Clases: RandomForest.	208
Ilustración 69. Diseño de Clases: MLP.	209
Ilustración 70. Diseño de Clases: LogisticRegressionModel.	209
Ilustración 71. Diseño de Clases: KNeighbors.	209
Ilustración 72. Diseño de Clases: DecisionTree.	210
Ilustración 73. Diseño de Clases: NaiveBayes.....	210
Ilustración 74. Diseño de Clases: KMeans.	211
Ilustración 75. Diseño de Clases: MAB.	213
Ilustración 76. Diseño de Clases: Transform.....	214
Ilustración 77. Diseño de Clases: ClearData.....	216
Ilustración 78. Diseño de Clases: ClearDataOur.	217
Ilustración 79. Diseño de Clases: ClearDataCIC2019.	218
Ilustración 80. Diseño de Clases: ClearDataCIC2018.	219
Ilustración 81. Diseño de Clases: ClearDataCIC2017.	220
Ilustración 82. Diseño de Clases: ApoloTrainer.....	221
Ilustración 83. Diseño de Clases: ApoloPredict.	222
Ilustración 84. Diseño de Clases: UtilsSave.	222
Ilustración 85. Diseño de Clases: UtilsLoad.	223
Ilustración 86. Diseño de Clases: DataSelector.....	224
Ilustración 87. Diseño de Clases: RedisService.....	225
Ilustración 88. Diseño de Clases: InfluxDBService.	225
Ilustración 89. Diseño de Clases: ScoreManager.	226
Ilustración 90. Diseño de Clases: DataCollector.	227
Ilustración 91. Patrón de Diseño: Strategy	227

Ilustración 92. Diagrama de Clases.....	230
Ilustración 93. FIFO (First In, First Out)	231
Ilustración 94. Estructuración interna de una Base de Datos InfluxDB	232
Ilustración 95. Validación cruzada	242
Ilustración 96. Primer escenario: Tasa de Detección (Recall) de los modelos y de Apolo entrenados con CIC-IDS-2017.....	245
Ilustración 97. Primer escenario: Tasa de Detección (Recall) de los modelos y de Apolo entrenados con CIC-IDS-2018.....	246
Ilustración 98. Primer escenario: Tasa de Detección (Recall) de los modelos y de Apolo entrenados con CIC-DDoS-2019.	247
Ilustración 99. Segundo escenario: Tasa de Detección (Recall) de los modelos y de Apolo. Utilizando el ataque ZOO.....	250
Ilustración 100. Segundo escenario: Precisión de los modelos y de Apolo. Utilizando el ataque ZOO.	251
Ilustración 101. Segundo escenario: Tasa de Detección (Recall) de los modelos y de Apolo. Utilizando el ataque HSJA.....	252
Ilustración 102. Segundo escenario: Precisión de los modelos y de Apolo. Utilizando el ataque HSJA.	253
Ilustración 103. Segundo escenario: Tasa de Detección (Recall) de los modelos y de Apolo. Utilizando el ataque W-GAN.	254
Ilustración 104. Segundo escenario: Precisión de los modelos y de Apolo. Utilizando el ataque W-GAN.....	254
Ilustración 105. Manual de Usuario Kibana: Inicio.....	264
Ilustración 106. Manual de Usuario Kibana: Dashboards.	265
Ilustración 107. Manual de Usuario Kibana: Visualización de Datos.....	265
Ilustración 108. Manual de Usuario Kibana: Creación de Gráficas.	266
Ilustración 109. Manual de Usuario Kibana: Panel Dashboards.....	267
Ilustración 110. Manual de Usuario Kibana: Gráfica Donut.....	267
Ilustración 111. Manual de Usuario Kibana: Múltiples Atributos en Gráficas (Parte I).	268
Ilustración 112. Manual de Usuario Kibana: Múltiples Atributos en Gráficas (Parte II).	269
Ilustración 113. Manual de Usuario Kibana: Panel con Gráficas.....	270
Ilustración 114. Manual de Usuario Kibana: Panel con Gráficas, Actualización.	270
Ilustración 115. Manual de Usuario Kibana: Guardado Panel.	271
Ilustración 116. Manual de Usuario Kibana: Paneles Guardados.	271
Ilustración 117. Manual de Usuario InfluxDB: Inicia.	272
Ilustración 118. Manual de Usuario InfluxDB: Panel Principal.	273
Ilustración 119. Manual de Usuario InfluxDB: Buckets.	273
Ilustración 120. Manual de Usuario InfluxDB: Data Explorer.....	274
Ilustración 121. Manual de Usuario InfluxDB: Datos Peticiones.	274
Ilustración 122. Manual de Usuario InfluxDB: Heatmap.	275
Ilustración 123. Manual de Usuario InfluxDB: Data.	276
Ilustración 124. Manual de Usuario InfluxDB: Tabla de Datos.	276
Ilustración 125. Manual de Usuario InfluxDB: Filtros.....	277
Ilustración 126. Manual de Usuario InfluxDB: Guardado (Parte I).	277
Ilustración 127. Manual de Usuario InfluxDB: Guardado (Parte II).	278
Ilustración 128. Manual de Usuario InfluxDB: Dashboards Guardados.	278
Ilustración 129. Planificación Final: Hitos.....	289
Ilustración 130. Planificación Final: Recursos y Roles.	290

Ilustración 131. Planificación Final: Tareas I	291
Ilustración 132. Planificación Final: Tareas II	292
Ilustración 133. Planificación Final: Tareas III	293
Ilustración 134. Planificación Final: Tareas IV	294
Ilustración 135. Planificación Final: Tareas V	294
Ilustración 136. Planificación Final: Tareas VI	295
Ilustración 137. Planificación Final: Tareas VII	295
Ilustración 138. Planificación Final: Diagrama de Gantt I	296
Ilustración 139. Planificación Final: Diagrama de Gantt II	296
Ilustración 140. Planificación Final: Diagrama de Gantt III	297
Ilustración 141. Planificación Final: Diagrama de Gantt IV	297
Ilustración 142. Planificación Final: Diagrama de Gantt V	298
Ilustración 143. Planificación Final: Diagrama de Gantt VI	299
Ilustración 144. Planificación Final: Diagrama de Gantt VII	300
Ilustración 145. Planificación Final: Diagrama de Gantt VIII	301
Ilustración 146. Planificación Final: Diagrama de Gantt IX	302
Ilustración 147. Planificación Final: Apolo	302
Ilustración 148. Planificación: Organigrama	303
Ilustración 149. Revista del artículo Harpe: Journal of Systems Architecture	341
Ilustración 150. Revista del artículo Apollon: Computers & Security	343

Índice de Tablas

Tabla 1. Resumen CIC-IDS 2017	49
Tabla 2. Resumen CIC-IDS 2018	50
Tabla 3. Tipo de máquinas involucradas en la obtención del dataset y sus respectivos sistemas operativos e IPs	51
Tabla 4. Nombres de los ataques ejecutados cada día, con sus respectivas horas	52
Tabla 5. Resumen de los clasificadores de IDS para el dataset CIC-IDS2017.....	53
Tabla 6. Resumen de los clasificadores de IDS para el dataset CSE-CIC-IDS2018.....	54
Tabla 7. Resumen de los clasificadores de IDS para el dataset CIC-DDoS2019.	54
Tabla 8. Personal.....	94
Tabla 9. Precio / hora personal	94
Tabla 10. Productividad personal	95
Tabla 11. Horas total productivas personal.....	95
Tabla 12. CI generales	96
Tabla 13. Amortizaciones.....	97
Tabla 14. Precio / Hora Personal (Parte I).....	97
Tabla 15. Precio / Hora Personal (Parte II).....	97
Tabla 16. Total CI	98
Tabla 17. Total Anual.....	99
Tabla 18. Partida 1 (Parte I).....	100
Tabla 19. Partida 1 (Parte II).....	100
Tabla 20. Partida 2 (Parte I).....	102
Tabla 21. Partida 2 (Parte II).....	103
Tabla 22. Partida 3 (Parte I).....	104
Tabla 23. Partida 3 (Parte II).....	105
Tabla 24. Presupuesto Empresa Agregado	106
Tabla 25. Presupuesto Empresa Detallado.....	107
Tabla 26. Presupuesto Cliente Agregado	108
Tabla 27. Presupuesto Cliente Detallado	108
Tabla 28. CU1: Identificarse UIX monitorización.....	124
Tabla 29. CU2: Identificarse UIX anomalías.	124
Tabla 30. CU3: Simular peticiones.	125
Tabla 31. CU4: Tratar las peticiones.	125
Tabla 32. CU5: Guardar peticiones.....	125
Tabla 33. CU6: Monitorizar peticiones.	126
Tabla 34. CU7: Clasificar Peticiones.	126
Tabla 35. CU8: Guardar clasificaciones.	126
Tabla 36. CU9: Enviar notificaciones.	127
Tabla 37. CU10: Mostrar clasificaciones.	127
Tabla 38. Descripción Caso de Uso Expandido 1.....	128
Tabla 39. Descripción Caso de Uso Expandido 2.....	129
Tabla 40. CUE1: Simular peticiones.	130
Tabla 41. CUE1: Dar formato inicial.	130
Tabla 42. CUE1: Recibir peticiones.	130
Tabla 43. CUE1: Tratar y Distribuir.....	131

Tabla 44. CUE1: Indexar.	131
Tabla 45. CUE2: Elector de módulos por clúster.	132
Tabla 46. CUE2: Predicción de los modelos.	132
Tabla 47. CUE2: Elección de los resultados de los modelos.	132
Tabla 48. CUE2: Guardar clasificaciones.	133
Tabla 49. CUE2: Monitorizar clasificaciones.	133
Tabla 50. CUE2: Gestión del Dashboard.	133
Tabla 51. Primer escenario: Métricas de los modelos y de Apolo entrenados con CIC-IDS-2017.	245
Tabla 52. Primer escenario: Métricas de los modelos y de Apolo entrenados con CIC-IDS-2018.	246
Tabla 53. Primer escenario: Métricas de los modelos y de Apolo entrenados con CIC-DDoS-2019.	247
Tabla 54. Segundo escenario: Tasa de Detección (Recall) y Precisión de los modelos y de Apolo. Utilizando el ataque ZOO.	250
Tabla 55. Segundo escenario: Tasa de Detección (Recall) y Precisión de los modelos y de Apolo. Utilizando el ataque HSJA.	252
Tabla 56. Segundo escenario: Tasa de Detección (Recall) y Precisión de los modelos y de Apolo. Utilizando el ataque W-GAN.	253
Tabla 57. Tercer Escenario: Resultados.	257
Tabla 58. Personal.	305
Tabla 59. Precio / hora personal.	306
Tabla 60. Productividad personal.	307
Tabla 61. Horas total productivas personal.	308
Tabla 62. Precio / Hora Personal (Parte I).	309
Tabla 63. Precio / Hora Personal (Parte II).	310
Tabla 64. Total CI.	311
Tabla 65. Total Anual.	311
Tabla 66. Partida 1 (Parte I).	312
Tabla 67. Partida 1 (Parte II).	313
Tabla 68. Partida 2 (Parte I).	313
Tabla 69. Partida 2 (Parte II).	314
Tabla 70. Partida 3 (Parte I).	315
Tabla 71. Partida 3 (Parte II).	315
Tabla 72. Partida 4 (Parte I).	316
Tabla 73. Partida 4 (Parte II).	317
Tabla 74. Partida 5 (Parte I).	318
Tabla 75. Partida 5 (Parte II).	318
Tabla 76. Partida 6 (Parte I).	319
Tabla 77. Partida 6 (Parte II).	320
Tabla 78. Partida 7 (Parte I).	321
Tabla 79. Partida 7 (Parte II).	321
Tabla 80. Partida 8 (Parte I).	322
Tabla 81. Partida 8 (Parte II).	322
Tabla 82. Partida 9 (Parte I).	323
Tabla 83. Partida 9 (Parte II).	324
Tabla 84. Partida 10 (Parte I).	324

Tabla 85. Partida 10 (Parte II).....	325
Tabla 86. Partida 11 (Parte I).....	326
Tabla 87. Partida 11 (Parte II).....	326
Tabla 88. Partida 12 (Parte I).....	326
Tabla 89. Partida 12 (Parte II).....	327
Tabla 90. Partida 13 (Parte I).....	327
Tabla 91. Partida 13 (Parte II).....	327
Tabla 92. Partida 14 (Parte I).....	328
Tabla 93. Partida 14 (Parte II).....	328
Tabla 94. Presupuesto Empresa	329
Tabla 95. Presupuesto Cliente	330
Tabla 96. Los 4 impactos medibles.....	346
Tabla 97. Tabla de cálculo de estimación del impacto de los riesgos.	347
Tabla 98. Rangos de color para el impacto de los riesgos.....	347
Tabla 99. Análisis de Riesgos: Ciberataque al sistema.	349
Tabla 100. Análisis de Riesgos: Rechazo de los Artículos Realizados.	351
Tabla 101. Análisis de Riesgos: Retraso en el Entrega.....	352
Tabla 102. Análisis de Riesgos: Calculo Erróneo del Presupuesto	353
Tabla 103. Análisis de Riesgos: Falta de Madurez del Personal.	354
Tabla 104. Análisis de Riesgos: Aumento de los Costes Eléctricos.....	355
Tabla 105. Análisis de Riesgos: Problemas con la Aceptación del Cliente.	356
Tabla 106. Análisis de Riesgos: Problemas con la Integración del Sistema.	357
Tabla 107. Análisis de Riesgos: Problemas con la Escalabilidad	358
Tabla 108. Análisis de Riesgos: Aparición de Competidores.....	359
Tabla 109. Estructura código.	371

Tabla de Ecuaciones

Ecuación 1. Formula Accuracy.....	237
Ecuación 2. Formula Recall.....	238
Ecuación 3. Formula TFP, Curva ROC	239
Ecuación 4. Formula Precision.....	240
Ecuación 5. Formula F1-score.	240

Capítulo 1. Definición del Proyecto

En el siguiente capítulo, se realizará una definición del proyecto, proporcionándole una introducción y contexto de la situación actual; la motivación para su realización; los objetivos del proyecto y la descripción formal y alcance de este. Por último, se introducirán el resto de los capítulos que forman parte de la memoria.



1.1. Introducción

La seguridad de los sistemas informáticos es de vital importancia debido al creciente número de amenazas y ataques ciberneticos que se producen a diario. Cada vez más organizaciones y usuarios individuales son víctimas de ataques informáticos, lo que puede resultar en la pérdida de información confidencial, la interrupción de servicios o un impacto negativo en su reputación. Sea cual sea el sistema informático, este es vulnerable frente a diversos ataques, ya sean ataques al hardware o al software del sistema, siendo estos últimos los más comunes y los que tienen más variantes. En el caso de que el sistema que se quiera defender tenga acceso a una red que comunique varios dispositivos, este se convierte en una potencial vulnerabilidad al ser un punto de acceso al sistema. Existen múltiples opciones para tratar de defenderse frente a estos ataques, pero una de las más extendidas es el uso de un *Sistema de Detención de Intrusos (IDS)* [1], el cual se encarga de monitorizar esta entrada al sistema y potencial vulnerabilidad.

Los sistemas de detección de intrusiones o *IDS* son un tipo de solución de seguridad que monitorean el tráfico de la red o un sistema informático en tiempo real para detectar y alertar sobre cualquier actividad sospechosa o maliciosa. Se sitúan en un punto de la red, como un *router* o un *switch*, y analizan todo el tráfico que pasa a través de ese punto. Estos sistemas se basan en un análisis y una comparación del tráfico de red, para ello utilizan una serie de reglas y patrones de comportamiento previamente definidos para identificar actividades que puedan ser indicativas de una amenaza de seguridad.

Tal y como se explica en el artículo de *Liao H. et al.* [2], existen dos tipos principales de sistemas *IDS* basados en red: los sistemas de detección de intrusiones en una red a tiempo real (*NIDS*) y los sistemas de detección de intrusiones de host de análisis de paquetes (*HIDS*).

Los *NIDS* monitorean el tráfico de la red en tiempo real y comparan el tráfico con una serie de reglas y patrones previamente definidos para identificar actividades sospechosas o maliciosas. Estos sistemas son capaces de detectar intrusiones en tiempo real, lo que les permite responder rápidamente a las amenazas de seguridad.

Los *HIDS*, por otro lado, monitorean el tráfico de la red y lo almacenan en un archivo para su posterior análisis. Este enfoque es útil para detectar intrusiones que pueden ser difíciles de detectar en tiempo real, como las intrusiones que se desarrollan lentamente a lo largo del tiempo.

Estos dos tipos no son excluyentes, de hecho, nuestro *sistema de detección de intrusos* es una combinación de un *HIDS* y de *NIDS*, por ello, a partir de ahora y en todo el documento, no se realizará una distinción entre estos dos tipos de *IDS*. Por lo que, en todo el documento, se referirá a ellos como *IDS*.

Las técnicas de detección que utilizan los *IDS*, por lo general, estaban basadas en patrones [2], análisis de comportamiento y en una serie de reglas que, en conjunto, pueden llegar a detectar peticiones a la red que son maliciosas. Es importante tener en cuenta que estos sistemas pueden generar falsos positivos o falsos negativos y deben ser parte de una estrategia integral de seguridad para ser efectivos. Debido a esto, los *IDS* suelen trabajar en conjunto con otras soluciones de seguridad, como *firewalls* y sistemas antivirus, para proporcionar una defensa completa contra las amenazas de seguridad, como vemos en el artículo de Ashoor A. et al. [1].

También, hay que tener en cuenta que los últimos avances tecnológicos han permitido realizar ataques mucho más potentes hacia los sistemas informáticos. Frente a estos avances, los *IDS* tradicionales (basados en reglas y patrones) no pueden detectar estos ataques, por lo que se genera una fuerte vulnerabilidad. Para intentar detectar estos ataques se ha optado por aplicar técnicas de *Machine Learning (ML)* en los *IDS*, esta solución parece que ha funcionado, tal y como vemos en el artículo de Vinayakumar R et al. [3], puesto que se han conseguido detectar muchas de estas peticiones maliciosas.

Las técnicas de *ML* en *IDS* ofrecen varias ventajas en comparación con los enfoques basados en reglas y patrones [3]. En primer lugar, son capaces de detectar patrones y características que pueden ser difíciles de identificar mediante reglas y patrones predefinidos. Además, son capaces de aprender y adaptarse continuamente a las nuevas amenazas de seguridad. Sin embargo, esto no las hace exentas de generar falsos positivos o falsos negativos y hay que tener en cuenta que están en una fase temprana de desarrollo.

1.2. Motivación del Proyecto

El uso de técnicas de *ML* ha supuesto una gran mejora frente a los enfoques más tradicionales (basado en reglas y patrones definidos), permitiendo una mayor adaptabilidad y precisión, reforzando enormemente a los *IDS* [3]. Sin embargo, el uso de estas técnicas no garantiza que el sistema sea invulnerable. Los nuevos avances en los ciberataques, como los ataques Adversariales basados en técnicas de *Machine Learning (AML)*, son capaces de engañar y evadir a estos *ML IDS*, tal y como afirma *Jmila H. et al.* [4].

Los ataques Adversariales basados en técnicas de *Machine Learning (AML)* [5], [6] hacia los *IDS* tradicionales son efectivos porque explotan las debilidades y limitaciones de los algoritmos y tecnologías utilizadas en estos. Estos ataques tienen como objetivo engañar a los *IDS* [4] y hacer que no detecten una intrusión o, en algunos casos, que genere una alarma falsa.

Los *IDS* que utilizan técnicas de *ML* no están exentos de estos ataques, de hecho, la mayoría de los ataques están pensados para vulnerar los modelos de inteligencia artificial que hay detrás de los *IDS*, pero es más probable que un *IDS* de este tipo sea capaz de resistir un ataque de *AML* que uno basado en un enfoque de reglas y patrones definidos, como afirma *Shroff J. et al.* [7].

Los ataques basados en técnicas de *AML* son muy eficaces contra los *IDS* que utilizan *ML*. Las principales razones de esta eficacia son las siguientes:

- Falta de conocimiento sobre el comportamiento real: Los *IDS* Pueden ser engañados por nuevas formas de ataque que no han sido previstas o para las que no han sido entrenados.
- Escasez de datos de entrenamiento: Muchos algoritmos de aprendizaje automático utilizados en los *IDS* están entrenados con conjuntos de datos limitados, lo que significa que pueden ser engañados por patrones que no han sido previstos [8]. Esto puede llevar a un sobreajuste del modelo, consiguiendo que solo sea capaz de predecir bien los datos del entrenamiento y no sea extrapolable a un entorno real.
- Falta de robustez a las perturbaciones: Muchos algoritmos de *ML* son sensibles a las pequeñas perturbaciones en los datos, lo que significa que pueden ser engañados por pequeñas modificaciones en el tráfico de la red. De hecho, muchos ataques se basan en esta técnica para vulnerar los

IDS [9]–[12], cogen un ataque reconocido y le añaden “ruido” (pequeñas perturbaciones) para que el *IDS* no lo reconozca como ataque.

- Dificultad para detectar ataques sutilmente diseñados: Los ataques adversariales a menudo son muy sutiles y están diseñados para ser indetectables por los *IDS* [4]. Los atacantes tienen la capacidad de aprender del comportamiento del *IDS*, probando diferentes combinaciones de ataques y peticiones. De esta forma, son capaces de replicar el algoritmo de clasificación del *IDS* para poder diseñar ataques que sean capaces de evadirlo.

La defensa frente a este nuevo tipo de ataques es un frente de investigación en alza, ya que no son peligrosos únicamente en *IDS*, sino que cualquier sistema que incorpore procesos de *ML* es vulnerable al *AML*. Algunos escenarios donde este tipo de ataques pueden ser peligrosos, además de los relacionados con la ciberseguridad, son la clasificación y detección de objetos en imágenes o la traducción y procesamiento de textos y audios.

1.3. Descripción y Alcance del Proyecto

En este proyecto se realiza el diseño de un nuevo Sistema de Detección de Intrusos (*IDS*) que utiliza técnicas de *Machine Learning (ML)* para la clasificación de tráfico red llamado **Apolo**. Este nuevo *IDS* cuenta con mecanismos de seguridad inteligentes que le permiten defenderse frente a las amenazas generadas con *Machine Learning Adversario (AML)*.

Apolo está compuesto por varias capas o mecanismos de seguridad que le proporcionan una mayor robustez y protección que los *IDS* tradicionales.

El primer mecanismo de seguridad que incorpora **Apolo** (y el último en actuar) incluye el uso de múltiples algoritmos de clasificación basados en *ML*, a diferencia de los *IDS* tradicionales, cuya respuesta la proporciona un único modelo. El propósito de usar múltiples clasificadores es dificultar a los posibles atacantes realizar réplicas del comportamiento del *IDS*.

La siguiente capa en actuar es la encargada de decidir qué clasificador o conjunto de clasificadores son los encargados de proporcionar la respuesta para cada petición de tráfico red. Para realizar esta toma de decisiones de una forma inteligente (en lugar de un enfoque estático basado en porcentajes o reglas), **Apolo** utiliza una solución basada en los *Multi-Armed Bandits (MAB)* [13]. Esta solución permite asignar a cada clasificador una puntuación que refleja su rendimiento detectando intrusos y que se puede actualizar de una forma dinámica. Una vez se obtienen las puntuaciones, se selecciona con el algoritmo *Thompson sampling* [14] el clasificador o conjunto de clasificadores.

Por último, y siendo la primera de las capas en actuar, se realiza una división de las peticiones en *clústeres* o grupos [15]. De esta forma, se puede agrupar el tráfico red y permite aplicar soluciones diferentes por cada grupo de peticiones similares entre sí.

El diseño de estos mecanismos de seguridad se ha reflejado como la propuesta de un artículo científico titulado "**Apollon: A Robust Defence System against Adversarial Machine Learning Attacks in Intrusion Detection Systems**", el cual esta adjuntado junto con este documento. Además, este artículo forma parte de la tesis doctoral del cotutor y estudiante de PhD Antonio Payá. Debido a que su tesis trata sobre el diseño de estrategias de refuerzo para sistemas basados en Perímetros Definidos por Software, y siendo los *IDS* un componente principal en este tipo de sistemas, este proyecto se complementa perfectamente con dicha tesis.

Para la parte de resultados de Apolo también hemos utilizado los conocimientos adquiridos en el artículo científico titulado “**Harpe: Using Adversarial Machine Learning to dodge Intrusion Detection Systems in Software-Defined Perimeters**”, el cual también forma parte de la tesis del cotutor y estudiante PhD Antonio Payá y está incluido en la sección de anexos. Este artículo trata sobre la creación de un ataque de *AML* enfocado a vulnerar *IDS* dentro de un Perímetro Definido por Software (*SDP*) [16].

Dichos artículos fueron realizados durante mi paso por la Fundación de la Universidad de Oviedo, en la cual realice mis prácticas de empresa junto con mi tutor y profesor titular en la Universidad de Oviedo Dr. Vicente García Díaz.

Cabe mencionar que también se está estudiando la posibilidad de realizar una propuesta de proyecto europeo, en colaboración con el Instituto Nacional de Ciberseguridad (INCIBE)¹. En esta propuesta se pretende continuar con la investigación realizada durante este proyecto, expandiéndola en otros campos.

Para poder demostrar el rendimiento y funcionalidad de **Apolo**, se ha incorporado el *IDS* como parte de una Infraestructura de red segura (*SNI*) (Ilustración 2)² ³, en el componente llamado “*IDS/IPS/SIEM*”. Esta infraestructura define los componentes que debe de tener una red para poder contar con un buen nivel de seguridad, siendo, con algunas variaciones dependiendo del caso de uso, la más utilizada en entornos donde la seguridad es crucial.

No obstante, en este proyecto únicamente se utilizará un subconjunto de los componentes de la *SNI*, siendo estos los componentes involucrados en el funcionamiento del *IDS*.

La arquitectura final sobre la que se probará nuestra solución se encuentra detallada en la Ilustración 1. En esta ilustración se dividen en colores los diferentes componentes del *SNI* que se utilizarán para probar **Apolo**. A continuación, una explicación de cada componente, en función del color:

- En primer lugar y de color verde se encuentra el *firewall* y *proxy* reverso que proporciona un único punto de acceso a la red (implementado con *NGINX*⁴).

¹ <https://www.incibe.es/>

² <https://ocw.uniovi.es/course/view.php?id=109>

³ <https://pciguru.files.wordpress.com/2013/12/200511-theultrasecurenetworkarchitecture.pdf>

⁴ <https://www.NGINX.com/>

- En color rojo se definen los servicios que simularán una funcionalidad básica de una aplicación genérica (dos APIs REST y una WebApp), el único cometido de estos servicios es ejemplificar el funcionamiento en un sistema real, no son el cometido principal del TFG.
- En azul se incluyen los servicios de gestión y procesamiento de los logs de las peticiones, estas peticiones las tratamos homogéneamente, gracias a esto, en el siguiente componente, conseguimos predecir la intención de cada petición, para ello usamos la herramienta del Canadian Institute for Cybersecurity, Cicflowmeter [17]. Esta sección tendrá una Interfaz Gráfica de Usuario que permita al administrador monitorizar todas las peticiones realizadas en la red.
- La información saliente del componente de color azul es la que se le proporciona al último componente, el **IDS (Apolo)**, en naranja. Los resultados de la clasificación de peticiones de tráfico red por parte de **Apolo** se almacenarán en una base de datos de series temporales. Estos resultados podrán visualizarse en una Interfaz Gráfica de Usuario y se contará con un sistema de alarmas para las peticiones clasificadas como intrusiones.

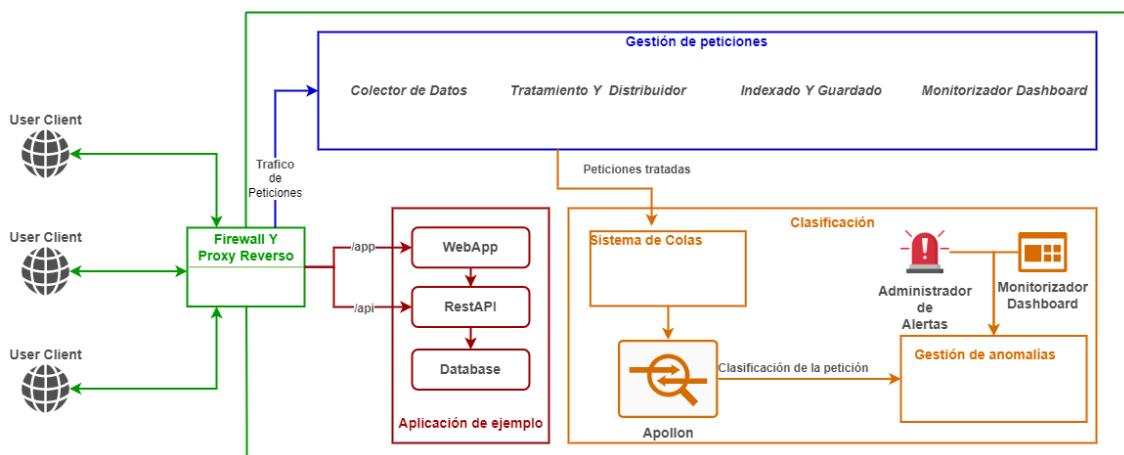


Ilustración 1. Representación abstracta de la arquitectura del sistema.

Apolo: Sistema inteligente de monitorización y detección de ataques basados en inteligencia artificial

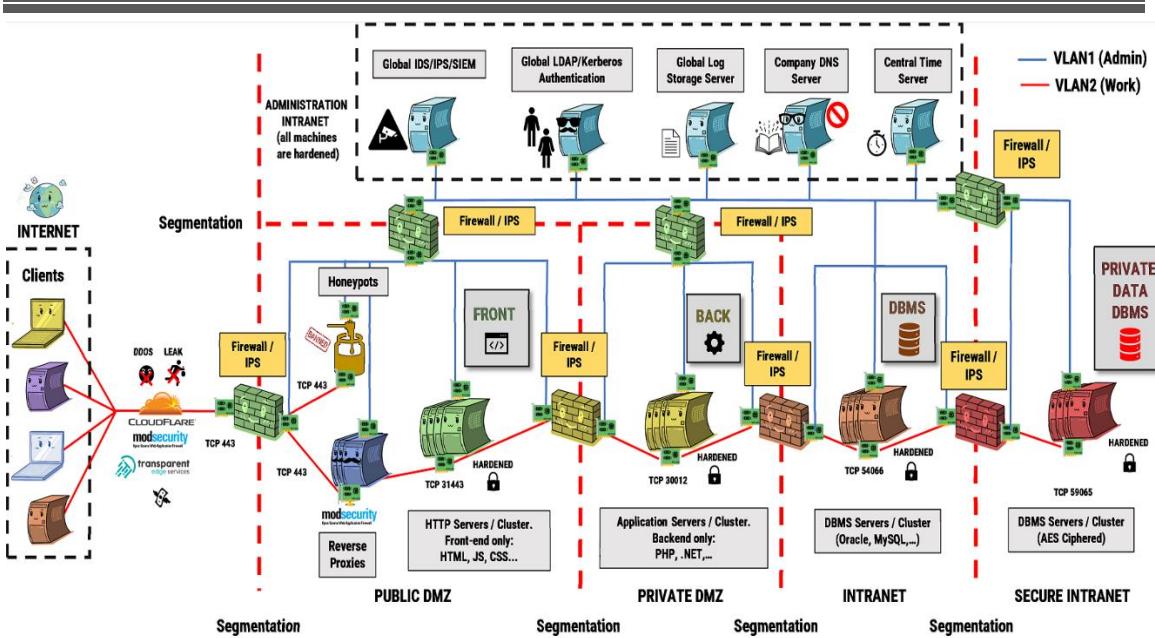


Ilustración 2. Infraestructura de red segura (SNI)

1.4. Objetivos del Proyecto

El objetivo principal de este proyecto es el diseño de un *IDS* capaz de protegerse frente a los ataques generados con técnicas de *AML*. Para conseguir cumplir con este objetivo principal, se definen los siguientes subobjetivos:

- Desarrollo de un *IDS* capaz de clasificar peticiones de tráfico red en benignas o malignas.
 - El *IDS* debe de poder clasificar correctamente el tráfico y los ataques normales.
 - El *IDS* debe de poder clasificar correctamente los ataques generados con *AML*.
- Integración del *IDS* en una infraestructura de red segura (*SNI*).
 - El *IDS* debe de poder incorporar únicamente los componentes de la *SNI* que intervienen en el funcionamiento del *IDS*.
 - El *IDS* debe de poder añadir a la *SNI* una aplicación web genérica con la que poder probar el *IDS*.
- Monitorización y visualización de los resultados generados por el *IDS*.
 - El *IDS* debe de poder visualizar de los resultados en tiempo real.
 - El *IDS* debe de tener un sistema de alertas para las peticiones clasificadas como tráfico maligno.

1.5. Resumen de Todos los Aspectos

En este apartado se resumirán brevemente los contenidos de cada sección que compone este documento y que se desarrollarán posteriormente:

- **Capítulo 1. Definición del Proyecto:** introducción al proyecto, motivación, descripción y alcance del mismo.
- **Capítulo 2. Estado Actual de los Conocimientos Científico-Técnicos:** En este capítulo se detalla el estado actual de los conocimientos científico técnicos relacionados con la temática del proyecto y proyectos relacionados.
- **Capítulo 3. Apolo:** Definición y explicación del funcionamiento del modelo de Apolo.
- **Capítulo 4. Planificación del Proyecto y Presupuesto Iniciales:** Se elabora una planificación inicial del proyecto y los presupuestos para la empresa y para el cliente.
- **Capítulo 5. Análisis:** Análisis del sistema a través de los requisitos, casos de uso e identificación de los componentes de la arquitectura.
- **Capítulo 6. Definición del Sistema:** Definición del sistema mediante descripciones de cada tecnología, herramienta, lenguaje, estándar y norma utilizado durante el desarrollo de los componentes y por ende del sistema.
- **Capítulo 7. Evaluación de Alternativas:** Alternativas que se valoraron para la realización del sistema, entre las alternativas dadas se encuentran las alternativas a Apolo, los componentes y el resto de las tecnologías utilizadas en el proyecto.
- **Capítulo 8. Diseño del Sistema:** Diagramas de la arquitectura, clases y diseño de las bases de datos.
- **Capítulo 9. Evaluación del Sistema:** Metodología, hardware y validación utilizada para la experimentación de Apolo.
- **Capítulo 10 Manuales del sistema:** Se ofrecen los manuales de instalación, ejecución, usuario y programador del sistema.
- **Capítulo 11. Conclusiones y Futuro Trabajo:** Se comentan las conclusiones del proyecto con respecto a los objetivos iniciales y el trabajo futuro que puede salir de este proyecto.
- **Capítulo 12. Planificación del Proyecto y Presupuesto finales:** Se ofrece la planificación y el presupuesto finales del proyecto a los que se ha llegado tras la elaboración de este.

- **Capítulo 13. Referencias bibliográficas:** Se enumeran las referencias bibliográficas utilizadas a lo largo del documento.
- **Apéndices:** En los apéndices se encuentra el análisis y gestión de los riesgos del proyecto, el glosario de términos utilizados y una descripción de la estructura y contenido del archivo adjunto.

Capítulo 2. Estado Actual de los Conocimientos Científico-Técnicos

En este capítulo se explicará de una forma más detallada todos los conocimientos científico-técnicos relacionados con el proyecto.

2.1. Infraestructura de red segura (SNI)

Una infraestructura de red segura (SNI)⁵⁶ [18], [19] es un conjunto de medidas de seguridad implementadas en una red de computadoras para protegerla contra amenazas internas y externas. La implementación de una infraestructura de red segura es esencial para proteger la información y los recursos críticos de una organización.

Para crear una infraestructura de red segura se utilizan varias medidas de seguridad. En primer lugar, es importante asegurar la ubicación física de los dispositivos de red, como routers, switches, servidores y otros dispositivos de red. Esto se puede hacer mediante la restricción del acceso a la habitación o edificio donde se encuentran estos dispositivos.

Otra medida de seguridad importante es el uso de *firewalls*. Los *firewalls* son sistemas de seguridad que se utilizan para controlar el tráfico de red entrante y saliente. Se pueden configurar para bloquear tráfico malicioso y permitir sólo el tráfico seguro., se pueden implementar en hardware o software.

Las redes privadas virtuales (VPN) [20] se utilizan para proporcionar una conexión segura y encriptada entre dos puntos de la red. Las VPN se utilizan comúnmente para permitir a los empleados acceder a la red de la empresa desde una ubicación remota.

Es importante implementar medidas de autenticación y autorización para controlar el acceso a los recursos de la red. Esto se puede hacer mediante el uso de contraseñas seguras, certificados digitales, tokens de seguridad y otros métodos de autenticación.

También es esencial monitorear el tráfico de la red y registrar las actividades de los usuarios para detectar posibles amenazas y responder a ellas de manera efectiva. Los registros de actividad también pueden ser útiles para fines de auditoría y cumplimiento.

Mantener actualizados todos los dispositivos de la red y aplicar parches de seguridad regularmente también es importante. Las vulnerabilidades en el

⁵ <https://ocw.uniovi.es/course/view.php?id=109>

⁶ <https://pciguru.files.wordpress.com/2013/12/200511-theultrasecurenetworkarchitecture.pdf>

software y el *firmware* pueden ser explotadas por los atacantes para comprometer la seguridad de la red.

Por último, las políticas y procedimientos de seguridad establecen las prácticas de seguridad de la red y las responsabilidades de los usuarios de la red. Estas políticas deben ser comunicadas y cumplidas por todos los empleados para mantener una infraestructura de red segura.

2.2. Sistemas de detección de intrusos

Un sistema de detección de intrusos (*IDS*) [2] es una tecnología de seguridad que detecta y alerta sobre accesos no autorizados o actividades maliciosas en una red. Los *IDS* pueden funcionar en tiempo real o analizar registros y otros datos almacenados. Utilizan diversas técnicas para detectar intrusiones y son una herramienta importante en la protección contra ciberamenazas.

Los *IDS* pueden funcionar en tiempo real, analizando el tráfico de red en el momento en que se produce, o pueden configurarse para analizar registros u otros datos almacenados. Utilizan diversas técnicas para detectar intrusiones, como el análisis de los patrones de tráfico de la red, el examen de los registros del sistema y la búsqueda de indicadores de actividades maliciosas conocidas. Los *IDS* desempeñan un papel fundamental en el mantenimiento de la seguridad y la integridad de una red, y suelen utilizarse junto con otras medidas de seguridad, como *firewalls* y software antivirus [1].

Las técnicas de aprendizaje automático (*ML*) se han utilizado cada vez más en los *IDS* debido a su capacidad para analizar grandes cantidades de datos e identificar patrones complejos [3], [8]. Los *IDS* basados en *ML* pueden aprender de datos anteriores y adaptarse a nuevos patrones de comportamiento, lo que los hace más eficaces a la hora de detectar amenazas nuevas y en evolución.

Los sistemas de detección de intrusos (*IDS*) pueden clasificarse en dos categorías principales [2]: basados en red (*NIDS*) y basados en host (*HIDS*). Los *NIDS* monitorean el tráfico de red para detectar actividades sospechosas o maliciosas, mientras que los *HIDS* monitorean los eventos en un solo host o sistema.

El uso del *Machine Learning (ML)* en los *IDS* tiene varias ventajas [3], [8]. Por ejemplo, el *ML* puede mejorar la precisión y eficacia de los *IDS* al reducir el número de falsos positivos y falsos negativos. Además, los *IDS* basados en *ML* pueden adaptarse a nuevos patrones de comportamiento y detectar amenazas nuevas y en evolución.

Sin embargo, también hay desventajas en el uso del *ML* en los *IDS* [3], [8]. Por ejemplo, el entrenamiento de modelos de *ML* puede requerir grandes cantidades de datos y tiempo. Además, los modelos de *ML* pueden ser vulnerables a ataques adversarios que buscan engañar al sistema.

2.2.1. Principales Datasets Utilizados en Sistemas de Detección de Intrusos

Los conjuntos de datos de los *sistemas de detección de intrusos* son un recurso importante para evaluar y comparar el rendimiento de los *IDS*. Estos conjuntos de datos suelen ser colecciones de ejemplos etiquetados de tráfico de red normal y anormal, que se utilizan para entrenar y probar la precisión y eficacia de los *IDS*. Hay varios conjuntos de datos disponibles, cada uno con sus propias características y puntos fuertes. En esta parte, se repasarán algunos de los conjuntos de datos más populares y se discutirán sus principales características y usos.

2.2.1.1. CIC-IDS2017

El conjunto de datos *CIC-IDS2017* [21] se creó en un entorno de red empresarial simulado y captura datos de tráfico de red durante un periodo de cinco días. Los datos se recopilaron para imitar el comportamiento de 25 usuarios y contienen aproximadamente 80 atributos significativos [22]. Actualmente es uno de los conjuntos de datos *IDS* más utilizados en la literatura moderna y contiene una alta proporción de ejemplos benignos frente a maliciosos (83% frente a 17%). Puede utilizarse individualmente o junto con otros conjuntos de datos, ya que refleja con exactitud la distribución normal de tráfico benigno y malicioso en una red [7]. Puede verse un resumen de sus características en la Tabla 1.

Dataset	CIC-IDS-2017
Tipo de dataset	Multiclas
Año de creación	2017
Duración de la captura (Días)	5
Infraestructura de Ataque	4 PCs, 1 router, 1 switch
Infraestructura de la víctima	3 server, 1 firewall, 2 switches, 10 PCs
Características	80

Tabla 1. Resumen CIC-IDS 2017⁷.

2.2.1.2. CSE-CIC-IDS2018

El conjunto de datos *CSE-CIC-IDS2018* [23], creado por el Instituto Canadiense de Ciberseguridad (C/C), se recopiló en un entorno de red empresarial simulado utilizando recursos de AWS en 2018. Contiene datos sobre siete tipos de ataques

⁷ <https://www.unb.ca/cic/datasets/ids-2017.html>

diferentes y tiene aproximadamente 79 características significativas. El conjunto de datos incluye datos de más de 450 dispositivos, incluidos servidores, ordenadores y otros dispositivos, y destaca por su gran tamaño y alto nivel de realismo [24]. Es similar al conjunto de datos CIC-IDS2017 en el sentido de que incluye análisis de paquetes de tráfico de red con flujo bidireccional, pero es más grande y completo. Como resultado, es ampliamente utilizado en la literatura para evaluar y comparar IDSs [24]. Puede verse un resumen de sus características en la Tabla 2.

Ataque	Herramientas	Duración (Días)	Atacante	Víctima
Ataque de fuerza bruta	FTP – Patator SSH – Patator	1	Kali Linux	Ubuntu 16.4 (Web Server)
Ataque DoS	Hulk, GoldenEye, Slowloris, Slowhttptest	1	Kali Linux	Ubuntu 16.4 (Apache)
Ataque DDoS	Heartbleech	1	Kali Linux	Ubuntu 12.04 (Open SSL)
Ataque Web	Damn Vulnerable Web App (DVWA) Framework interno de selenium (XSS y Brute-force)	2	Kali Linux	Ubuntu 16.4 (Web Server)
Ataque Infiltración de	Primer nivel: Descarga de Dropbox en una máquina Windows Segundo nivel: Nmap y portscan	2	Kali Linux	Windows Vista y Macintosh
Ataque Botnet	Ares (desarrollado por Python): shell remoto, archive de subida y bajada, captura, screenshots y key logging	1	Kali Linux	Windows Vista, 7, 8.1, 10 (32-bit) y 10 (64-bit)
DDoS+PortScan	Low Orbit Ion Canon (LOIC) para UDP, TCP, o HTTP requests	2	Kali Linux	Windows Vista, 7, 8.1, 10 (32-bit) y 10 (64-bit)

Tabla 2. Resumen CIC-IDS 2018 ⁸.

⁸ <https://www.unb.ca/cic/datasets/ids-2018.html>

2.2.1.3. CIC-DDoS2019

El conjunto de datos *CIC-DDoS2019* [25] se creó para abordar la falta de representación de todos los subtipos de ataques *DDoS* en los conjuntos de datos existentes. Aunque el conjunto de datos incluye tráfico de red simulado, se ha intentado crear datos benignos realistas. Incluye 13 tipos de ataques *DDoS* y tiene más de 80 características significativas. Sin embargo, el conjunto de datos está muy desequilibrado, con 50.006.249 registros de ataques *DDoS* y sólo 56.863 registros de tráfico benigno, lo que dificulta su uso para entrenar un modelo en ambos tipos de datos. Por lo tanto, se recomienda utilizar este conjunto de datos junto con otros, como *CIC-IDS2017* o *CSE-CIC-IDS2018*, para entrenar un modelo más robusto [7]. Puede verse un resumen de sus características en la Tabla 3 y en la Tabla 4.

Máquina	SO	IPs
Server	<i>Ubuntu 16.04 (Web Server)</i>	192.168.50.1 (<i>primer día</i>) 192.168.50.4 (<i>segundo día</i>)
Firewall	Fortinet	205.174.165.81
PCs (primer día)	Win 7 Win Vista Win 8.1 Win 10	192.168.50.8 192.168.50.5 192.168.50.6 192.168.50.7
PCs (segundo día)	Win 7 Win Vista Win 8.1 Win 10	192.168.50.9 192.168.50.6 192.168.50.7 192.168.50.8

Tabla 3. Tipo de máquinas involucradas en la obtención del dataset y sus respectivos sistemas operativos e IPs ⁹.

⁹ <https://www.unb.ca/cic/datasets/ddos-2019.html>

Días	Ataques	Horas de los ataques
Primer Día	PortMap	9:43 - 9:51
	NetBIOS	10:00 - 10:09
	LDAP	10:21 - 10:30
	MSSQL	10:33 - 10:42
	UDP	10:53 - 11:03
	UDP-Lag	11:14 - 11:24
	SYN	11:28 - 17:35
Segundo Día	NTP	10:35 - 10:45
	DNS	10:52 - 11:05
	LDAP	11:22 - 11:32
	MSSQL	11:36 - 11:45
	NetBIOS	11:50 - 12:00
	SNMP	12:12 - 12:23
	SSDP	12:27 - 12:37
	UDP	12:45 - 13:09
	UDP-Lag	13:11 - 13:15
	WebDDoS	13:18 - 13:29
	SYN	13:29 - 13:34
	TFTP	13:35 - 17:15

Tabla 4. Nombres de los ataques ejecutados cada día, con sus respectivas horas ¹⁰.

2.2.1.4. Datasets Descartados

Existen muchos otros conjuntos de datos de *IDS*, como *Darpa 1998/99* [26], *KDD 99* [27] o *NSL-KDD* [27], que no se han utilizado en este proyecto. Los conjuntos de datos *Darpa 1998/99* y *KDD 99* ya no se utilizan habitualmente para evaluar y comparar *IDS* debido a su carácter obsoleto. Estos conjuntos de datos se crearon a finales de los 90 y principios de los 2000, y no reflejan con exactitud el panorama actual de amenazas y comportamientos en la red. En particular, el conjunto de datos *KDD 99* ha sido criticado por su alta tasa de falsos positivos y su falta de realismo, lo que lo hace menos útil para probar el rendimiento de los *IDS* modernos [28], [29].

Por otro lado, conjuntos de datos como el *CIC-IDS2017* son más utilizados debido a su mayor tamaño, mayor realismo y distribución más equilibrada de ejemplos benignos y maliciosos.

¹⁰ <https://www.unb.ca/cic/datasets/ddos-2019.html>

2.2.2. Clasificadores Utilizados en los IDS

Con el aumento en número y sofisticación de las amenazas de malware, se vuelve cada vez más crucial contar con sistemas *IDS* eficaces para proteger las redes. Para lograr esta eficacia, los investigadores realizan estudios y revisiones bibliográficas para identificar y solucionar posibles vulnerabilidades en los sistemas *IDS*. Estos estudios son continuos y esenciales para el desarrollo constante de sistemas *IDS* capaces de mantenerse al día con la evolución del panorama de amenazas.

En las tablas: Tabla 5, Tabla 6 y Tabla 7; se presentan los clasificadores *IDS* junto con sus puntuaciones de precisión, *F1* y *AUC* para los conjuntos de datos *CIC-IDS2017*, *CSECIC-IDS2018* y *CIC-DDoS2019*, respectivamente. Los clasificadores más utilizados y con mejores resultados en estos conjuntos de datos, así como en los conjuntos de datos más populares como *KDD99* y *NSL-KDD*, son *Random Forest (RF)* [30], *Decision Trees (DT)* [31] y *Support Vector Machine (SVM)* [32]. En particular, *RF* se ha popularizado gracias a su capacidad para manejar grandes conjuntos de datos y su rendimiento sólido incluso con ruido y valores perdidos en los datos. Los *DT* son populares debido a su interpretabilidad y simplicidad, lo que permite una visualización clara del proceso de toma de decisiones. *SVM* también es una opción favorita por su capacidad para manejar datos de alta dimensión y su versatilidad en la clasificación de diferentes tipos de tareas. En general, estos clasificadores han demostrado un excelente rendimiento y son comúnmente utilizados en el campo de los *IDS*.

Clasificador	Media	F1	AUC	Referencias
IGAN-IDS	99.79	99.79	99.98	[33]
ANN	99.28	99.17	99.28	[34]
Fuzziness-based NN	99.61	99.57	99.83	[33], [35]
CNN	99.48	99.44	99.94	[33], [34]
RTIDS	99.35	99.17	99.83	[35]
MLP	99.46	99.46	99.41	[24], [36]
ROS + MLP	99.55	99.55	99.88	[33]
Random Forest	99.79	99.78	99.98	[24], [33], [34], [37], [38]
Decision Trees	99.62	99.57	99.56	[24], [33], [34]
SMOTE+SVM	97.00	97.04	98.97	[33]
SVM	96.97	96.99	98.98	[24], [33]–[35], [38]
ROS + SVM	96.98	97.04	98.96	[33]
RUS + SVM	96.45	96.55	98.96	[33]
Naive Bayes	93.90	93.53	97.49	[33], [34], [38]

Tabla 5. Resumen de los clasificadores de *IDS* para el dataset *CIC-IDS2017*.

Clasificador	Media	F1	AUC	Referencias
MLP	62.00	98.00	100.00	[24], [36]

Random Forest	92.00	94.00	100.00	[24], [33], [34], [37], [38]
Decision Tree	88.00	91.00	100.00	[24], [33], [34]
SVM	61.00	66.00	100.00	[24], [33]–[35], [38]

Tabla 6. Resumen de los clasificadores de IDS para el dataset CSE-CIC-IDS2018.

Clasificador	Media	F1	AUC	Referencias
Fuzziness-based NN	95.55	95.50	95.63	[33], [35]
RTIDS	98.58	98.48	98.66	[35]
SVM	94.02	94.88	94.24	[24], [33]–[35], [38]

Tabla 7. Resumen de los clasificadores de IDS para el dataset CIC-DDoS2019.

2.3. Adversarial Machine Learning (AML)

Adversarial Machine Learning se refiere al uso de técnicas de Machine Learning para engañar o confundir a los modelos [5], [6].

2.3.1. Ataques en Función del Conocimiento del Modelo

Como vemos en el artículo de Qiu S. et al. [5], en función del nivel de conocimiento del atacante sobre el modelo, los ataques adversarios pueden clasificarse en tres tipos, caja blanca, caja gris y caja negra. En la Ilustración 3 se pueden ver el funcionamiento de los ataques de caja negra/gris y caja blanca.

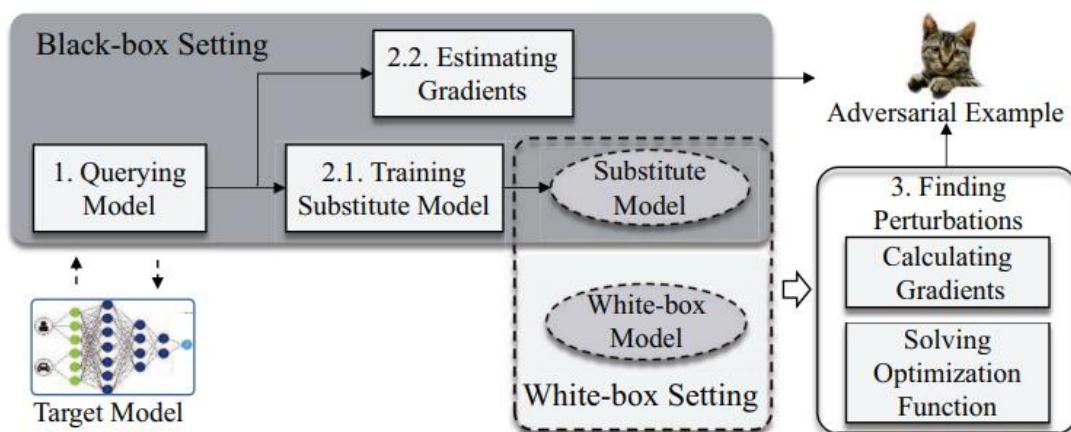


Ilustración 3. Ataque de caja negra y de caja blanca, extraído del artículo de He Y. et al. [6]

2.3.1.1.1. Ataques de Caja Blanca

Los ataques de caja blanca son considerados los más poderosos dentro de los ataques AML, ya que permiten al atacante conocer todos los detalles del clasificador IDS y los datos de entrenamiento. Con esta información, el atacante puede crear ataques altamente específicos y complejos que pueden evadir las defensas del sistema y lograr sus objetivos.

Sin embargo, los ataques de caja blanca son también los menos realistas, ya que requieren que el atacante tenga un conocimiento detallado del sistema y sus vulnerabilidades, lo cual no es siempre posible. En la mayoría de las situaciones, el atacante sólo tendrá un conocimiento limitado del sistema, lo que hace que los ataques de caja blanca sean impracticables. Por lo tanto, los ataques de caja

blanca son poco comunes en la vida real y generalmente se realizan en operaciones muy específicas y bien planificadas.

2.3.1.1.2. Ejemplos de Ataques de Caja Blanca

Algunos ejemplos de ataques de caja blanca comúnmente utilizados en *IDS* incluyen el Método Rápido de Señal Gradiente (*FGSM*) [39], *Deep-Fool* [40], el ataque *Carlini & Wagner (C&W)* [41], el Ataque de Mapa de Saliencia Basado en Jacobianos (*JSMA*) [42], el Método Iterativo Básico (*BIM*) [43], y el Descenso Gradiente Proyectado (*PGD*) [44].

2.3.1.2. Ataques de caja negra/gris

A diferencia de los ataques de caja blanca, los ataques de caja negra/gris son más realistas ya que no requieren que el atacante tenga conocimientos detallados sobre el modelo objetivo. Sin embargo, estos ataques suelen ser menos efectivos que los de caja blanca, debido a que el conocimiento limitado del atacante sobre el sistema restringe su capacidad para crear ataques muy específicos y sofisticados. En su lugar, tienen que depender de técnicas más generales que pueden resultar menos efectivas para evadir las defensas del sistema.

2.3.1.2.1. Redes Generativas Adversariales (GAN)

Las Redes Generativas Adversariales (*GANs*) [45] son ampliamente utilizadas en ataques de caja negra y caja gris para generar ejemplos adversarios. Las *GANs* son un tipo de modelo de aprendizaje automático que consta de dos redes neuronales: una red generadora y una red discriminadora. La red generadora se entrena para producir datos sintéticos que se parezcan a los datos reales, mientras que la red discriminadora se entrena para distinguir entre datos reales y sintéticos.

En el contexto de los ataques de caja negra y caja gris, la red generadora puede generar ejemplos adversarios específicamente diseñados para evadir las defensas del sistema objetivo. El atacante puede tener acceso a las puntuaciones del modelo del sistema (caja gris) o sólo a una salida binaria que indique si la entrada fue aceptada o rechazada (caja negra). Esta información puede utilizarse para guiar el entrenamiento de la red generadora, mejorando su capacidad para producir ejemplos adversarios más eficaces.

2.3.1.2.2. Ejemplos de Ataques de Caja Negra/Gris

Ataques de caja gris/negra recientes a *IDS* han utilizado diversas técnicas, como *attackGAN* [46], *DIGFuPAS* [47], *IDSGAN* [9], *VulnerGAN* [48], el ataque *ZOO* [48], el ataque *Boundary* [49] y el ataque *HotSkipJump (HSJA)* [50]. El ataque *ZOO* es una técnica basada en puntuaciones que estima gradientes para generar tráfico adverso en entornos de caja gris/negra. Por otro lado, el ataque *Boundary* y *HSJA* son ataques basados en decisiones que utilizan únicamente retroalimentación binaria para crear entradas adversarias.

Los ataques *IDSGAN*, *attackGAN* y *DIGFuPAS* son técnicas de caja gris/negra que emplean *Wasserstein-GAN* para generar tráfico adversario. Este tipo de *GAN* se entrena con una función objetivo diferente llamada distancia *Wasserstein*, la cual mide la distancia entre dos distribuciones de probabilidad y tiene propiedades como la suavidad y la continuidad. Algunas implementaciones recientes de *WGAN* utilizan la regularización del gradiente penalizado [51] para mejorar la convergencia del entrenamiento.

No obstante, los ataques con *WGANS*, como *IDSGAN*, generan ejemplos adversarios mediante el entrenamiento de un generador con ruido aleatorio como entrada. Este enfoque no resulta efectivo debido a que la red generadora altera características clave de un ataque en un intento de evadir la detección [52]. Esta no es una táctica viable, ya que estas características son fundamentales para que el ataque sea exitoso.

2.3.2. Ataques en Función de su Taxonomía

Tal y como argumenta He Y. et al. [6], dependiendo de la taxonomía del ataque, se pueden clasificar en ataques de evasión, inferencia, envenenamiento y de extracción.

Se puede ver una exemplificación de estos 4 tipos de ataque en la Ilustración 4 y una más específica de envenenamiento, inferencia y extracción en la Ilustración 5.

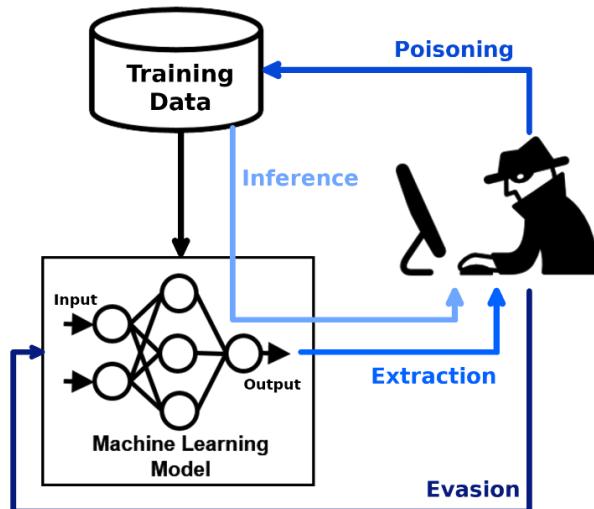


Ilustración 4. Adversarial Machine Learning attacks por Nicolae M. et al. [53]

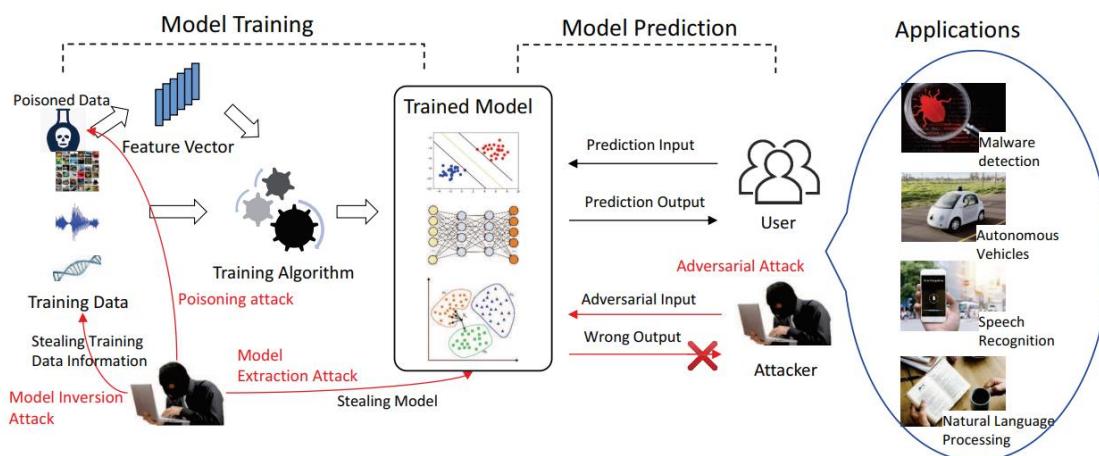


Ilustración 5. Adversarial Machine Learning attacks por He Y. et al. [6]

2.3.2.1. Ataques de Evasión

Los ataques de evasión [5], [6], [53] tienen como objetivo generar entradas diseñadas específicamente para eludir la detección del modelo. Estas entradas, conocidas como ejemplos adversarios, son versiones modificadas de entradas normales que han sido ligeramente perturbadas de una manera que no es fácilmente perceptible para los humanos, pero que puede hacer que el modelo las clasifique erróneamente. Los ataques de evasión pueden realizarse mediante diversas técnicas, como la optimización basada en gradientes o el ML. En la

Ilustración 6 y en la Ilustración 7; se pueden observar varios ejemplos ataques de evasión.

La Ilustración 6 es una demostración de un ataque de evasión, esta muestra como añadiendo pequeñas perturbaciones a una en la entrada de una red neuronal puede conseguir que se clasifique incorrectamente.

En este ejemplo en particular, empezaron con una imagen de un “panda”. En un primer instante la red neuronal lo clasifica correctamente como un “panda”. Al generar una capa de “ruido” diseñada específicamente para confundir a la red neuronal, esta pasa a clasificar erróneamente la imagen como “gibbon”.

Este ruido es calculado cuidadosamente para cambiar la clasificación de la imagen sin cambiar el aspecto de la imagen de manera perceptible para los humanos. El resultado sigue pareciendo un “panda” para un humano, pero la red neuronal ahora lo clasifica erróneamente.

En la Ilustración 7 hay otro ejemplo parecido al anteriormente descrito. En este caso, el sistema está diseñado para reconocer y clasificar números escritos a mano. La red neuronal puede comenzar reconociendo correctamente una imagen de un número “1”. Sin embargo, si un atacante sabe cómo está configurado el sistema, puede introducir una pequeña perturbación o cambiar unos pocos píxeles de la imagen. A simple vista, la imagen aún parecería un “1” para un observador humano, pero la red neuronal puede ser engañada para que la clasifique erróneamente como un “4”.

Esto muestra cómo las redes neuronales pueden ser extremadamente sensibles a pequeñas perturbaciones en sus entradas. Este tipo de ataques pueden ser un problema importante en ciertos contextos, donde un error de clasificación podría tener consecuencias graves.

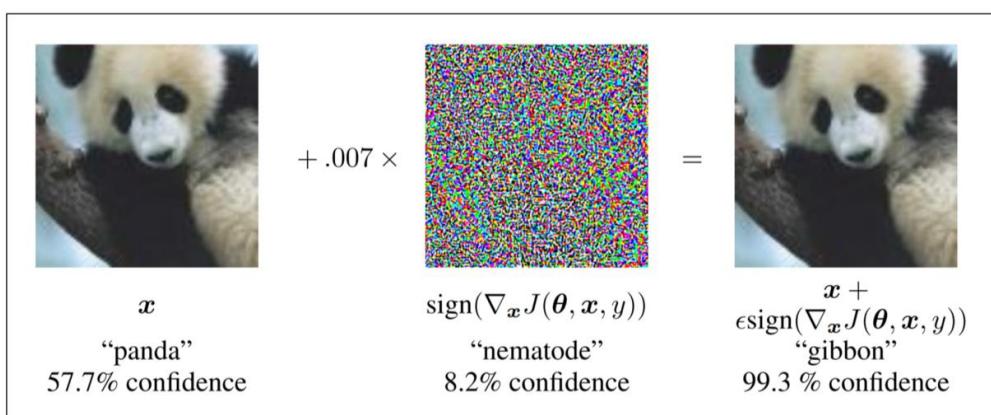


Ilustración 6. Ataque de evasión, extraído del artículo de Qiu S. et al. [5]

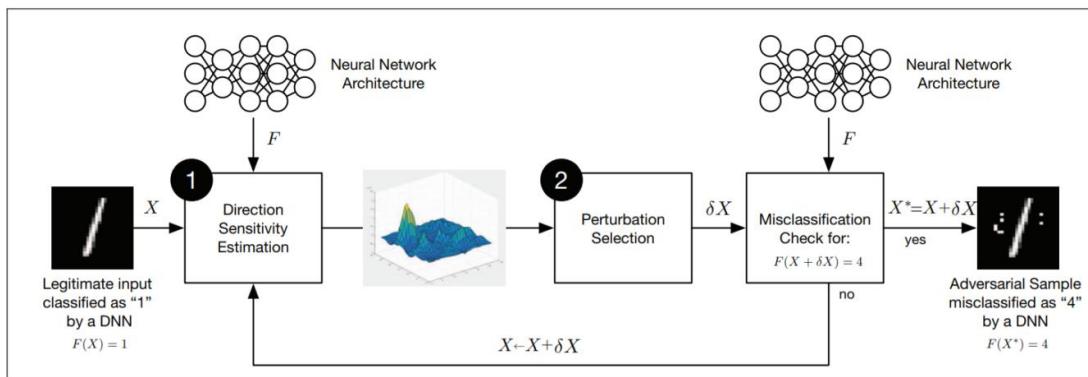


Ilustración 7. Ataque de evasión, extraído del artículo de Qiu S. et al. [5]

2.3.2.2. Ataques de Inferencia

Los ataques de inferencia [5], [6], [53] consisten en invertir el sentido de la información en un modelo de *Machine Learning*. El objetivo es obtener información del modelo, que no se pretendía compartir. Hay tres subtipos principales de ataque de inferencia, el Ataque de Inferencia de Membresía (*MIA*) [54], [55] que consiste en descubrir si un dato ha sido utilizado o no en el entrenamiento, el Ataque de Inferencia de Propiedades (*PIA*) [56], [57] que intenta inferir parámetros y estadísticas del modelo, y reconstrucción de datos, que intenta replicar las muestras de entrenamiento.

En la Ilustración 8 se pueden observar los tres enfoques diferentes que hay en los ataques por inferencia y que comentamos anteriormente: *MIA*, *PIA* y reconstrucción de datos.

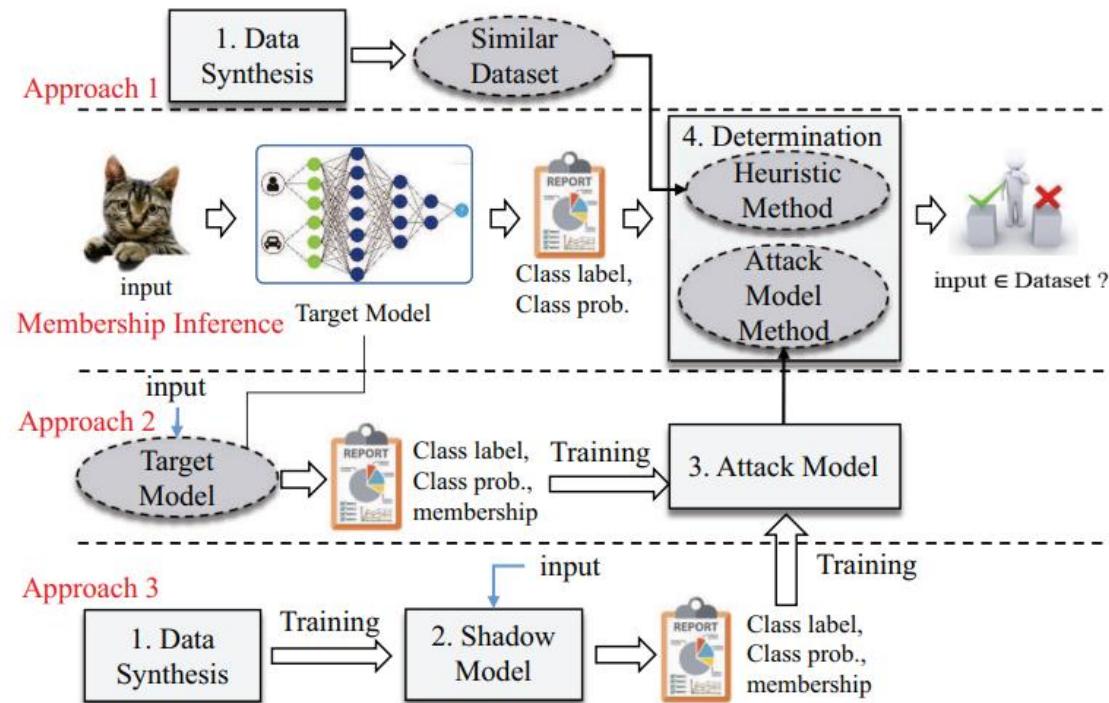


Ilustración 8. Ataque de inversión, extraído del artículo de He Y. et al. [6]

2.3.2.3. Ataques de Envenenamiento

El ataque de envenenamiento [5], [6], [53] implica la manipulación malintencionada de los datos de entrenamiento para afectar negativamente el rendimiento del modelo. El objetivo es hacer que el modelo aprenda relaciones incorrectas entre las entradas y las salidas, lo que puede llevar a resultados incorrectos o incluso peligrosos. Los datos maliciosos pueden ser diseñados para parecer normales a simple vista, pero contienen información oculta que puede afectar negativamente el rendimiento del modelo. En la Ilustración 9 se puede ver un ejemplo de ataque de envenenamiento.

En la Ilustración 9 se puede observar que mediante el uso de un dataset envenenado puede lograrse que se inviertan las etiquetas de los objetos del dataset original. Por ejemplo, si en el dataset original hay imágenes de coches que están etiquetadas como coches, en el dataset envenenado puede haber imágenes de coches etiquetadas como elefantes, estas incongruencias hacen que el modelo se confunda y clasifique mal.

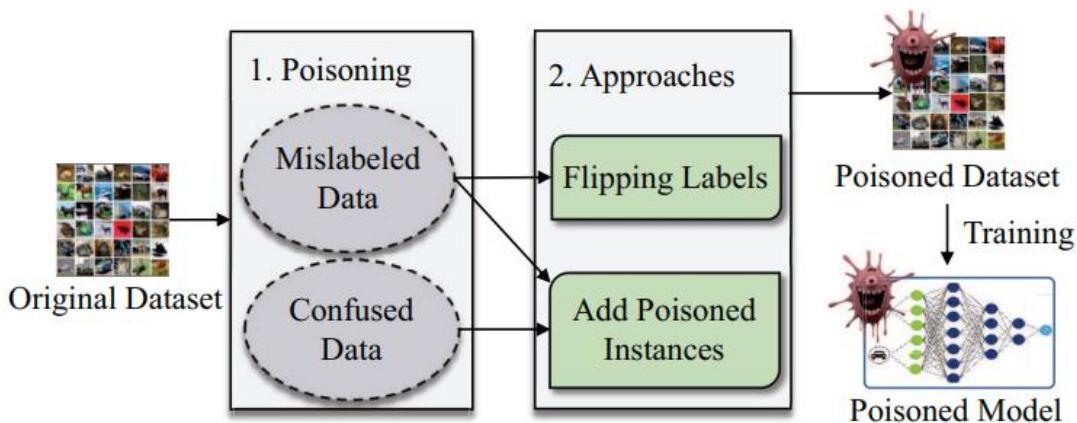


Ilustración 9. Ataque de envenenamiento, extraído del artículo de He Y. et al. [6]

2.3.2.4. Ataque de Extracción

El ataque de extracción [5], [6], [53] de modelo implica la extracción no autorizada del modelo a partir de consultas al sistema. El objetivo es obtener información sobre el modelo subyacente para replicarlo o utilizarlo en otros ataques. Esto se logra mediante la realización de consultas al sistema y el análisis de las respuestas para inferir información sobre el modelo. Las consultas pueden ser diseñadas para obtener información específica sobre el modelo, como su arquitectura o sus parámetros.

En la Ilustración 10 se puede ver un ejemplo de ataque de extracción de un modelo. Mediante el uso de peticiones va replicando el modelo lentamente, haciendo uso de los diferentes enfoques: *Equation Solving (ES)*, *Training Metamodel (TM)* y *Training Substitute Model (TSM)*; los cuales se especializan en extraer la información que compone un modelo de *ML*.

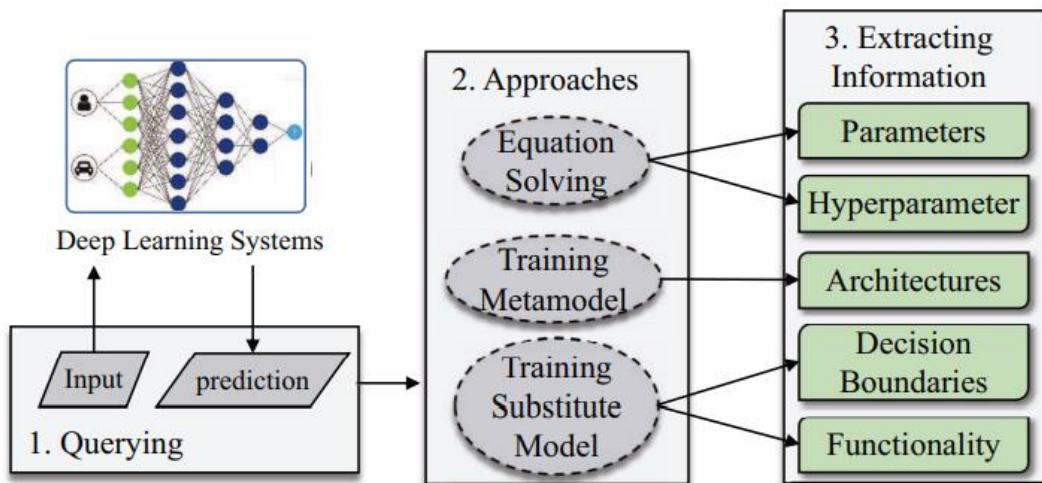


Ilustración 10. Ataque de extracción, extraído del artículo de He Y. et al. [6]

2.4. Multi-Armed Bandits (MAB)

Multi-Armed Bandits (MAB) es un problema clásico en teoría de probabilidad y Aprendizaje Automático, donde un agente debe asignar un conjunto limitado de recursos entre opciones en competencia que tienen recompensas inciertas [58]. El agente enfrenta un equilibrio entre explotar las opciones que tienen las recompensas esperadas más altas según la información actual y explorar nuevas opciones que podrían brindar recompensas más altas en el futuro.

El problema de *MAB* tiene muchas aplicaciones prácticas en diversos dominios, como ensayos clínicos, *adaptive routing*, diseño de carteras financieras y publicidad en línea. Se han propuesto varios algoritmos para resolver el problema de *MAB*, como la inicialización optimista [59], el límite superior de confianza (*UCB*) [60] y el muestreo de *Thompson* [61]. Estos algoritmos difieren en cómo equilibran la exploración y la explotación, y cómo estiman las recompensas esperadas de cada opción.

Para entender mejor el concepto de elección mediante recompensas inciertas daremos un ejemplo mediante una metáfora del mundo real. Imagina que estás en un casino y tienes frente a ti varias máquinas tragamonedas (cada una es un “brazo” del *bandit*). Cada máquina proporciona una recompensa según una distribución de probabilidad desconocida. El objetivo es desarrollar una estrategia para jugar a las máquinas de manera que maximices tus recompensas totales en una serie de juegos.

Para ello contamos con las técnicas de exploración y explotación:

- Exploración: implica probar cada “brazo” (máquina tragamonedas) para obtener más información sobre las recompensas potenciales que cada uno puede dar.
- Explotación: una vez que tienes cierta información sobre qué brazos podrían ser más prometedores, explotarlos significa utilizar esos brazos para obtener la mayor recompensa.

El algoritmo de Muestreo de *Thompson* (*Thompson Sampling*) es una estrategia para el problema del *MAB* que equilibra la exploración y la explotación.

La idea básica del muestreo de *Thompson* es mantener una distribución de probabilidad para la recompensa de cada “brazo” basada en las recompensas observadas de ese brazo hasta ahora.

La forma en que estimamos las recompensas esperadas es manteniendo una distribución de probabilidad para cada “brazo”. Usualmente se inicia con una distribución previa (*prior*) que puede ser uniforme (cada recompensa es igualmente probable) o basada en algún conocimiento previo.

Cada vez que jugamos un “brazo”, actualizamos la distribución de probabilidad de la recompensa para ese “brazo” basada en la recompensa observada. Esta actualización se hace a través de la inferencia bayesiana.

A lo largo del tiempo, a medida que jugamos más, nuestras distribuciones de probabilidad se vuelven más precisas, y nuestra estrategia se inclina más hacia la explotación de los “brazos” que han demostrado dar altas recompensas. Sin embargo, siempre hay una posibilidad de exploración, porque siempre hay alguna probabilidad de que un “brazo” que no hemos jugado mucho podría dar una recompensa alta.

Se ha demostrado que este enfoque es efectivo en muchas aplicaciones y tiene una justificación teórica sólida en términos de minimizar el arrepentimiento. El muestreo de *Thompson* ha ganado popularidad en los últimos años debido a su capacidad para equilibrar la exploración y la explotación de manera fundamentada [62]. Al muestrear las distribuciones de probabilidad sobre las distribuciones de recompensa, el muestreo de *Thompson* fomenta la exploración de todas las opciones mientras favorece las opciones con recompensas esperadas más altas. Además, el marco bayesiano permite la incorporación de conocimiento previo sobre las distribuciones de recompensa, lo cual puede ser especialmente útil en escenarios con datos limitados.

El problema de *MAB* está intrínsecamente relacionado con el ámbito del *Aprendizaje por Refuerzo (RL)*. En el *RL*, un agente inteligente se esfuerza por adquirir conocimiento y desarrollar una estrategia, conocida como política, que maximice sus recompensas totales a lo largo de su interacción con un entorno. En los últimos años, el *RL* ha exhibido un éxito notable en diversos dominios. Específicamente, ha encontrado una utilidad significativa en el ámbito de la predicción de la demanda [63], [64].

En el contexto de nuestro trabajo, utilizamos el algoritmo de *MAB* para seleccionar el mejor clasificador (*IDS*) para cada solicitud de tráfico de red. Esto es similar a cómo se utiliza *MAB* en la predicción de la demanda para seleccionar

el modelo de predicción óptimo o determinar los mejores hiperparámetros para un modelo de predicción dado. Al utilizar *MAB*, podemos equilibrar el compromiso entre explotar los clasificadores que tienen la precisión esperada más alta según la información actual y explorar nuevos clasificadores que podrían proporcionar una mayor precisión en el futuro.

Capítulo 3. Apolo

En este capítulo explicaremos con más detalle cómo funciona el módulo de clasificación de peticiones **Apolo**, ya que es la pieza fundamental del TFG.

El sistema de defensa propuesto, llamado **Apolo**, está diseñado para proteger un *IDS* contra ataques *AML* utilizando múltiples capas para proporcionar una mejor seguridad que los *IDS* tradicionales y los trabajos anteriores que se basan únicamente en el entrenamiento con tráfico adversario. El sistema incluye múltiples clasificadores, un algoritmo *Multi-Armed Bandits (MAB)* y agrupación de peticiones para proporcionar una defensa robusta contra los ataques *AML*.

La primera capa de **Apolo** implica el uso de múltiples clasificadores, en lugar de un único clasificador que se utiliza normalmente en *IDS*. El uso de múltiples clasificadores aumenta la dificultad para los atacantes potenciales que intentan replicar el modelo *IDS*, ya que no pueden predecir qué modelo específico será responsable de clasificar una solicitud determinada. Para seleccionar dinámicamente el clasificador o conjunto de clasificadores óptimos para cada entrada, **Apolo** utiliza un algoritmo *MAB* con muestreo de *Thompson*. El algoritmo *MAB* selecciona el brazo (clasificador) que debe utilizarse para cada solicitud en función del estado actual del sistema.

Las solicitudes también se agrupan en *clústeres*, y hay una versión de cada clasificador para cada *clúster*, entrenada sólo con la información de ese *clúster*. La agrupación añade otra capa de incertidumbre al sistema, ya que el atacante no puede predecir de forma sencilla a qué *clúster* pertenece una solicitud.

Con **Apolo**, el rendimiento de los *IDS* basado en usar un único clasificador, cuando se trata de tráfico de red sin ataques *AML*, se mantiene, ya que, utiliza los mejores clasificadores para cada tipo de petición. Además, al meter la capa de incertidumbre, **Apolo** ofrece una solución para evitar que los atacantes aprendan fácilmente del comportamiento de los clasificadores.

En la Ilustración 13 se puede observar el funcionamiento de **Apolo** y el flujo que sigue una solicitud de tráfico de red hasta que es clasificada. En la Ilustración 11 se muestra el resultado de un ejemplo de proceso de entrenamiento de **Apolo** con dos *clústeres* y tres clasificadores.

Primero se dividen los datos de entrenamiento entre los dos *clústeres*, se entrena los clasificadores para cada *clúster* y, finalmente, se asignan las recompensas. En la Ilustración 12 se muestra un ejemplo de proceso de selección del clasificador para cada solicitud, el valor predicho y el valor real.

Apolo es un sistema de defensa robusto que combina múltiples capas, incluyendo múltiples clasificadores, un algoritmo *MAB*, y agrupación de peticiones para proporcionar una defensa segura contra ataques *AML*. La arquitectura del sistema proporciona un enfoque dinámico y adaptativo para seleccionar el clasificador o conjunto de clasificadores óptimo para cada solicitud de tráfico de red, garantizando un alto nivel de seguridad contra posibles atacantes.

A continuación, vamos a describir cada una de las 3 capas de **Apolo**, desde la más interna (clasificadores) a la más externa (*clúster*).

```
Cluster 0: Len of request 274888
Training arm 0 on cluster 0
Training arm 1 on cluster 0
Training arm 2 on cluster 0
Cluster 1: Len of request 358525
Training arm 0 on cluster 1
Training arm 1 on cluster 1
Training arm 2 on cluster 1

Setting reward_sums arm 0 on cluster 0
Setting reward_sums arm 1 on cluster 0
Setting reward_sums arm 2 on cluster 0
Setting reward_sums arm 0 on cluster 1
Setting reward_sums arm 1 on cluster 1
Setting reward_sums arm 2 on cluster 1
```

Ilustración 11. Ejemplo del proceso de entrenamiento de Apolo.

Selected arm: 0.0	Predicted:0	Actual:0
Selected arm: 0.0	Predicted:0	Actual:0
Selected arm: 2.0	Predicted:0	Actual:0
Selected arm: 0.0	Predicted:0	Actual:0
Selected arm: 1.0	Predicted:1	Actual:1
Selected arm: 2.0	Predicted:0	Actual:0

Ilustración 12. Ejemplo del proceso de selección de los clasificadores de Apolo.

Apolo: Sistema inteligente de monitorización y detección de ataques basados en inteligencia artificial

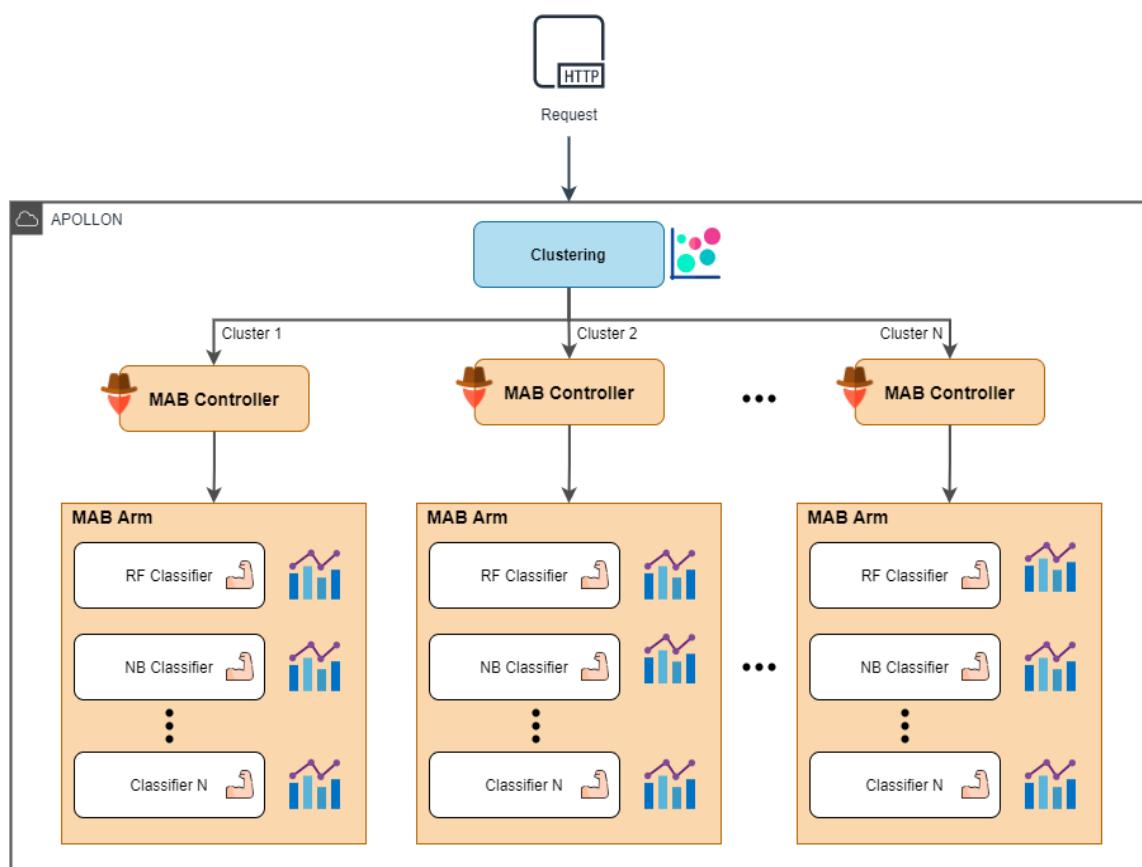


Ilustración 13. El funcionamiento de Apolo.

3.1. Modelos

En esta sección explicaremos la primera capa de **Apolo**, para ello primero tenemos que describir que datos estamos usando y como son obtenidos, como preprocesamos estos datos y que clasificadores permitimos utilizar en **Apolo**.

3.1.1. CIC-FlowMeter

CIC-FlowMeter [17] es una herramienta de código abierto desarrollada por el *Canadian Institute for Cybersecurity (CIC)* para generar características de tráfico de red a partir del tráfico de red en tiempo real o a partir de archivos “pcap” (captura de paquetes). El objetivo de estas características es proporcionar un conjunto de datos para el análisis de seguridad, en particular para la detección de anomalías o ataques.

CIC-FlowMeter genera más de 80 características de tráfico de red. A continuación, proporcionamos algunas de las características más comunes junto con una breve descripción:

1. **Duración:** El tiempo total de la conexión entre el envío del primer paquete y el reconocimiento del último paquete.
2. **Protocolo:** El protocolo utilizado para la transmisión de paquetes (por ejemplo, *TCP*, *UDP*, *ICMP*, etc.)
3. **Paquetes de origen:** El número total de paquetes enviados desde el origen al destino.
4. **Paquetes de destino:** El número total de paquetes enviados desde el destino al origen.
5. **Bytes de origen:** El número total de bytes enviados desde el origen al destino.
6. **Bytes de destino:** El número total de bytes enviados desde el destino al origen.
7. **Tasa de paquetes de origen/destino:** El número de paquetes por segundo (p/s) enviados desde el origen/destino.
8. **Tasa de bytes de origen/destino:** El número de bytes por segundo (b/s) enviados desde el origen/destino.
9. **Tamaño medio del paquete:** El tamaño promedio de los paquetes transmitidos durante la conexión.
10. **Flujo de bytes:** Número total de bytes transmitidos en la conexión.
11. **Flujo de paquetes:** Número total de paquetes transmitidos en la conexión.

- 12. Tiempo de inicio de flujo:** Tiempo en que comenzó la conexión.
- 13. Tiempo de finalización del flujo:** Tiempo en que finalizó la conexión.
- 14. Puerto de origen y destino:** Los puertos utilizados por el origen y el destino.
- 15. Dirección IP de origen y destino:** Las direcciones IP del origen y el destino.

Estas son solo algunas de las muchas características que *CIC-FlowMeter* puede capturar. En general, todas estas características son valiosas para el análisis de tráfico de red y la detección de posibles amenazas de seguridad.

3.1.2. Preprocesado

El preprocesamiento de las peticiones es fundamental para garantizar que los datos sean adecuados para el entrenamiento y la evaluación de los modelos de clasificación.

En primer lugar, una de las tareas de preprocesamiento más importantes es la eliminación de elementos categóricos de las peticiones. Esto se debe a que los modelos de clasificación, en general, requieren datos numéricos para su entrenamiento y evaluación. Por lo tanto, los elementos categóricos, como las *IDS* de las peticiones y los nombres de protocolos, deben ser codificados numéricamente o eliminados, en nuestro caso hemos realizado un estudio de las variables del dataset, haciendo uso de técnicas de interpretabilidad como *Partial Dependence Plots (PDP)* [65], con estas técnicas hemos concluido que el nombre de los protocolos no aportaba un valor notable a la clasificación final, por lo que hemos decidido eliminar tanto los *IDS* (estos nunca aportan información, de hecho siempre hay que eliminarlos), como los nombres de los protocolos.

Además, es importante eliminar los valores faltantes y erróneos de los datos, ya que pueden afectar negativamente la precisión de los modelos de clasificación. En el caso de valores faltantes (*Nans*), existen varias técnicas para tratarlos, como la eliminación de filas con valores faltantes, la sustitución de valores faltantes utilizando la media, la mediana o un valor constante, o la imputación utilizando técnicas más avanzadas, como *KNN* o modelos de regresión. En el caso de **Apolo**, se utiliza la sustitución de valores faltantes por la constante “0”.

Por otro lado, los valores erróneos pueden ser eliminados o imputados utilizando técnicas similares a las utilizadas para los valores faltantes. En el caso de **Apolo**, se utiliza la sustitución de valores faltantes por la constante “0”, ya que se ha observado que esta técnica proporciona buenos resultados en la mayoría de los casos.

Una vez que se han eliminado los elementos categóricos y los valores faltantes y erróneos, es importante escalar y normalizar los datos. Esto se debe a que las diferentes características de las peticiones pueden tener diferentes escalas y varianzas, lo que puede afectar negativamente la precisión de los modelos de clasificación. En el caso de **Apolo**, se utiliza un “*RobustScaler*” para escalar los datos, lo que reduce la influencia de los valores atípicos en la escala de las características. Luego, se normalizan los datos para asegurarse de que todas las características tengan una media de cero y una varianza unitaria.

3.1.3. Clasificadores

La capa inicial de **Apolo** despliega una multitud de clasificadores en lugar de un único clasificador, que es lo que se emplea tradicionalmente en los *IDS*. El objetivo principal de utilizar múltiples clasificadores es aumentar el nivel de dificultad para que el atacante replique el modelo *IDS*. Al emplear un conjunto variado de clasificadores, el sistema se vuelve más resistente, generando una mayor incertidumbre para los atacantes, lo que se traduce en una mejor puntuación de rendimiento para el sistema **Apolo**. Los clasificadores utilizados en **Apolo** pueden ser de varios tipos, incluyendo los basados en *Deep Learning*, *Machine Learning* o técnicas de previsión de peticiones de tráfico de red, así como los clásicos que utilizan sistemas de reglas. La diversidad en la selección de clasificadores es una característica clave de **Apolo**, que mejora su seguridad y eficacia.

Para **Apolo** utilizamos 7 tipos de clasificadores diferentes, *Random Forest (RF)*, árboles de decisión (*DT*), *Naive Bayes (NB)*, *MultiLayer Perceptron (MLP)*, *Support Vector Classification (SVC)*, *K-Nearest Neighbors (KNN)* y Regresión Logística (*LR*). A continuación, explicaremos cómo funciona cada uno de ellos, específicamente en los problemas de clasificación.

Para la mayoría de los modelos hemos utilizado los parámetros por defecto, puesto que el objetivo de **Apolo** no es maximizar el funcionamiento de los modelos individualmente, si no que el conjunto de estos modelos unidos sea capaz de conseguir que el algoritmo *MAB* tenga muchas clasificaciones entre las que poder elegir, aumentando así la incertidumbre y retrasando el punto en el que el atacante pueda replicar **Apolo**.

3.1.3.1. Árboles de Decisión

Los árboles de decisión (*DT*) [31] son un algoritmo de *ML* utilizado en problemas de clasificación y otras tareas de minería de datos. Se basan en una estructura de árbol en la que cada nodo representa una pregunta o una prueba sobre una característica del conjunto de datos.

En un árbol de decisión, se divide el conjunto de datos en función de las características relevantes en cada nodo. La división se realiza buscando la característica que proporciona la mayor ganancia de información o la reducción más significativa en la impureza de los datos. La impureza se refiere a la mezcla de diferentes clases de datos en un nodo y se puede medir mediante métricas como la entropía o el índice *Gini*, nosotros utilizamos este último, ya que, hacemos uso de la librería *sklearn*¹¹ que se basa en dicho índice.

Cada rama del árbol representa una opción o un valor posible para la característica en ese nodo. El proceso de división y crecimiento del árbol continúa recursivamente hasta alcanzar ciertos criterios de parada, como una profundidad máxima del árbol o un número mínimo de muestras en un nodo.

Durante la etapa de clasificación, los datos se propagan a través del árbol siguiendo las condiciones de prueba en cada nodo. Finalmente, los datos llegan a las hojas del árbol, donde se asigna una etiqueta de clase o una predicción basada en la mayoría de las muestras pertenecientes a esa hoja.

Los árboles de decisión tienen varias ventajas. Son fáciles de entender e interpretar, ya que su estructura se asemeja a un diagrama de flujo de decisiones. Además, los árboles de decisión pueden manejar características categóricas y numéricas, y pueden manejar conjuntos de datos grandes y complejos.

Sin embargo, los árboles de decisión también tienen limitaciones. Pueden ser propensos al sobreajuste si no se controla su crecimiento adecuadamente. Además, pueden ser sensibles a pequeños cambios en los datos de entrenamiento, lo que puede resultar en árboles diferentes. Esto se conoce como inestabilidad del árbol.

Se decidió elegir *DT* para **Apolo**, ya que, al ser tan eficiente y rápido no supone un gran coste computacional y así *MAB* cuenta con un enfoque más simple que *RF* para tomar las decisiones.

¹¹ <https://scikit-learn.org/stable/>

3.1.3.2. Random Forest

Random Forest (RF) [30], es un algoritmo de aprendizaje automático que se utiliza comúnmente en problemas de clasificación. Es una técnica basada en la combinación de múltiples árboles de decisión individuales para tomar decisiones colectivas.

En un *RF*, se construye un conjunto de árboles de decisión independientes entre sí. Cada árbol se entrena utilizando una muestra aleatoria del conjunto de datos original, seleccionando aleatoriamente características del conjunto total de características disponibles en cada división de nodo. Esta aleatoriedad aporta diversidad a los árboles individuales y evita el sobreajuste al reducir la correlación entre ellos.

El proceso de entrenamiento de un *RF* implica la creación de un gran número de árboles de decisión mediante un enfoque de “ensamblado” o combinación. Durante la clasificación, cada árbol en el conjunto emite su propia predicción, y la predicción final se determina por votación mayoritaria o promediando las predicciones de todos los árboles.

La fortaleza de los *RF* radica en su capacidad para manejar conjuntos de datos complejos y de alta dimensionalidad, y para lidiar con características irrelevantes o ruidosas. Además, los *RF* son menos propensos a sufrir sobreajuste en comparación con un solo árbol de decisión.

El modelo de *RF* es el que mejor precisión obtiene en **Apolo**, por lo que en la mayoría de los casos *MAB* escoge el *RF* en contraposición del resto de modelos.

3.1.3.3. Naive Bayes

Naive Bayes (NB) [66] es un algoritmo de aprendizaje automático basado en el teorema de *Bayes* y la suposición de independencia condicional entre las características. Se utiliza comúnmente en problemas de clasificación y se basa en la probabilidad y la estadística.

El algoritmo *NB* asume que las características son independientes entre sí, lo que significa que la presencia o ausencia de una característica no está relacionada con la presencia o ausencia de otras características. Esta suposición de independencia condicional simplifica el cálculo de la probabilidad conjunta de las características y permite realizar predicciones de manera eficiente.

Durante la etapa de entrenamiento, se estiman las probabilidades a priori y las probabilidades condicionales de cada característica para cada clase en el conjunto de datos de entrenamiento. Estas probabilidades se utilizan para construir un modelo probabilístico del problema.

Durante la etapa de clasificación, se utiliza el modelo probabilístico para calcular la probabilidad de pertenencia a cada clase para una nueva instancia. La clase con la mayor probabilidad a posteriori se elige como la predicción final.

NB es especialmente útil en problemas de clasificación con conjuntos de datos grandes y dimensionalidad alta, ya que su suposición de independencia condicional reduce la complejidad computacional. Además, el algoritmo es robusto frente a la presencia de características irrelevantes o redundantes.

Sin embargo, la suposición de independencia condicional puede ser poco realista en algunos casos, lo que puede afectar la precisión del algoritmo. Además, *NB* puede verse afectado por el problema de “cero frecuencia”, donde una característica no aparece en una clase particular en el conjunto de entrenamiento, lo que resulta en una probabilidad de cero.

En **Apolo** se han preprocesado las características irrelevantes o redundantes, por lo que *NB* no da el mejor resultado posible. Lo hemos elegido debido a su baja complejidad y a su rápida ejecución. *MAB* no lo suele elegir, pero nunca está de menos tener un enfoque tan diferente al resto como es *NB*.

3.1.3.4. MultiLayer Perceptron

Las *MLP* (*MultiLayer Perceptron*) [67] son una clase de redes neuronales artificiales utilizadas en el aprendizaje automático para resolver problemas de clasificación. Se basan en el concepto de neuronas artificiales interconectadas en capas.

Una *MLP* consta de una capa de entrada, una o más capas ocultas y una capa de salida. Cada capa está compuesta por un conjunto de neuronas o nodos, y cada nodo está conectado con los nodos de la capa siguiente mediante conexiones ponderadas. Estas conexiones ponderadas representan los pesos que determinan la influencia de las señales de entrada en la salida de cada neurona.

Durante el entrenamiento de una *MLP*, los pesos se ajustan iterativamente para minimizar una función de costo, que mide la discrepancia entre las salidas predichas por la red y las salidas deseadas. Este proceso se realiza utilizando algoritmos de optimización, como el descenso de gradiente, que actualizan los pesos en función de la magnitud del error de predicción.

La capacidad de las *MLP* para aprender relaciones no lineales las hace adecuadas para resolver problemas de clasificación en los que las fronteras de decisión entre las clases son complejas. Las *MLP* necesitan una alta dimensionalidad de datos para poder generalizar y que no haya un sobreajuste en el entrenamiento.

En **Apolo** utilizamos una red neuronal con 3 capas de 32 neuronas cada una. Hemos añadido el parámetro de “*early_stopping*” a “*True*”, debido a que al haber tantos datos el entrenamiento llevaba mucho tiempo (en comparación con el resto de los modelos). El “*batch_size*” está en “200”, para que escoja *batchs* muy altos. El resto de los parámetros están por defecto. Aun con esta configuración la *MLP* lleva mucho tiempo de ejecución.

3.1.3.5. Regresión Logística

La Regresión Logística (*LR*) [68] es un algoritmo de aprendizaje automático utilizado para resolver problemas de clasificación binaria y multiclas. Aunque el nombre “regresión” está presente, la regresión logística se utiliza para clasificar datos en lugar de predecir valores continuos.

El objetivo de la regresión logística es modelar la probabilidad de que una instancia pertenezca a una clase específica en función de sus características de entrada. Se basa en el modelo logístico, que utiliza la función logística (también conocida como función sigmoide) para mapear las entradas a un rango entre 0 y 1, que se interpreta como la probabilidad de pertenencia a una clase.

El modelo de regresión logística asume que la relación entre las características de entrada y la probabilidad de pertenencia a una clase sigue una función logística. Durante el entrenamiento, se ajustan los parámetros del modelo para que se ajuste mejor a los datos de entrenamiento y maximice la verosimilitud de los datos observados.

El algoritmo utiliza el método de máxima verosimilitud para estimar los parámetros del modelo. Se busca encontrar los valores de los coeficientes que maximicen la probabilidad de que el modelo genere los resultados observados.

Una vez entrenado, el modelo de regresión logística se utiliza para predecir la clase de nuevas instancias en función de sus características. El modelo calcula la probabilidad de pertenencia a cada clase y luego asigna la clase con la probabilidad más alta como la predicción final.

La regresión logística es especialmente útil cuando se trata de problemas binarios o de múltiples clases y cuando se desea obtener probabilidades de clasificación en lugar de una simple etiqueta de clase.

Se utilizó la regresión logística en **Apolo** debido a que este modelo es muy rápido y simple en términos de computaciones, lo que favorece a tener otra clasificación para que *MAB* pueda tener un espectro mayor a la hora de elegir.

3.1.3.6. SVC

El *Support Vector Classification* (SVC) [69] es un algoritmo de aprendizaje automático utilizado para resolver problemas de clasificación. Está basado en la teoría de máquinas de soporte vectorial (SVM) [32], que se fundamenta en la búsqueda de un hiperplano óptimo que separe las diferentes clases de datos.

SVC busca encontrar un hiperplano en un espacio de alta dimensión que maximice la separación entre las muestras de diferentes clases. El hiperplano óptimo es aquel que tiene la máxima distancia a las muestras de entrenamiento más cercanas de cada clase, llamadas vectores de soporte.

La separación lograda por el hiperplano óptimo se logra a través de la técnica de margen máximo. El margen es la distancia perpendicular más corta entre el hiperplano y las muestras de entrenamiento. El objetivo del SVC es maximizar este margen para obtener una clasificación más robusta y generalizable.

Durante el entrenamiento del SVC, se busca encontrar los parámetros que definen el hiperplano óptimo. Esto se realiza mediante técnicas de optimización convexa que buscan minimizar una función de costo que penaliza el error de clasificación y maximiza el margen.

Una vez entrenado, el modelo SVC puede clasificar nuevas muestras asignándolas a una de las clases en función de su posición con respecto al hiperplano óptimo.

El mayor problema de SVC es el alto tiempo de entrenamiento que este conlleva, siendo el que más tiempo tarda en entrenar de todos los modelos escogidos.

3.1.3.7. K-Nearest Neighbors

El algoritmo de *K-Nearest Neighbors* (KNN) [70] es un método utilizado en el aprendizaje automático para resolver problemas de clasificación y regresión. Se basa en la idea de que los elementos similares tienden a estar cerca unos de otros en el espacio de características.

El algoritmo de *KNN* asigna una instancia no etiquetada a una clase en función de las etiquetas de sus vecinos más cercanos en el espacio de características. El número de vecinos considerados, denotado como “*K*”, es un parámetro que se elige de antemano. En el caso de **Apolo**, se utiliza el valor por defecto “*K=3*”.

Durante el proceso de clasificación, el algoritmo calcula la distancia entre la instancia no etiquetada y todos los puntos de entrenamiento en el espacio de características. Luego, selecciona los *K* puntos más cercanos a la instancia no etiquetada. La clase mayoritaria entre estos vecinos se asigna como la clase de la instancia no etiquetada. Para nuestro caso, en la clasificación binaria, si *K* es un número impar, se evita la indecisión en la asignación de clases. Si *K* es un número par y se presenta un empate, se puede utilizar un criterio adicional, como la distancia promedio a los vecinos, para desempatar.

Al igual que los modelos de *MLP* y *SVC* el problema de *KNN*, el elevado tiempo de entrenamiento que tiene el modelo para datasets grandes (nuestro caso).

3.1.4. Ejemplo

En la Ilustración 14, se puede observar el funcionamiento de la capa de modelos. Una vez se han elegido que modelos han de clasificar una petición “X” (ver 3.3. “Clusterización”), estos modelos pasan a formar parte de los brazos de *MAB* y todos ellos clasifican la petición “X” obteniendo un resultado entre 0 y 100.

Apolo: Sistema inteligente de monitorización y detección de ataques basados en inteligencia artificial

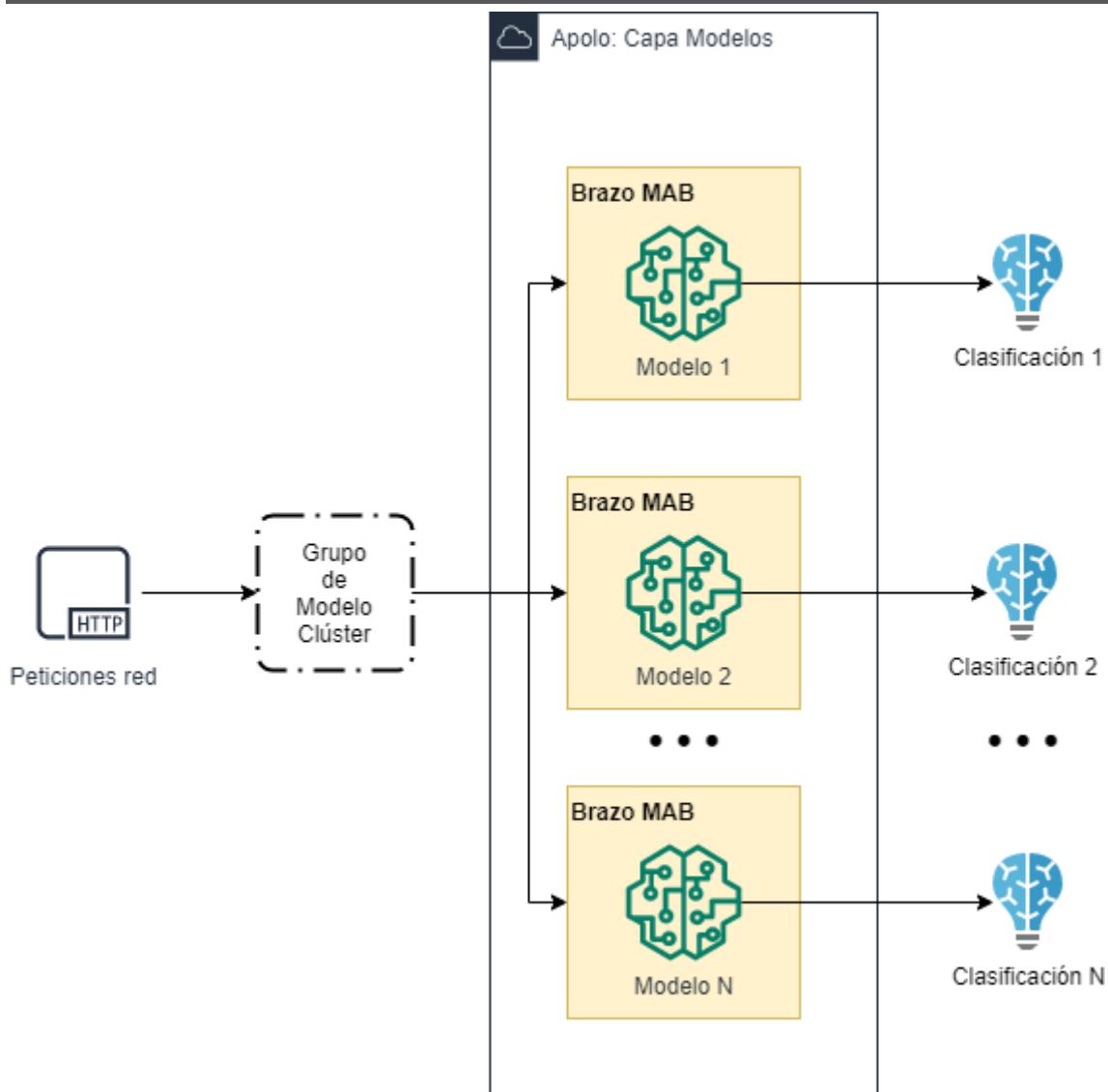


Ilustración 14. Apolo: Capa Modelos

3.2. MAB

La segunda capa del sistema de defensa **Apolo** utiliza un algoritmo *Multi-Armed Bandit (MAB)* para seleccionar el clasificador o conjunto de clasificadores más adecuado para cada solicitud de tráfico de red. Esta capa elimina la necesidad de ajustar manualmente los umbrales o pesos de cada clasificador. El algoritmo *MAB* elige el mejor clasificador o conjunto de clasificadores seleccionando el brazo, o clasificador, que tiene la mayor probabilidad de proporcionar la clasificación correcta. El muestreo de *Thomson*, un algoritmo popular para resolver el problema *MAB*, se utiliza en **Apolo** para equilibrar la exploración y la explotación de los clasificadores disponibles. Esto garantiza que el sistema seleccione el clasificador o conjunto de clasificadores óptimos sin dejar de responder a tipos de tráfico nuevos y desconocidos.

El algoritmo *MAB* de **Apolo** tiene en cuenta los diferentes tipos de clasificadores utilizados en el sistema. Por ejemplo, si el clasificador *Random Forest* tiene una alta probabilidad de ser correcto, pero los clasificadores *Naive Bayes* y *Logistic Regression* tienen probabilidades más bajas, el algoritmo *MAB* seleccionará el clasificador *Random Forest* para esa solicitud concreta. Esto garantiza que el sistema seleccione el conjunto óptimo de clasificadores para cada solicitud, mejorando la precisión general de la clasificación. El algoritmo *MAB* actualiza las probabilidades de los distintos clasificadores en función de su rendimiento anterior, lo que permite al sistema adaptarse a los cambios en los patrones de tráfico a lo largo del tiempo y mantenerse al día de los últimos tipos de ataques.

En la Ilustración 15 se puede observar un algoritmo que describe el proceso de actualización utilizado por el algoritmo *MAB* en **Apolo**. S_i y F_i representan el número de éxitos y fracasos observados para el brazo i , respectivamente, mientras que θ_{ai} es la probabilidad estimada de obtener una recompensa positiva del brazo i . El algoritmo utiliza la distribución *Beta* como distribución a priori para los parámetros θ_{ai} y la actualiza con los datos observados utilizando el teorema de Bayes. El algoritmo selecciona el brazo (clasificador *ML*) que tiene la mayor probabilidad de ser el mejor según las muestras de la distribución posterior.

```
Iniciarizar  $S_i = 0$  y  $F_i = 0$  para cada brazo  $i$ 
for  $t = 1, 2, \dots$  do
    Para cada brazo  $i$ , muestra  $\theta_i$  de la distribución Beta
    ( $S_i + 1, F_i + 1$ )
    Elige el brazo  $I_t$  que maximiza  $\theta_i$ 
    Observa la recompensa  $X_t$  del brazo  $I_t$ .
    if  $X_t = 1$  then
        Incrementa  $S_{I_t}$  en uno
    else
        Incrementa  $F_{I_t}$  en uno
    end if
end for
```

Ilustración 15. Algoritmo: Apolo Thompson Sampling

Apolo utiliza un algoritmo *MAB* que permite seleccionar el mejor clasificador o conjunto de clasificadores para cada solicitud de tráfico de red, lo que permite al sistema responder mejor a los nuevos tipos de ataque. El muestreo *Thomson* se emplea para garantizar un equilibrio entre exploración y explotación, lo que mejora la tasa global de detección de ataques de la clasificación. La figura Ilustración 16 muestra el diagrama del algoritmo *MAB* con varios clasificadores.

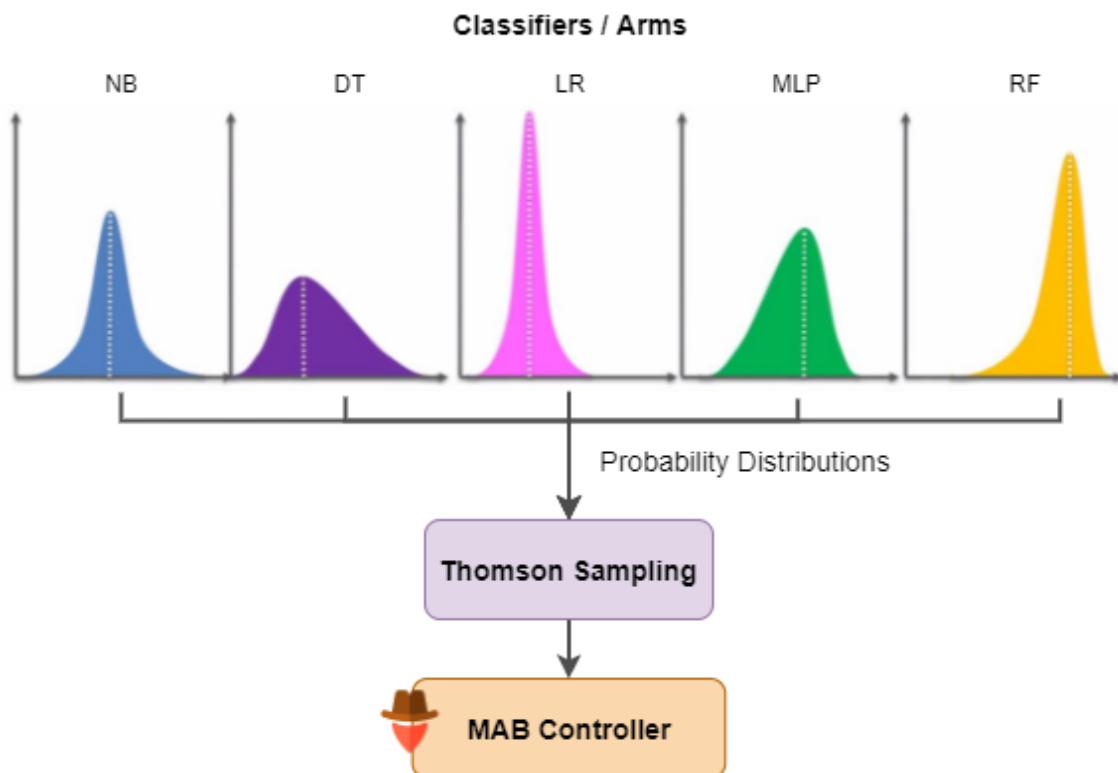


Ilustración 16. Apolo: El algoritmo Multi-Armed Bandits con varios clasificadores.

3.2.1. Ejemplo

En la Ilustración 17, se puede ver un ejemplo del funcionamiento de MAB. Dentro del flujo entrada/salida de MAB, las entradas serían las clasificaciones que han realizado cada uno de los “brazos” (modelos) de MAB. Todas estas clasificaciones son evaluadas por el muestreo de *Thompson*, ya sea por exploración o por explotación. Una vez *Thompson* decide que clasificador va a utilizar (no tiene por qué ser el que de mejor resultado), envia este clasificador como salida de esta capa, para que se pueda guardar su clasificación.

Apolo: Sistema inteligente de monitorización y detección de ataques basados en inteligencia artificial

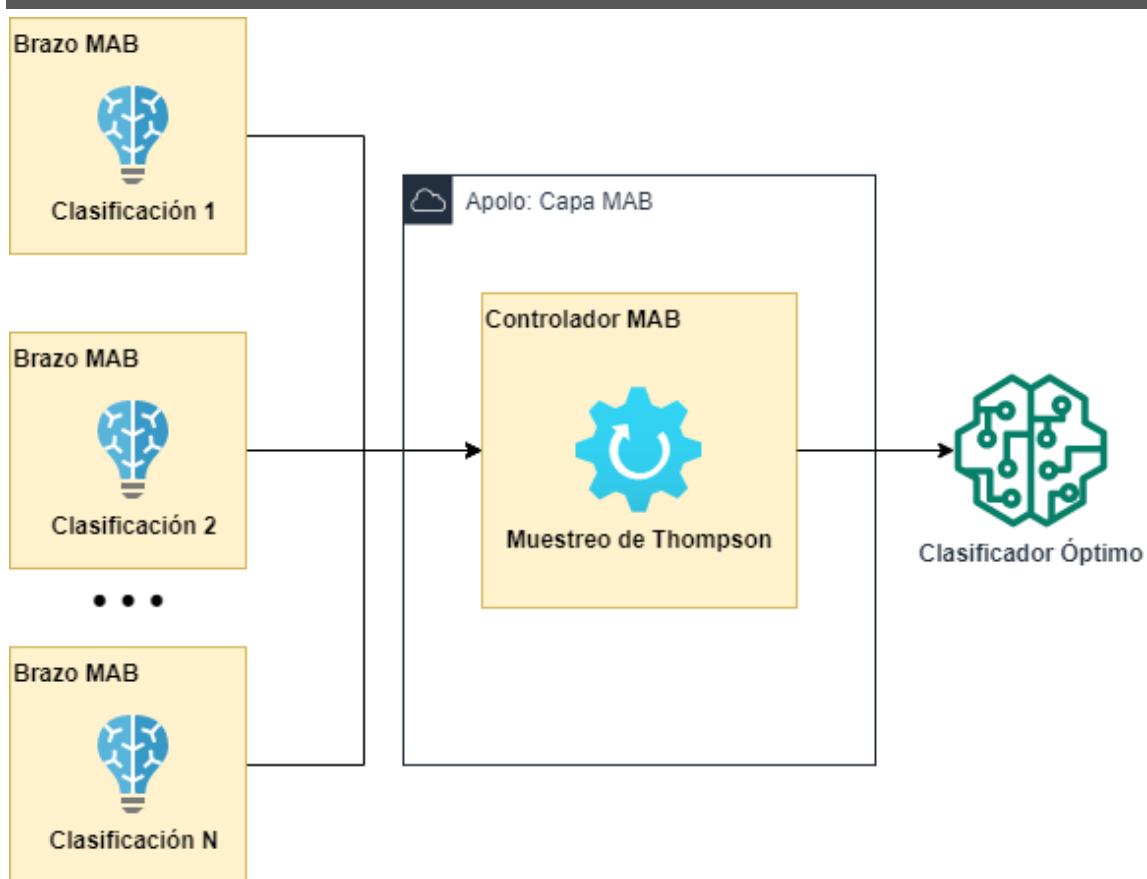


Ilustración 17. Apolo: Capa MAB

3.3. Clusterización

La capa final del sistema de defensa **Apolo** involucra agrupar las solicitudes de tráfico de red en *clústeres* basados en sus características y luego entrenar una versión separada de cada clasificador para cada *clúster*. De esta manera, el sistema logra una mayor precisión para cada *clúster* al entrenar los clasificadores específicamente para los patrones de tráfico en ese *clúster*.

Las solicitudes de tráfico se agrupan utilizando el algoritmo *K-Means* [71], que es un popular algoritmo de agrupamiento utilizado en muchas aplicaciones de *Machine Learning*. El algoritmo *K-Means* funciona seleccionando aleatoriamente K puntos como centros iniciales, luego va actualizando iterativamente estos centros hasta que los *clústeres* convergen.

En **Apolo**, utilizamos el algoritmo *K-Means* para agrupar las solicitudes de tráfico de red en función de sus características, asegurando que las solicitudes con características similares se agrupen juntas. Estas características son las mismas que utilizan los clasificadores para evaluar las solicitudes, garantizando que el agrupamiento se base en la misma información que los clasificadores utilizan para tomar sus decisiones.

Para cada *clúster*, se entrena una versión separada de cada clasificador utilizando solo las solicitudes de tráfico en ese *clúster*. Esto garantiza que cada clasificador esté ajustado específicamente a los patrones de tráfico en ese *clúster*.

Cuando llega una nueva solicitud de tráfico de red a **Apolo**, se clasifica en el *clúster* apropiado en función de sus características. Luego, se utiliza el algoritmo de *MAB* para seleccionar el conjunto óptimo de clasificadores para ese *clúster*, teniendo en cuenta el rendimiento de cada clasificador en ese *clúster* específico. Luego, el clasificador o conjunto de clasificadores seleccionados evalúan la solicitud y determinan si es benigna o maligna.

Al agrupar las solicitudes de tráfico de red y entrenar versiones separadas de cada clasificador para cada *clúster*, **Apolo** permite que el algoritmo de *MAB* genere múltiples distribuciones de probabilidad para cada clasificador, dependiendo del tipo de solicitud recibida. Esta combinación de técnicas dificulta que posibles atacantes identifiquen el clasificador que proporciona la respuesta, reduciendo la probabilidad de una imitación exitosa.

3.3.1. Ejemplo

En la Ilustración 18, se puede observar un ejemplo del flujo de la capa de clusterización de **Apolo**. En dicha imagen se ve como llegan las peticiones de red a **Apolo** (estas ya han sido tratadas), llamemos a la petición “X”. El clúster en función de sus atributos y estructura interna, la asigna a un grupo de modelos de clasificación. El resultado que obtenemos es que para la petición “X” tenemos un grupo de modelos óptimos que la han de clasificar.

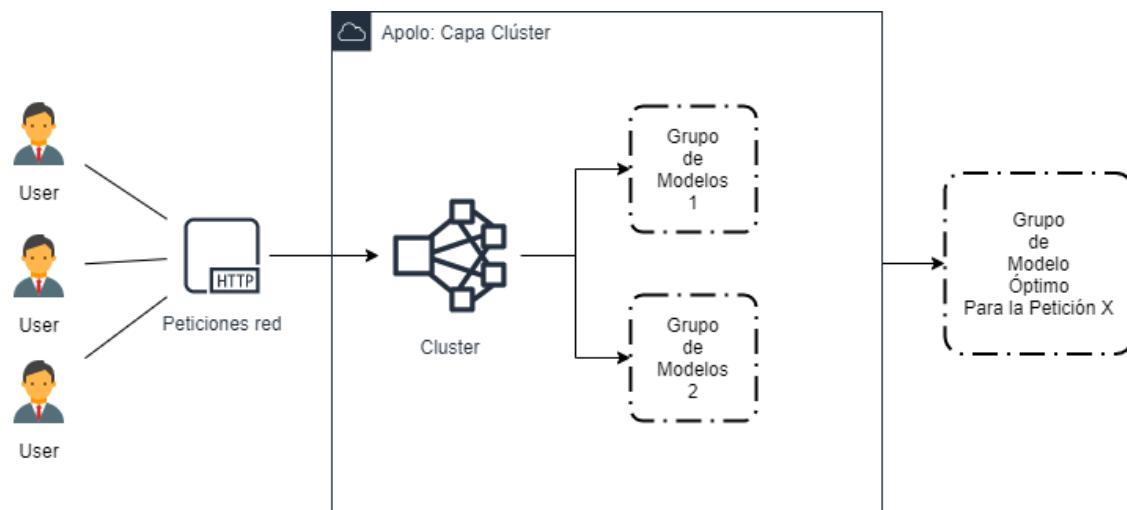


Ilustración 18. Apolo: Capa Clúster

Capítulo 4. Planificación y Presupuesto del Proyecto (Inicial)

En este capítulo, se tratará la planificación del proyecto inicial y el presupuesto del proyecto inicial. Para la realización de estas dos partes aun no tenía los suficientes conocimientos sobre estos campos para realizar un buen trabajo, por lo que en el Capítulo 12. "Planificación y Presupuesto del Proyecto " está la planificación y presupuestos finales.



4.1. Planificación Inicial

Una vez definido el alcance y los objetivos de este proyecto, se ha definido una planificación inicial, la cual se seguirá a lo largo de este proyecto. Al final del proyecto se realizará otra planificación, con todas las variaciones que hayan ocurrido durante el proyecto.

La planificación muestra una serie de tareas planificadas, las cuales se muestran en Ilustración 19, Ilustración 20 e Ilustración 21. En dichas tareas se puede observar (de izquierda a derecha) el nombre de la propia tarea, la estimación de cuánto se tardará el realizar, las fechas en las que se realizará, las tareas predecesoras que son necesarias realizar antes de hacer la propia tarea y el recurso que necesitan utilizar, en este caso solo hay un recurso “Sergio Arroni del Riego” y es el que se va a realizar todas las tareas.

También, se ha realizado un diagrama de Gantt (Ilustración 22, Ilustración 23 e Ilustración 24) el cual ayuda a la estimación del tiempo de las tareas, ya que permite ver una visión más global del proyecto al señalar todas las tareas representadas en función de su duración y ordenadas en función del inicio de la tarea.

En la duración de las tareas se han utilizado dos escalas de tiempo (días y horas). Debido a que este proyecto se ha tenido que compaginar con la universidad y el trabajo, un día en el proyecto puede equivaler a unas 3 horas de trabajo medio en el proyecto. Por lo que si el proyecto tiene una duración total de 114 días (desde el 2/1/23 hasta el 9/6/23) se podría decir que el proyecto tiene una estimación de 342 horas en total.

El horario en el que se realizarán las 3 horas fluctuará en función del día y semana que sea, principalmente de lunes a miércoles será por la mañana y de jueves a viernes será por la tarde, la franja de horas exacta me es imposible darla, ya que al teletrabajar mi horario de trabajo también fluctúa.

Al inicio del proyecto se realizaron dos reuniones, las cuales sirvieron para sentar las bases del proyecto y para organizar el desarrollo del proyecto.

Cada semana, los miércoles, se organiza una reunión con los directores del proyecto. El cometido de dicha reunión es comentarles los avances realizados, así como los bloqueos que haya tenido durante la semana. La duración es de una 1 hora, aunque si por algún motivo durara más tiempo no habría ningún problema.

La fase de desarrollo se subdivide en diferentes fases (diagrama de Gantt y tareas):

- Análisis: En esta fase será una toma de contacto con el entorno y el tema del proyecto, para ello habrá que investigar sobre las posibles herramientas, arquitecturas, sistemas y sobre todo el estado del arte del tema del proyecto, ya que al ser un proyecto con carácter investigador es muy importante que esta parte se realice con meticulosidad.
- Diseño: En esta fase se recoge todos los datos extraídos del análisis y se comenzará a pensar en cómo desarrollarlos, se harán diagramas y bocetos del sistema.
- Implementación: una vez se haya diseñado todo correctamente, ahora hay que realizar todas las partes del sistema.
- Pruebas: En esta fase se probará el sistema que se acaba de implementar.

Luego va la fase de despliegue de toda la arquitectura y la realización de la demostración. Esta fase, a no ser que salga algo mal y haya fallos acumulados, debería de ser relativamente rápida.

Por último, va la documentación, esta parte puede ser la parte que más varíe respecto a la planificación final, ya que la voy realizando paralelamente a la realización de cada etapa, pero esto no siempre es posible por falta de tiempo.

Apolo: Sistema inteligente de monitorización y detección de ataques basados en inteligencia artificial

Nombre de tarea	Duración	Comienzo	Fin	Predecesoras	Nombres de los recursos
• Reuniones iniciales	2,13 días	lun 02/01/23	mié 04/01/23		Sergio Arroni del Riego
Reunión de planificación	1 hora	mié 04/01/23	mié 04/01/23		Sergio Arroni del Riego
Reunión de arranque	1 hora	lun 02/01/23	lun 02/01/23		Sergio Arroni del Riego
• Reuniones semanales	105,13 días	mié 11/01/23	mié 07/06/23	4;5	Sergio Arroni del Riego
Reunión semanal	1 hora	mié 11/01/23	mié 11/01/23		Sergio Arroni del Riego
Reunión semanal	1 hora	mié 18/01/23	mié 18/01/23		Sergio Arroni del Riego
Reunión semanal	1 hora	mié 25/01/23	mié 25/01/23		Sergio Arroni del Riego
Reunión semanal	1 hora	mié 01/02/23	mié 01/02/23		Sergio Arroni del Riego
Reunión semanal	1 hora	mié 08/02/23	mié 08/02/23		Sergio Arroni del Riego
Reunión semanal	1 hora	mié 15/02/23	mié 15/02/23		Sergio Arroni del Riego
Reunión semanal	1 hora	mié 22/02/23	mié 22/02/23		Sergio Arroni del Riego
Reunión semanal	1 hora	mié 01/03/23	mié 01/03/23		Sergio Arroni del Riego
Reunión semanal	1 hora	mié 08/03/23	mié 08/03/23		Sergio Arroni del Riego
Reunión semanal	1 hora	mié 15/03/23	mié 15/03/23		Sergio Arroni del Riego
Reunión semanal	1 hora	mié 22/03/23	mié 22/03/23		Sergio Arroni del Riego
Reunión semanal	1 hora	mié 29/03/23	mié 29/03/23		Sergio Arroni del Riego
Reunión semanal	1 hora	mié 05/04/23	mié 05/04/23		Sergio Arroni del Riego
Reunión semanal	1 hora	mié 12/04/23	mié 12/04/23		Sergio Arroni del Riego
Reunión semanal	1 hora	mié 19/04/23	mié 19/04/23		Sergio Arroni del Riego
Reunión semanal	1 hora	mié 26/04/23	mié 26/04/23		Sergio Arroni del Riego
Reunión semanal	1 hora	mié 03/05/23	mié 03/05/23		Sergio Arroni del Riego
Reunión semanal	1 hora	mié 10/05/23	mié 10/05/23		Sergio Arroni del Riego
Reunión semanal	1 hora	mié 17/05/23	mié 17/05/23		Sergio Arroni del Riego
Reunión semanal	1 hora	mié 24/05/23	mié 24/05/23		Sergio Arroni del Riego
Reunión semanal	1 hora	mié 31/05/23	mié 31/05/23		Sergio Arroni del Riego
Reunión semanal	1 hora	mié 07/06/23	mié 07/06/23		Sergio Arroni del Riego

Ilustración 19. Planificación de las tareas de reuniones semanales.

Apolo: Sistema inteligente de monitorización y detección de ataques basados en inteligencia artificial

Nombre de tarea	Duración	Comienzo	Fin	Predecesoras	Nombres de los recursos
« Proyecto	114 días	mar 03/01/23	vie 09/06/23		Sergio Arroni del Riego
« Reuniones	112,13 días	lun 02/01/23	mié 07/06/23		Sergio Arroni del Riego
« Reuniones iniciales	2,13 días	lun 02/01/23	mié 04/01/23		Sergio Arroni del Riego
« Reuniones semanales	105,13 días	mié 11/01/23	mié 07/06/23	4;5	Sergio Arroni del Riego
« Desarrollo	88 días	mié 04/01/23	vie 05/05/23	3	Sergio Arroni del Riego
« Análisis	26 días	mié 04/01/23	mié 08/02/23	7;8;9;10;11	Sergio Arroni del Riego
Análisis de las herramientas	7 días	mié 04/01/23	jue 12/01/23		Sergio Arroni del Riego
Análisis del sistema	3 días	vie 13/01/23	mar 17/01/23		Sergio Arroni del Riego
Análisis de la arquitectura	3 días	mié 18/01/23	vie 20/01/23		Sergio Arroni del Riego
Análisis del estado del arte	13 días	lun 23/01/23	mié 08/02/23		Sergio Arroni del Riego
« Diseño	15 días	jue 09/02/23	mié 01/03/23	11;12;13;14	Sergio Arroni del Riego
Diseño arquitectura	5 días	jue 09/02/23	mié 15/02/23		Sergio Arroni del Riego
Diseño interfaz de la monito	5 días	jue 16/02/23	mié 22/02/23		Sergio Arroni del Riego
Diseño red neuronal	5 días	jue 23/02/23	mié 01/03/23		Sergio Arroni del Riego
« Implementación	33 días	jue 02/03/23	lun 17/04/23	15;16;17;18;19;20	Sergio Arroni del Riego
Arquitectura	2 días	jue 02/03/23	vie 03/03/23		Sergio Arroni del Riego
Interfaz de la monitorización	1 día	lun 06/03/23	lun 06/03/23		Sergio Arroni del Riego
Red neuronal	20 días	mar 07/03/23	lun 03/04/23		Sergio Arroni del Riego
Integración de todos los com	10 días	mar 04/04/23	lun 17/04/23		Sergio Arroni del Riego
« Pruebas	14 días	mar 18/04/23	vie 05/05/23	20;21;22;23	Sergio Arroni del Riego
Diseño	1 día	mar 18/04/23	mar 18/04/23		Sergio Arroni del Riego
Desarrollo	10 días	mié 19/04/23	mar 02/05/23		Sergio Arroni del Riego
Accesibilidad	1 día	mié 03/05/23	mié 03/05/23		Sergio Arroni del Riego
Usabilidad	1 día	jue 04/05/23	jue 04/05/23		Sergio Arroni del Riego
Usuarios	1 día	vie 05/05/23	vie 05/05/23		Sergio Arroni del Riego
« Despliegue y demostración	3 días	mié 17/05/23	lun 22/05/23	24;25	Sergio Arroni del Riego

Ilustración 20. Planificación de las tareas de desarrollo.

Nombre de tarea	Duración	Comienzo	Fin	Predecesoras	Nombres de los recursos
« Proyecto	114 días	mar 03/01/23	vie 09/06/23		Sergio Arroni del Riego
« Reuniones	112,13 días	lun 02/01/23	mié 07/06/23		Sergio Arroni del Riego
« Desarrollo	88 días	mié 04/01/23	vie 05/05/23	3	Sergio Arroni del Riego
« Despliegue y demostración	3 días	mié 17/05/23	lun 22/05/23	24;25	Sergio Arroni del Riego
« Documentación	114 días	mar 03/01/23	vie 09/06/23	25;26;27;28;7;4;5	Sergio Arroni del Riego
Formato del documento	2 días	mié 10/05/23	jue 11/05/23		Sergio Arroni del Riego
Capítulo 1	2 días	vie 12/05/23	lun 15/05/23		Sergio Arroni del Riego
Capítulo 2	1 día	lun 15/05/23	lun 15/05/23		Sergio Arroni del Riego
Capítulo 3	2 días	mar 16/05/23	mié 17/05/23		Sergio Arroni del Riego
Capítulo 4	1 día	mar 03/01/23	mar 03/01/23		Sergio Arroni del Riego
Capítulo 5	2 días	lun 22/05/23	mar 23/05/23		Sergio Arroni del Riego
Capítulo 6	2 días	mar 23/05/23	mié 24/05/23		Sergio Arroni del Riego
Capítulo 7	2 días	jue 25/05/23	vie 26/05/23		Sergio Arroni del Riego
Capítulo 8	2 días	lun 29/05/23	mar 30/05/23		Sergio Arroni del Riego
Capítulo 9	1 día	mié 31/05/23	mié 31/05/23		Sergio Arroni del Riego
Capítulo 10	1 día	jue 01/06/23	jue 01/06/23		Sergio Arroni del Riego
Capítulo 11	1 día	vie 02/06/23	vie 02/06/23		Sergio Arroni del Riego
Capítulo 12	2 días	lun 05/06/23	mar 06/06/23		Sergio Arroni del Riego
Capítulo 13	2 días	mié 07/06/23	jue 08/06/23		Sergio Arroni del Riego
Apéndices	1 día	vie 09/06/23	vie 09/06/23		Sergio Arroni del Riego

Ilustración 21. Planificación de las tareas de documentación.

Apolo: Sistema inteligente de monitorización y detección de ataques basados en inteligencia artificial

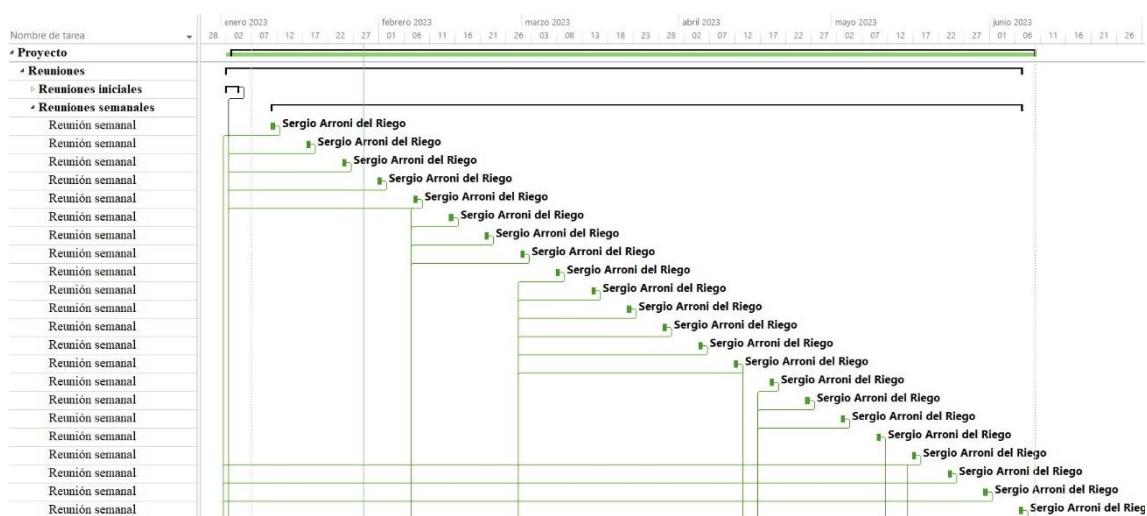


Ilustración 22. Diagrama de Gantt de las reuniones semanales.

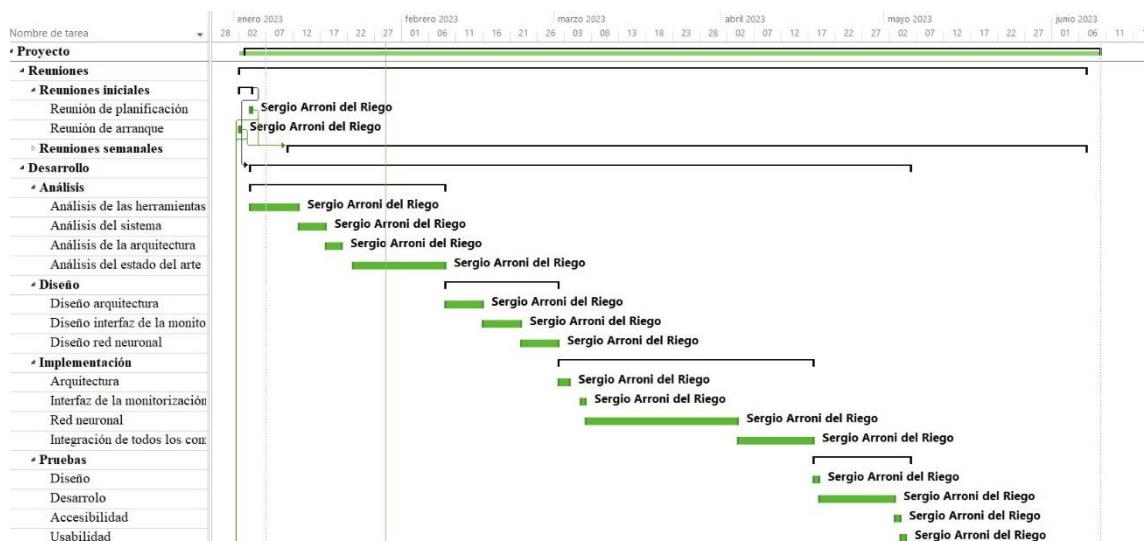


Ilustración 23. Diagrama de Gantt del desarrollo.

Apolo: Sistema inteligente de monitorización y detección de ataques basados en inteligencia artificial

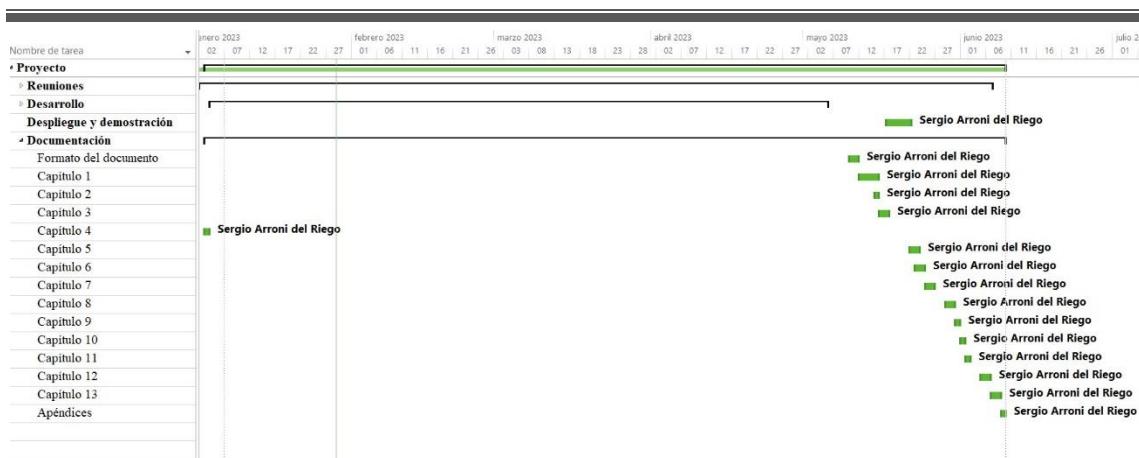


Ilustración 24. Diagrama de Gantt de la documentación.

4.2. Presupuesto Inicial

En esta sección daremos un presupuesto inicial para el Proyecto, para ello se ha seguido la guía del PMBOK [72], la Norma UNE 157801 y por consiguiente la Norma UNE 157001.

4.2.1. Definición de la Empresa

Vamos a empezar definiendo la empresa y sus gastos generales anuales. Estos gastos deben ser tenidos en cuenta para no incurrir en pérdidas y que al final de cada año fiscal se pueda tener un gran margen de beneficio, es por eso por lo que son de vital importancia en la realización del presupuesto.

4.2.1.1. Personal

Para la realización de este Proyecto contamos con una persona trabajando el recurso “Sergio Arroni del Riego”, Tabla 8. Recibe un sueldo bruto anual resultado de multiplicar las 1.776 horas de trabajo laborables al año por el sueldo bruto/hora que tiene asignado. No obstante, a ese sueldo bruto anual hay que añadirle el 25% extra en concepto de contingencias comunes para obtener la cantidad total real que le cuesta a la empresa el poder pagarle a su empleado dicho sueldo.

Por lo que para una empresa el empleado Sergio Arroni del Riego costaría 16,25 € / horas, lo que da un total de 28.860,00 € anualmente, Tabla 9.

Personal	Cantidad
Sergio Arroni del Riego	1

Tabla 8. Personal

Personal	Sueldo bruto hora	Sueldo bruto año	Coste salario hora	Coste salario año
Sergio Arroni del Riego	13,00 €	23.088,00 €	16,25 €	28.860,00 €

Tabla 9. Precio / hora personal

4.2.1.1.1. Productividad del Personal

En general, como empleado, “Sergio Arroni del Riego” no podría mantener un nivel de productividad al 100% durante su jornada laboral diaria de 3 horas. Es probable que solo dedique estrictamente 2 horas y cuarto a generar valor al proyecto, lo que significa que los otros 45 minutos serían pagados, pero no generaría beneficios para la empresa. Por lo tanto, al vender nuestros servicios, debemos tener en cuenta el costo indirecto que representa para nosotros cada empleado que no es productivo al 100%, Tabla 10.

Para calcular esto, se estima un porcentaje de productividad, y se aplica ese porcentaje a su salario bruto anual. De esta manera, se obtiene el coste directo, que representa el coste real del servicio que presta cada empleado. Por otro lado, la diferencia entre el porcentaje de productividad y el 100% representa el porcentaje de coste indirecto que, aplicado al salario bruto anual, resultaría en el coste total indirecto de cada empleado. La Tabla 11 muestra los diferentes costes asociados con cada empleado.

Personal	Total	Prod (%)	Coste Directo	CI (%)	Coste Indirecto
Sergio Arroni del Riego	28.860,00 €	75,00%	21.645,00 €	25%	7.215,00 €

Tabla 10. Productividad personal

Personal	Productividad (%)	Horas / Año	Horas productivas / Año (por persona)	Horas productivas (Total Empresa)
Sergio Arroni del Riego	75,00%	1776	1332,00	1332,00

Tabla 11. Horas total productivas personal

4.2.1.2. Costes Indirectos

Por otro lado, la empresa cuenta con diferentes gastos generales al mes, en diferentes conceptos, tales como limpieza, agua, etc.; todo ello supone un conjunto de costes indirectos que deben ser tenidos en cuenta a la hora de generar el presupuesto final, ya que suponen un incremento de las necesidades de facturación anual de la empresa. Estos costes se pueden ver en la Tabla 12.

Servicios	Costes (Mes)	Costes (Año)	Proyecto
Limpieza	799,98 €	9.599,76 €	
Tributos y tasa diversas	1.400,00 €	16.800,00 €	
Alquiler o arrendamientos de inmuebles	2.500,00 €	30.000,00 €	
Gastos de mantenimiento, reparación y conservación	600,00 €	7.200,00 €	
Pequeño utillaje y herramientas	100,12 €	1.201,44 €	
Consumo de electricidad (excepto consumos para producción)	499,98 €	5.999,76 €	
Consumo de combustibles para calefacción	199,98 €	2.399,76 €	
Consumo de agua	150,00 €	1.800,00 €	
Primas de seguros	1.750,02 €	21.000,24 €	
Portes y gastos de transporte	100,02 €	1.200,24 €	
Gastos en comunicaciones	850,00 €	10.200,00 €	
Gastos financieros	1.000,02 €	12.000,24 €	
Dotación amortización del inmovilizado	1.350,00 €	16.200,00 €	
Internet	50,00 €	600,00 €	
TOTAL:	11.300,12 €	135.601,44 €	

Tabla 12. CI generales

4.2.1.3. Amortizaciones

Para la elaboración del proyecto que vamos a abordar, se necesitan diferentes componentes, hay unos que se pueden amortizar a largo plazo en varios años y otros que se tienen que pagar como alquiler al año.

La empresa cuenta con un equipo de desarrollo, además de un portátil (para facilitar el trabajo en remoto). Estas prestaciones van a ser amortizadas en un plazo de 4 años, por lo que no generan un gasto importante en el presente. También contamos con un servidor CPD. Por otro lado, se cuenta con 3 licencias de desarrollo.

Pese al coste total de 23.600€ que supone disponer de estos medios, en el año en curso solo se genera un coste de 9.650,00€, gracias a que hay materiales que se amortizarán en 4 años.

A continuación, se puede ver en la Tabla 13 un resumen de estas amortizaciones.

Equipo / Licencia	Unidades	Precio unidad	Coste (Total)	Coste (Año)	Tipo	Plazo
Equipo de desarrollo	1	1.100,00 €	1.100,00 €	275,00 €	Amortización	4
Portátiles	1	1.000,00 €	1.000,00 €	250,00 €	Amortización	4
Licencias de desarrollo	2	250,00 €	500,00 €	500,00 €	Alquiler	-
Servidor CPD	1	3.000,00 €	3.000,00 €	750,00 €	Amortización	4
TOTAL:			23.600,00 €	9.650,00 €		

Tabla 13. Amortizaciones.

4.2.1.4. Precio Por Hora del Personal

A el coste de la Tabla 11 hay que añadirle los costes indirectos que le cuesta a la empresa mantener a ese recurso, el beneficio que esperamos obtener, así como el margen que nos queremos guardar para posibles contingencias. Este nuevo precio por hora, que se puede ver en la Tabla 14 y en la Tabla 15, sería el precio que habría que pasarle al cliente en caso de que nos pida el costo de nuestro recurso. Para la realización de las partidas se utilizará el coste por hora de la Tabla 11.

Personal	Precio Hora /	Horas productivas (Total empresa)	Coste beneficios sin	Precio/Hora (Con beneficios del 25%)
Sergio Arroni del Riego	110,00 €	1332,00	146.520,00 €	137,50 €

Tabla 14. Precio / Hora Personal (Parte I)

Personal	Coste con beneficio	Precio / Hora (Con margen de facturación 5%)	Facturación
Sergio Arroni del Riego	183.150,00 €	144,38 €	192.307,50 €

Tabla 15. Precio / Hora Personal (Parte II)

4.2.1.5. Total

A continuación, vamos a sacar el total de costes indirectos y costes anuales de la empresa.

4.2.1.5.1. Total Costes Indirectos

Primero vamos a calcular el coste indirecto de la empresa, para ello vamos a sumar los Costes *Indirectos (CI)* generales, las amortizaciones y el coste de la productividad del personal, se puede observar estos costes en la Tabla 16.

Motivo	Importe
CI generales	135.601,44 €
Amortizaciones	1.775,00 €
Personal	7.215,00 €
TOTAL CI	144.591,44 €

Tabla 16. Total CI

4.2.1.5.2. Total Anual

Ahora vamos a calcular los costes anuales de la empresa, para ello vamos a añadir un beneficio deseado del 25% y un margen de facturación del 5%. Al final nos sale un total de 218.185.33€ de costes indirectos anuales, se puede ver en la Tabla 17.

Estos gastos son el importe de dinero que tiene que generar la empresa este año para poder obtener beneficios, estos gastos son independientes a los proyectos que se realicen dentro de la empresa y por ende al de este documento.

Gracias a estos costes, podemos saber cuál es el precio / hora, del personal, el cual tenemos que utilizar a la hora de hacer los costes directos para poder contrarrestar sus costes indirectos y cuál es el precio / hora que necesitamos para poder contar con el beneficio y el margen y hemos de pasar al cliente en caso de que nos lo pida.

CONCEPTO	IMPORTE
Total CD	21.645,00 €
Total CI	144.591,44 €
Suma de los costes	166.236,44 €
Beneficio deseado (25%)	41.559,11 €
Coste Total	207.795,55 €
Porcentaje del margen entre Facturación y Coste Total	5,00%
Margen entre Facturación y Coste Total	10.389,78 €
Facturación	218.185,33 €

Tabla 17. Total Anual

4.2.2. Costes Directos

A continuación, vamos con el proyecto per se, en esta sección daremos los costes directos del proyecto **Apolo**. Está dividido en diferentes partidas que representan las partes de alto nivel de la planificación y estas a su vez están divididas en tareas, toda esta información esta extraída de la planificación.

4.2.2.1. Partida 1

La primera partida hace referencia a las Reuniones, en ella encontramos dos grupos, las iniciales y las semanales, estas se realizan los martes de cada semana. Nos da un total de 2.750,00 € en total, se puede ver en la Tabla 18 y en la Tabla 19.

Reuniones					
<i>I1</i>	<i>I2</i>	<i>I3</i>	<i>Descripción</i>	<i>Cantidad</i>	<i>Unidades</i>
01			<i>Reunión inicial</i>		
	001		Reunión inicial		
		01	Sergio Arroni del Riego	2	horas
01			<i>Reuniones semanales</i>		
	001		Reunión semanal		
		01	Sergio Arroni del Riego	23	horas

Tabla 18. Partida 1 (Parte I)

Reuniones						
<i>I1</i>	<i>I2</i>	<i>I3</i>	<i>Precio</i>	<i>Subtotal (3)</i>	<i>Subtotal (2)</i>	<i>Total</i>
01						220,00 €
	001				220,00 €	
		01	110,00 €	220,00 €		
01						2.530,00 €
	001				2.530,00 €	
		01	110,00 €	2.530,00 €		
TOTAL						2.750,00 €

Tabla 19. Partida 1 (Parte II)

4.2.2.2. Partida 2

La segunda partida hace referencia al Desarrollo del Proyecto, esta es la que más tareas engloba y la que más horas lleva, está dividida en Análisis, Diseño,

Apolo: Sistema inteligente de monitorización y detección de ataques basados en inteligencia artificial

Implementación, Pruebas y Despliegue, lo que da una suma total de 20.020,00 €, se puede ver en la Tabla 20 y en la Tabla 21.

Desarrollo					
I1	I2	I3	Descripción	Cantidad	Unidades
01			Análisis		
	001		Análisis de las herramientas y tecnologías		
		001	Sergio Arroni del Riego	14	horas
	002		Análisis del sistema		
		001	Sergio Arroni del Riego	6	horas
	003		Análisis de la arquitectura		
		001	Sergio Arroni del Riego	6	horas
	004		Análisis del estado del arte		
		001	Sergio Arroni del Riego	26	horas
02			Diseño		
	001		Diseño arquitectura		
		001	Sergio Arroni del Riego	10	horas
	002		Diseño interfaz de la monitorización		
		001	Sergio Arroni del Riego	10	horas
	003		Diseño red neuronal		
		001	Sergio Arroni del Riego	10	horas
02			Implementación		
	001		Arquitectura		
		001	Sergio Arroni del Riego	4	horas
	002		Interfaz de la monitorización		
		001	Sergio Arroni del Riego	2	horas
	003		Red neuronal		
		001	Sergio Arroni del Riego	40	horas
	004		Integración de todos los componentes		
		001	Sergio Arroni del Riego	20	horas
01			Pruebas		
	001		Diseño		
		001	Sergio Arroni del Riego	2	horas
	002		Desarrollo		
		001	Sergio Arroni del Riego	20	horas
	003		Accesibilidad		

		001	Sergio Arroni del Riego	2	horas
	004		Usabilidad		
		001	Sergio Arroni del Riego	2	horas
	005		Usuarios		
		001	Sergio Arroni del Riego	2	horas
01			Despliegue demostración y		
	001		Despliegue demostración y		
		001	Sergio Arroni del Riego	6	horas

Tabla 20. Partida 2 (Parte I)

Desarrollo						
I1	I2	I3	Precio	Subtotal (3)	Subtotal (2)	Total
01						5.720,00 €
	001				1.540,00 €	
		001	110,00 €	1.540,00 €		
	002				660,00 €	
		001	110,00 €	660,00 €		
	003				660,00 €	
		001	110,00 €	660,00 €		
	004				2.860,00 €	
		001	110,00 €	2.860,00 €		
02						3.300,00 €
	001				1.100,00 €	
		001	110,00 €	1.100,00 €		
	002				1.100,00 €	
		001	110,00 €	1.100,00 €		
	003				1.100,00 €	
		001	110,00 €	1.100,00 €		
02						7.260,00 €
	001				440,00 €	
		001	110,00 €	440,00 €		
	002				220,00 €	
		001	110,00 €	220,00 €		
	003				4.400,00 €	
		001	110,00 €	4.400,00 €		
	004				2.200,00 €	

		001	110,00 €	2.200,00 €		
01						3.080,00 €
	001				220,00 €	
		001	110,00 €	220,00 €		
	002				2.200,00 €	
		001	110,00 €	2.200,00 €		
	003				220,00 €	
		001	110,00 €	220,00 €		
	004				220,00 €	
		001	110,00 €	220,00 €		
	005				220,00 €	
		001	110,00 €	220,00 €		
01						660,00 €
	001				660,00 €	
		001	110,00 €	660,00 €		
					TOTAL	20.020,00 €

Tabla 21. Partida 2 (Parte II)

4.2.2.3. Partida 3

Por último, tenemos la partida dedicada a la Documentación del Proyecto. Esta parte lleva poco tiempo y por ende poco coste, con un total de 1.540,00 €, se puede ver en la Tabla 22 y en la Tabla 23.

Documentación					
I1	I2	I3	Descripción	Cantidad	Unidades
01			Documentación		
	001		Formato del documento		
		001	Sergio Arroni del Riego	4	horas
	002		Capítulo 1		
		001	Sergio Arroni del Riego	4	horas
	003		Capítulo 2		
		001	Sergio Arroni del Riego	2	horas
	004		Capítulo 3		
		001	Sergio Arroni del Riego	4	horas
	005		Capítulo 4		
		001	Sergio Arroni del Riego	2	horas
	006		Capítulo 5		
		001	Sergio Arroni del Riego	4	horas
	007		Capítulo 6		
		001	Sergio Arroni del Riego	4	horas
	008		Capítulo 7		
		001	Sergio Arroni del Riego	4	horas
	009		Capítulo 8		
		001	Sergio Arroni del Riego	4	horas
	010		Capítulo 9		
		001	Sergio Arroni del Riego	2	horas
	011		Capítulo 10		
		001	Sergio Arroni del Riego	2	horas
	012		Capítulo 11		
		001	Sergio Arroni del Riego	2	horas
	013		Capítulo 12		
		001	Sergio Arroni del Riego	4	horas
	014		Capítulo 13		
		001	Sergio Arroni del Riego	4	horas
	015		Apéndices		
		001	Sergio Arroni del Riego	2	horas

Tabla 22. Partida 3 (Parte I)

Documentación						
I1	I2	I3	Precio	Subtotal (3)	Subtotal (2)	Total
01						1.540,00 €
	001				440,00 €	
		001	110,00 €	440,00 €		
	002				440,00 €	
		001	110,00 €	440,00 €		
	003				220,00 €	
		001	110,00 €	220,00 €		
	004				440,00 €	
		001	110,00 €	440,00 €		
	005				220,00 €	
		001	110,00 €	220,00 €		
	006				440,00 €	
		001	110,00 €	440,00 €		
	007				440,00 €	
		001	110,00 €	440,00 €		
	008				440,00 €	
		001	110,00 €	440,00 €		
	009				440,00 €	
		001	110,00 €	440,00 €		
	010				220,00 €	
		001	110,00 €	220,00 €		
	011				220,00 €	
		001	110,00 €	220,00 €		
	012				220,00 €	
		001	110,00 €	220,00 €		
	013				440,00 €	
		001	110,00 €	440,00 €		
	014				440,00 €	
		001	110,00 €	440,00 €		
	015				220,00 €	
		001	110,00 €	220,00 €		
TOTAL						1.540,00 €

Tabla 23. Partida 3 (Parte II)

4.2.3. Resúmenes

En esta sección daremos los resúmenes del presupuesto, tanto la versión para nuestra empresa (antes de beneficios y margen), como la versión para el cliente (con los beneficios y margen ya incluidos).

4.2.3.1. Costes (Empresa)

Empezamos con el resumen de la empresa, daremos una versión agregada (Tabla 24) y una detallada (Tabla 25), estas se diferencian en que la versión agregada solo tiene las tareas de más alto nivel, mientras que la detallada desarrolla un nivel más de tareas.

Hemos tenido en cuenta un beneficio del 25% y un margen del 5%, dicho margen es tanto para hacer frente a los más que seguros inconvenientes que ocurran, como para tener un margen de maniobra en caso de que ocurra algún contratiempo.

Nos da un total de 31.906,88€ a facturar durante el proyecto.

Presupuesto de costes (Agregado)	
Reuniones	2.750,00 €
Desarrollo	20.020,00 €
Documentación	1.540,00 €
Coste sin beneficio	24.310,00 €
Margen de beneficio	25%
Beneficio	6.077,50 €
Coste con beneficio	30.387,50 €
Margen de facturación	5%
Margen	1.519,38 €
Total a Facturar	31.906,88 €

Tabla 24. Presupuesto Empresa Agregado

Presupuesto de costes (Detallado)	
Reuniones	2.750,00 €
Desarrollo	
Análisis	5.720,00 €
Diseño	3.300,00 €
Implementación	7.260,00 €
Pruebas	3.080,00 €
Despliegue y demostración	660,00 €
Documentación	1.540,00 €
Coste sin beneficio	24.310,00 €
Margen de beneficio	25%
Beneficio	6.077,50 €
Coste con beneficio	30.387,50 €
Margen de facturación	5%
Margin	1.519,38 €
Total a Facturar	31.906,88 €

Tabla 25. Presupuesto Empresa Detallado

4.2.3.2. Costes (Cliente)

Para finalizar, tenemos el resumen del cliente, para la cual también daremos una versión agregada (Tabla 26) y una detallada (Tabla 27), se diferencian en que la versión agregada solo tiene las tareas de más alto nivel, mientras que la detallada desarrolla un nivel más de tareas. El proyecto tiene un 21% de IVA.

Nos da un total de 38.608,38€ a facturar durante el Proyecto.

Resumen del Presupuesto del Cliente	
Reuniones	3.609,38 €
Desarrollo	26.276,25 €
Documentación	2.021,25 €
Presupuesto estimado antes de impuestos	31.906,88 €
Porcentaje de IVA aplicado	21%
IVA	6.700,44 €
Presupuesto estimado después de impuestos	38.607,32 €

Tabla 26. Presupuesto Cliente Agregado

Presupuesto del Cliente (Detallado)	
Reuniones	3.609,38 €
Desarrollo	
Análisis	7.507,50 €
Diseño	4.331,25 €
Implementación	9.528,75 €
Pruebas	4.042,50 €
Despliegue y demostración	866,25 €
Documentación	2.021,25 €
Presupuesto estimado antes de impuestos	31.906,88 €
Porcentaje de IVA aplicado	21%
IVA	6.700,44 €
Presupuesto estimado después de impuestos	38.607,32 €

Tabla 27. Presupuesto Cliente Detallado

Capítulo 5. Análisis

Para esta sección nos hemos querido enfocar en **Apolo** y en su interacción con el resto de la arquitectura, para ello se detallarán los requisitos del sistema, el diagrama de contexto, así como sus casos de uso y escenarios derivados y los componentes de la arquitectura.

5.1. Requisitos del Sistema

En esta sección se recogerán tanto los requisitos funcionales como los no funcionales. Para ello se va a seguir como referencia el capítulo 3 del estándar IEE 830-1998 "3. Specific requirements"¹².

5.1.1. Interfaces Externas

En esta sección se tratarán requisitos referentes a las diferentes interfaces del sistema: usuario, hardware, software y comunicaciones.

5.1.1.1. Interfaz de Usuario

RIUsuario-1. El Sistema será accesible a los usuarios mediante los navegadores de LISTA_NAV.

RIUsuario-1.1. El valor inicial de LISTA_NAV será de: Google, Firefox, Safari, Opera, Internet Explorer.

RIUsuario-1.2. El valor de LISTA_NAV podrá ser configurado por el administrador.

5.1.1.2. Interfaces de Hardware

RIGHardware-1. El sistema deberá poder ser navegado usando diferentes medios.

RIGHardware-1.1. Usando un ratón.

RIGHardware-1.2. Usando un teclado.

RIGHardware-2. El sistema deberá de poder desplegarse siguiendo una estructura basada en microservicios.

RIGHardware-2.1. Varios microservicios para **Apolo**.

RIGHardware-2.1.1. Un microservicio para el IDS.

RIGHardware-2.1.2. Un microservicio para la base de datos de series temporales.

RIGHardware-2.1.3. Un microservicio para el sistema de colas.

RIGHardware-2.2. Dos microservicios para la web simulada.

RIGHardware-2.2.1. La WebApp.

RIGHardware-2.2.2. La REST API.

¹² <https://s3-us-west-2.amazonaws.com/ieeeshutpages/xplore/xplore-shut-page.html>

- RIHardware-2.3. Un microservicio para proxy reverso y el *firewall* de red.
- RIHardware-2.4. Varios microservicios para el sistema de monitorización de *logs*.
 - RIHardware-2.4.1. Un microservicio para la recopilación de peticiones de tráfico red.
 - RIHardware-2.4.2. Un microservicio para la agrupación de las peticiones e ingesta en la base de datos de *logs*.
 - RIHardware-2.4.3. Un microservicio con la base de datos de *logs*.
 - RIHardware-2.4.4. Una interfaz gráfica para el monitoreo de los *logs*.

5.1.1.3. Interfaces de Software

- RISoftware-1. El sistema estará disponible en varios idiomas.
 - RISoftware-1.1. Español.
 - RISoftware-1.2. Inglés.

5.1.1.4. Interfaces de Comunicaciones

- RIComunicaciones-1. El sistema se deberá comunicar con el proveedor de *hosting*.
- RIComunicaciones-2. El sistema se deberá comunicar con el proveedor de *dominio*.
- RIComunicaciones-3. El sistema deberá realizar todas las comunicaciones seguras utilizando el protocolo *HTTPS*.

5.1.2. Requisitos Funcionales

- RF-IDS-1. El sistema permitirá al usuario no identificado, identificarse en el sistema.
 - RF-IDS-1.1. El sistema solicitará al usuario no identificado una serie de campos.
 - RF-IDS-1.1.1. Un identificador único.
 - RF-IDS-1.1.1.1. Este campo es obligatorio.
 - RF-IDS-1.1.2. Una contraseña.
 - RF-IDS-1.1.2.1. Este campo es obligatorio.
 - RF-IDS-1.2. El sistema comprobará que dicho usuario este registrado en la base de datos.
 - RF-IDS-1.2.1. En el caso en que no lo este, dará un fallo.

- RF-IDS-1.2.1.1. Enviará una notificación al administrador del sistema.
- RF-IDS-1.2.1.2. Denegará el acceso al usuario no identificado.
- RF-IDS-1.2.2. En el caso de que lo este, permitirá el correcto uso del sistema al usuario.
- RF-IDS-1.3. El sistema guardará todos los intentos de identificación quedaran guardados en un historial.
- RF-IDS-1.3.1. El sistema deberá de dejar acceder al administrador al historial de intentos de identificación.
- RF-IDS-2. El sistema deberá poder clasificar una petición.
- RF-IDS-2.1. El sistema elegirá, en base a como es la petición, que conjunto de modelos, serán usados para predecir dicha petición.
- RF-IDS-2.1.1. El conjunto de modelos escogidos podrá ir desde 1 hasta el máximo de modelos usados en el entrenamiento.
- RF-IDS-2.1.1.1. El administrador podrá cambiar el número de modelos usados.
- RF-IDS-2.1.2. Cada modelo elegido por el sistema deberá de identificar el tipo de la petición dada.
- RF-IDS-2.1.2.1. Para identificar la petición, darán una puntuación de 0 a 100.
- RF-IDS-2.1.2.1.1. Siendo 0 la petición más benigna posible.
- RF-IDS-2.1.2.1.2. Siendo 100 la petición más maligna posible.
- RF-IDS-2.2. El sistema elegirá, mediante un componente de incertidumbre, que conjunto de modelos, será el elegido para la predicción de la petición.
- RF-IDS-2.2.1. El número de modelos usados en el conjunto podrá ser modificado por el administrador.
- RF-IDS-2.2.2. Inicialmente es 1 un único modelo.
- RF-IDS-2.3. Si durante la predicción de la petición se generase un fallo por cualquier motivo.
- RF-IDS-2.3.1. El sistema dejaría de predecir peticiones.
- RF-IDS-2.3.2. Se notificaría al administrador mediante una notificación.
- RF-IDS-2.4. Si durante la predicción todo transcurrió sin ningún fallo.
- RF-IDS-2.4.1. Si la petición es benigna, se sigue permitiendo al usuario acceder a la página.
- RF-IDS-2.4.2. Si la petición es maligna, se envía una notificación de urgencia al administrador ver RF-IDS-4.
- RF-IDS-2.5. El sistema almacenará todas las peticiones, ordenadas en orden de llegada, con la puntuación predicha por el sistema, en un historial.
- RF-IDS-2.5.1. El sistema deberá de dejar acceder al administrador al historial de puntuaciones.
- RF-IDS-3. El sistema deberá poder administrar peticiones que se produzcan de manera simultánea.

RF-IDS-3.1. El sistema deberá de almacenar las peticiones en orden de llegada.

RF-IDS-3.1.1. El sistema irá sacando las peticiones en orden de llegada.

RF-IDS-3.1.1.1. El sistema guardará cada petición en el *Log stack*, que guardará un historial de todas las peticiones realizadas.

RF-IDS-3.1.1.2. El sistema identificará cada petición (RF-IDS-2).

RF-IDS-3.2. El sistema deberá poder mostrar al administrador el historial de peticiones.

RF-IDS-3.3. El sistema dejará al administrador crear elementos que muestren las peticiones en un *dashboard*.

RF-IDS-3.3.1. Elementos como tablas.

RF-IDS-3.3.2. Elementos como graficas.

RF-IDS-3.4. Dichos elementos podrán ser gestionados por el administrador.

RF-IDS-3.4.1. Crearlos.

RF-IDS-3.4.1.1. El sistema solicitará al administrador que parámetros de las peticiones quiere mostrar en el elemento.

RF-IDS-3.4.2. Modificarlos.

RF-IDS-3.4.2.1. El sistema le solicitará al administrador que parámetros de las peticiones quiere modificar del elemento.

RF-IDS-3.4.3. Eliminarlos.

RF-IDS-3.4.3.1. El sistema solicitará al administrador que parámetros de las peticiones quiere eliminar en el elemento.

RF-IDS-3.4.3.2. El sistema dará la opción de eliminar completamente el elemento.

RF-IDS-4. El sistema deberá contar con un sistema de alertas.

RF-IDS-4.1. Dicho sistema deberá de actuar inmediatamente después de que ocurra una incidencia.

RF-IDS-4.2. El sistema enviará un mensaje al administrador.

RF-IDS-4.2.1. El sistema enviará el mensaje que contenga la información mínima para que el administrador sepa que hay una incidencia.

RF-IDS-4.2.2. Fecha de la incidencia.

RF-IDS-4.2.3. Hora de la incidencia.

RF-IDS-4.2.4. Predicción de la puntuación dada por el modelo.

RF-IDS-4.2.5. IP del equipo del atacante.

RF-IDS-4.2.6. Petición realizada por el atacante.

RF-IDS-4.3. El sistema deberá de guardar en un historial todas las alertas mandadas.

RF-IDS-4.3.1. El sistema deberá de permitir al administrador acceder a este histórico.

RF-IDS-5. El sistema deberá poder integrarse en una arquitectura de defensa en profundidad.

5.1.3. Requisitos de rendimiento

- RRend-1 El sistema deberá responder a la petición del cliente mostrando la web en menos de T_MAX_RESPUESTA.
- RRend-1.1 T_MAX_RESPUESTA será inicialmente de 2 segundos.
- RRend-1.2 Este valor puede ser modificado por el administrador.
- RRend-1.3 En caso de no poder ofrecer la respuesta en ese tiempo, se genera un *Log stack* que contendrá los siguientes puntos:
- RRend-1.3.1 Fecha del fallo.
 - RRend-1.3.2 Dirección web a la que se intentó acceder.
 - RRend-1.3.3 IP del usuario.
 - RRend-1.3.4 Motivo del fallo.
- RRend-1.4 El sistema deberá permitir que el *Log stack* pueda ser accedido por el administrador.
- RRend-1.5 El sistema intentará seguir cargando la página durante T_CARGA_RESPUESTA.
- RRend-1.5.1 T_CARGA_RESPUESTA será inicialmente de 7 segundos.
- RRend-1.5.2 Este valor puede ser modificado por el administrador.
- RRend-1.5.3 En caso de no poder cargar la página en T_CARGA_RESPUESTA, se mostrará un mensaje de error.

5.1.4. Lógicos de BD

N/A.

5.1.5. Restricciones de Diseño

- RD-1 El sistema deberá de guardar las peticiones en orden de llegada, para ello usará una pila FIFO.

5.1.6. Atributos del Sistema

5.1.6.1. Seguridad

- At-Sist-Sec-1 Los datos personales de los usuarios deberán estar cifrados mediante SHA-256.

5.1.6.2. Mantenimiento

- At-Sist-Mant-1 El sistema podrá ser reiniciado únicamente por el administrador.
- At-Sist-Mant-2 El sistema deberá garantizar la integridad de los datos que maneje.
- At-Sist-Mant-3 El sistema deberá ofrecer opciones de recuperación y reinicio en caso de fallo.
- At-Sist-Mant-4 Para poder actualizar el sistema, **Apolo** tendrá un tiempo al día en el que el sistema no funcionará.
- At-Sist-Mant-4.1 El tiempo será inicialmente de 1 hora.
- At-Sist-Mant-4.2 El tiempo solamente podrá ser configurado por el administrador.

5.1.7. Otros Requisitos

N/A.

5.2. Identificación de Actores del Sistema

Para el sistema se han identificado los siguientes actores:

- **Usuario no identificado:** Este actor representa al usuario del sistema de monitoreo y de anomalías, que aún no está identificado en el sistema, solo podrá acceder a las pantallas de identificación. No se pueden registrar nuevos usuarios, por lo que no hay pantalla de registro.
- **Usuario identificado, o lo que es lo mismo el administrador:** Una vez el usuario no identificado se haya identificado, se convierte en el administrador, tan solo habrá uno en el sistema y no se podrán crear nuevos, este actor tiene permisos para ejecutar todas las acciones citadas en este apartado. Cabe mencionar que no vamos a hacer un gestor de contraseñas puesto que se sale del ámbito de este proyecto, por lo que únicamente habrá una contraseña.
- **Módulo User Experience (UIX) del cliente (Simulador de datos):** Este actor es el sistema formado por una *WebApp* y una *Rest API*, el cual se encarga de recibir todas las peticiones generadas para simular un flujo de datos real.
- **Módulo de gestión de peticiones:** Este actor es el encargado de recibir las peticiones simuladas, tratarlas de tal forma que en los próximos pasos puedan ser utilizadas y mostradas en una *User Experience (UIX)* accedida únicamente por el administrador.
- **Submódulo del sistema de colas:** Se encarga de guardar las peticiones ya tratadas.
- **Submódulo de gestión de anomalías:** En este actor se realizan las acciones de gestión de las anomalías detectadas en la clasificación de las peticiones, se permitirá al administrador acceder a una *UIX* que muestre el resultado de dichas clasificaciones. También, será el encargado de notificar al administrador las posibles anomalías perjudiciales que se encuentre el sistema durante la clasificación.

5.3. Identificación de los Componentes de la Arquitectura del Sistema

En esta sección identificaremos los diferentes componentes de la arquitectura, así como daremos una explicación de cada uno de ellos. Vamos a integrar **Apolo** en una arquitectura de gestión y monitorización de peticiones, siendo **Apolo** el modelo que clasificara las peticiones en malignas o benignas.

5.3.1. Descripción de los Componentes

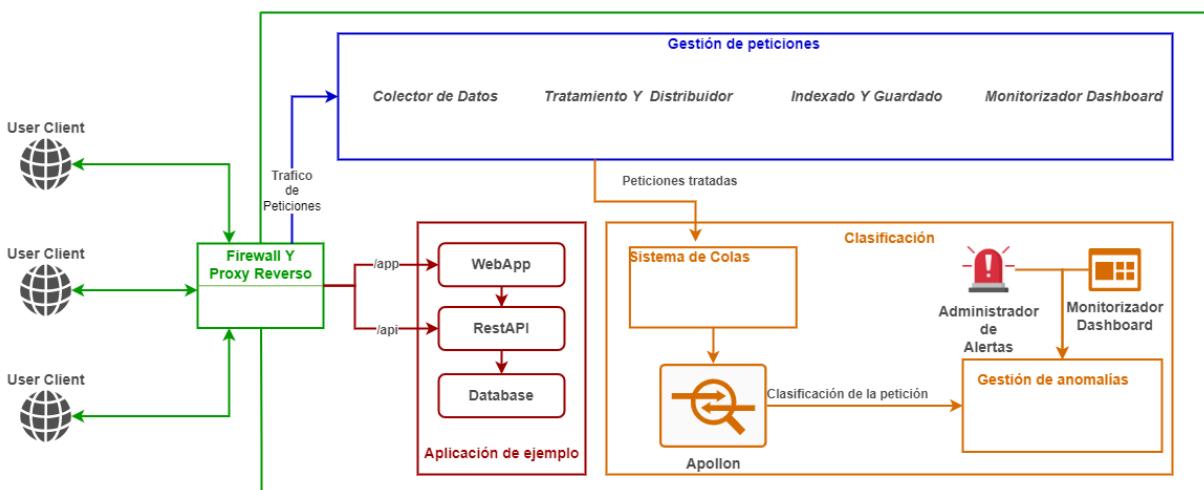


Ilustración 25. Identificación de la Arquitectura del Sistema.

Tal y como vemos en la Ilustración 25, tenemos 4 grandes componentes bien diferenciados, los cuales se pueden descomponer en diferentes partes, tal y como se describe a continuación.

5.3.1.1. Firewall y Proxy Reverso

El *firewall* es un componente crucial dentro del sistema que tiene como objetivo principal salvaguardar la infraestructura de posibles ataques externos. Su función consiste en establecer y aplicar políticas de seguridad para controlar el tráfico de red que ingresa y sale del sistema. El *firewall* analiza los paquetes de datos en función de reglas predefinidas, como direcciones IP, puertos, protocolos y patrones de tráfico, y decide qué conexiones permitir o bloquear. Esto ayuda a prevenir intrusiones no autorizadas, filtrar contenido malicioso y proteger los recursos y datos sensibles del sistema.

Por otro lado, el *proxy* reverso es otro componente importante en términos de seguridad y rendimiento. Actúa como intermediario entre los clientes externos y los servidores internos. En lugar de permitir que los clientes se conecten directamente a los servidores, el *proxy* reverso recibe las solicitudes de los clientes y las reenvía a los servidores correspondientes. Esto brinda varios beneficios, como ocultar la información sobre los servidores internos y proporcionar una capa adicional de protección contra ataques directos.

Además de la seguridad, el *proxy* reverso también mejora el rendimiento del sistema. Al actuar como una caché, puede almacenar en memoria las respuestas a solicitudes frecuentes de los clientes y entregarlas directamente sin tener que acceder a los servidores internos cada vez. Esto reduce la carga en los servidores y acelera la entrega de contenido, mejorando la experiencia del usuario.

5.3.1.2. Aplicación de Ejemplo

Esta aplicación será accedida por los clientes y estos realizarán peticiones tanto a la *WebApp* como a la *Rest API* de la aplicación. Como ya he comentado en otras ocasiones esta aplicación solamente sirve de ejemplo para simular las peticiones al sistema.

Este componente se conforma por 3 partes diferenciadas:

- *WebApp*: Proporciona una interfaz de usuario gráfica.
- *Rest API*: Expone los datos y la lógica de negocio a través de una *API*.
- *Database*: Almacena y gestiona los datos.

5.3.1.3. Gestión de Peticiones

Este componente desempeña un papel fundamental en la recolección, procesamiento y almacenamiento de las solicitudes entrantes en el sistema de IDS. El componente de Gestión de Peticiones se compone de cuatro partes diferenciadas que trabajan de manera conjunta para llevar a cabo estas tareas:

- Colector de Datos: Esta parte del componente se encarga de recopilar los datos entrantes. El colector de datos garantiza que toda la información necesaria para el análisis esté disponible para su posterior procesamiento.

- Tratamiento y Distribuidor: Una vez recopilados los datos, esta parte del componente se encarga de procesarlos y distribuirlos según su destino. Se encarga de enviar los datos al sistema de colas y a la etapa de indexado y guardado.
- Indexado y Guardado: En esta etapa, los datos procesados se almacenan en un gestor de datos. Se utilizan técnicas de indexación para facilitar el acceso y la búsqueda eficiente de la información almacenada. El objetivo es almacenar los datos de manera segura y organizada, asegurando su disponibilidad para su posterior monitorización.
- Monitorización *dashboard*: El componente de Gestión de Peticiones también incluye una interfaz gráfica de usuario, conocida como *dashboard*, que permite supervisar y analizar los datos recopilados y procesados. El *dashboard* proporciona visualizaciones en tiempo real de los eventos de seguridad, estadísticas, gráficos y datos relevantes. Esta interfaz permite al administrador monitorear la actividad de la red y tomar decisiones informadas sobre la respuesta a incidentes.

A través de sus componentes de colecta de datos, tratamiento y distribución, indexado y guardado, así como la monitorización *dashboard*, se logra una gestión eficiente de la información y se proporciona una base sólida para el análisis y la detección de intrusiones en el sistema de *IDS*.

5.3.1.4. Clasificación

Este componente es el más importante de todos ya que es el que contiene al modelo de clasificación **Apolo**.

El componente de clasificación se encarga de guardar las peticiones en el sistema de colas, en función de su orden de llegada, a su vez **Apolo** las va extrayendo, preprocesando y clasificando en los modelos escogidos por *MAB*.

Este componente se conforma por 3 partes: Sistema de Colas, **Apolo** y Gestión de Anomalías.

5.3.1.4.1. Sistema de Colas

El sistema de colas en la arquitectura de **Apolo** desempeña un papel crucial al recibir y encolar de forma continua los paquetes de tráfico de red para su

posterior procesamiento. Cada paquete se añade a la cola en el orden de llegada, asegurando un procesamiento secuencial.

La implementación de un sistema de colas en la arquitectura de **Apolo** ofrece numerosos beneficios. En primer lugar, permite gestionar eficientemente grandes volúmenes de tráfico de red, evitando la pérdida de paquetes y garantizando que todos ellos sean procesados en algún momento.

Además, el sistema de colas es altamente escalable. Esto significa que se puede agregar recursos de procesamiento adicionales, como servidores o nodos de cómputo en la nube, para hacer frente a picos de tráfico o incrementos en la carga de **Apolo**, sin que ello afecte el funcionamiento del servicio. Esto asegura que **Apolo** pueda adaptarse a las demandas cambiantes de tráfico sin interrupciones en el servicio.

5.3.1.4.2. Apolo

Apolo se encarga de clasificar las peticiones que llegan del sistema de colas, para ello las preprocesa y predice en los modelos escogidos por *MAB*.

Apolo está conformado por 3 partes:

- Modelos: Esta parte se encarga de preprocesar las peticiones para que los diferentes clasificadores puedan tratar las peticiones.
 - Preprocesado: Preprocesa las peticiones recibidas.
 - Clasificadores: Son los modelos de *ML* que clasifican las peticiones.
- *MAB*: Gestiona la elección de los modelos que van a realizar la clasificación de las peticiones. Funciona como un sistema de toma de decisiones en tiempo real, seleccionando de forma dinámica los clasificadores más adecuados en función de su rendimiento y eficacia. *MAB* introduce una capa de incertidumbre en el proceso de clasificación, lo cual ayuda a mejorar la defensa frente al *AML*.
- Clúster: Elige qué grupo de clasificadores va a clasificar la petición, en función de la estructura de la petición. Al permitir que diferentes grupos de clasificadores se encarguen de las diferentes peticiones, se aumenta la robustez y adaptabilidad del sistema de clasificación.

En conjunto, estas tres partes de **Apolo** (modelos, *MAB* y *clúster*) trabajan en armonía para procesar, preprocesar y clasificar las peticiones. A través del preprocesamiento, se preparan los datos de entrada para que los clasificadores puedan trabajar de manera efectiva. Los clasificadores, basados en modelos *ML*, aplican algoritmos y técnicas para determinar la categoría de la petición, es decir, si es benigna o maligna.

El *MAB* desempeña un papel crucial al gestionar la selección de los modelos de clasificación. Evalúa y compara el rendimiento de los clasificadores en tiempo real, teniendo en cuenta métricas como la tasa de detección y la precisión. Esto permite adaptar dinámicamente la elección de los clasificadores más efectivos en función de las características del tráfico y mejorar continuamente la precisión del sistema de clasificación.

El *clúster*, por su parte, agrega una capa adicional de incertidumbre al proceso de clasificación al seleccionar el grupo de clasificadores más adecuado para una petición en particular. Esto se basa en la estructura de la petición y puede considerar atributos específicos o información sobre la comunicación. Al asignar grupos especializados en la detección de amenazas específicas, se mejora la capacidad del sistema para identificar y responder a diferentes tipos de intrusiones.

5.3.1.4.3. Gestión de Anomalías

El componente de Gestión de Anomalías se encarga de guardar las clasificaciones en función de la estampa del tiempo en el que se realizaron, mostrarlas en un *dashboard* y notificar en caso de anomalía al administrador.

El componente de Gestión de Anomalías se encarga de gestionar y administrar las clasificaciones realizadas por el modelo de clasificación **Apolo**.

Una de las funciones principales del componente de Gestión de Anomalías es almacenar las clasificaciones realizadas por **Apolo** en función de la estampa de tiempo en la que se llevaron a cabo. Esto permite tener un registro histórico de las clasificaciones y facilita el análisis posterior de los eventos de red.

Además del almacenamiento de clasificaciones, el componente de Gestión de Anomalías también se encarga de presentar estas clasificaciones en un *dashboard*. El *dashboard* proporciona una interfaz gráfica que permite visualizar y analizar la información recopilada. Puede mostrar estadísticas, tendencias, alertas y otra información relevante para el monitoreo y análisis del tráfico de red. Esto permite a los administradores del sistema obtener una visión general y

detallada de la actividad de la red, identificar patrones sospechosos y tomar medidas preventivas o correctivas en consecuencia.

Además de la visualización en el *dashboard*, el componente de Gestión de Anomalías también tiene la capacidad de enviar notificaciones al administrador en caso de detectar una anomalía. La detección de anomalías puede basarse en reglas predefinidas o umbrales establecidos, que indican desviaciones significativas del comportamiento normal de la red. Al recibir estas notificaciones, el administrador puede tomar medidas inmediatas para investigar y responder a la posible intrusión o actividad maligna.

5.4. Diagrama de Contexto

En esta sección realizaremos el diagrama de contexto del sistema (Ilustración 26), lo hemos realizado enfocado en **Apolo**, por lo que el resto de los componentes de la arquitectura interactúan con él. Posteriormente, detallaremos todos los Casos de Uso dentro del sistema, se puede ver en las tablas: Tabla 28, Tabla 29, Tabla 30, Tabla 31, Tabla 32, Tabla 33, Tabla 34, Tabla 35, Tabla 36 y Tabla 37.

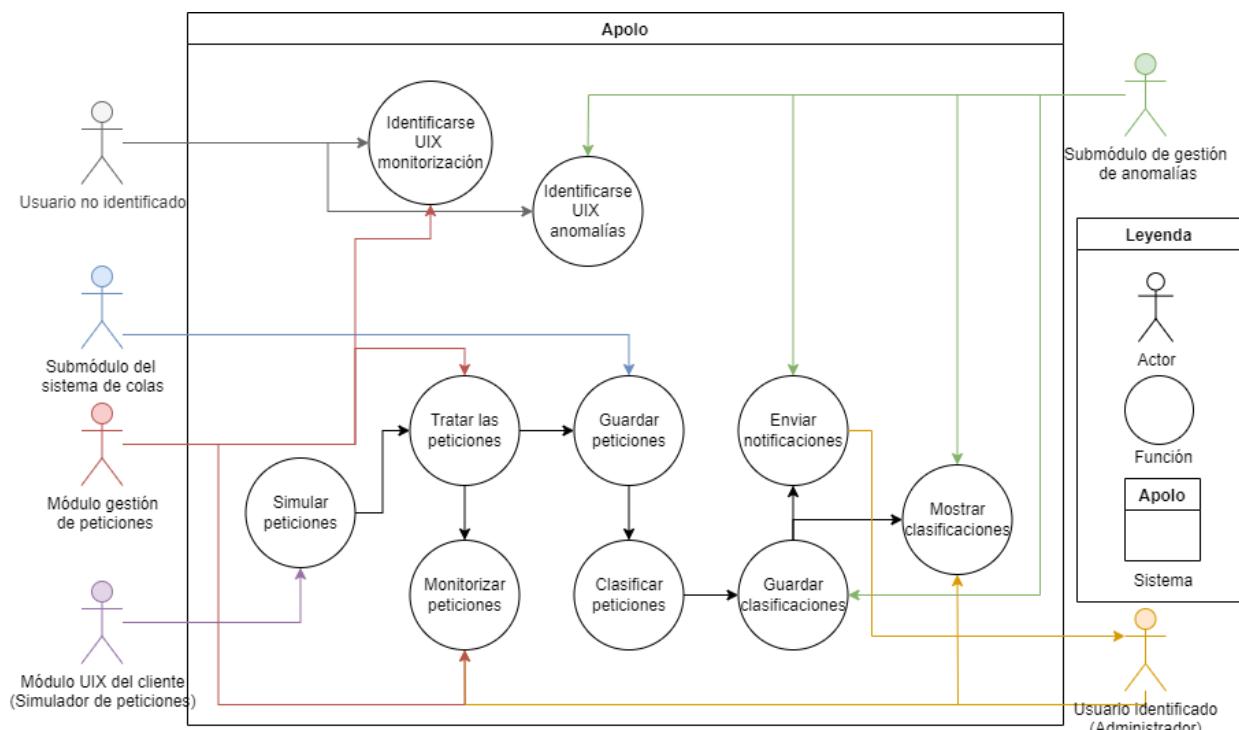


Ilustración 26. Diagrama de Contexto.

5.4.1. Caso de Uso: Identificarse UIX Monitorización

IDENTIFICARSE UIX MONITORIZACIÓN	
IDENTIFICADOR	CU1
PRECONDICIONES	<i>El usuario debe de ser el usuario administrados registrado en módulo de gestión de peticiones y saberse su usuario y contraseña.</i>
POSTCONDICIONES	<i>El usuario no identificado, se convertirá en administrador y tendrá sus permisos asociados.</i>
ACTORES	<i>Usuario no identificado, Módulo de gestión de peticiones.</i>
DESCRIPCIÓN	<i>El usuario no identificado: Accederá a la Interfaz de inicio de sesión del sistema de monitorización. El sistema le solicitará sus credenciales. Si son correctas accederá a la UIX con los permisos de administrador. Si son incorrectas le denegara el paso al sistema.</i>
EXCEPCIONES	<i>El sistema este caído: si el sistema está caído no podrá acceder a las UIX. Notificar un error asociado al problema encontrado.</i>

Tabla 28. CU1: Identificarse UIX monitorización.

5.4.2. Caso de Uso: Identificarse UIX Anomalías

IDENTIFICARSE UIX ANOMALÍAS	
IDENTIFICADOR	CU2
PRECONDICIONES	<i>El usuario debe de ser el usuario administrados registrado en el submódulo del sistema de anomalías y saberse su usuario y contraseña.</i>
POSTCONDICIONES	<i>El usuario no identificado, se convertirá en administrador y tendrá sus permisos asociados.</i>
ACTORES	<i>Usuario no identificado, Submódulo de gestor de anomalías.</i>
DESCRIPCIÓN	<i>El usuario no identificado podrá realizar dos tareas: Accederá a la Interfaz de inicio de sesión del sistema de anomalías. El sistema le solicitará sus credenciales. Si son correctas accederá a la UIX con los permisos de administrador. Si son incorrectas le denegara el paso al sistema.</i>
EXCEPCIONES	<i>El sistema este caído: si el sistema está caído no podrá acceder a las UIX. Notificar un error asociado al problema encontrado.</i>

Tabla 29. CU2: Identificarse UIX anomalías.

5.4.3. Caso de Uso: Simular Peticiones

SIMULAR PETICIONES	
IDENTIFICADOR	CU3
PRECONDICIONES	-
POSTCONDICIONES	Que existan peticiones en el almacén de logs.
ACTORES	Módulo UIX del cliente (Simulador de datos).
DESCRIPCIÓN	Se simularán las peticiones y se recogerán en un sistema de logs (aún sin tratar).
EXCEPCIONES	<p>El sistema este caído: si el sistema está caído no se podrán simular las peticiones ya que no habría UIX.</p> <p>Notificar un error asociado al problema encontrado.</p>

Tabla 30. CU3: Simular peticiones.

5.4.4. Caso de Uso: Tratar las Peticiones

TRATAR LAS PETICIONES	
IDENTIFICADOR	CU4
PRECONDICIONES	Que existan peticiones en los logs.
POSTCONDICIONES	Que las peticiones tengan el formato deseado.
ACTORES	Módulo de gestión de logs.
DESCRIPCIÓN	Se recogerán las peticiones simuladas y se les tratará con la herramienta CIC-FlowMeter.
EXCEPCIONES	<p>El sistema este caído: si el sistema está caído no se podrán tratar las peticiones con la herramienta.</p> <p>Notificar un error asociado al problema encontrado.</p>

Tabla 31. CU4: Tratar las peticiones.

5.4.5. Caso de Uso: Guardar Peticiones

GUARDAR PETICIONES	
IDENTIFICADOR	CU5
PRECONDICIONES	Que haya peticiones y estén debidamente tratadas.
POSTCONDICIONES	Que estén en el sistema de colas.
ACTORES	Submódulo del sistema de colas.
DESCRIPCIÓN	El sistema de colas deberá de introducir en orden de llegada todas las peticiones tratadas.
EXCEPCIONES	<p>El sistema este caído: si el sistema está caído no se podrán guardar las peticiones en el sistema de colas.</p> <p>Notificar un error asociado al problema encontrado.</p>

Tabla 32. CU5: Guardar peticiones.

5.4.6. Caso de Uso: Monitorizar Peticiones

MONITORIZAR PETICIONES	
IDENTIFICADOR	CU6
PRECONDICIONES	Que haya peticiones y estén debidamente tratadas.
POSTCONDICIONES	Que se puedan observar en UIX.
ACTORES	Módulo de gestión de peticiones y Usuario identificado (Administrador).
DESCRIPCIÓN	El sistema de monitorización deberá de ir mostrando todas las peticiones tratadas y el administrador podrá observarlas y agruparlas en diferentes dashboard.
EXCEPCIONES	El sistema este caído: si el sistema está caído no se podrán monitorizar las peticiones. Notificar un error asociado al problema encontrado.

Tabla 33. CU6: Monitorizar peticiones.

5.4.7. Caso de Uso: Clasificar Peticiones

CLASIFICAR PETICIONES	
IDENTIFICADOR	CU7
PRECONDICIONES	Que existan peticiones en el sistema de colas y que estén debidamente tratadas.
POSTCONDICIONES	Las peticiones tengan una clasificación numérica de 0 a 100.
ACTORES	-
DESCRIPCIÓN	El sistema extraerá del sistema de colas las peticiones y Apolo las irá clasificando, dándoles un valor numérico entre 0 y 100, desentendiendo de si son o no benignas o malignas.
EXCEPCIONES	El sistema este caído: si el sistema está caído no se podrán clasificar las peticiones en Apolo. Notificar un error asociado al problema encontrado.
NOTAS	Esta función la realiza internamente el sistema de Apolo.

Tabla 34. CU7: Clasificar Peticiones.

5.4.8. Caso de Uso: Guardar Clasificaciones

GUARDAR CLASIFICACIONES	
IDENTIFICADOR	CU8
PRECONDICIONES	Las peticiones tengan un valor numérico asignado entre 0 y 100.
POSTCONDICIONES	Que las peticiones clasificadas estén guardadas en módulo de anomalías.
ACTORES	Módulo de gestión de anomalías.
DESCRIPCIÓN	Una vez se hayan clasificado las peticiones, el módulo de gestión de anomalías las guardara teniendo en cuenta la estampa de la fecha de cada petición.
EXCEPCIONES	El sistema este caído: si el sistema está caído no se podrán guardar las peticiones de clasificaciones. Notificar un error asociado al problema encontrado.

Tabla 35. CU8: Guardar clasificaciones.

5.4.9. Caso de Uso: Enviar Notificaciones

ENVIAR NOTIFICACIONES	
IDENTIFICADOR	CU9
PRECONDICIONES	Las peticiones tengan un valor numérico asignado entre 0 y 100; y estén guardas en el módulo de gestión de anomalías.
POSTCONDICIONES	Que se haya enviado una notificación al correo del administrador, en el caso de que haya una anomalía.
ACTORES	Módulo de gestión de anomalías y usuario identificado (Administrador).
DESCRIPCIÓN	El módulo de gestión de anomalías detectara cuando una petición se ha clasificado como anómala y le enviara un correo al administrador con la información básica para que este sepa cómo actuar.
EXCEPCIONES	El sistema este caído: si el sistema está caído no se podrán enviar las notificaciones al administrador. Notificar un error asociado al problema encontrado.

Tabla 36. CU9: Enviar notificaciones.

5.4.10. Caso de Uso: Mostrar Clasificaciones

MOSTRAR CLASIFICACIONES	
IDENTIFICADOR	CU10
PRECONDICIONES	Las peticiones tengan un valor numérico asignado entre 0 y 100; y estén guardas en el módulo de gestión de anomalías.
POSTCONDICIONES	Que se puedan observar en UIX.
ACTORES	Módulo de gestión de anomalías y usuario identificado (Administrador).
DESCRIPCIÓN	El módulo de gestión de anomalías mostrara en una UIX todas las peticiones que se han clasificado, así como en qué fecha se han realizado y la puntuación que han obtenido.
EXCEPCIONES	El sistema este caído: si el sistema está caído no se podrán mostrar las clasificaciones en la UIX. Notificar un error asociado al problema encontrado.

Tabla 37. CU10: Mostrar clasificaciones.

5.5. Análisis de Casos de Uso y Escenarios

En esta sección se hará hincapié sobre algunos de los Casos de Uso anteriormente descritos, se desarrollarán más y se darán los escenarios de cada uno de ellos.

5.5.1. Diagramas de Casos de Uso

En este apartado, nos centraremos en expandir los Casos de Uso más relevantes que se derivan del Diagrama de Contexto, nos hemos centrado en dos grandes partes del sistema, el tratamiento de las peticiones (Ilustración 27 y Tabla 38) y la clasificación de estas (Ilustración 28 y Tabla 39).

5.5.1.1. Caso de Uso Expandido: Tratamiento de las Peticiones

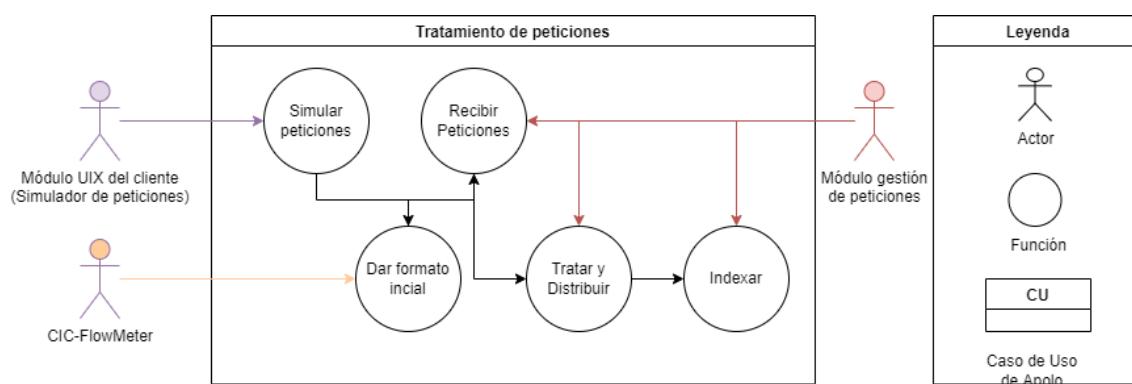


Ilustración 27. Caso de Uso Expandido 1.

Tratamiento de las peticiones
CUE1
Descripción
El sistema contará con la participación del actor adicional "CIC-FlowMeter", encargado de dar formato a las peticiones. Estas peticiones serán procesadas y pasadas por los subsistemas de tratamiento y distribución, así como por el subsistema de indexado.

Tabla 38. Descripción Caso de Uso Expandido 1.

5.5.1.2. Caso de Uso Expandido: Clasificación

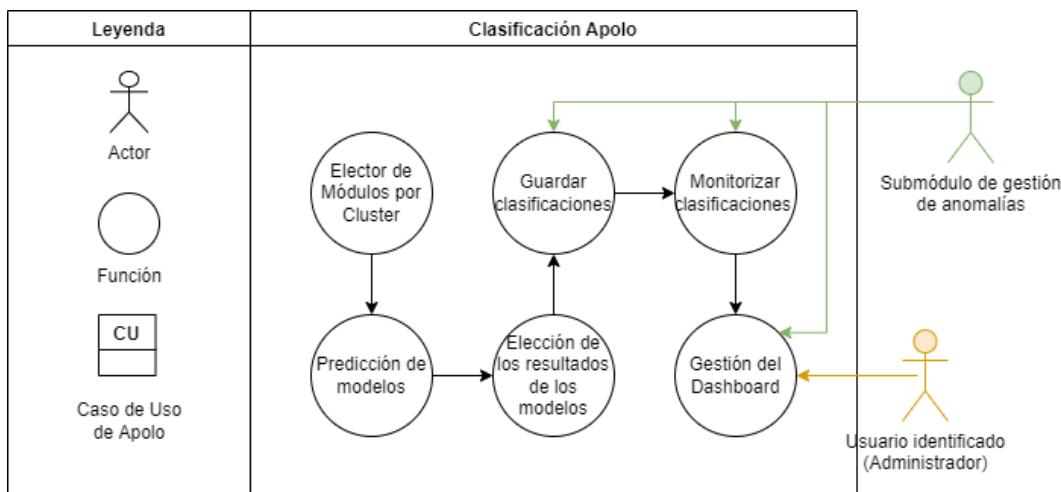


Ilustración 28. Caso de Uso Expandido 2.

Gestión de anomalías
CUE2
Descripción
Una vez tratadas las peticiones, el sistema clasificara las peticiones haciendo uso del modelo Apolo , esto hace que se ejecuten las diferentes capas de Apolo , como son la elección mediante Clúster de que modelos clasificarán las peticiones, la propia predicción de las peticiones y la elección de las clasificaciones haciendo uso de MAB. Una vez se ejecute Apolo , se guardarán las clasificaciones y se mostrarán para la correcta monitorización realizada por el administrador.

Tabla 39. Descripción Caso de Uso Expandido 2.

5.5.2. Escenarios

En esta sección, se realizarán los escenarios de los Casos de Uso Expandidos anteriormente descritos.

5.5.2.1. Escenario: Tratamiento de las Peticiones

En este apartado, se encuentran todos los escenarios de la sección 5.5.1.1. de este mismo capítulo “Caso de Uso Expandido: Tratamiento de las Peticiones”, se pueden ver en las tablas: Tabla 40, Tabla 41, Tabla 42, Tabla 43 y Tabla 44.

5.5.2.1.1. Caso de Uso Expandido: Simular Peticiones

SIMULAR PETICIONES	
IDENTIFICADOR	CUE1.1
PRECONDICIONES	-
POSTCONDICIONES	Que existan peticiones en el almacén de logs.
ACTORES	Módulo UIX del cliente (Simulador de datos).
DESCRIPCIÓN	Se simularán las peticiones y se recogerán en un sistema de logs (aún sin tratar).
EXCEPCIONES	<i>El sistema este caído: si el sistema está caído no se podrán simular las peticiones ya que no habría UIX.</i> <i>Notificar un error asociado al problema encontrado.</i>

Tabla 40. CUE1: Simular peticiones.

5.5.2.1.2. Caso de Uso Expandido: Dar Formato Inicial

DAR FORMATO INICIAL	
IDENTIFICADOR	CUE1.2
PRECONDICIONES	Que existan peticiones en los logs.
POSTCONDICIONES	Que las peticiones tengan el formato deseado.
ACTORES	CIC-FlowMeter.
DESCRIPCIÓN	Se tratará las peticiones con de la herramienta CIC-FlowMeter .
EXCEPCIONES	<i>El sistema este caído: si el sistema está caído no se podrán tratar las peticiones con la herramienta CIC-FlowMeter.</i> <i>Notificar un error asociado al problema encontrado.</i>

Tabla 41. CUE1: Dar formato inicial.

5.5.2.1.3. Caso de Uso Expandido: Recibir Peticiones

RECIBIR PETICIONES	
IDENTIFICADOR	CUE1.3
PRECONDICIONES	Que existan peticiones en los logs y estén tratadas con CIC-FlowMeter.
POSTCONDICIONES	Que haya peticiones en el módulo de gestión de peticiones.
ACTORES	Módulo de gestión de peticiones.
DESCRIPCIÓN	Recoger las peticiones simuladas y tratadas con CIC-FlowMeter.
EXCEPCIONES	<i>El sistema este caído: si el sistema está caído no se podrán recoger las peticiones en el módulo de gestión de peticiones.</i> <i>Notificar un error asociado al problema encontrado.</i>

Tabla 42. CUE1: Recibir peticiones.

5.5.2.1.4. Caso de Uso Expandido: Tratar y Distribuir

TRATAR Y DISTRIBUIR	
IDENTIFICADOR	CUE1.4
PRECONDICIONES	Que haya peticiones en el módulo de gestión de peticiones.
POSTCONDICIONES	Que las peticiones se hayan distribuido al sistema de colas y al indexado.
ACTORES	Módulo de gestión de peticiones.
DESCRIPCIÓN	Se recogerán las peticiones recibidas y tratadas con CIC-FlowMeter, se les dará un tratamiento superfluo para que el módulo de gestión de peticiones pueda distribuirlas adecuadamente. Enviara las peticiones al sistema de colas y a la etapa de Indexación del módulo de gestión de peticiones.
EXCEPCIONES	El sistema este caído: si el sistema está caído no se podrán tratar y distribuir las peticiones en el módulo de gestión de peticiones. Notificar un error asociado al problema encontrado.

Tabla 43. CUE1: Tratar y Distribuir.

5.5.2.1.5. Caso de Uso Expandido: Indexar

INDEXAR	
IDENTIFICADOR	CUE1.5
PRECONDICIONES	Que haya peticiones en el módulo de gestión de peticiones.
POSTCONDICIONES	Que las peticiones se hayan indexado en el módulo de gestión de peticiones.
ACTORES	Módulo de gestión de peticiones.
DESCRIPCIÓN	Se indexarán las peticiones en el módulo de gestión de peticiones, para que estas sean más fáciles de gestionar y monitorizar en el dashboard.
EXCEPCIONES	El sistema este caído: si el sistema está caído no se podrán indexar las peticiones en el módulo de gestión de peticiones. Notificar un error asociado al problema encontrado.

Tabla 44. CUE1: Indexar.

5.5.2.2. Escenario: Clasificación

En este apartado, se encuentran todos los escenarios de la sección 5.5.1.2. de este mismo capítulo “Caso de Uso Expandido: Clasificación”, se pueden ver en las tablas: Tabla 45, Tabla 46, Tabla 47, Tabla 48, Tabla 49 y Tabla 50.

5.5.2.2.1. Caso de Uso Expandido: Elector de Módulos por Clúster

ELECTOR DE MÓDULOS POR CLUSTER	
IDENTIFICADOR	CUE2.1
PRECONDICIONES	Debe de existir al menos una petición en el sistema de colas, la cual tiene el formato indicado para poder clasificar la petición.
POSTCONDICIONES	El sistema haya escogido que modelos van a clasificar la petición.
ACTORES	
DESCRIPCIÓN	Apolo elegirá mediante su capa de clúster que modelos van a clasificar la petición, dicha elección se realizará teniendo en cuenta la estructura interna de la petición.
EXCEPCIONES	<p>El sistema este caído: si el sistema está caído no podrá ejecutar sus funciones de Apolo.</p> <p>Notificar un error asociado al problema encontrado.</p>

Tabla 45. CUE2: Elector de módulos por clúster.

5.5.2.2.2. Caso de Uso Expandido: Predicción de los modelos

PREDICCIÓN DE LOS MODELOS	
IDENTIFICADOR	CUE2.2
PRECONDICIONES	Debe de existir al menos una petición en el sistema de colas, la cual tiene el formato indicado para poder clasificar la petición.
POSTCONDICIONES	La petición esté clasificada por al menos un modelo.
ACTORES	
DESCRIPCIÓN	Apolo clasificara la petición en los modelos elegidos por el clúster.
EXCEPCIONES	<p>El sistema este caído: si el sistema está caído no podrá ejecutar sus funciones de Apolo.</p> <p>Notificar un error asociado al problema encontrado.</p>

Tabla 46. CUE2: Predicción de los modelos.

5.5.2.2.3. Caso de Uso Expandido: Elección de los Resultados de los Modelos

ELECCIÓN DE LOS RESULTADOS DE LOS MODELOS	
IDENTIFICADOR	CUE2.3
PRECONDICIONES	Debe de existir al menos una petición en el sistema de colas, la cual tiene el formato indicado para poder clasificar la petición.
POSTCONDICIONES	La petición esté clasificada por al menos un modelo.
ACTORES	
DESCRIPCIÓN	Apolo haciendo uso del algoritmo MAB, elegirá que modelo o combinación de modelos, es el más preciso para la petición dada.
EXCEPCIONES	<p>El sistema este caído: si el sistema está caído no podrá ejecutar sus funciones de Apolo.</p> <p>Notificar un error asociado al problema encontrado.</p>

Tabla 47. CUE2: Elección de los resultados de los modelos.

5.5.2.2.4. Caso de Uso Expandido: Guardar Clasificaciones

GUARDAR CLASIFICACIONES	
IDENTIFICADOR	CUE2.4
PRECONDICIONES	Las peticiones tengan un valor numérico asignado entre 0 y 100.
POSTCONDICIONES	Que las peticiones clasificadas estén guardadas en el módulo de anomalías.
ACTORES	Módulo de gestión de anomalías.
DESCRIPCIÓN	Una vez se hayan clasificado las peticiones, el módulo de gestión de anomalías las guardará teniendo en cuenta la estampa de la fecha de cada petición.
EXCEPCIONES	<p><i>El sistema este caído: si el sistema está caído no se podrán guardar las peticiones de clasificaciones.</i></p> <p><i>Notificar un error asociado al problema encontrado.</i></p>

Tabla 48. CUE2: Guardar clasificaciones.

5.5.2.2.5. Caso de Uso Expandido: Monitorizar Clasificaciones

MONITORIZAR CLASIFICACIONES	
IDENTIFICADOR	CUE2.5
PRECONDICIONES	Las peticiones tengan un valor numérico asignado entre 0 y 100; y estén guardadas en el módulo de gestión de anomalías.
POSTCONDICIONES	Que se puedan observar en UIX.
ACTORES	Módulo de gestión de anomalías.
DESCRIPCIÓN	El módulo de gestión de anomalías mostrara en una UIX todas las peticiones que se han clasificado, así como en qué fecha se han realizado y la puntuación que han obtenido.
EXCEPCIONES	<p><i>El sistema este caído: si el sistema está caído no se podrán mostrar las clasificaciones en la UIX.</i></p> <p><i>Notificar un error asociado al problema encontrado.</i></p>

Tabla 49. CUE2: Monitorizar clasificaciones.

5.5.2.2.6. Caso de Uso Expandido: Gestión del Dashboard

GESTIÓN DEL DASHBOARD	
IDENTIFICADOR	CUE2.6
PRECONDICIONES	Se pueden observar las clasificaciones en la UIX.
POSTCONDICIONES	
ACTORES	Módulo de gestión de anomalías y usuario identificado (Administrador)
DESCRIPCIÓN	El administrador puede crear diferentes tablas o gráficas para monitorizar y gestionar las clasificaciones del sistema.
EXCEPCIONES	<p><i>El sistema este caído: si el sistema está caído no se podrán mostrar las clasificaciones en la UIX.</i></p> <p><i>Notificar un error asociado al problema encontrado.</i></p>

Tabla 50. CUE2: Gestión del Dashboard.

Capítulo 6. Definición del Sistema

En este capítulo definiremos el sistema, para ello explicaremos el gestor de microservicios qué usaremos, cuáles han sido las tecnologías y herramientas usadas para cada componente de la arquitectura del sistema, describiremos cómo funciona la comunicación entre los diferentes componentes de la arquitectura, las normas y estándares utilizados durante el desarrollo del proyecto y los lenguajes, entornos y herramientas usadas durante todo el proyecto.

Al explicar todas las tecnologías y herramientas utilizadas, daremos una pertinente justificación de cada uso de estas. En el Capítulo 7. "Evaluación de Alternativas" damos diferentes alternativas que hemos sopesado.

6.1. Microservicios

En primera instancia vamos a utilizar *Docker*¹³ y *Docker Compose*¹⁴ para gestionar los microservicios descritos en los requisitos del sistema.

Docker es una plataforma abierta que ayuda a automatizar el despliegue, la escalabilidad y la administración de aplicaciones. *Docker* permite encapsular una aplicación y sus dependencias en un contenedor aislado, que puede ejecutarse en cualquier servidor. Esto ayuda a asegurar que la aplicación se ejecute de la misma manera, sin importar el entorno en el que se encuentre.

Los contenedores *Docker* son livianos y se inician rápidamente. Se parecen a las máquinas virtuales, pero en lugar de crear un sistema operativo completo, *Docker* permite que las aplicaciones usen el mismo kernel de Linux que el sistema que las está ejecutando, solo requiere que la aplicación tenga sus propias bibliotecas y configuraciones de sistema.

En nuestro caso *Docker* va a servir para desplegar cada uno de contenedores que encapsulan los microservicios del sistema.

Como tenemos varios contenedores *Docker*, necesitamos una herramienta que nos permita gestionarlos. Para esa tarea contamos con *Docker Compose* que es una herramienta que justamente sirve para gestionar aplicaciones compuestas por múltiples contenedores *Docker*, aplicaciones multi-contenedor.

Con *Docker Compose*, se puede especificar la configuración de los contenedores en un archivo *YAML*¹⁵ para configurar las aplicaciones de tu servicio. También puedes administrar, iniciar, pausar o detener todos los servicios definidos en el archivo de configuración simultáneamente con un solo comando.

Estas herramientas nos son de mucha utilidad ya que están pensadas para ser utilizadas en proyectos de este gramaje. Proyectos que conlleven muchos contenedores, donde tanto la comunicación como la seguridad son de vital importancia para el correcto funcionamiento del sistema.

Docker Compose es especialmente útil en situaciones donde necesitas correr múltiples contenedores como un único servicio. En nuestro caso tenemos una aplicación que requiere tener 10 contenedores *Docker* levantados

¹³ <https://www.Docker.com/>

¹⁴ <https://docs.Docker.com/compose/>

¹⁵ <https://YAML.org/>

simultáneamente. Con *Docker Compose*, puedes definir los 10 contenedores en un solo archivo y luego ejecutar los 10 juntos con un solo comando.

La arquitectura del sistema **Apolo**, como se especifica en el requisito RIHardware-2, se basa en una estructura de microservicios. A continuación, se describen en detalle:

1. Microservicio modelo **Apolo** (RIHardware-2.1.1.): Este contenedor *Docker* se encarga de alojar el modelo **Apolo** ya entrenado, véase el IDS del sistema de gestión y monitorización de peticiones. Este contenedor es el encargado de clasificar las peticiones en malignas y benignas.
2. Base de Datos de Series Temporales y monitorización de anomalías (RIHardware-2.1.2.): Este contenedor aloja una base de datos especialmente diseñada para manejar información de series temporales. También incluye una interfaz para poder monitorizar las anomalías encontradas en las clasificaciones de las peticiones.
3. Sistema de Colas (RIHardware-2.1.3.): En este contenedor se implementa un servicio de colas.
4. *WebApp* (RIHardware-2.2.1.): Este contenedor aloja la aplicación web de ejemplo que los usuarios finales utilizan para interactuar con el sistema.
5. *REST API* (RIHardware-2.2.2.): Este contenedor aloja la *API REST* de ejemplo que se utiliza para comunicarse con la *WebApp*.
6. *Proxy reverso y Firewall* (RIHardware-2.3.): Este contenedor implementa un proxy reverso y un firewall de red para proteger el sistema y redirigir el tráfico de red.
7. Recopilación de Peticiones de Tráfico Red (RIHardware-2.4.1.): Este microservicio es responsable de recolectar las peticiones de tráfico de red.
8. Agrupación de Peticiones e Ingesta en la Base de Datos de *Logs* (RIHardware-2.4.2.): Este microservicio toma las peticiones de tráfico de red recopiladas, las agrupa y las ingresa en la base de datos de *logs*.
9. Base de Datos de *Logs* (RIHardware-2.4.3.): Este contenedor aloja la base de datos que almacena todos los *logs* del sistema.
10. Interfaz Gráfica para el Monitoreo de los *Logs* (RIHardware-2.4.4.): Este contenedor aloja una interfaz de usuario gráfica que permite a los administradores del sistema visualizar y monitorear los *logs* del sistema de manera fácil y conveniente.

Cada uno de estos contenedores *Docker* se puede desplegar, escalar y gestionar de forma independiente, lo que proporciona al sistema **Apolo** una alta flexibilidad y resistencia.

Docker ofrece una serie de ventajas que lo hacen ideal para la implementación y gestión de sistemas. Su capacidad de proporcionar aislamiento y portabilidad es uno de sus principales beneficios. Esto se logra encapsulando todas las dependencias del sistema en contenedores independientes. Cada componente se ejecuta en su propio entorno aislado, evitando conflictos y garantizando la estabilidad del sistema. Además, los contenedores son portátiles, permitiendo un despliegue sin problemas en diversos entornos y sistemas operativos sin la preocupación de diferencias de configuración.

Además de la portabilidad y el aislamiento, Docker ofrece escalabilidad y flexibilidad. Permite escalar cada componente del sistema de forma independiente, facilitando la asignación de recursos donde más se necesiten para manejar cargas de trabajo intensas. *Docker Compose* mejora esta flexibilidad, facilitando la gestión de múltiples contenedores interconectados y permitiendo que el sistema se adapte y escale según las necesidades específicas de gestión y monitorización de peticiones y detección de anomalías.

La facilidad de despliegue y gestión es otra ventaja significativa de Docker. Los contenedores pueden ser creados y configurados de manera consistente, lo que simplifica enormemente el despliegue en entornos de desarrollo, pruebas y producción. Con *Docker Compose*, la configuración de los contenedores se define en un archivo *YAML*, facilitando la gestión de la infraestructura y permitiendo reproducir el entorno en diferentes máquinas con facilidad.

Docker también cuenta con un ecosistema robusto y una comunidad de usuarios activa. Hay una amplia variedad de recursos, documentación y ejemplos disponibles para asistir en el desarrollo, implementación y resolución de problemas del sistema. *Docker Hub*¹⁶ proporciona un repositorio centralizado de imágenes de contenedores, lo que facilita la búsqueda y el uso de imágenes preconfiguradas para acelerar el proceso de implementación.

¹⁶ <https://hub.Docker.com/>

Estas herramientas permiten crear un entorno eficiente, flexible y confiable para el análisis y control de tráfico web, facilitando la detección y respuesta a posibles anomalías de manera ágil y efectiva.

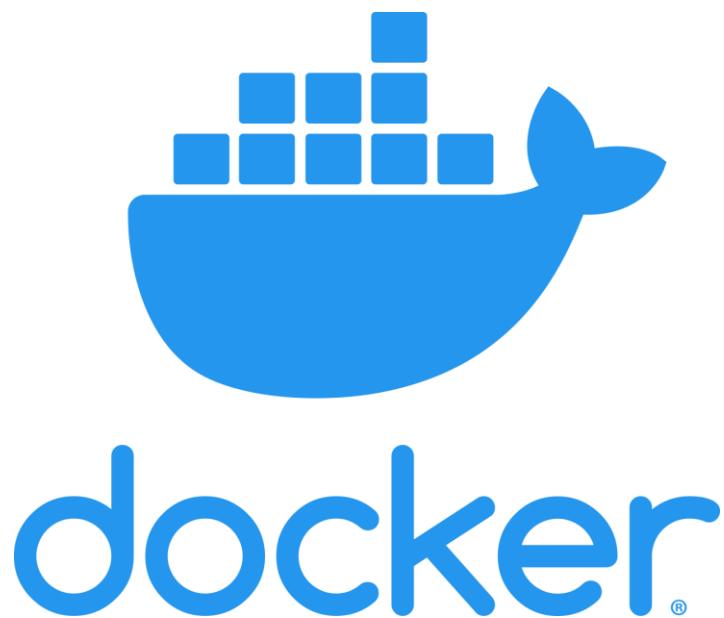


Ilustración 29. Logo de Docker.

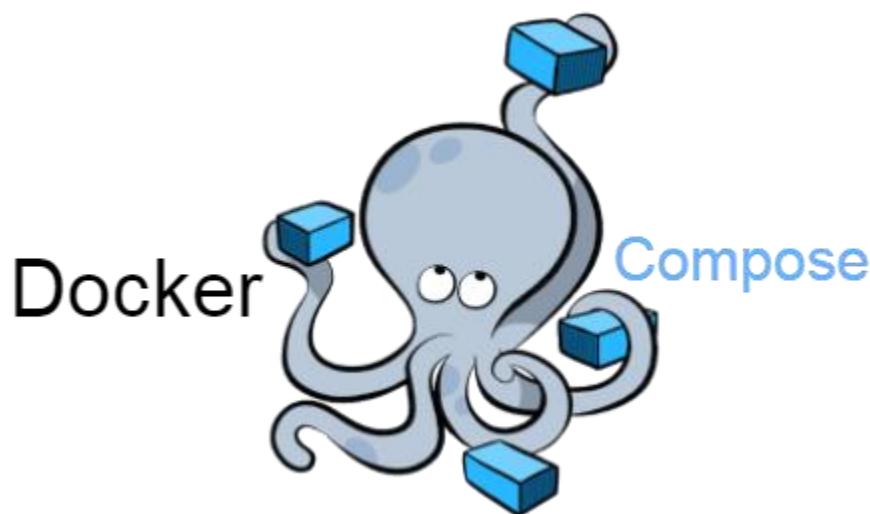


Ilustración 30. Logo de Docker Compose.

6.2. Definición de los Componentes de la Arquitectura

En este apartado vamos a explicar que tecnologías y herramientas hemos utilizado en cada componente.

En la Ilustración 31 se puede observar la arquitectura del sistema con las tecnologías utilizadas en cada componente, la explicación de qué función realiza cada componente está en el Capítulo 5. “Análisis”, más concretamente en el punto 5.3. “Identificación de los Componentes de la Arquitectura del Sistema”.

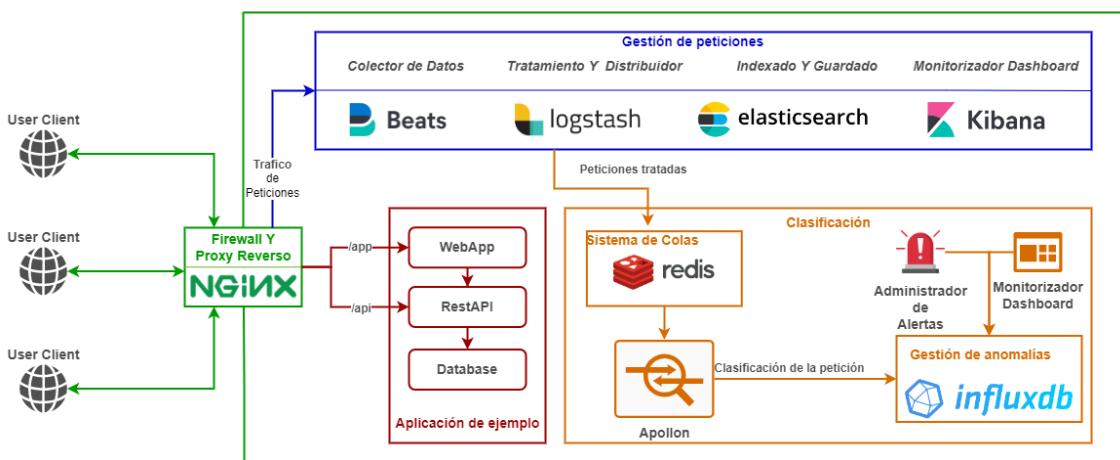


Ilustración 31. Arquitectura del Sistema.

6.2.1. Firewall y Proxy Reverso

Para este componente decidimos utilizar *NGINX*¹⁷. *NGINX* es un servidor web de alto rendimiento y proxy reverso de código abierto. Es conocido por su capacidad para manejar cargas de tráfico webs pesadas y proporcionar un rendimiento rápido y escalable.

NGINX actúa como un intermediario entre los clientes y los servidores *backend*, funcionando como un *proxy* reverso. Cuando un cliente realiza una solicitud, *NGINX* recibe la solicitud y puede realizar varias acciones, como enrutar el tráfico a diferentes servidores *backend*, equilibrar la carga entre múltiples servidores o cachear contenido estático para una entrega más rápida. En nuestro caso, *NGINX* envía las peticiones al componente de gestión de peticiones. También, puede

¹⁷ <https://www.nginx.com/>

trabajar como *firewall*, lo que la convierte en una tecnología multifacética que hace que encaje a la perfección con nuestro sistema.

NGINX nos da varias ventajas para este caso en el que la seguridad y la eficiencia son muy importantes y no puede prevalecer una por encima de la otra. A continuación, una serie de ventajas del uso de *NGINX* en nuestro sistema:

- Rendimiento y escalabilidad: *NGINX* es conocido por su alto rendimiento y capacidad de escalar eficientemente para manejar cargas de tráfico pesadas. Esto es especialmente importante en un sistema de monitorización en tiempo real (**Apolo**), donde se espera un flujo constante de peticiones de tráfico web. *NGINX* está diseñado para manejar grandes cantidades de conexiones simultáneas y optimizar la entrega de contenido estático y dinámico, lo que asegura una respuesta rápida y eficiente en tiempo real.
- Balanceo de carga y tolerancia a fallos: *NGINX* es capaz de realizar el balanceo de carga entre múltiples servidores *backend*, lo que permite distribuir la carga de tráfico y garantizar una alta disponibilidad del sistema. Esto es esencial en un entorno de monitorización y gestión de anomalías donde se requiere una infraestructura robusta y resistente a fallos. *NGINX* también puede realizar comprobaciones de salud en los servidores *backend* y redirigir el tráfico a nodos sanos en caso de fallos, asegurando así la continuidad del servicio.
- Configuración flexible y extensible: *NGINX* ofrece una configuración flexible que permite adaptar el sistema a las necesidades específicas del sistema de monitorización y gestión de anomalías. Es altamente personalizable y admite una amplia gama de módulos y complementos para agregar funcionalidades adicionales según sea necesario. Esto facilita la integración con otras herramientas y servicios utilizados en el sistema, como sistemas de almacenamiento, bases de datos y herramientas de análisis.
- Seguridad y protección: *NGINX* incluye características de seguridad sólidas que ayudan a proteger el sistema de monitorización y gestión de anomalías del tráfico web. Puede actuar como un *proxy reverso* y proporcionar una capa adicional de seguridad al ocultar la estructura interna del sistema y protegerlo de posibles ataques directos. *NGINX*

también admite la configuración de reglas de *firewall* y políticas de acceso para controlar y filtrar el tráfico entrante.



Ilustración 32. Logo de NGINX.

6.2.2. Aplicación de Ejemplo

El componente de la aplicación de ejemplo es un componente especial, ya que, está pensado para que se pueda intercambiar con cualquier aplicación creada. El único requisito que tiene este componente es que la aplicación elegida se pueda desplegar en *Docker*.

Para nuestra demostración hemos escogido una aplicación sencilla¹⁸, para que dé la menor cantidad de problemas posibles. Para probar la arquitectura, se ha utilizado una aplicación muy simple, la cual se creó para una asignatura de la carrera (*Software y Estándares para la Web*), esta aplicación no tiene una lógica muy compleja, pero para nuestro ejemplo es válida.

Que este componente sea tan intercambiable y personalizable favorece el cometido del sistema que es que se pueda desplegar en entornos reales de empresas, para que gestionen y monitoricen sus peticiones y posibles anomalías.

6.2.3. Gestión de Peticiones

Para este componente decidimos utilizar el *Stack de ELK*¹⁹. Este *Stack* es uno de los más extendidos en el mundo y encaja a la perfección con nuestro sistema, ya

¹⁹ <https://www.elastic.co/es/what-is/elk-stack>

que necesitas almacenar, distribuir, gestionar y monitorizar las diferentes peticiones web que lleguen al sistema.

ELK Stack es un conjunto de tres herramientas de código abierto - *Elasticsearch*, *Logstash* y *Kibana* - que juntas forman una poderosa plataforma para la gestión de *logs* y datos de tiempo real. En nuestro caso, también se incorpora una cuarta herramienta conocida como *Beats*²⁰. A continuación, explicaremos el funcionamiento de cada parte del *Stack*:

- *Beats*: Es una plataforma para agentes de recopilación de datos ligeros y de un solo propósito, llamados “*Beats*”, que se instalan como agentes en los servidores para enviar datos operativos a *Logstash*. *Beats* puede enviar diferentes tipos de datos, como *logs* de sistema, métricas de red, archivos de *log* de aplicación y más. En nuestro caso, *Beats* envía los *logs* de las peticiones web.
- *Logstash*: Es una herramienta de procesamiento de *logs* que se utiliza para recibir los datos desde *Beats*, transformarlos y luego enviarlos a diferentes destinos, en nuestro caso a *Elasticsearch* y a *Redis*²¹. *Logstash* proporciona un marco para la ingestión de datos de diferentes fuentes y la transformación de esos datos, como extracción de campos, enriquecimiento de datos o normalización, para que puedan ser más fácilmente consumidos y analizados por *Elasticsearch*.
- *Elasticsearch*: Es un motor de búsqueda y análisis de código abierto, distribuido y basado en *REST*. *Elasticsearch* se utiliza para indexar, buscar y analizar los datos en tiempo real. Es altamente escalable, permitiendo la búsqueda en grandes cantidades de datos de manera rápida y eficiente.
- *Kibana*: Es una interfaz de usuario web que se utiliza para visualizar los datos en *Elasticsearch*. *Kibana* proporciona una variedad de herramientas y gráficos interactivos para crear paneles personalizados, tablas, gráficos y mapas. *Kibana* permite realizar consultas *ad hoc*, crear visualizaciones dinámicas y compartir los resultados con otros usuarios. También ofrece funciones de búsqueda y filtrado avanzadas para explorar los datos de manera eficiente. *Kibana* es una herramienta clave para la visualización y el análisis de datos en tiempo real.

²⁰ <https://www.elastic.co/es/beats/>

²¹ <https://redis.io/>

En el contexto de su proyecto, *ELK Stack* encaja perfectamente ya que permite manejar, almacenar, monitorear y analizar las diferentes peticiones web que llegan al sistema. *Beats* recoge y envía los datos operativos pertinentes a *Logstash*; *Logstash* facilita la recopilación y transformación de los datos de las peticiones; *Elasticsearch* proporciona la funcionalidad de almacenamiento y análisis; y *Kibana* permite visualizar y monitorear estos datos en tiempo real.

A continuación, daremos una serie de ventajas que nos han hecho decantarnos por utilizar el *Stack* de *ELK* más *Beats* para la gestión de peticiones en nuestro sistema:

- Recopilación de datos eficiente: *Logstash* y *Beats* son excelentes para recopilar y enviar datos de las peticiones web a *Elasticsearch*. *Beats* recopila los *logs*, mientras que *Logstash* los trata y distribuye a *Elasticsearch* y a *Redis*. En el contexto de un *IDS*, esto significa que se puede recoger y procesar un amplio rango de información de tráfico en tiempo real.
- Almacenamiento y búsqueda de datos en tiempo real: *Elasticsearch* es ideal para el almacenamiento y la búsqueda de datos en tiempo real. Con *Elasticsearch*, los datos de las peticiones web y del tráfico de red pueden ser indexados y buscados en tiempo real, lo que es crítico para la detección y respuestas rápidas a potenciales amenazas en un *IDS*.
- Análisis de datos: *Elasticsearch* proporciona una capacidad de búsqueda y análisis altamente eficiente, lo que permite realizar consultas y búsquedas complejas en los datos recopilados. Puedes buscar, filtrar y visualizar datos de diferentes formas para identificar patrones, tendencias y anomalías. Esto puede ayudar a detectar actividades sospechosas o anomalías en el tráfico de la red.
- Visualización de datos: *Kibana* ofrece visualizaciones en tiempo real y cuadros de mando personalizados que pueden ser utilizados para monitorizar el tráfico de la red y las peticiones web. Esto puede proporcionar una visión instantánea del estado del *IDS*, permitiendo a los administradores del sistema responder rápidamente a cualquier problema.

- Escalabilidad: *ELK Stack* está diseñado para escalarse horizontalmente, lo que significa que se puede agregar capacidad adicional a medida que aumenta la carga de trabajo. Esto permite manejar volúmenes crecientes de datos de tráfico web y mantener un rendimiento óptimo. Además, *Elasticsearch* proporciona opciones de configuración para lograr alta disponibilidad y tolerancia a fallos, lo que garantiza que el sistema esté disponible y en funcionamiento incluso en caso de problemas.
- Comunidad y soporte: *ELK Stack* es una solución ampliamente adoptada y respaldada por una comunidad activa y un soporte robusto. Esto asegura que haya recursos disponibles, documentación detallada, tutoriales y actualizaciones periódicas para mantener el sistema actualizado y seguro.

El Stack *ELK* más *Beats*, proporciona una solución completa para manejar y analizar grandes volúmenes de datos en tiempo real, esto lo convierte en una elección ideal para gestionar las peticiones del tráfico de un *IDS*.



Ilustración 33. Logo de Stack ELK incluyendo Beats.

6.2.4. Clasificación

En este apartado vamos a definir dos partes del componente de clasificación: el Sistema de Colas y la Gestión de Anomalías. **Apolo** está definido y explicado en el Capítulo 3. “Apolo”.

6.2.4.1. Sistema de Colas

Para este subcomponente hemos optado por la herramienta *Redis*²². *Redis* (*Remote Dictionary Server*) es un almacén de estructura de datos en memoria, de código abierto, que se utiliza como base de datos, caché y agente de mensajes. Es conocido por su alto rendimiento, flexibilidad, naturaleza rica en características y capacidades de escalado horizontal.

En nuestro caso lo utilizamos como sistema de colas de mensajes de alta velocidad para la gestión de colas y la comunicación asíncrona entre el gestor de peticiones y **Apolo**. Se basa en una estructura de datos clave-valor en memoria, lo que le permite ofrecer un rendimiento extremadamente rápido y una latencia baja.

En *Redis*, las colas se implementan mediante la utilización de pilas, donde los elementos se agregan al final de la lista y se pueden eliminar tanto del principio como del final. En nuestro caso lo utilizamos como una pila *FIFO* (*First In, First Out*), esto permite asegurarnos un orden de llegada eficiente y ordenado de las peticiones.

Redis tiene varias características clave:

- Almacenamiento en memoria: *Redis* mantiene sus datos en memoria y, por lo tanto, puede ofrecer una latencia muy baja y un rendimiento de lectura y escritura muy alto.
- Estructuras de datos ricas: A diferencia de otros sistemas de colas que pueden soportar un conjunto limitado de tipos de datos, *Redis* soporta una amplia variedad de estructuras de datos, como cadenas, listas, conjuntos, conjuntos ordenados, mapas, *hyperlogs*, *bitmaps* y flujos de datos.
- Persistencia de datos: Aunque *Redis* es un almacén de datos en memoria, también proporciona mecanismos para persistir los datos en el disco. Esto puede proteger contra la pérdida de datos en caso de que el sistema se caiga o se reinicie.

²² <https://redis.io/>

- Replicación y particionamiento de datos: *Redis* admite la replicación maestro-esclavo y proporciona soporte para la partición de datos para permitir un escalado horizontal.
- *Pub/Sub*: *Redis* soporta capacidades de publicación y suscripción, lo que lo hace útil para la mensajería en tiempo real.
- Compatibilidad con transacciones: *Redis* también ofrece soporte para transacciones, lo que permite ejecutar un grupo de comandos de manera atómica.

Redis es conocido por su alta velocidad y baja latencia, estas cualidades son esenciales para un sistema de monitorización en tiempo real que requiere procesar y responder rápidamente a los eventos de tráfico web. Su capacidad de configurar clústeres y agregar nodos permite una escalabilidad horizontal eficiente a medida que aumenta la carga de tráfico web.

Redis ofrece persistencia en disco para garantizar la durabilidad y consistencia en la gestión de eventos. Proporciona funciones avanzadas para la manipulación de colas y tiene una amplia compatibilidad con *Python*.

Cuenta con años de bagaje ya que es de las tecnologías más utilizadas en este tipo de casos, por lo que cuenta con una comunidad activa y documentación amplia que facilita su implementación y mantenimiento.

Hemos decidido implementar *Redis* en el sistema, ya que las características citadas anteriormente encajan a la perfección con la solución que estábamos buscando para este componente.



Ilustración 34. Logo de Redis.

6.2.4.2. Gestión de Anomalías

Puesto que este componente debe de cumplir con múltiples funciones, guardado de clasificaciones, monitorización y gestión de anomalías y notificación de alertas, necesitamos una tecnología capaz de realizar todas estas funciones de la manera más eficiente posible, por ello hemos decidido utilizar la tecnología de *InfluxDB*²³.

InfluxDB es una base de datos de series de tiempo de código abierto desarrollada por *InfluxData*. Está diseñada específicamente para manejar series de tiempo con alta disponibilidad y requisitos de rendimiento, lo que la hace útil para aplicaciones que deben procesar grandes cantidades de datos basados en tiempo, como la monitorización y gestión de anomalías en tiempo real, la telemetría y los análisis de rendimiento.

InfluxDB utiliza un modelo de datos basado en series temporales, donde los datos se almacenan en medidas, campos y etiquetas. Las medidas representan los tipos de datos, los campos contienen los valores numéricos o de texto asociados y las etiquetas proporcionan metadatos para organizar y filtrar los datos. Esta estructura permite una fácil consulta y análisis de datos basados en criterios de tiempo y otras dimensiones relevantes.

InfluxDB es una base de datos de series de tiempo altamente optimizada, diseñada para manejar eficientemente grandes volúmenes de datos basados en tiempo, lo que resulta en consultas rápidas y un rendimiento de lectura y escritura extremadamente rápido.

Se diferencia de las bases de datos tradicionales en su capacidad para manejar una gran cantidad de puntos de datos por segundo, y tiene la capacidad de escalar horizontalmente para acomodar aún más datos. También, cuenta con un potente lenguaje de consulta de datos, *InfluxQL*, similar a *SQL*, pero adaptado para consultas basadas en tiempo, lo que permite un análisis de datos muy flexible. Posee características de compresión de datos integradas para almacenar más datos en menos espacio, y también permite la configuración de políticas de retención de datos para gestionar automáticamente la duración de los datos almacenados.

InfluxDB se beneficia de una comunidad activa de usuarios y desarrolladores que proporcionan una amplia gama de recursos, documentación y soporte técnico,

²³ <https://www.influxdata.com/>

Apolo: Sistema inteligente de monitorización y detección de ataques basados en inteligencia artificial

garantizando soluciones a problemas comunes, actualizaciones regulares y un mejor rendimiento y seguridad, especialmente valioso en entornos críticos como sistemas de monitorización y gestión de anomalías.

Gracias a sus altas prestaciones, *InfluxDB* es ideal para la captura y almacenamiento de eventos de tráfico web en un sistema de monitorización y gestión de anomalías.



Ilustración 35. Logo de InfluxData.



Ilustración 36. Logo de InfluxDB.

6.3. Descripción de la Comunicación entre Componentes de la Arquitectura

Como se puede observar en la Ilustración 31, cada subsistema está aislado y sólo se puede acceder a través del componente de *firewall* y *proxy* reverso (*NGINX*).

El sistema está pensado para poder desplegarse en una arquitectura de red real, pero para realizar esta “prueba” usamos *Docker Compose*. Por lo que cada componente, y en algunos casos partes de estos, se encuentra en un contenedor *Docker* separado, lo que permite controlar los puertos abiertos de cada módulo, así como su acceso.

La aplicación de ejemplo no está conectada con otros componentes del sistema, las solicitudes son enviadas directamente desde *NGINX* al Stack de *ELK*. A este componente se puede acceder a través del sistema de Monitorización, pero requiere una contraseña de acceso.

Las solicitudes son procesadas y almacenadas en *Redis* y **Apolo** las clasifica en orden de llegada. Una vez clasificadas, se envían a *InfluxDB* para su almacenamiento, con la clasificación correspondiente y la estampa de tiempo. Gracias a que *InfluxDB* cuenta con una interfaz de monitorización se pueden realizar diferentes *dashboards* que pueden ser personalizables por el administrador. Además, en el caso de que la petición fuera clasificada como anomalía *InfluxDB* puede notificar al administrador.

6.4. Estándares y Normas Seguidos

En esta sección describiremos los estándares y normas utilizados durante la realización del proyecto.

6.4.1. Estándares

Principalmente hemos usado estándares centrados en la investigación y desarrollo de proyectos tecnológicos. Estos estándares son ampliamente usados en proyectos informáticos.

6.4.1.1. IEE 830-1998

El estándar *IEEE 830-1998*, “Recommended Practice for Software Requirements Specifications”¹², es un estándar desarrollado por el *Institute of Electrical and Electronics Engineers (IEEE)*. Este estándar proporciona recomendaciones sobre cómo se deben definir, organizar y presentar las especificaciones de requisitos de software. Abarca aspectos como la introducción, los requisitos generales, las funcionalidades específicas, los apéndices y el índice.

6.4.2. Normas

Principalmente hemos usado normas centradas en la investigación y desarrollo de proyectos tecnológicos. Estas normas son ampliamente usadas en el ámbito de proyectos informáticos.

6.4.2.1. UNE 157801

La norma *UNE 157801*, denominada “Criterios generales para la elaboración de proyectos de I+D+i”²⁴, es una norma desarrollada por la *Asociación Española de Normalización (UNE)*. Esta norma define los criterios generales que deben seguirse para la elaboración de proyectos de *Investigación, Desarrollo e Innovación (I+D+i)*. Cubre las distintas etapas de un proyecto de *I+D+i*, incluyendo la identificación de oportunidades, la definición de objetivos, la planificación, la ejecución y el control, así como la evaluación final del proyecto.

²⁴ <https://www.une.org/encuentra-tu-norma/busca-tu-norma/norma?c=N0039367>

6.4.2.2. UNE 157001

La norma *UNE 157001*, “Especificaciones de prestación de servicios de archivos”²⁵, es una norma española que establece los requisitos mínimos que deben cumplir los servicios de archivos en términos de la organización y las responsabilidades, los procedimientos y las instalaciones. Cubre todos los aspectos del servicio de archivos, desde la adquisición y preservación de los documentos hasta la organización y accesibilidad de la información.

²⁵ <https://www.une.org/encuentra-tu-norma/busca-tu-norma/norma?c=N0052985>

6.5. Lenguajes, Entornos y Herramientas

En este apartado definiremos los lenguajes, entornos y herramientas utilizados durante el desarrollo del proyecto. Hemos valorado otras alternativas a las utilizadas para este proyecto, estas se recogen en el Capítulo 7. “Evaluación de Alternativas”.

6.5.1. Lenguajes

En esta sección, se encuentran los lenguajes de programación, intérpretes de comandos y ficheros de configuración utilizados durante el desarrollo del proyecto.

6.5.1.1. Python

*Python*²⁶ es un lenguaje de programación de alto nivel que es conocido por su legibilidad y simplicidad. Es un lenguaje de programación interpretado y de propósito general, lo que significa que puede utilizarse para desarrollar casi cualquier tipo de aplicación o software. Se utilizó la versión 3.10.0 de *Python*.

Python cuenta con varias características que lo hacen especialmente útil para nuestro proyecto:

- Dominio de la inteligencia artificial y el aprendizaje automático: *Python* es uno de los lenguajes más utilizados en el campo de la inteligencia artificial y el aprendizaje automático. Sus poderosas bibliotecas y marcos de trabajo para el aprendizaje automático y el análisis de datos, como *TensorFlow*, *PyTorch*, *Scikit-learn*, *Pandas*, y *NumPy*, hacen que la implementación de técnicas complejas de inteligencia artificial y análisis de datos sean mucho más sencillas.
- Enfoque multiparadigma: *Python* es un lenguaje multiparadigma, lo que significa que soporta varios estilos de programación, incluyendo programación orientada a objetos y programación funcional. Esto nos permitió escoger el estilo de programación que mejor se adaptaba a cada problema. Gracias a esto, realizamos tanto clases y subclases (utilizando patrones de diseño como *Strategy* y *Decorator*), como funciones *lambda*, para conectar todos los componentes del sistema.

²⁶ <https://www.python.org/>

- Facilidad de integración con bases de datos: *Python* proporciona librerías para poder integrar y comunicarse con *InfluxDB* y *Redis* de manera sencilla.
- Código eficiente y mantenible: *Python* es conocido por su sintaxis limpia y legible, que promueve la escritura de código que es eficiente y fácil de mantener. Al ser tan permisivo y dejar la responsabilidad de programar limpio y eficiente al desarrollador, cayó en nosotros la responsabilidad de programas lo más legible y eficiente posible. Gracias a esta permisibilidad, pude aplicar todo lo aprendido durante mi desarrollo en el grado.
- Interoperabilidad con *Docker* y otras tecnologías: *Python* se integra bien con *Docker* y *Docker Compose*, facilitando la gestión de múltiples contenedores y servicios. Además, *Python* es compatible con una amplia gama de otras tecnologías y plataformas que hemos usado durante el desarrollo del proyecto. *Python* facilita la integración de las aplicaciones desarrolladas con otros sistemas y servicios.

Python nos ha permitido implementar técnicas avanzadas de inteligencia artificial, manejar la comunicación con bases de datos, gestionar contenedores *Docker*, y mantener su código eficiente y fácil de mantener. Todo ello demuestra que *Python* es la elección acertada para el proyecto.



Ilustración 37. Logo de Python.

6.5.1.2. Shell Script

*Shell Script*²⁷ es un programa de computadora diseñado para ser ejecutado por el intérprete de comandos *Unix shell*. Un *shell* es un intérprete de comandos que proporciona una interfaz de usuario para sistemas operativos basados en *Unix*. Los *shell Scripts* pueden incluir comandos que se ejecutarían desde la línea de comandos, y son comúnmente utilizados para automatizar tareas repetitivas.

Aquí están algunas razones por las cuales hemos utilizado *Shell Script* en nuestro proyecto:

- Automatización: Una de las ventajas más significativas de los *shell Scripts* es la capacidad de automatizar tareas repetitivas. Si tienes una serie de comandos que necesitas ejecutar con regularidad, puedes escribir un *Script* para hacerlo por ti. Con esto se consigue ahorrar mucho tiempo y reducir la posibilidad de fallos.
- Control de flujo: Al igual que otros lenguajes de programación, los *shell Scripts* permiten estructuras de control de flujo, como bucles y condicionales. Esto permite escribir *Scripts* más complejos que pueden tomar decisiones basadas en las condiciones actuales en el sistema.
- Interacción con el sistema operativo: Los *shell Scripts* se ejecutan en el *shell* de un sistema operativo, lo que te da un control significativo sobre el sistema operativo mismo. Gracias a esto se puede manipular archivos, iniciar y detener servicios, y realizar otras tareas a nivel del sistema operativo.
- Portabilidad: Los *shell Scripts* pueden ejecutarse en cualquier sistema que tenga un intérprete de comandos compatible, lo que significa que puedes escribir un *Script* en un sistema y esperar que se ejecute en cualquier otro sistema *Unix-like*.
- Integración con otras herramientas: Muchas herramientas, especialmente en el ámbito del desarrollo y operaciones de software (*DevOps*), se pueden controlar a través de la línea de comandos, lo que significa que puedes utilizar *shell Scripts* para interactuar con estas herramientas y automatizar tareas comunes.

²⁷ https://es.wikipedia.org/wiki/Script_de_shell

En nuestro caso, únicamente hemos utilizado *Shell Script* para utilizar la herramienta *C/C-FlowMeter*, la cual, dependiendo de una variable del sistema, se ejecuta de diferentes formas.

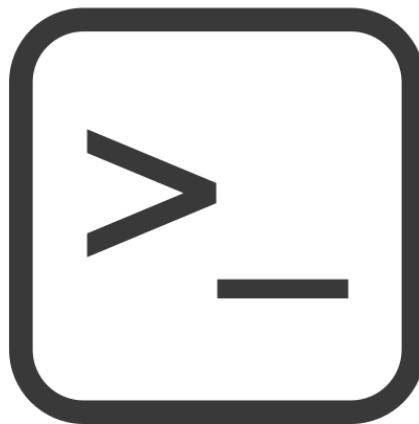


Ilustración 38. Logo de *Shell Script*.

6.5.1.3. YAML

*YAML (YAML Ain't Markup Language)*²⁸ es un formato de datos legible por humanos, comúnmente utilizado para la configuración de aplicaciones, pero también se utiliza en muchas aplicaciones donde se requieren datos a ser almacenados y transmitidos. YAML es un superconjunto de *JSON*²⁹, lo que significa que cualquier documento *JSON* válido también será un documento *YAML* válido.

Aquí están algunas razones por las cuales hemos decidido utilizar *YAML* en nuestro proyecto:

- Uso en *Docker* y *Docker Compose*: *Docker Compose* utiliza archivos *YAML* para definir servicios, redes y volúmenes. En un archivo *Compose*, puedes definir una aplicación multi-contenedor en un solo archivo, y luego lanzar la aplicación con un solo comando (“*Docker-compose up*”) y también, puedes detenerlo con un solo comando (“*Docker-compose down*”). Esto

²⁸ <https://yaml.org/>

²⁹ <https://www.JSON.org/JSON-es.html>

simplifica considerablemente la gestión de aplicaciones multi-contenedor.

- Legibilidad: *YAML* está diseñado para ser fácilmente legible por humanos. Su diseño simple y consistente hace que sea fácil de escribir y entender.
- Interoperabilidad: Dado que *YAML* es un superconjunto de *JSON*, es compatible con la mayoría de los lenguajes modernos de programación. Esto significa que puedes utilizar *YAML* en una variedad de contextos y con una variedad de herramientas, lo que nos fue útil durante el desarrollo del proyecto, ya que este es considerablemente grande y complejo (implica múltiples tecnologías).
- Soporte de estructuras complejas: *YAML* soporta estructuras de datos complejas como mapas, listas y escalares. También permite el uso de referencias, lo que puede ayudar a evitar la duplicación en los archivos de configuración.

Por lo tanto, *YAML* es una elección sólida para nuestro proyecto, ya que, facilita la configuración y la definición de servicios como *Docker* y *Docker Compose*, servicios que utilizamos en nuestro proyecto. Su simplicidad, legibilidad y flexibilidad lo convierten en una muy buena elección para la gestión de aplicaciones y servicios multi-contenedor.



Ilustración 39. Logo de *YAML*.

6.5.2. Entornos

En esta sección, se encuentran los entornos de ejecución utilizados durante el desarrollo del proyecto.

6.5.2.1. Visual Studio Code

*Visual Studio Code (VS Code)*³⁰ es un editor de código fuente desarrollado por Microsoft que se ha vuelto extremadamente popular en los últimos años. VS Code es ligero, rápido y gratuito, y está disponible para Windows, Mac y Linux. Es compatible con una amplia gama de lenguajes de programación y tecnologías lo que lo hace muy versátil. En este proyecto, se ha utilizado su versión 1.78.2 para el desarrollo de todo el sistema.

VS Code ofrece una serie de características útiles para nuestro proyecto:

- Soporte de *Python*: VS Code proporciona soporte integrado para *Python*, con características como resaltado de sintaxis, *linting* (búsqueda de fallos), depuración, *IntelliSense* (autocompletado), formateo de código, entre otros. Esto lo convierte en una herramienta ideal para cualquier proyecto de *Python*.
- Soporte de *Docker*: VS Code incluye soporte para *Docker* a través de una extensión, lo que nos permite construir, administrar y desplegar contenedores de *Docker* directamente desde el editor.
- Integración de *Git*³¹: VS Code se integra muy bien con *Git*, y con la extensión *GitLens*³², puedes ver quién hizo cada cambio en el código y cuándo.
- Extensible: VS Code admite una amplia gama de extensiones que permiten personalizar el entorno de desarrollo para satisfacer nuestras necesidades específicas. Estas extensiones nos ayudaron a mejorar nuestra productividad.

³⁰ <https://code.visualstudio.com/>

³¹ <https://git-scm.com/>

³² <https://gitlens.amod.io/>

- Edición en tiempo real y depuración: Con VS Code, se puede editar, ejecutar y depurar el código en tiempo real. Esto nos ayudó a identificar y solucionar problemas más rápidamente.
- Terminal integrada: VS Code tiene una terminal integrada, lo que significa que puedes ejecutar comandos de *shell* directamente desde el editor. Esto es especialmente útil, ya que, trabajamos con *Docker* y con herramientas de línea de comandos de *Python* como *pip*.

VS Code proporciona todas las herramientas que necesitas para desarrollar, probar, depurar y desplegar tu aplicación, todo en un solo lugar. Además, gracias a su gran comunidad y la amplia selección de extensiones disponibles hemos podido encontrar la ayuda que necesitamos en cada caso.

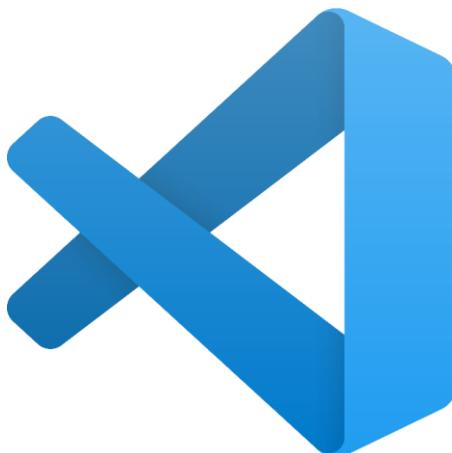


Ilustración 40. Logo de Visual Studio Code.

6.5.2.2. Jupyter Notebook

*Jupyter Notebook*³³ es una aplicación web de código abierto que permite crear y compartir documentos que contienen código en vivo, ecuaciones, visualizaciones y texto narrativo. Sus usos principales son la limpieza y transformación de datos, la simulación numérica, el modelado estadístico, la visualización de datos, el aprendizaje automático, entre otras. Es un ambiente de trabajo interactivo que permite a los usuarios escribir y ejecutar código en una variedad de lenguajes de programación, aunque es más popularmente utilizado con *Python*.

³³ <https://Jupyter.org/>

Jupyter Notebook tiene muchas características que lo hacen útil para nuestro proyecto:

- Código interactivo: *Jupyter* permite la ejecución de código en tiempo real, lo que facilita la experimentación, el “debuggeo” y la iteración rápida, aspectos clave en el proyecto.
- Narración de historias con datos: Puedes combinar casillas de código, con casillas de texto (*Markdown*³⁴). Esto favorece la legibilidad del código y la documentación del proyecto.
- Visualización de datos: *Jupyter* se integra bien con bibliotecas de visualización como *Matplotlib* y *Seaborn*, lo que te permite crear gráficos y visualizaciones interactivas para entender mejor los datos y resultados de los modelos.
- Compartir y reproducir: Los *Notebooks* de *Jupyter* son documentos JSON que pueden ser fácilmente compartidos, reproducidos, y versionados. Esto fue útil para la colaboración entre autor y cotutor.

Jupyter es una gran herramienta para la ciencia de datos y el *ML*, especialmente cuando se trabaja con *Python*. Te permite explorar y entender tus datos, construir, entrenar, y evaluar modelos de *ML*, visualizar resultados, y documentar tu proceso de una manera reproducible y compatible.

³⁴ <https://markdown.es/>



Ilustración 41. Logo de Jupyter Notebook.

6.5.3. Herramientas

En esta sección, se encuentran las herramientas y tecnologías utilizadas durante el desarrollo del proyecto.

6.5.3.1. Git

*Git*³⁵ es un sistema de control de versiones distribuido de código abierto que fue diseñado para manejar diferentes tipos de proyectos con velocidad y eficiencia. *Git* es fácil de aprender y alcanza un rendimiento muy alto.

En nuestro caso, usamos el entorno *Visual Studio Code*, el cual cuenta con un plugin llamado *GitLens*³⁶ que facilita el uso de *Git*. Otras razones por la que decidimos utilizar *Git* fueron:

- Seguimiento de cambios: *Git* te permite rastrear los cambios en el código a lo largo del tiempo. Esto significa que puedes ver exactamente qué cambios se hicieron, quién los hizo, y cuándo se hicieron. Esto nos fue útil para entender cómo ha evolucionado el código, también nos sirvió de gran utilidad para rastrear las fuentes de los fallos.

³⁵ <https://git-scm.com/>

³⁶ <https://gitlens.amod.io/>

- *Branching y Merging:* Puedes crear una nueva rama en el repositorio para trabajar en una nueva característica o solucionar un fallo, y luego puedes fusionar esos cambios de nuevo en la rama principal cuando estén listos.
- Integración con herramientas de desarrollo: *Git* se integra bien con una serie de herramientas de desarrollo, incluyendo *Visual Studio Code*, *GitLens* y *GitKraken*³⁷. *Visual Studio Code* es un editor de código que tiene soporte integrado para *Git*, permitiéndote realizar tareas comunes de *Git* directamente desde el editor. *GitLens* es una extensión de *Visual Studio Code* que mejora la funcionalidad de *Git* en el editor. *GitKraken* es un cliente de *Git* que proporciona una interfaz gráfica para interactuar con nuestros repositorios.

Git es una herramienta esencial para el desarrollo de software que permite un seguimiento de cambios eficiente, colaboración sin problemas, desarrollo paralelo a través del *branching* y *merging*, e integración con una serie de otras herramientas de desarrollo. Su uso es casi imprescindible en cualquier proyecto moderno de desarrollo de software.



Ilustración 42. Logo de *Git*.

6.5.3.2. Docker Hub

*Docker Hub*³⁸ es un servicio de hosting en la nube para imágenes de *Docker*. Proporciona una ubicación centralizada para que los desarrolladores suban, descarguen y compartan imágenes de *Docker*. Gracias a esto, se puede acceder a cualquier imagen de *Docker* con facilidad.

³⁷ <https://www.gitkraken.com/>

³⁸ <https://hub.docker.com/>

Hemos decidido utilizar *Docker Hub* en el desarrollo de proyecto por varias razones:

- Acceso a una amplia gama de imágenes de *Docker* preconfiguradas: *Docker Hub* alberga una vasta cantidad de imágenes oficiales de *Docker*, proporcionadas por proveedores de software, así como imágenes de la comunidad creadas por otros desarrolladores.
- Integración con *Docker* y *Docker Compose*: *Docker Hub* se integra de manera nativa con *Docker* y *Docker Compose*. Puedes especificar imágenes de *Docker Hub* en tu archivo *Docker Compose* y *Docker* las descargará automáticamente cuando necesites construir o desplegar tus contenedores.
- Automatización de compilaciones y pruebas: *Docker Hub* ofrece funcionalidades de integración y entrega continuas (*CI/CD*), permitiéndote automatizar la construcción de imágenes y la ejecución de pruebas cada vez que haces un cambio en el código.
- Seguridad y conformidad: *Docker Hub* ofrece características de seguridad, como el escaneo de imágenes de *Docker* para vulnerabilidades, lo que contribuye a mantener la integridad y seguridad del proyecto.

Como en nuestro proyecto usamos *Docker* y *Docker Compose*, utilizar *Docker Hub* nos facilita el manejo de las imágenes de *Docker* y de la integración con *Docker Compose*.



Ilustración 43. Logo de Docker Hub.

6.5.3.3. Draw.io

Draw.io³⁹ es una herramienta de dibujo y diagramación que permite crear diagramas de manera fácil y rápida. Es una aplicación que ofrece una amplia gama de opciones para diseñar y representar visualmente diversos tipos de diagramas, como diagramas de *Lenguaje Unificado de Modelado (UML)*, diagramas de paquetes, diagramas de entidad-relación, diagramas de red, diagramas de clase, entre otros.

Hemos decidido utilizar *Draw.io* en el desarrollo de **Apolo** por varias razones:

- Facilidad de uso: *Draw.io* es una herramienta intuitiva y de fácil manejo, lo que nos permitió crear y editar diagramas de forma sencilla.
- Amplia variedad de plantillas y elementos gráficos: *Draw.io* proporciona una amplia biblioteca de plantillas predefinidas y elementos gráficos que pueden ser arrastrados y soltados en el lienzo, lo que facilita la creación de diagramas de manera rápida y eficiente.
- Integración con otras herramientas y formatos: *Draw.io* es compatible con múltiples formatos de archivo, lo que facilita la importación y exportación de diagramas en diferentes formatos.
- Almacenamiento en la nube: *Draw.io* ofrece la posibilidad de almacenar los diagramas en la nube, lo que permite acceder a ellos desde cualquier lugar y en cualquier momento, sin necesidad de preocuparse por la pérdida de archivos o problemas de compatibilidad.

Al utilizar una herramienta especializada como *Draw.io*, se facilita la creación y modificación de los diagramas, lo que contribuye a un desarrollo más eficiente y organizado del proyecto **Apolo**.

³⁹ <https://app.diagrams.net/>



Ilustración 44. Logo de Draw.io

6.5.3.4. PlantUML

*PlantUML*⁴⁰ es una herramienta de código abierto que permite generar diagramas *UML* a partir de un lenguaje de descripción textual. Utiliza una sintaxis sencilla y legible para crear diagramas de clase, de despliegue, de secuencia, de actividad y otros tipos de diagramas *UML*.

Hemos decidido utilizar *PlantUML* en el desarrollo de **Apolo** por varias razones:

- Legibilidad y mantenibilidad del código: Al utilizar una sintaxis basada en texto para describir los diagramas *UML*, *PlantUML* proporciona una forma clara y legible de representar visualmente la estructura y relaciones del sistema. El código fuente de los diagramas se puede mantener en un repositorio de control de versiones, lo que facilita la colaboración y el seguimiento de cambios a lo largo del tiempo.
- Automatización y generación de diagramas: *PlantUML* permite generar automáticamente los diagramas *UML* a partir del código fuente. Al utilizar *PlantUML*, se pueden generar fácilmente los diagramas actualizados, ahorrando tiempo y esfuerzo.
- Flexibilidad y extensibilidad: Proporciona una amplia gama de características y opciones para personalizar los diagramas generados. Esto nos permitió adaptar los diagramas *UML* a las necesidades específicas del proyecto y mostrar la información relevante de manera clara y concisa.
- Documentación y comunicación: Los diagramas *UML* generados con *PlantUML* son una forma efectiva de documentar y comunicar la

⁴⁰ <https://plantuml.com/es/>

estructura y comportamiento del sistema **Apolo**. Proporcionan una representación visual clara y concisa que ayuda a comprender el diseño y la arquitectura del sistema.

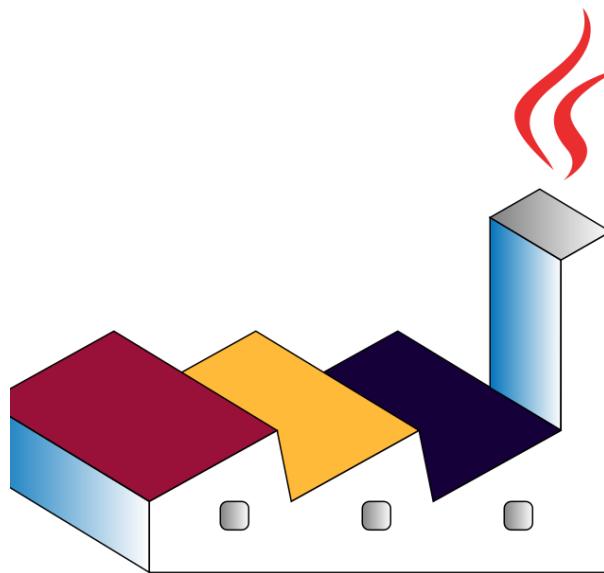


Ilustración 45. Logo de PlantUML

Capítulo 7. Evaluación de Alternativas

En este capítulo daremos las alternativas que hemos tenido en cuenta a la hora de decidir que lenguajes, herramientas y tecnologías utilizar durante el desarrollo del proyecto. También, valoramos diferentes enfoques que sopesamos al inicio del desarrollo sobre la implementación de **Apolo**.

Para realizar el análisis de las alternativas hemos realizado dos listas una de ventajas y otra de desventajas y al final una breve conclusión de por qué se decidió no utilizar dicha alternativa.

7.1. Alternativas a Apolo

En este apartado se encuentran las alternativas relativas al propio modelo de **Apolo**, describiremos diferentes aproximaciones que valoramos al inicio del desarrollo del sistema. La definición de **Apolo** se encuentra en el Capítulo 3. "Apolo".

7.1.1. Forecasting

El *forecasting*, o pronóstico, es una técnica que se utiliza para predecir el futuro basándose en los datos históricos. Este método es comúnmente utilizado en muchas áreas, como las finanzas, la planificación de ventas, la gestión de la cadena de suministro, el clima y muchas más.

Esta fue una de las principales alternativas que sopesamos, ya que, añadir un módulo de *forecasting* al sistema daría otra perspectiva de las anomalías del tráfico red.

Aplicar *forecasting* en los *IDS* no es algo nuevo [73], [74], pero nuestra idea no era únicamente aplicar *forecasting*, si no que fuera otro punto de vista para el análisis de las anomalías del sistema.

Este módulo actuaría como un predictor de futuro y la idea sería que mediante un sistema de ponderaciones se tuvieran en cuenta tanto el enfoque de *forecasting* como el enfoque de *MAB*.

El principal objetivo de añadir *forecasting* al sistema es el enfoque diferente que aportan este tipo de aproximaciones, se podría realizar un análisis en función de la estampa de tiempo de cada petición.

Realizar un análisis de series temporales aportaría una visión diferente frente a los posibles ataques que hagan al sistema (*DDoS*, *Botnet* ...), este análisis nos haría poder en qué momentos del día hay una mayor o menor carga de peticiones, con este dato se podría detectar anomalías en el uso del sistema.

Gracias al análisis de series temporales, se podría detectar una carga anómala en tiempo real, con dicha información, y haciendo uso del sistema de gestión de alertas, el administrador podría aplicar medidas en tiempo real para paliar esta posible amenaza.

El motivo por el que decidimos no aplicar este mecanismo al sistema es debido a que complicaba el desarrollo del sistema y se escapaba al tiempo dedicado al proyecto.

Aun así, esta implementación está recogida en el posible trabajo futuro.

7.1.2. Reentrenar con AML

Otro enfoque que se sopesó fue el reentrenamiento con datos generados de AML. Este enfoque consistía en reentrenar los clasificadores con datos generados haciendo uso de técnicas de AML, para ello se utilizaría una WGAN parecida a la creada para el paper realizado junto a mis directores “**Harpe: Using Adversarial Machine Learning to dodge Intrusion Detection Systems in Software-Defined Perimeters**”, pero no tan compleja en términos de entrenamiento.

Esto haría que nuestros clasificadores sean más robustos frente a este tipo de ataques, puesto que estarían entrenados con ellos.

El principal problema que se encontró a este enfoque fue que los datos generados con AML, serían datos benignos con ruido en algunos de sus atributos. Esto hasta podría desencadenar un resultado adverso al sistema, ya que daría más falsos positivos. Esto es debido a que, al entrenar los modelos con más datos, la proporción de datos benignos bajaría.

Otro problema que detectamos fue el posible sobreajuste que podría desencadenar este entrenamiento, ya que estaría especializándose en la WGAN que usemos para generar los datos, no generalizaría (presuntamente) a otras WGAN u otros métodos de generación de AML.

Esta alternativa es muy extendida en la literatura [7], [8], [75], [76].

Aun así, esta implementación está recogida en el posible trabajo futuro.

7.2. Alternativas a los Componentes de la Arquitectura

En esta sección se encuentran las alternativas relativas a los componentes de la arquitectura del sistema, dichos componentes se encuentran definidos dentro del Capítulo 6. “Definición del Sistema”, más concretamente en la sección 6.2. “Definición de los Componentes de la Arquitectura”.

7.2.1. Firewall y Proxy Reverso

En esta sección se encuentran las alternativas al componente de firewall y proxy reverso.

7.2.1.1. Apache HTTP Server

*Apache HTTP Server*⁴¹ es un popular servidor web de código abierto ampliamente utilizado en diversos sistemas.



Ilustración 46. Logo de Apache HTPP Server.

A continuación, se presentan algunas ventajas y desventajas asociadas a su uso en este contexto.

7.2.1.1.1. Ventajas

- Estabilidad y fiabilidad: *Apache HTTP Server* tiene una larga trayectoria y una reputación de estabilidad y fiabilidad. Ha sido ampliamente probado y utilizado en entornos de producción durante muchos años, lo que

⁴¹ <https://httpd.apache.org/>

demuestra su capacidad para manejar altas cargas de tráfico web y mantener la disponibilidad del sistema.

- Alto rendimiento: *Apache HTTP Server* está diseñado para ofrecer un alto rendimiento, lo que es esencial para un sistema de monitorización y gestión de anomalías del tráfico web en tiempo real. Puede manejar eficientemente solicitudes concurrentes y responder rápidamente a las peticiones, minimizando la latencia y garantizando una experiencia fluida para los usuarios finales.
- Flexibilidad y personalización: *Apache HTTP Server* ofrece una amplia gama de módulos y configuraciones que permiten una gran flexibilidad y personalización. Puede ser adaptado para cumplir con los requisitos específicos del sistema de monitorización y gestión de anomalías, permitiendo la implementación de reglas y filtros personalizados.
- Amplia comunidad de soporte: *Apache HTTP Server* cuenta con una gran comunidad de desarrolladores y usuarios que proporcionan soporte y contribuyen con mejoras continuas. Esto garantiza la disponibilidad de recursos, documentación y posibles soluciones en caso de problemas o necesidades específicas.

7.2.1.1.2. Desventajas

- Configuración inicial compleja: La configuración inicial de *Apache HTTP Server* puede resultar compleja, especialmente si se requiere una personalización exhaustiva o la implementación de características avanzadas. Se necesita un conocimiento técnico sólido y experiencia para realizar una configuración adecuada.
- Gestión y mantenimiento continuos: Un servidor Apache HTTP requiere una gestión y mantenimientos continuos para garantizar su seguridad y rendimiento óptimo. Esto incluye la aplicación de parches de seguridad, actualizaciones regulares y supervisión constante para identificar y resolver posibles problemas.
- Consumo de recursos: *Apache HTTP Server* puede requerir una cantidad significativa de recursos de hardware, como memoria y capacidad de procesamiento, especialmente cuando se enfrenta a cargas de tráfico

webs intensivas. Esto puede aumentar los requisitos de infraestructura y los costos asociados.

- Limitaciones en la escalabilidad horizontal: Si bien Apache HTTP Server puede escalar verticalmente para manejar cargas de tráfico más altas, la escalabilidad horizontal puede ser un desafío. Agregar nuevos nodos y equilibrar la carga entre ellos puede requerir una configuración adicional y la implementación de soluciones de balanceo de carga.

7.2.1.1.3. Conclusión

Apache HTTP Server es una opción muy popular para servir sitios web, pero hemos decidido descartarla, ya que, no se adecua a las necesidades que tenemos en el proyecto como son su configuración compleja y sus posibles problemas de rendimiento, ya que, el rendimiento es una característica crucial en nuestro sistema.

7.2.1.2. Caddy

Caddy⁴² es un servidor web y un proxy reverso de código abierto, escrito en Go⁴³, que se utiliza para servir sitios web y aplicaciones web. Fue diseñado con la simplicidad y la seguridad en mente, y es conocido por ser el primer servidor web en habilitar HTTPS automáticamente y por defecto.

⁴² <https://Caddyserver.com/docs/quick-starts/reverse-proxy>

⁴³ <https://go.dev/>



Caddy®
THE ULTIMATE SERVER

Ilustración 47. Logo de Caddy.

A continuación, se presentan algunas ventajas y desventajas asociadas a su uso en este proyecto.

7.2.1.2.1. Ventajas

- Configuración sencilla: *Caddy* se destaca por su configuración fácil y sencilla. Utiliza una sintaxis clara y concisa que simplifica el proceso de configuración del servidor web y del proxy reverso. Esto puede ahorrar tiempo y esfuerzo en la implementación y mantenimiento del sistema.
- Automatización de certificados SSL/TLS: *Caddy* cuenta con integración nativa con *Let's Encrypt*, lo que permite obtener y renovar automáticamente certificados SSL/TLS para los sitios web protegidos. Esto simplifica la configuración de seguridad y garantiza conexiones cifradas.
- Soporte nativo para *HTTP/2* y *HTTPS*: *Caddy* ofrece soporte nativo para *HTTP/2* y *HTTPS*, lo que permite aprovechar las últimas mejoras en rendimiento y seguridad de los protocolos web. Esto puede mejorar la velocidad de carga de las páginas y la seguridad de la comunicación.
- Interfaz de usuario intuitiva: *Caddy* proporciona una interfaz de usuario web intuitiva que facilita la configuración y administración del servidor. Esta interfaz simplifica las tareas de monitoreo y gestión de anomalías.

del tráfico web, brindando una experiencia más amigable para los administradores.

7.2.1.2.2. Desventajas

- Menos opciones de personalización: Aunque *Caddy* es conocido por su configuración sencilla, esto puede limitar las opciones de personalización en comparación con otros servidores web más avanzados. Si se requieren configuraciones altamente personalizadas, puede ser necesario recurrir a alternativas más flexibles.
- Menor cantidad de complementos y módulos: *Caddy* tiene una selección más limitada de complementos y módulos en comparación con servidores web más establecidos como *NGINX* o *Apache*. Esto puede ser una limitación si se necesitan funcionalidades específicas o integraciones con tecnologías específicas.
- Rendimiento: Si bien *Caddy* es eficiente para servir sitios web y aplicaciones, el monitoreo constante de las peticiones web para detectar anomalías podría afectar su rendimiento. Otras soluciones más dedicadas están diseñadas para manejar este tipo de carga sin afectar el rendimiento del sistema.
- Seguridad: Aunque *Caddy* es seguro por diseño, no está diseñado para proteger contra ataques específicos o para detectar comportamientos anómalos en las peticiones web.

7.2.1.2.3. Conclusión

Caddy es una excelente opción para servir sitios web y aplicaciones, pero puede que no sea la mejor opción para un sistema de monitoreo y gestión de anomalías en peticiones web debido a su falta de funcionalidades específicas, su extensibilidad limitada, posibles problemas de rendimiento. La característica que nos hace tomar la decisión final es su enfoque de seguridad menos robusto en comparación con otras soluciones, ya que, la seguridad es de vital importancia para nuestro sistema.

7.2.2. Gestión de peticiones

En esta sección se encuentran las alternativas al componente de gestión de peticiones.

7.2.2.1. Fluentd, Grafana y Prometheus

*Fluentd*⁴⁴, *Grafana*⁴⁵ y *Prometheus*⁴⁶ son herramientas de código abierto que, cuando se combinan, pueden proporcionar una solución de monitoreo y gestión de anomalías en peticiones web.

Fluentd: Es un colector de datos y un administrador de logs de código abierto. Su trabajo principal es unificar la recogida y el consumo de datos para una mejor utilización y comprensión de estos. *Fluentd* ofrece flexibilidad y permite que los datos se transfieran entre diferentes tipos de sistemas de almacenamiento, índices y aplicaciones de visualización de datos.



Ilustración 48. Logo de Fluentd.

Grafana: Es una herramienta de visualización de datos de código abierto que se utiliza para el monitoreo en tiempo real de métricas de sistemas distribuidos. *Grafana* permite a los usuarios crear, explorar y compartir dashboards que muestran los datos de las aplicaciones, la infraestructura y el sensor. *Grafana* es compatible con una variedad de bases de datos, incluyendo *Prometheus*, *MySQL*, y *Elasticsearch*, entre otras.

⁴⁴ <https://docs.fluentd.org/v/0.12/articles/monitoring-prometheus>

⁴⁵ <https://grafana.com/grafana/dashboards/13042-fluentd-1-x/>

⁴⁶ <https://prometheus.io/>



Ilustración 49. Logo de Grafana.

Prometheus: Es un sistema de monitoreo y alerta de código abierto que se utiliza para el monitoreo de métricas en tiempo real. *Prometheus* recoge métricas de sus objetivos a intervalos específicos, evalúa las reglas de alerta, muestra los resultados y puede activar alertas si alguna condición se observa como verdadera. Es muy eficiente y se utiliza en muchos sistemas de monitoreo de microservicios debido a su modelo de datos multidimensional, su consulta potente y su capacidad para integrarse con una variedad de fuentes de datos.



Ilustración 50. Logo de Prometheus.

A continuación, se presentan algunas ventajas y desventajas asociadas a su uso en este proyecto.

7.2.2.1.1. Ventajas

- Código abierto: Todas estas herramientas son de código abierto, lo que significa que son gratuitas y tienen una comunidad activa de desarrolladores que las respaldan.
- Flexible y extensible: *Fluentd* puede recoger datos de una variedad de fuentes y enviarlos a múltiples destinos. *Prometheus* es un sistema de monitoreo y alerta que puede manejar métricas multidimensionales. *Grafana* permite visualizar los datos de múltiples fuentes en un solo panel.
- Escalable: Estas herramientas están diseñadas para ser escalables y pueden manejar grandes volúmenes de datos.
- Integración: *Fluentd*, *Grafana* y *Prometheus* se integran bien entre sí y con otras herramientas y plataformas.

7.2.2.1.2. Inconvenientes

- Complejidad de configuración: Aunque estas herramientas son poderosas, pueden ser complejas de configurar y mantener, especialmente en entornos grandes y dinámicos.
- Soporte limitado: Aunque estas herramientas tienen una comunidad activa de desarrolladores, el soporte puede ser limitado en comparación con las soluciones comerciales.
- Rendimiento: Si bien estas herramientas son escalables, el monitoreo constante de las peticiones web para detectar anomalías podría afectar su rendimiento, especialmente en entornos con mucho tráfico.

7.2.2.1.3. Conclusión

Aunque *Fluentd*, *Grafana* y *Prometheus* pueden proporcionar una visión valiosa de los datos de tráfico web, no son la mejor opción para el proyecto. Esto es debido a la complejidad de su configuración y mantenimiento, posibles problemas de rendimiento y soporte limitado; siendo los problemas de rendimiento la característica que nos hace descartar esta opción, ya que, para nuestro proyecto es de vital importancia esta característica.

7.2.2.2. Graylog

Graylog⁴⁷ es una plataforma de gestión de logs de código abierto que ayuda a los desarrolladores a recoger, indexar y analizar los datos de logs de sus aplicaciones y sistemas. Graylog se utiliza comúnmente para el monitoreo de aplicaciones, la seguridad de la información y la resolución de problemas de infraestructura.



Ilustración 51. Logo de Graylog.

A continuación, se presentan algunas ventajas y desventajas asociadas a su uso en este proyecto.

7.2.2.2.1. Ventajas

- Recopilación y búsqueda eficiente de registros: Puede recibir y procesar grandes volúmenes de registros, lo que facilita la gestión de un alto tráfico web y la detección de anomalías de forma rápida y precisa.
- Interfaz de usuario intuitiva: Graylog ofrece una interfaz de usuario intuitiva y fácil de usar. Permite visualizar y analizar los registros de manera clara y estructurada, facilitando la identificación de patrones, problemas y anomalías en el tráfico web.
- Funcionalidades de alerta y notificación: Graylog permite configurar alertas personalizadas basadas en eventos y condiciones específicas. Esto permite recibir notificaciones en tiempo real cuando se detectan anomalías en el tráfico web, lo que ayuda a tomar medidas rápidas y mitigar posibles amenazas.

⁴⁷ <https://www.Graylog.org/>

- Integración con otras herramientas: *Graylog* se integra fácilmente con otras herramientas de monitorización y análisis, como *Grafana* y *Elasticsearch*. Esto permite una mayor flexibilidad y la posibilidad de aprovechar características adicionales para visualizar y analizar los registros de forma más avanzada.

7.2.2.2.2. Desventajas

- Configuración inicial compleja: La configuración inicial de *Graylog* puede ser compleja, especialmente en escenarios avanzados o en la integración con diferentes sistemas y fuentes de datos. Requiere un conocimiento técnico adecuado para aprovechar al máximo todas las funcionalidades y realizar configuraciones personalizadas.
- Requerimientos de infraestructura: *Graylog* puede requerir recursos de hardware y capacidad de almacenamiento considerables, especialmente cuando se trata de grandes volúmenes de registros y tráfico web. Es importante asegurarse de tener la infraestructura adecuada para admitir la carga de trabajo requerida.
- Curva de aprendizaje inicial: Al ser una plataforma completa de gestión de registros, *Graylog* puede tener una curva de aprendizaje inicial para usuarios nuevos. La comprensión de los conceptos, la configuración y las mejores prácticas puede llevar tiempo y esfuerzo adicional.
- Dependencia de Elasticsearch: *Graylog* depende de Elasticsearch para la indexación y búsqueda de logs.

7.2.2.2.3. Conclusión

Graylog puede proporcionar una visión valiosa de los datos de tráfico web, pero no es la mejor opción para nuestro sistema. Esto se debe principalmente a la dependencia con *Elasticsearch*, para utilizar un gestor de peticiones que depende de *Elasticsearch*, preferimos utilizar el Stack de *ELK*, el cual está desarrollado y optimizado para usar en conjunto.

7.2.3. Clasificación

En esta sección se encuentran las alternativas al componente de clasificación, más concretamente los componentes del sistema de colas y el gestor de anomalías, las alternativas a **Apolo** están recogidas en el apartado 7.1. “Alternativas a Apolo”, de este mismo capítulo.

7.2.3.1. Sistema de colas

En esta sección se encuentran las alternativas al componente de sistema de colas.

7.2.3.1.1. Hazelcast

Hazelcast⁴⁸ es una plataforma de computación en memoria de código abierto que proporciona una variedad de características de almacenamiento y procesamiento de datos distribuidos. Está diseñada para ser altamente escalable y proporcionar un rendimiento de baja latencia, lo que la hace ideal para aplicaciones que requieren un procesamiento de datos rápido y en tiempo real.



Ilustración 52. Logo de Hazelcast.

A continuación, se presentan algunas ventajas y desventajas asociadas a su uso en este proyecto.

⁴⁸ <https://hazelcast.com/>

7.2.3.1.1.1. Ventajas

- Alta escalabilidad: Es altamente escalable y puede manejar fácilmente el crecimiento de los datos y las demandas de procesamiento simplemente añadiendo más nodos al *clúster*.
- Almacenamiento en memoria: Almacena los datos en la memoria (en lugar de en el disco), lo que permite un acceso y un procesamiento de datos mucho más rápidos.
- Resiliencia: Está diseñado para ser tolerante a fallos y la replicación de datos, lo que significa que, si un nodo experimenta un problema, otros nodos pueden continuar realizando las operaciones sin interrupción, garantizando la disponibilidad y la continuidad del sistema.
- Cómputo distribuido: Permite ejecutar tareas y algoritmos en paralelo en los nodos de un *clúster*, lo que puede mejorar significativamente el rendimiento de las operaciones de procesamiento de datos.
- Integración: Se puede integrar con una variedad de lenguajes de programación y tecnologías, incluyendo *Python*.

7.2.3.1.1.2. Desventajas

- Complejidad de configuración: La configuración inicial y el despliegue de un *clúster* de *Hazelcast* pueden ser complejos, lo que puede requerir conocimientos técnicos y un tiempo adicional de implementación y puesta en marcha.
- Costo de recursos: Dado que *Hazelcast* almacena los datos en memoria, puede consumir una cantidad significativa de recursos del sistema, especialmente en entornos con mucho tráfico o grandes volúmenes de datos.
- Curva de aprendizaje: Si el equipo no está familiarizado con *Hazelcast*, puede llevar tiempo aprender y familiarizarse con sus conceptos y características específicas, lo que puede requerir capacitación adicional o soporte externo.

- Mantenimiento y gestión: Al utilizar Hazelcast en un entorno de producción, se requerirá un mantenimiento regular y una gestión adecuada del clúster para garantizar su rendimiento, disponibilidad y seguridad.
- Falta de persistencia de datos: Aunque Hazelcast es excelente para el procesamiento en tiempo real, no está diseñado para el almacenamiento a largo plazo de datos. Si necesitas mantener los datos de registro durante un período de tiempo prolongado para el análisis histórico, podrías necesitar una solución de almacenamiento de datos adicional.

7.2.3.1.1.3. Conclusión

En conclusión, aunque Hazelcast puede proporcionar un rendimiento de baja latencia y manejar grandes volúmenes de datos, puede no ser la mejor opción para el sistema debido a la complejidad de su configuración y mantenimiento, su consumo de recursos y su falta de persistencia de datos, sobre todo esta última, ya que, es de vital importancia que haya una buena persistencia de datos.

7.2.3.1.2. Azure Cosmos DB

Azure Cosmos DB⁴⁹ es un servicio de base de datos multimodelo y distribuido globalmente de Microsoft Azure. Proporciona escalabilidad horizontal, alta disponibilidad y latencia baja, y permite a los desarrolladores trabajar con datos en una variedad de modelos de datos populares.



Ilustración 53. Logo de Azure Cosmos DB.

⁴⁹ <https://Azure.microsoft.com/es-es/products/cosmos-db>

A continuación, se presentan algunas ventajas y desventajas asociadas a su uso en este proyecto.

7.2.3.1.2.1. Ventajas

- Escalabilidad global: Permite escalar horizontalmente la capacidad y el rendimiento de la base de datos a nivel global, lo que permite gestionar grandes volúmenes de peticiones en tiempo real.
- Baja latencia: Con la capacidad de distribuir los datos en múltiples regiones geográficas, *Azure Cosmos DB* ofrece una baja latencia en el acceso a los datos, lo que es esencial para un sistema de monitorización y gestión de peticiones en tiempo real.
- Alta disponibilidad: Ofrece una alta disponibilidad de los datos (99.999% de disponibilidad para lecturas y escrituras) y garantiza la latencia baja en cualquier lugar del mundo.
- Consistencia múltiple: *Cosmos DB* ofrece cinco modelos de consistencia, desde “eventual” hasta “fuerte”, lo que te permite elegir el nivel de consistencia que mejor se adapte a tus necesidades.
- Integración con el ecosistema de *Azure*: Como parte de la familia de productos de *Azure*, *Cosmos DB* se integra bien con otros servicios de *Azure*.
- Multimodelo: *Cosmos DB* es una base de datos multimodelo, lo que significa que puede manejar datos de documentos, datos de gráficos, datos clave-valor y datos de columnas anchas.

7.2.3.1.2.2. Desventajas

- Costo: El uso de *Azure Cosmos DB* puede resultar costoso en comparación con otras opciones de bases de datos, especialmente cuando se manejan grandes volúmenes de datos o se requiere un alto rendimiento.
- Curva de aprendizaje: *Azure Cosmos DB* tiene una curva de aprendizaje, especialmente si no se está familiarizado con el servicio o los modelos de

datos no relacionales. Se requiere tiempo y esfuerzo para comprender y utilizar eficazmente todas las características y capacidades que ofrece.

- Dependencia de la nube de Azure: Si se elige *Azure Cosmos DB*, es importante tener en cuenta que existe una dependencia de la infraestructura y servicios de la nube de Azure. Esto puede limitar la portabilidad y la flexibilidad en términos de alojamiento y migración a otras plataformas.

7.2.3.1.2.3. Conclusión

Azure Cosmos DB puede proporcionar un rendimiento de baja latencia y manejar grandes volúmenes de datos, pero no es la mejor opción para nuestro sistema debido a su costo potencialmente alto, la complejidad de su configuración y mantenimiento, y su dependencia de Azure. Principalmente es debido a su alto coste, ya que este proyecto no tiene tanto presupuesto.

7.2.3.2. Gestión de Anomalías

En esta sección se encuentran las alternativas al componente de gestión de anomalías.

7.2.3.2.1. Prometheus

*Prometheus*⁵⁰ es un sistema de monitoreo y alerta de código abierto que se utiliza para el monitoreo de métricas en tiempo real. Fue desarrollado por SoundCloud y es ahora parte del proyecto *Cloud Native Computing Foundation*, que incluye proyectos populares como Kubernetes y Fluentd.



Ilustración 54. Logo de Prometheus.

⁵⁰ <https://prometheus.io/>

A continuación, se presentan algunas ventajas y desventajas asociadas a su uso en este proyecto.

7.2.3.2.1.1. Ventajas

- Arquitectura escalable: Utiliza una arquitectura escalable que permite manejar grandes volúmenes de datos de métricas en tiempo real. Puede adaptarse fácilmente a entornos de alta disponibilidad y escalar verticalmente para satisfacer las demandas de monitoreo en sistemas complejos.
- Recolección y almacenamiento de métricas flexibles: Ofrece una amplia gama de opciones para la recolección de métricas, incluyendo el propio agente de recolección de *Prometheus* y bibliotecas de cliente para diferentes lenguajes de programación. Además, almacena todas las métricas recogidas en el disco local para un análisis a largo plazo.
- Consultas y alertas flexibles: Proporciona una poderosa consulta de datos y un lenguaje de alertas que permite realizar consultas complejas y personalizadas sobre las métricas recopiladas. Esto facilita la detección temprana de anomalías y la configuración de alertas para notificar a los administradores del sistema.
- Integración con sistemas de visualización: *Prometheus* se integra con diferentes sistemas de visualización, como *Grafana*, lo que permite crear paneles y gráficos personalizados para visualizar y analizar las métricas en tiempo real. Esto facilita la comprensión y el seguimiento de los patrones y tendencias de las anomalías.

7.2.3.2.1.2. Desventajas

- Almacenamiento a largo plazo: *Prometheus* no está diseñado para el almacenamiento a largo plazo de datos. Si necesitas mantener los datos de registro durante un período de tiempo prolongado para el análisis histórico, podrías necesitar una solución de almacenamiento de datos adicional.
- Escalabilidad horizontal: Aunque *Prometheus* es escalable verticalmente, la escalabilidad horizontal puede ser un desafío. Agregar nuevos nodos requiere una configuración y gestión adicionales, y puede haber ciertos

límites en términos de la cantidad de métricas y datos que pueden ser procesados por un *clúster*.

- Configuración inicial compleja: La configuración inicial de *Prometheus* puede ser compleja, especialmente si se requiere una configuración personalizada para adaptarse a las necesidades específicas del sistema de monitorización y gestión de anomalías. Esto puede requerir conocimientos técnicos y tiempo adicional de implementación.

7.2.3.2.1.3. Conclusión

Prometheus puede proporcionar un monitoreo de métricas en tiempo real y manejar una gran cantidad de métricas, pero no es la mejor opción para el sistema debido a los desafíos con el almacenamiento a largo plazo de datos, la complejidad de su configuración y mantenimiento, y los posibles desafíos de escalabilidad. Principalmente es debido a los problemas de almacenamiento a largo plazo.

7.2.3.2.2. TimescaleDB

*Timescale*⁵¹ es una base de datos de series temporales de código abierto que está diseñada para manejar datos de series temporales a gran escala. Se construye como una extensión de *PostgreSQL*, lo que significa que hereda todas las características de una base de datos relacional completa, incluyendo el soporte para transacciones, la consistencia de datos, un lenguaje de consulta potente (*SQL*) y la capacidad de unir datos de series temporales con otros datos relacionales.



Timescale

Ilustración 55. Logo de Timescale.

⁵¹ <https://www.Timescale.com/>

A continuación, se presentan algunas ventajas y desventajas asociadas a su uso en este proyecto.

7.2.3.2.2.1. Ventajas

- Optimizada para series temporales: Está diseñada específicamente para el almacenamiento y la consulta eficiente de datos de series temporales. Proporciona estructuras de índices y algoritmos optimizados que permiten un rápido acceso y análisis de datos en tiempo real.
- Escalabilidad: Está diseñado para manejar grandes volúmenes de datos de series temporales y proporciona una escalabilidad horizontal y vertical.
- Funcionalidades SQL completas: A diferencia de muchas otras bases de datos de series temporales, *TimescaleDB* soporta SQL completo, lo que permite a los usuarios realizar consultas complejas y unir datos de series temporales con otros datos relacionales.
- Compresión de datos: *TimescaleDB* ofrece una compresión de datos nativa, lo que puede reducir significativamente el almacenamiento de datos y mejorar la velocidad de las consultas.
- Compatibilidad con PostgreSQL: Como extensión de PostgreSQL, *TimescaleDB* es compatible con una amplia gama de herramientas y bibliotecas.

7.2.3.2.2.2. Desventajas

- Complejidad de configuración y mantenimiento: Aunque *TimescaleDB* ofrece muchas características, configurar y mantener una base de datos *TimescaleDB* puede ser complejo y requerir un conocimiento técnico considerable. Esto puede ser especialmente desafiante en entornos grandes y dinámicos.
- Consumo de recursos: Aunque *TimescaleDB* es eficiente en términos de almacenamiento y rendimiento de consultas, aún puede consumir una cantidad significativa de recursos del sistema, especialmente en entornos con mucho tráfico o grandes volúmenes de datos.

- Dependencia de PostgreSQL: *TimescaleDB* es una extensión de *PostgreSQL*, lo que significa que hereda tanto las fortalezas como las debilidades de *PostgreSQL*. Si bien esto proporciona una gran cantidad de funcionalidades, también puede introducir complejidad adicional y posibles limitaciones, especialmente en términos de escalabilidad y rendimiento para grandes volúmenes de datos.
- Falta de funcionalidades: No cumple todas las necesidades que necesitamos para el correcto funcionamiento del componente, funcionalidades como el monitoreo y la gestión de alertas entre otras.

7.2.3.2.2.3. Conclusión

Aunque *TimescaleDB* puede proporcionar un rendimiento de baja latencia y manejar grandes volúmenes de datos, no es la mejor opción para nuestro sistema debido a la complejidad de su configuración y mantenimiento, su dependencia de *PostgreSQL* y su consumo potencial de recursos. Concretamente la falta de funcionalidades para nuestro problema en concreto es lo que nos hace tomar esta decisión.

7.3. Alternativas Generales

En esta sección se valorarán diferentes alternativas a los lenguajes, herramientas y entornos utilizados durante la realización del proyecto. Estas se encuentran definidas dentro del Capítulo 6. “Definición del Sistema”, más concretamente en la Sección 6.1. “Microservicios” y la sección 6.5. “Lenguajes, Entornos y Herramientas”.

7.3.1. Microservicios

En esta sección se encuentran las alternativas a las tecnologías de microservicios utilizadas.

7.3.1.1. Podman

*Podman*⁵² es una herramienta de código abierto que permite administrar contenedores y *pods* sin la necesidad de un *daemon*. Fue desarrollada por Red Hat⁵³ y es una alternativa a *Docker*. *Podman* puede manejar todas las características de *Docker* y añade varias mejoras.



Ilustración 56. Logo de Podman.

A continuación, se presentan algunas ventajas y desventajas asociadas a su uso en este contexto.

7.3.1.1.1. Ventajas

- Compatibilidad con *Docker*: *Podman* es completamente compatible con las imágenes y contenedores de *Docker*. Esto significa que puede reutilizar las imágenes de *Docker* existentes y ejecutarlas con *Podman* sin necesidad de realizar cambios significativos.

⁵² <https://Podman.io/>

⁵³ <https://www.redhat.com/es>

- No requiere un *daemon*: A diferencia de *Docker*, *Podman* no requiere un *daemon* en ejecución. Esto mejora la seguridad y simplifica la gestión de contenedores.
- Ejecución de contenedores como usuarios sin privilegios: Permite ejecutar contenedores como usuarios sin privilegios, lo que mejora la seguridad al reducir la superficie de ataque en caso de una brecha en el contenedor.
- Integración con herramientas de *Kubernetes*⁵⁴: Está diseñado para trabajar con *Kubernetes* y *OpenShift*⁵⁵, lo que puede ser útil si decide escalar su sistema de monitorización a un entorno de *Kubernetes*.
- Soporte para *pods*: *Podman* puede agrupar varios contenedores en un *pod*, lo que facilita la gestión de aplicaciones que constan de varios contenedores.
- Integración con otras herramientas de *Red Hat*: Se integra bien con otras herramientas de *Red Hat*, como *Buildah*⁵⁶ para la construcción de imágenes de contenedores y *Skopeo*⁵⁷ para la distribución de imágenes.

7.3.1.1.2. Desventajas

- Menos maduro que *Docker*: *Docker* sigue siendo más maduro y tiene una comunidad más grande y un ecosistema más establecido. Esto puede traducirse en más recursos de aprendizaje y soporte disponibles para *Docker*.
- Falta de algunas características avanzadas de *Docker*: *Podman* es compatible con las características básicas de *Docker*, pero puede que no soporte todas las características avanzadas de *Docker*.
- Diferencias en la interfaz de línea de comandos: Existen algunas diferencias, con la interfaz de *Docker*, que podrían requerir cambios en los Scripts y las herramientas existentes.

⁵⁴ <https://kubernetes.io/es/>

⁵⁵ <https://www.redhat.com/en/technologies/cloud-computing/openshift>

⁵⁶ <https://buildah.io/>

⁵⁷ <https://www.redhat.com/es/topics/containers/what-is-skopeo>

- Potenciales problemas de rendimiento con modelos de *ML*: Dependiendo de cómo estén configurados los modelos de *ML* y algoritmos de agrupamiento y selección, podría experimentar problemas de rendimiento al ejecutarlos en contenedores *Podman*.

7.3.1.1.3. Conclusión

En conclusión, *Podman* es una tecnología para tener en cuenta por su rápido crecimiento en los últimos tiempos, pero aún le falta bagaje y madurez, lo que nos hace decantarnos por el uso de *Docker*.

7.3.2. Entornos

En esta sección se encuentran las alternativas a los entornos utilizados.

7.3.2.1. PyCharm

*PyCharm*⁵⁸ es un entorno de desarrollo integrado (*IDE*) para *Python* desarrollado por *JetBrains*. Proporciona una serie de herramientas y características que facilitan la programación en *Python*, incluyendo la edición de código, la depuración, el análisis de código, la integración con sistemas de control de versiones y la gestión de entornos virtuales de *Python*.

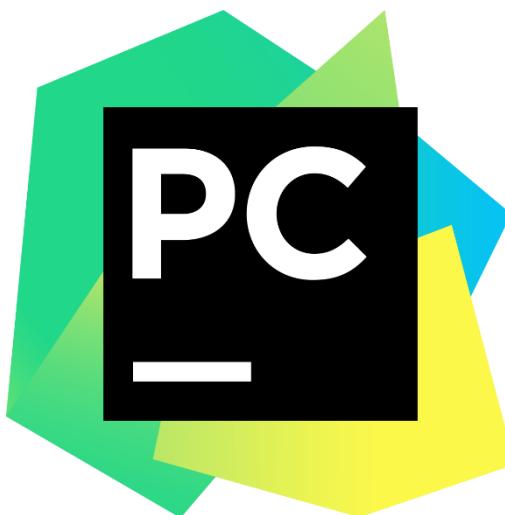


Ilustración 57. Logo de PyCharm.

⁵⁸ <https://www.jetbrains.com/es-es/PyCharm/>

A continuación, se presentan algunas ventajas y desventajas asociadas a su uso en este contexto.

7.3.2.1.1. Ventajas

- Entorno de desarrollo integrado (*IDE*) completo: *PyCharm* ofrece un entorno de desarrollo completo para programar en *Python*. Proporciona características avanzadas como finalización de código, resaltado de sintaxis, depuración, refactorización, control de versiones, entre otros. Esto mejora la productividad y facilita el desarrollo de aplicaciones *Python*.
- Soporte completo para *Python*: *PyCharm* está diseñado específicamente para *Python* y ofrece un soporte completo para el lenguaje. Tiene una integración profunda con el intérprete de *Python*, lo que permite un desarrollo más eficiente y la detección de errores en tiempo real.
- Amplia gama de funcionalidades: *PyCharm* ofrece una amplia gama de características y herramientas que facilitan el desarrollo de proyectos *Python* de cualquier escala. Esto incluye soporte para *virtualenv*, administración de paquetes, pruebas unitarias, análisis de código, autocompletado inteligente, generación de documentación y más. Además, tiene una amplia gama de complementos y extensiones disponibles para personalizar y ampliar sus capacidades.
- Integración con otras herramientas y servicios: *PyCharm* se integra con una variedad de herramientas y servicios populares en el ecosistema de desarrollo de *Python*. Esto incluye integración con sistemas de control de versiones como *Git*, integración con frameworks y bibliotecas populares, como *NumPy* y *Pandas*, y compatibilidad con herramientas de implementación y automatización, como *Docker*.

7.3.2.1.2. Desventajas

- Consumo de recursos: *PyCharm* es una aplicación robusta y completa, lo que puede llevar a un consumo relativamente alto de recursos de la computadora. Es especialmente notable cuando se trabaja en proyectos grandes o con múltiples proyectos abiertos simultáneamente. Esto puede

afectar el rendimiento general del sistema, especialmente en máquinas con recursos limitados.

- Licencia de pago: Aunque *PyCharm Community Edition* es de código abierto y gratuito, algunas de las características más avanzadas y potentes están disponibles en *PyCharm Professional Edition*, que es una licencia comercial de pago.

7.3.2.1.3. Conclusión

PyCharm es muy útil para el desarrollo de software en *Python*, pero no es la mejor opción para nuestro sistema debido a su consumo potencial de recursos del sistema y su costo. Sobre todo, por su costo, ya que, existen alternativas gratuitas.

7.3.3. Lenguajes

En esta sección se encuentran las alternativas a los lenguajes utilizados.

7.3.3.1. R

*R Project*⁵⁹ es un lenguaje de programación y un entorno de software para el análisis estadístico y la visualización de datos. Fue desarrollado en 1993 por *Ross Ihaka* y *Robert Gentleman* en la Universidad de Auckland, Nueva Zelanda. Desde entonces, se ha convertido en una herramienta popular en campos que requieren análisis de datos, como la estadística, la ciencia de datos y *ML*.



Ilustración 58. Logo de R.

⁵⁹ <https://www.r-project.org/>

A continuación, se presentan algunas ventajas y desventajas asociadas a su uso en este contexto.

7.3.3.1.1. Ventajas

- Amplio ecosistema de paquetes: Cuenta con una amplia variedad de paquetes y librerías especializadas en análisis de datos, *ML* y visualización. Esto proporciona una gran flexibilidad y capacidad de elección para implementar modelos de *ML*.
- Análisis estadístico avanzado: Proporciona una amplia gama de técnicas estadísticas y numéricas, incluyendo modelado lineal y no lineal, pruebas estadísticas clásicas, análisis de series temporales, clasificación, agrupación y otros.
- Programación de alto nivel: *R* es un lenguaje de programación de alto nivel, lo que significa que es más fácil de aprender y usar que los lenguajes de bajo nivel.
- Visualización de datos: Ofrece potentes capacidades de visualización de datos a través de paquetes como *ggplot2* y *plotly*. Esto permite generar gráficos interactivos y visualizaciones informativas para analizar y presentar los resultados del sistema.

7.3.3.1.2. Desventajas

- Rendimiento computacional: Aunque *R* es eficiente para el análisis estadístico y la manipulación de datos, en comparación con lenguajes como *Python*, puede presentar cierta limitación en cuanto al rendimiento computacional en tareas intensivas de procesamiento o análisis en tiempo real.
- Concurrencia y paralelismo: *R* no fue diseñado con un fuerte soporte para la concurrencia y el paralelismo, lo que puede ser un desafío en un sistema IDS que necesita manejar múltiples flujos de datos simultáneamente.

7.3.3.1.3. Conclusión

En conclusión, aunque *R* puede ser útil para algunas casuísticas, para nuestro caso hay mejores alternativas que cuentan con un mayor soporte y bibliotecas,

un rendimiento potencialmente superior y no cuentan con problemas de limitaciones para la concurrencia y el paralelismo.

7.3.3.2. Julia

*Julia*⁶⁰ es un lenguaje de programación de alto nivel y alto rendimiento para la computación técnica. Fue diseñado para abordar la necesidad de un lenguaje que sea tanto fácil de aprender y escribir (como *Python* o *R*) y capaz de ejecutar cálculos numéricos y científicos a alta velocidad (como *C* o *Fortran*).



Ilustración 59. Logo de *Julia*.

A continuación, se presentan algunas ventajas y desventajas asociadas a su uso en este contexto.

7.3.3.2.1. Ventajas

- Rendimiento de alto nivel: *Julia* fue diseñado para ser rápido. Su sintaxis es fácil de aprender para los usuarios de otros lenguajes de programación de alto nivel, pero también permite la ejecución de código a velocidades comparables a las de los lenguajes de bajo nivel.
- Cómputo paralelo y distribuido: *Julia* tiene soporte incorporado para la computación paralela y distribuida, lo que permite a los usuarios aprovechar al máximo los recursos de hardware disponibles.
- Llamadas a funciones de otros lenguajes: *Julia* puede llamar a funciones de *C*, *Fortran* y *Python*, lo que permite a los usuarios utilizar bibliotecas existentes en esos lenguajes.

⁶⁰ [https://es.wikipedia.org/wiki/Julia_\(lenguaje_de_programaci%C3%B3n\)](https://es.wikipedia.org/wiki/Julia_(lenguaje_de_programaci%C3%B3n))

- Tipos definidos por el usuario: *Julia* permite a los usuarios definir sus propios tipos de datos, lo que puede mejorar la legibilidad y la eficiencia del código.
- Macros y meta programación: *Julia* soporta meta programación, lo que permite a los usuarios escribir programas que generan o manipulan otros programas (en *Julia*), y macros que proporcionan una forma de incluir código generado en el lugar de la llamada de la macro.

7.3.3.2.2. Desventajas

- Madurez del ecosistema: Aunque el lenguaje *Julia* ha experimentado un crecimiento significativo, su ecosistema de paquetes y herramientas aún no es tan extenso como el de lenguajes más establecidos como *Python* o *R*.
- Madurez del lenguaje: *Julia* es un lenguaje de programación relativamente nuevo en comparación con otros lenguajes. Esto podría llevar a problemas de estabilidad o compatibilidad, y también podría significar que hay menos soporte y recursos disponibles.

7.3.3.2.3. Conclusión

Julia puede ser útil para algunos proyectos estrictamente científicos, pero no para nuestro proyecto, esto es debido a su ecosistema de paquetes relativamente pequeño y su madurez como lenguaje de programación. Sobre todo, es debido a que no cuenta con todas las librerías que necesitamos para el desarrollo del sistema.

Capítulo 8. Diseño del Sistema

En este capítulo contendrá el diseño del sistema realizado, para ello hemos tenido en cuenta el análisis del Capítulo 5. “Análisis” y el Capítulo 6. “Definición del Sistema”. Se ofrecerán los diferentes diagramas diseñados para facilitar el entendimiento del sistema y arquitectura del sistema. También, se explicará el diseño del sistema de colas y de la base de datos basada en series temporales.

8.1. Diagramas de Paquetes

Para este diagrama solo tenemos en cuenta la carpeta “ids”, ya que es la única en la que se encuentra código fuente, el resto de las carpetas de carpetas se encuentran ficheros de configuración y variables de entorno.

En la Ilustración 60 se puede observar la estructura global de paquetes del sistema. Esta se divide en dos partes, los paquetes de “apolo” y los paquetes independientes de “ids”. A continuación, vamos a entrar en cada parte y explicar cada paquete que la componga.

También, mostraremos las conexiones que existen entre paquetes.

Apolo: Sistema inteligente de monitorización y detección de ataques basados en inteligencia artificial

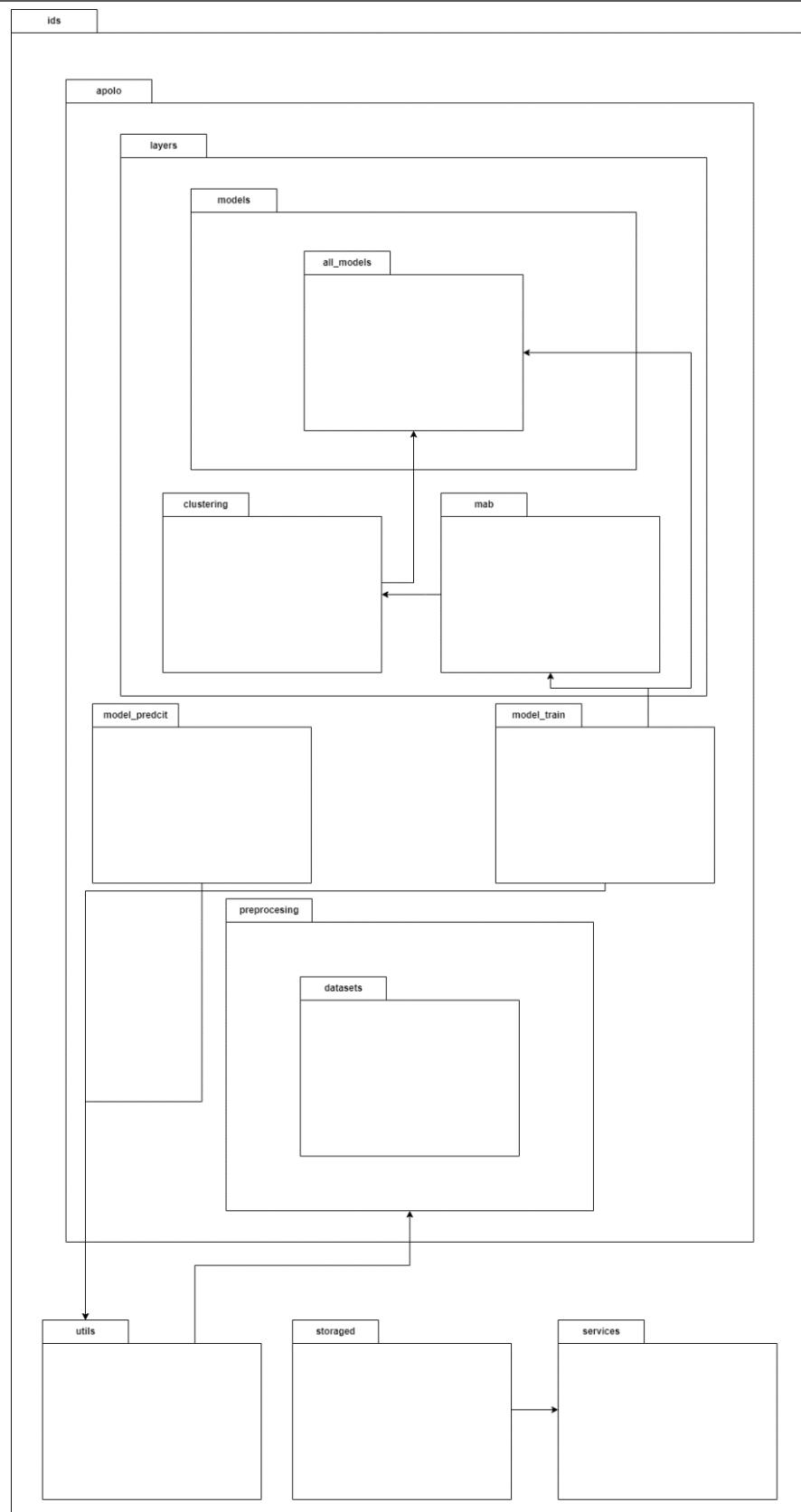


Ilustración 60. Diagrama Paquetes: Global

8.1.1. Paquete Apolo

En esta sección explicaremos los paquetes que conforman el paquete de “apolo”, el cual está principalmente dividido en dos grandes zonas, el paquete de “layers” y el grupo de paquetes de “preprocessing”, “model_train” y “model_predict”.

8.1.1.1. Layers

En la Ilustración 61 se puede observar el paquete “layers”, en este paquete se albergan las diferentes capas de **Apolo**, las cuales están descritas en el Capítulo 3. “Apolo”. Este paquete se subdivide en 3 paquetes, los cuales son:

- “models”: Este paquete cuenta con la clase abstracta “Model” de la cual heredan todas las clases de los modelos del subpaquete “all_models”.
- “clustering”: En este paquete se encuentra el modelo de *clustering*.
- “mab”: Este paquete es de los más importantes, ya que es el que implementa el algoritmo *Multi-Armed Bandits (MAB)*, para ello necesita el resto de las capas de **Apolo**.

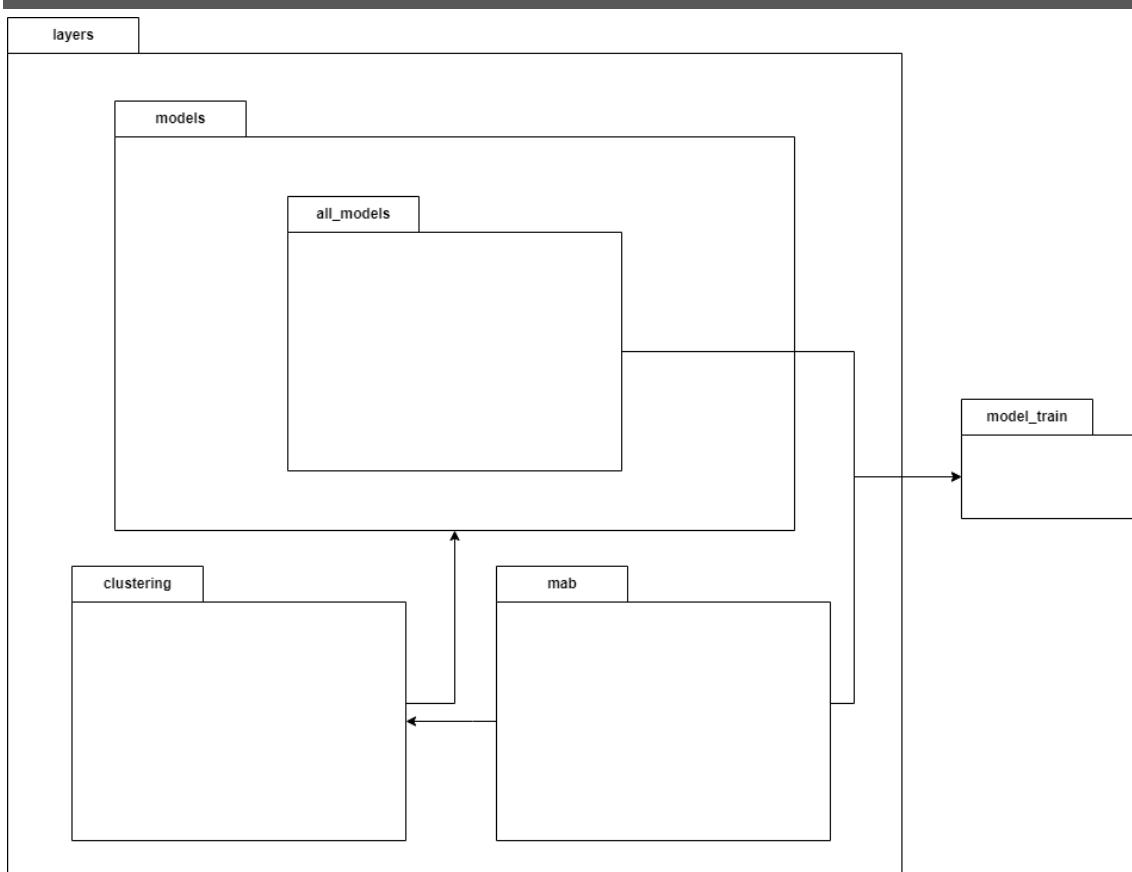


Ilustración 61. Diagrama Paquetes: Layers

8.1.1.2. Paquetes Preprocesing, Model Train y Model Predict

En la Ilustración 62 se pueden observar tres paquetes iniciales:

- “preprocessing”: Este paquete es el encargado de preprocesar y transformar las peticiones, está pensado para facilitar la adhesión de nuevos datasets, por lo que en su subpaquete “datasets”, se albergan los preprocesados específicos para cada dataset utilizado.
- “model_train”: Este paquete contiene la clase necesaria para entrenar **Apolo**, para ello tiene que hacer de nexo de muchos paquetes, como son “mab”, “models” y “utils”.
- “model_predict”: Este paquete contiene la clase necesaria para predecir una petición un dataset utilizando el modelo **Apolo**, para ello carga el modelo (utilizando el paquete “utils”) y predice la petición dada.

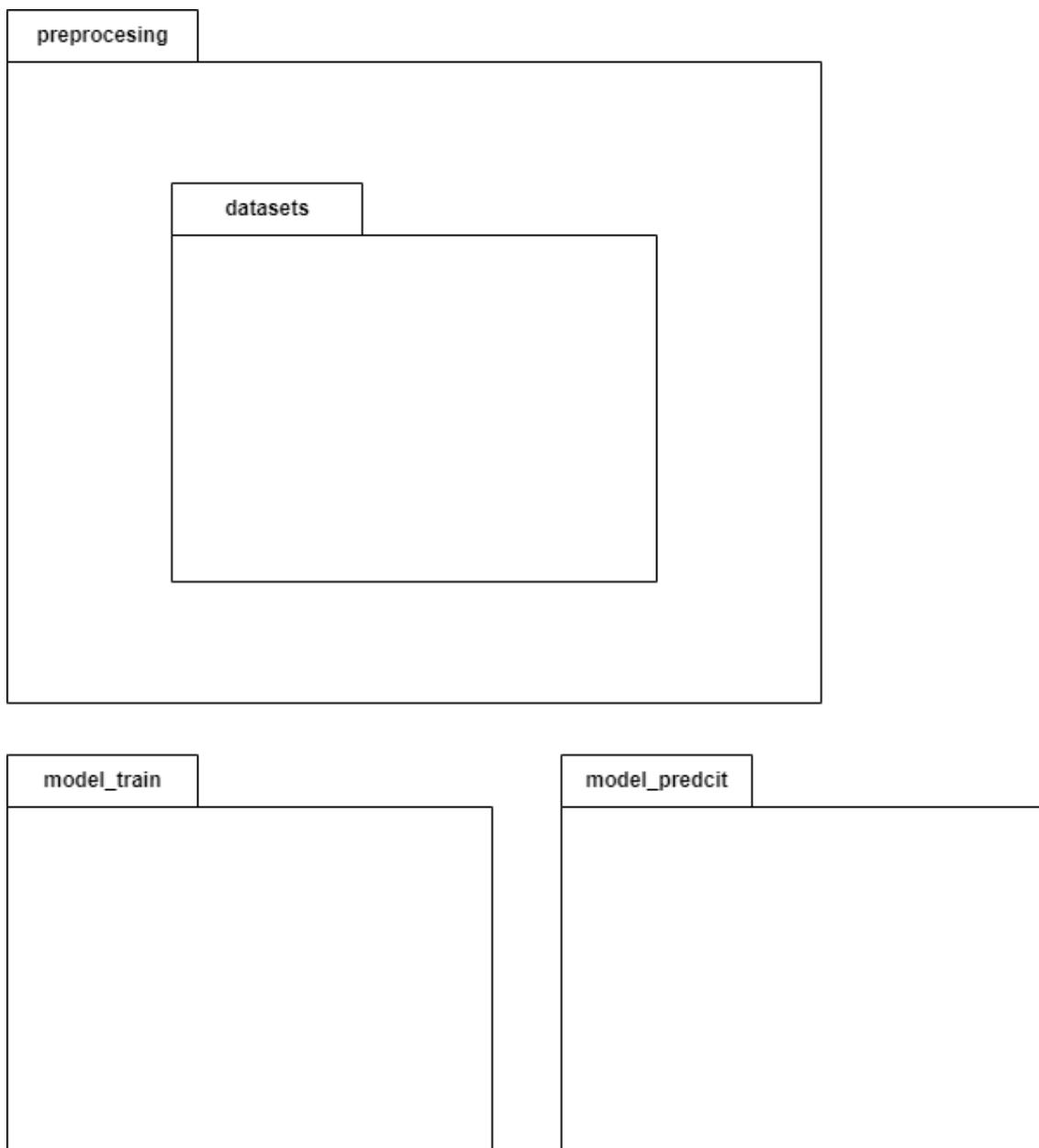


Ilustración 62. Diagrama Paquetes: Preprocesing, model train y model predict

8.1.2. Paquetes Utils, Services y Storage

En la Ilustración 63 se pueden ver estos tres paquetes, que no son código propio del paquete “apolo”, son paquetes necesarios para implementar **Apolo** en toda la arquitectura, como son los paquetes “services” y “storage”; y para funciones externas al propio Apolo, como el paquete “utils”, el cual contiene funciones de carga y guardado de modelos, datasets y peticiones.

Apolo: Sistema inteligente de monitorización y detección de ataques basados en inteligencia artificial

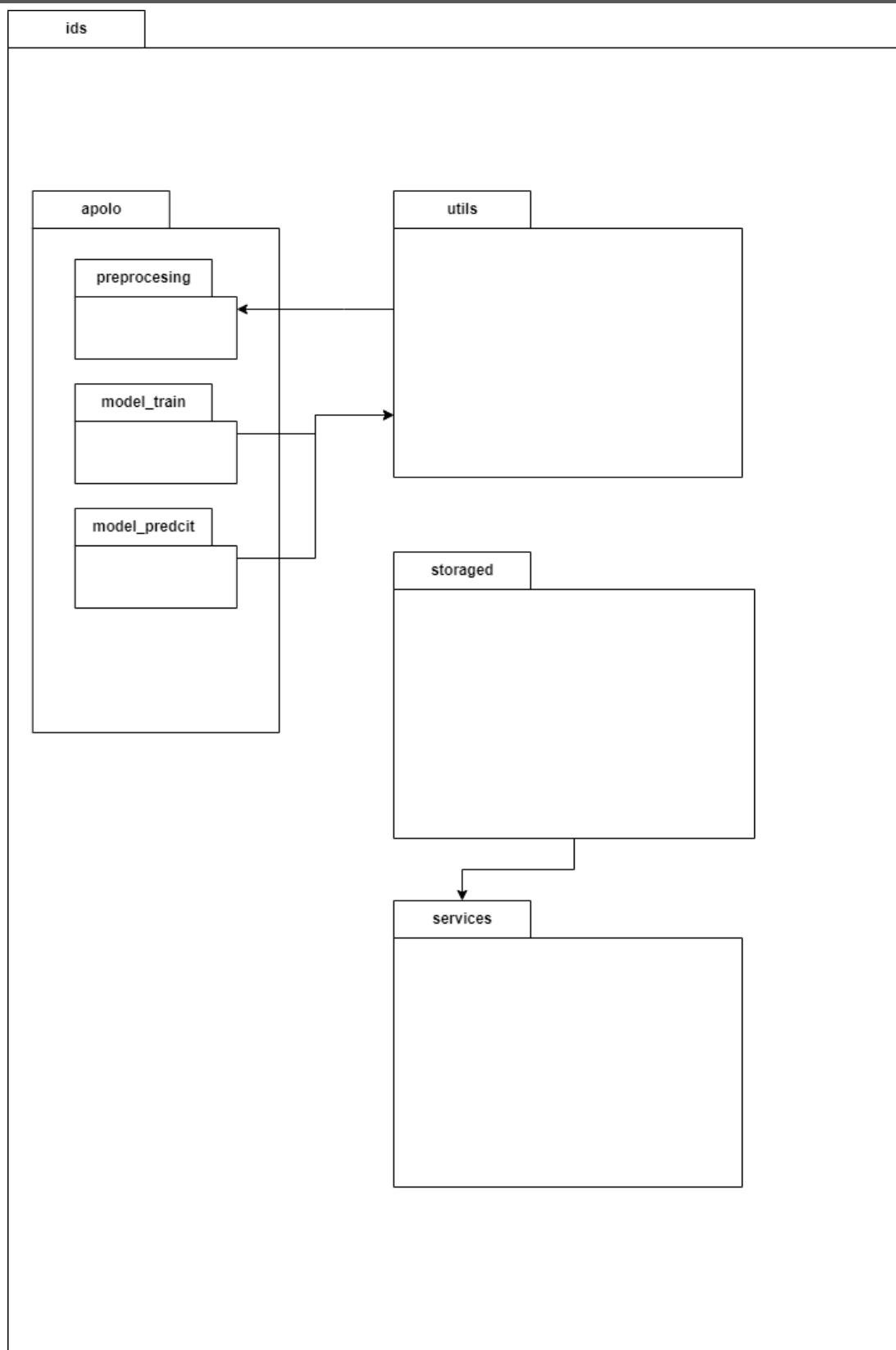


Ilustración 63. Diagrama Paquetes: Paquetes independientes al paquete "apolo".

8.2. Diagramas de Despliegue

En este apartado realizaremos el diagrama de despliegue del sistema. En la Ilustración 64 se puede observar cómo se comunican los diferentes componentes del sistema.

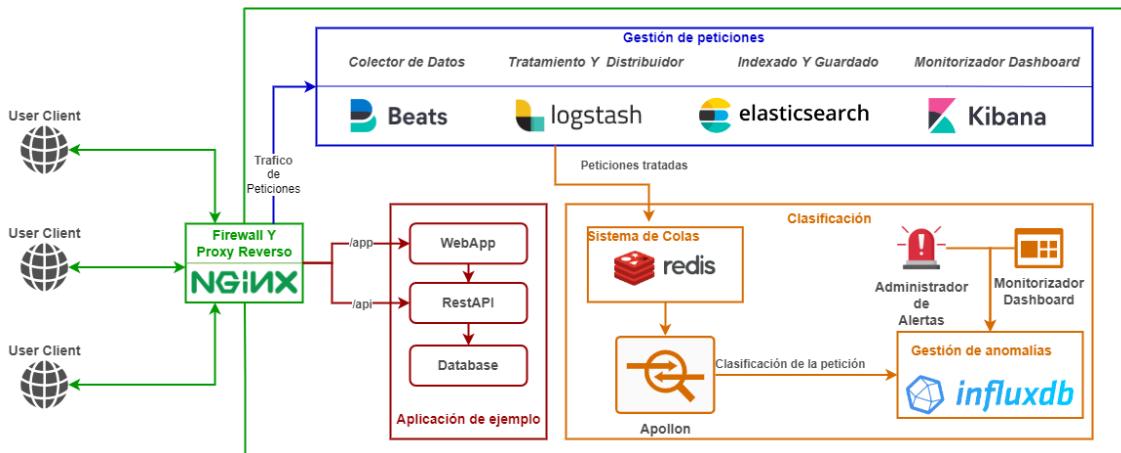


Ilustración 64. Diagrama de despliegue.

El despliegue de este sistema se realiza mediante el uso de la herramienta *Docker-Compose*, siendo cada uno de los componentes de la arquitectura un microservicio implementado con un contenedor *Docker*.

Todos los contenedores *Docker* están comunicados entre si mediante una red virtual de tipo “Bridge”. A esta red virtual la hemos llamado “tfg” y la utilizan todos los componentes de la arquitectura, esto se puede observar en la Ilustración 65.

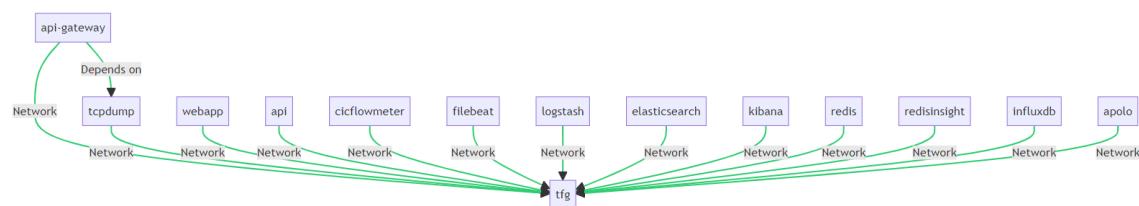


Ilustración 65. Diagrama de Despliegue: Red Virtual

La descripción detallada de cada uno de estos componentes se ha realizado en el Capítulo 6. “Definición del Sistema”, más concretamente en el punto 6.2. “Definición de los Componentes de la Arquitectura”.

8.3. Diagrama de Clases

En este apartado explicaremos todas las clases que contienen los paquetes descritos en el punto 8.1. “Diagramas de Paquetes”. En primer lugar, daremos una explicación clase por clase, explicaremos los patrones de diseño utilizados y por último daremos una vista general de todas las clases.

8.3.1. Clases del Sistema

Vamos a realizar la misma división que en el diseño de paquetes, clases del paquete de “apolo” y clases independientes a dicho paquete.

8.3.1.1. Paquete Apolo

A continuación, se detallarán las clases contenidas en el paquete “apolo”, más concretamente en los subpaquetes “layers”, “preprocesing”, “model_train” y “model_predict”.

8.3.1.1.1. Layers

A continuación, se detallarán las clases contenidas en los subpaquetes de “layers”, los paquetes “models”, “clustering” y “mab”.

8.3.1.1.1.1. Models

En el paquete “models” tenemos la clase llamada “Model”, la cual es una clase abstracta, para facilitar la adhesión de nuevos modelos *ML*, se creó el subpaquete “all_models”, el cual contiene todos los modelos *ML* utilizados. Hay una clase padre llamada “Model” y el resto de las clases hijas son muy parecidas entre sí, cambia el modelo *ML* que implementan y los parámetros que tienen, específicos para ese modelo.

La clase “Model” (Ilustración 66) se puede usar para crear un modelo genérico de *ML*. Esta clase proporciona la estructura básica para entrenar, probar y realizar predicciones con un modelo.

La clase “Model” tiene varios atributos importantes: “predictions” para almacenar las predicciones de los datos de prueba, “time_total” para registrar el tiempo total de entrenamiento y prueba del modelo, “model_trained” para el modelo entrenado, “dataset” para el nombre del conjunto de datos, “x_train” y

“y_train” para los datos de entrenamiento y sus etiquetas, “x_test” y “y_test” para los datos de prueba y sus etiquetas, y “seed” para el estado aleatorio.

El método “`__init__`” de la clase “Model” es el método constructor que inicializa estos atributos. El método “`exe`” es un método de ejecución que invoca al método “`model_train_test`”. El método “`model_train_test`” se encarga de llamar al método de entrenamiento del modelo específico (es decir, “`expecific_model().fit`”) y de registrar el tiempo total de entrenamiento y predicción. También almacena las predicciones en el atributo “`predictions`”.

El método “`predict`” se utiliza para realizar predicciones sobre los datos de prueba. Si se proporciona un conjunto de datos de prueba diferente, este método puede usarlo en lugar del conjunto de datos de prueba por defecto. Finalmente, el método “`expecific_model`” es un método de marcador de posición que debe ser redefinido por cualquier subclase de la clase “Model” para definir el modelo específico a utilizar. En resumen, la clase “Model” proporciona una plantilla genérica para trabajar con modelos de aprendizaje automático.

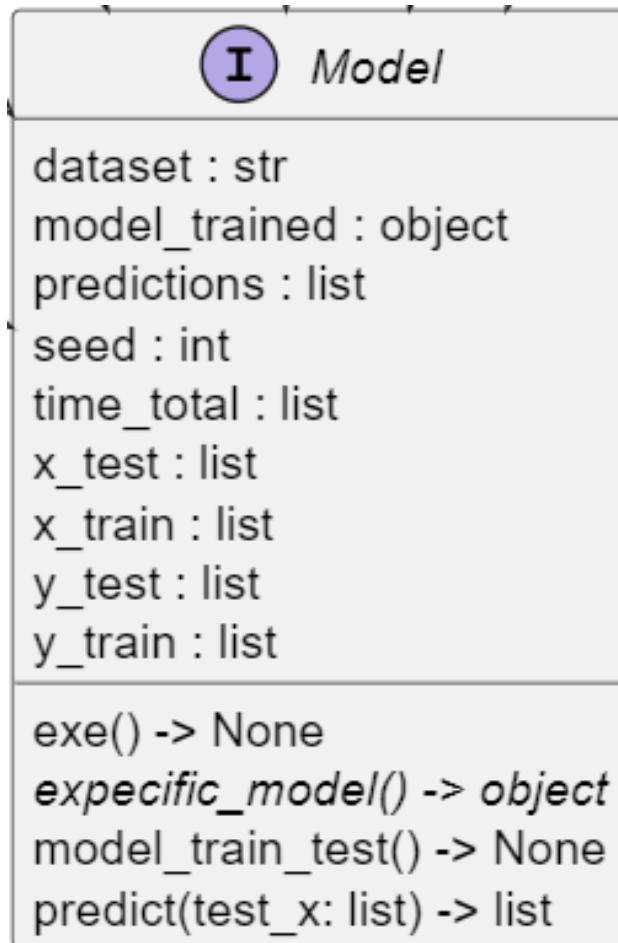


Ilustración 66. Diseño de Clases: *Model*.

La clase “SVCModel” (Ilustración 67), que hereda de la clase “Model” y se utiliza para crear un modelo de *Máquina de Soporte Vectorial* (SVC) utilizando la biblioteca *sklearn*. En el método de inicialización, se pasan los conjuntos de entrenamiento y prueba, el nombre del conjunto de datos y una semilla para el estado aleatorio. Si el parámetro “exe” es verdadero, el modelo se entrena y se prueba automáticamente durante la inicialización. La clase sobrescribe el método “expecific_model” de la clase padre para devolver una instancia de SVC con la semilla proporcionada. Además, sobrescribe el método “__str__” para devolver el nombre de la clase.

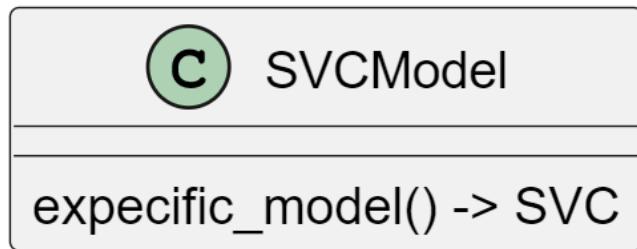


Ilustración 67. Diseño de Clases: *SVC*.

La clase “RandomForest” (Ilustración 68) que hereda de “Model” y utiliza la biblioteca *sklearn* para crear un modelo de clasificación de *Random Forest*. Durante la inicialización, se establecen el número de estimadores y el número de trabajos paralelos, y si el argumento “exe” es verdadero, el modelo se entrena y se prueba automáticamente. El método “expecific_model” está sobrescrito para devolver una instancia de *RandomForestClassifier* con los parámetros establecidos. La función “__str__” ha sido modificada para incluir información adicional sobre el número de estimadores y trabajos en la cadena de retorno.

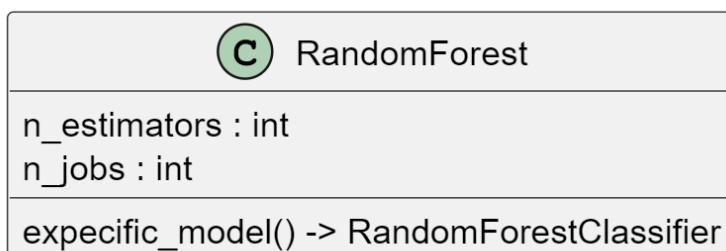


Ilustración 68. Diseño de Clases: *RandomForest*.

La clase “MLP” (Ilustración 69) es una subclase de “Model”, para implementar una *Red Neuronal Multicapa (MLP)* utilizando la biblioteca *sklearn*. En su inicialización, si el argumento “exe” es verdadero, el modelo se entrena y prueba automáticamente. El método “expecific_model” está sobrescrito para devolver una instancia de *MLPClassifier* con una arquitectura de tres capas ocultas de 32 neuronas cada una, y otros parámetros de configuración específicos, como la función de activación “relu”, el optimizador “adam” y el uso de detención temprana para el entrenamiento.

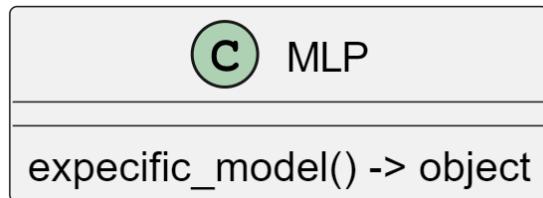


Ilustración 69. Diseño de Clases: MLP.

La clase “LogisticRegressionModel” (Ilustración 70) hereda de “Model”, y se usa para implementar un modelo de regresión logística a partir de la biblioteca *sklearn*. No introduce nuevos atributos, pero sobrescribe el método “expecific_model” para devolver una instancia de *LogisticRegression*. Similarmente a las otras clases, si se establece “exe” a *True*, el entrenamiento y la prueba del modelo se inician automáticamente durante la creación de la instancia.

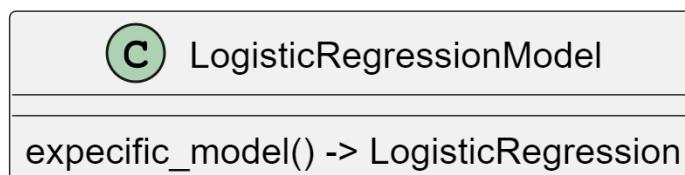


Ilustración 70. Diseño de Clases: LogisticRegressionModel.

La clase “KNeighbors” (Ilustración 71), la cual es una subclase de “Model” y se utiliza para crear y manejar un modelo de clasificación de vecinos más cercanos (*KNeighborsClassifier*) de *sklearn*. Introduce un atributo “k” que determina el número de vecinos a considerar. El método “expecific_model” se sobrescribe para devolver una instancia de *KNeighborsClassifier* con el número de vecinos “k” especificado. Si se establece la opción “exe”, la clase inicia automáticamente el entrenamiento y prueba del modelo al crearse la instancia.

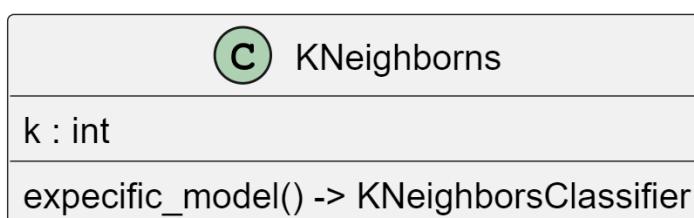


Ilustración 71. Diseño de Clases: KNeighbors.

La clase “DecisionTree” (Ilustración 72) hereda de la clase “Model”. Esta clase personaliza la funcionalidad de “Model” para implementar un modelo de árbol de decisión usando *sklearn*. Esencialmente, inicializa el modelo, realiza el entrenamiento si se especifica, y redefine el método “*expecific_model*” para devolver una instancia de “DecisionTreeClassifier”, proporcionando así un marco para trabajar con árboles de decisión.

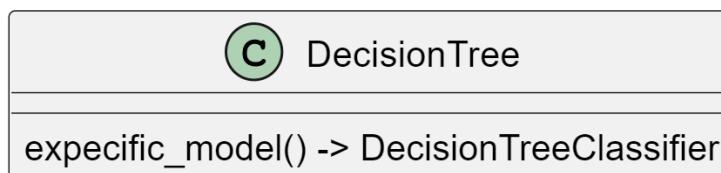


Ilustración 72. Diseño de Clases: *DecisionTree*.

La clase “NaiveBayes” (Ilustración 73), que hereda de “Model”, sirve para implementar un modelo de clasificación basado en el algoritmo *Naive Bayes Gaussiano* utilizando la biblioteca *sklearn*. Durante su inicialización, si el argumento “*exe*” es verdadero, el modelo se entrena y prueba automáticamente. El método “*expecific_model*” está sobreescrito para devolver una instancia de *GaussianNB*, que es la implementación del *Naive Bayes Gaussiano* en *sklearn*.

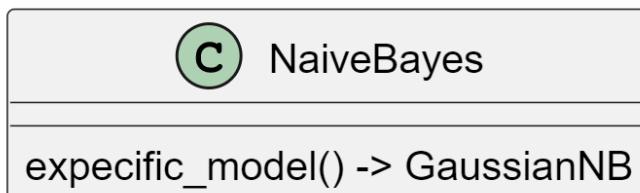


Ilustración 73. Diseño de Clases: *NaiveBayes*.

8.3.1.1.1.2. Clustering

En el paquete “clustering” tenemos la clase llamada “KMeansCluster”.

La clase “KMeansCluster” (Ilustración 74) hereda de la clase “Model”. Esta clase está diseñada para adaptar la funcionalidad del “Model” a un modelo de agrupamiento *KMeans* de *sklearn*. Introduce un nuevo atributo “*k*” que representa el número de *clusters*. El método “*expecific_model*” se sobrescribe para devolver una instancia de *KMeans* con el número especificado de clústeres. Si se especifica la opción “*exe*”, la clase inicia automáticamente el proceso de entrenamiento al crear una instancia.

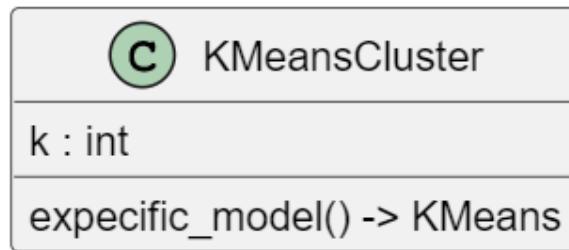


Ilustración 74. Diseño de Clases: **KMeans**.

8.3.1.1.1.3. MAB

En el paquete “mab” tenemos la clase llamada “MAB”.

La clase “MAB” (Ilustración 75) implementa el algoritmo *Multi-Armed Bandit* (*MAB*) para la exploración y explotación en el *ML*. Entre sus atributos principales se encuentran “n_arms”, que representa el número de “brazos” u opciones disponibles, “n_clusters”, que denota el número de *clústeres* en los que se dividen los datos, y “arms”, una lista de los modelos de *ML* que se utilizan como “brazos” en *MAB*. También almacena los centros de los *clústeres*, las asignaciones de los *clústeres* y las sumas de las recompensas de cada “brazo” en cada *clúster*.

A continuación, detallaremos de los métodos principales de la clase “MAB”:

- “*__init__*”: Es el método constructor de la clase. Inicializa los atributos del objeto “MAB” con valores predeterminados o con los valores proporcionados durante la creación del objeto. También se inicializa el objeto “KMeansCluster” que se utilizará para agrupar los datos.
- “*train*”: Este método entrena el algoritmo *MAB*. Para cada *clúster*, selecciona los datos de entrenamiento que pertenecen a ese *clúster* y luego entrena cada “brazo” (o modelo) con estos datos. Además, calcula las recompensas (es decir, la precisión del modelo) de cada “brazo” en cada *clúster* utilizando los datos de prueba. Los “brazos” son entrenados usando tanto los datos específicos del *clúster* como el conjunto de datos de entrenamiento completo, en el caso de que los datos específicos del *clúster* sean demasiado limitados.
- “*select_arm*”: Este método selecciona el “brazo” (o modelo) que debe usarse para realizar predicciones para un *clúster* específico. La elección del “brazo” se basa en las recompensas calculadas durante el

entrenamiento, utilizando un algoritmo de selección que incluye cierto grado de aleatoriedad para equilibrar la exploración y la explotación.

- “predict”: Este método realiza predicciones para un conjunto de datos de prueba. Para cada punto de datos, primero determina a qué *clúster* pertenece y luego utiliza el “brazo” seleccionado para ese *clúster* para hacer la predicción.
- “test”: Este método evalúa el rendimiento del algoritmo *MAB*. Hace predicciones para un conjunto de datos de prueba y luego compara estas predicciones con las etiquetas reales, registrando varias métricas de rendimiento como precisión, *recall*, *F1* y *ROC AUC*.
- “print_arms_test”: Este método imprime en un archivo los “brazos” seleccionados para cada muestra de los datos de prueba, junto con la predicción hecha y la etiqueta real.

Los métodos de la clase “MAB” permiten entrenar, seleccionar modelos y realizar predicciones utilizando el algoritmo *Multi-Armed Bandit*, así como probar y registrar los resultados de su rendimiento.

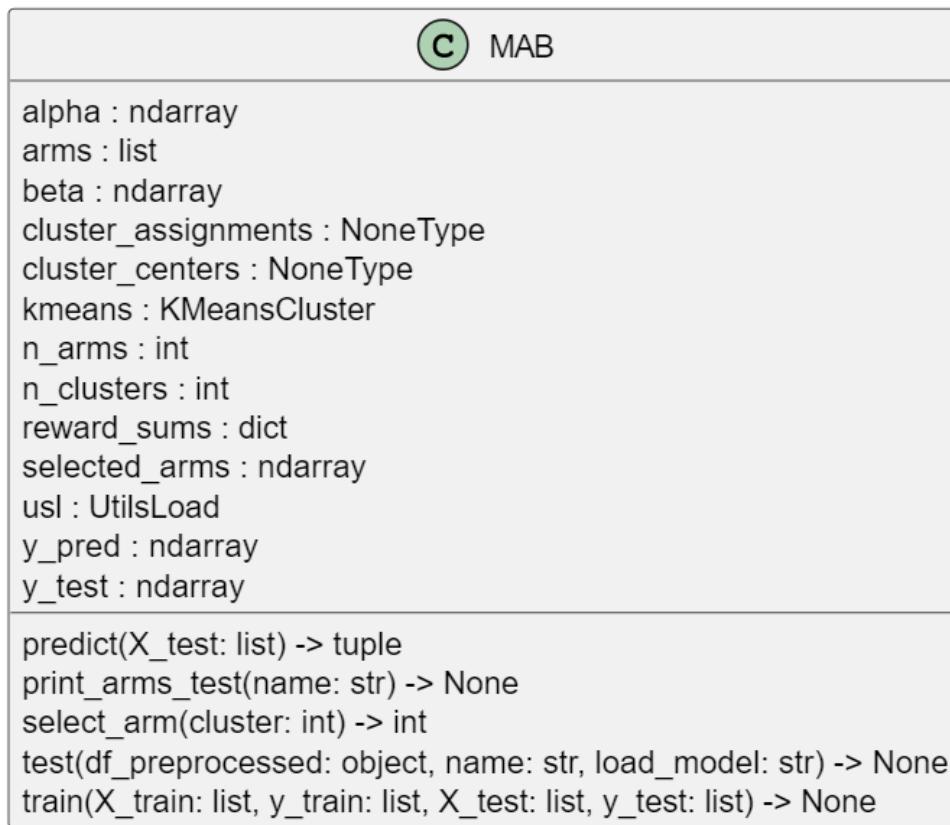


Ilustración 75. Diseño de Clases: MAB.

8.3.1.1.2. Paquetes Preprocesing, Model Train y Model Predict

A continuación, se detallarán las clases contenidas en los paquetes “preprocesing”, “model_train” y “model_predict”.

8.3.1.1.2.1. Preprocesing

En el paquete “preprocesing” tenemos varias clases como son “Transform”, “ClearData” que es una clase abstracta y todas las clases hijas de “ClearData”, que son: “ClearDataOur”, “ClearDataCIC2017”, “ClearDataCIC2018” y “ClearDataCIC2019”.

La clase “Transform” (Ilustración 76) proporciona métodos para la transformación de datos. Tiene varios atributos de instancia que incluyen “x” y “y”, que son listas de características y etiquetas respectivamente, “size” que determina el tamaño de la partición de prueba en la división de entrenamiento/prueba, “x_train”, “x_test”, “y_train” y “y_test” que son los

conjuntos de entrenamiento y prueba divididos, “seed” que es la semilla para el generador de números aleatorios, y “df” que es un “DataFrame” de pandas.

El método “transform” divide los datos en conjuntos de entrenamiento y prueba, luego aplica una transformación “RobustScaler” y “Normalizer” a estos conjuntos. “RobustScaler” hace que los datos sean menos sensibles a los valores atípicos al centrar y escalar los datos usando estadísticas robustas, mientras que “Normalizer” redimensiona cada muestra de datos para tener una longitud de unidad norma (es decir, escala las características de entrada individualmente a la norma unitaria).

El método “transform_request” realiza una operación similar al método “transform”, pero se aplica solo a “x”, que representa las características de una solicitud en lugar de un conjunto completo de datos. La operación de transformación incluye la aplicación de las transformaciones “RobustScaler” y “Normalizer” a los datos.

La clase “Transform” encapsula operaciones de preprocesamiento de datos esenciales para la preparación de los datos para el aprendizaje automático.

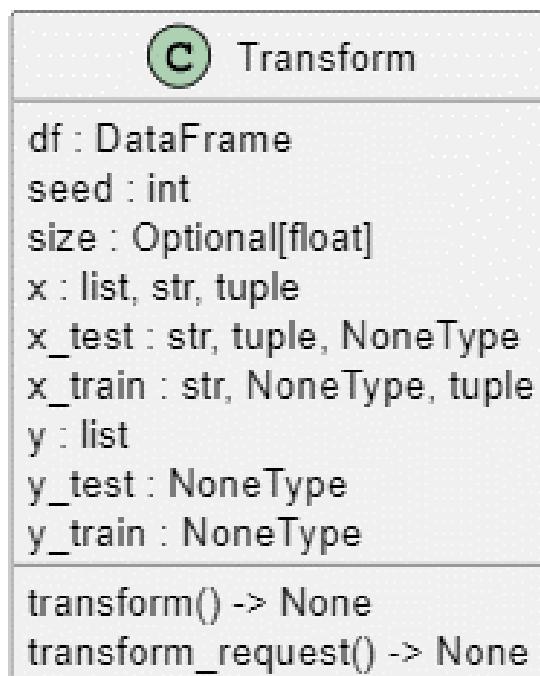


Ilustración 76. Diseño de Clases: Transform.

La clase “ClearData” (Ilustración 77) provee métodos para la limpieza y el preprocesamiento de datos. Tiene varios atributos de instancia que incluyen “df” que es un “DataFrame” de *pandas* para limpiar, “quantile”, “max_mediana”, y “log_unic” que determinan varios umbrales para la limpieza de datos, “label_f” que es el número de etiquetas más frecuentes a conservar en las características categóricas, “x” y “y” que son las características y las etiquetas respectivamente después de la limpieza, “do_log” y “save” que determinan si aplicar la función logarítmica y guardar el *dataframe* preprocesado respectivamente, y “seed” que es la semilla para el generador de números aleatorios.

- El método “best_features_func” selecciona las mejores características de los datos. Aplica una serie de funciones de limpieza y preprocesamiento, que incluyen la eliminación de características con un solo valor, columnas duplicadas y elementos indeseables, la gestión de valores atípicos, la aplicación de la función logarítmica a características con muchos valores únicos, y la selección de las etiquetas más frecuentes en características categóricas.
- Los métodos “drop_bad_elements” y “drop_bad_elements_x” eliminan los elementos indeseables de “df” y “x” respectivamente, reemplazando los valores infinitos y los valores faltantes por cero.
- El método “clear_valores_atipicos” gestiona los valores atípicos recortando cualquier valor que sea un número determinado de veces mayor que la mediana a un percentil determinado.
- El método “clear_log” aplica la función logarítmica a las características con más de un número especificado de valores únicos.
- El método “clear_freq” selecciona las etiquetas más frecuentes de las características y establece todas las demás como “-”.
- El método “reduce_tam” reduce el tamaño del *dataframe* cambiando los tipos de datos a versiones que ocupan menos memoria.
- El método “loss_data” calcula la pérdida de datos.
- El método “cast_time” convierte una columna de tiempo a un formato específico.
- El método “one_shotear” aplica la codificación *One-Hot* a las columnas especificadas.
- El método “replace” reemplaza los valores de las etiquetas. Estandariza estos valores a “Benign” para el tráfico normal y “Malicious” para el tráfico malicioso.

La clase “ClearData” encapsula operaciones de preprocesamiento de datos esenciales para la preparación de los datos para el aprendizaje automático.

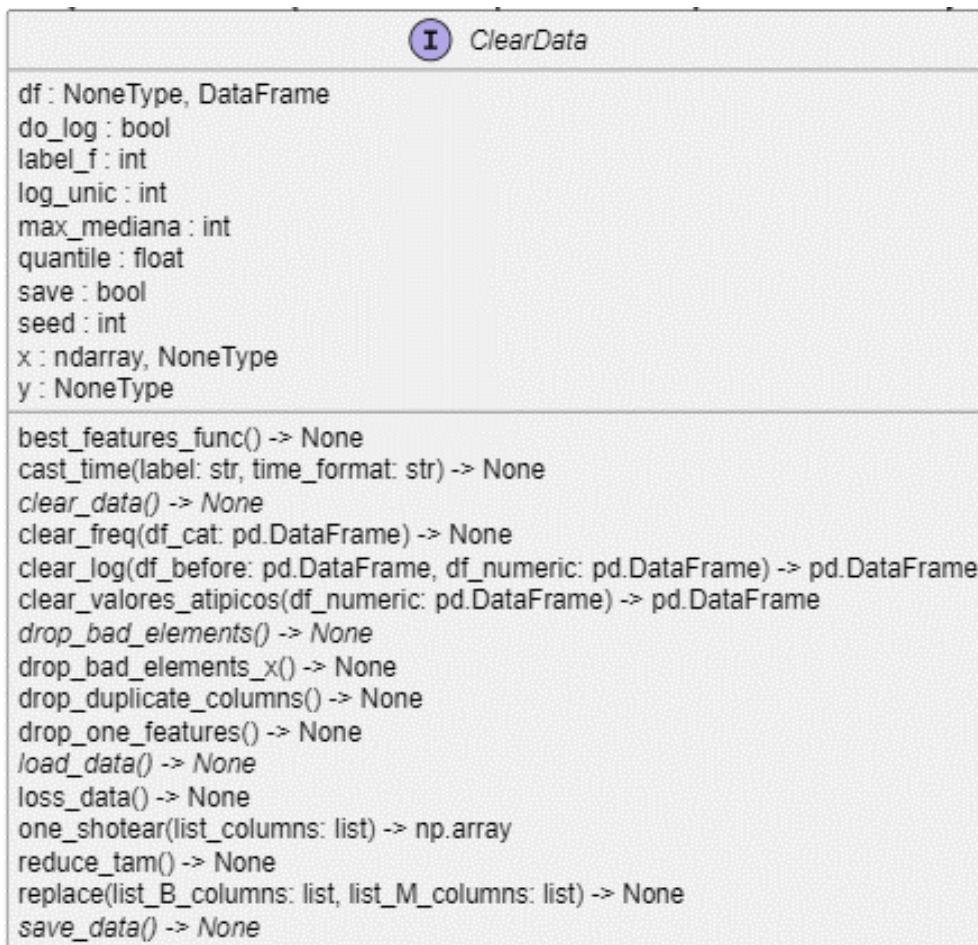


Ilustración 77. Diseño de Clases: `ClearData`.

La clase “`ClearDataOur`” (Ilustración 78) es una extensión de la clase “`ClearData`” y se usa para limpiar los datos procedentes de nuestra arquitectura.

Los atributos de esta clase incluyen “`df`”, que es un *DataFrame* de *pandas* que contiene los datos a limpiar, “`do_save`”, un *booleano* que determina si los datos limpios deben ser guardados o no, “`name_save`”, una cadena de caracteres que define el nombre del archivo en el que se guardarán los datos limpios, “`name_load`”, una cadena de caracteres que define el nombre del archivo a cargar y “`seed`” que es la semilla para el generador de números aleatorios.

El método “`__init__`” es utilizado para inicializar la clase “`ClearDataOur`”. Acepta un *DataFrame*, una variable *booleana* que indica si se deben guardar los datos o no, una semilla para el generador de números aleatorios, y los nombres de los archivos de guardar y cargar.

El método “clear_data” se utiliza para limpiar los datos en nuestra arquitectura. Elimina las columnas duplicadas y los elementos indeseables, divide el *DataFrame* en características y etiquetas, y elimina la etiqueta “No Label” si está presente. Si la opción “do_save” está habilitada, los datos limpios se guardan utilizando el método “save_data”.

El método “save_data” guarda los datos limpios en un archivo “.csv”. Crea una copia del *DataFrame*, elimina la columna “Label” si está presente, y si hay etiquetas disponibles, las guarda en un archivo .csv separado. Luego, concatena el *DataFrame* original y las etiquetas en un solo *DataFrame* y lo guarda en un archivo .csv.

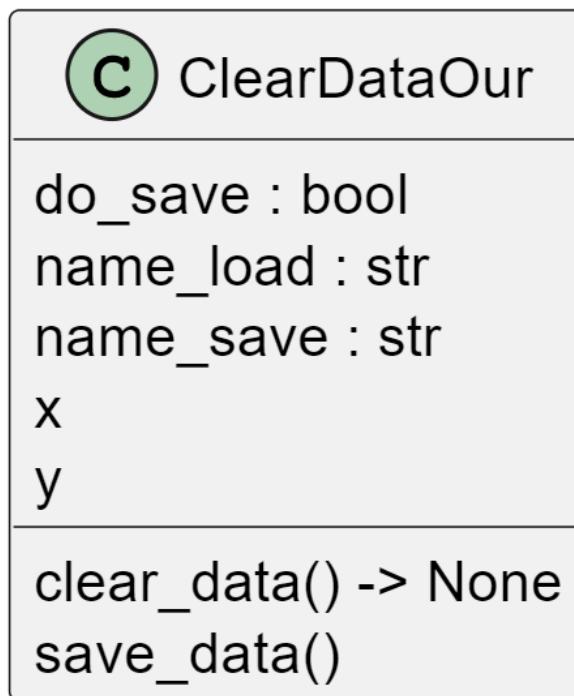


Ilustración 78. Diseño de Clases: *ClearDataOur*.

La clase “ClearDataCIC2019” (Ilustración 79) es una extensión de la clase “ClearData”, destinada específicamente a la limpieza de datos del conjunto de datos *CIC* 2019. Entre sus atributos, tiene un *DataFrame* de pandas “df” para los datos a limpiar, “do_save”, que indica si los datos limpios deben ser guardados, y “name_save” y “name_load” para los nombres de los archivos donde se guardarán y cargarán los datos, respectivamente.

Los métodos clave de la clase incluyen “clear_data”, que realiza la limpieza de los datos eliminando columnas innecesarias, reduciendo el tamaño del conjunto

de datos y modificando las etiquetas para un formato binario; “save_data”, que guarda los datos limpios en un archivo .csv; y “load_data”, que carga los datos limpios y sus etiquetas correspondientes desde las rutas de archivos predeterminadas. Esta clase es esencial para el preprocesamiento de datos en el análisis de seguridad de la red utilizando el conjunto de datos *CIC 2019*.

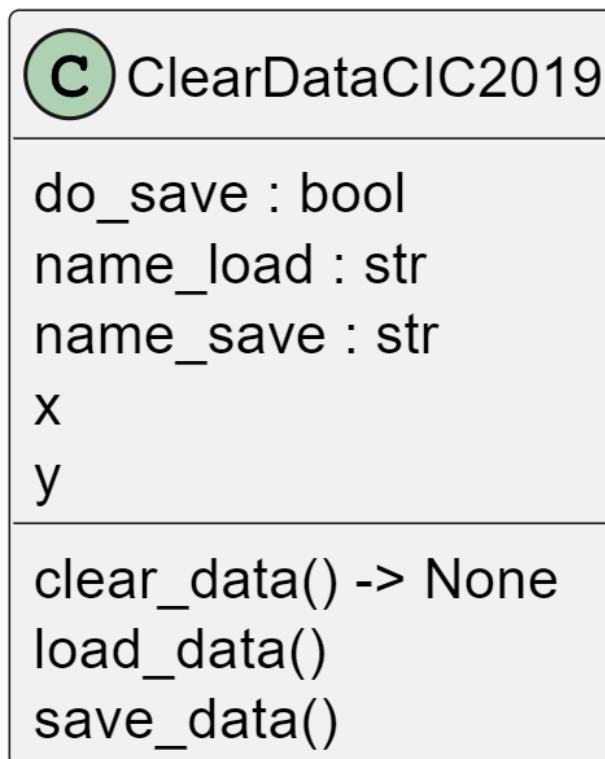


Ilustración 79. Diseño de Clases: *ClearDataCIC2019*.

La clase “`ClearDataCIC2018`” (Ilustración 80), derivada de la clase “`ClearData`”, está diseñada para limpiar los datos del conjunto de datos *CIC 2018*. Tiene atributos como “`df`” para los datos que se van a limpiar, “`do_save`” para determinar si guardar los datos después de la limpieza, y “`name_save`” y “`name_load`” para los nombres de los archivos donde se guardarán y cargarán los datos.

Los métodos principales de esta clase incluyen “`clear_data`”, que ejecuta la limpieza de los datos, eliminando la columna de tiempo, reduciendo la dimensión del conjunto de datos, eliminando columnas redundantes y caracterizando las etiquetas de una forma binaria; “`save_data`”, que guarda los datos procesados en archivos .csv; y “`load_data`”, que carga los datos limpios y sus respectivas etiquetas desde los archivos especificados. En resumen, esta clase es vital para

el preprocesamiento de los datos en el análisis de seguridad de la red utilizando el conjunto de datos *CIC 2018*.

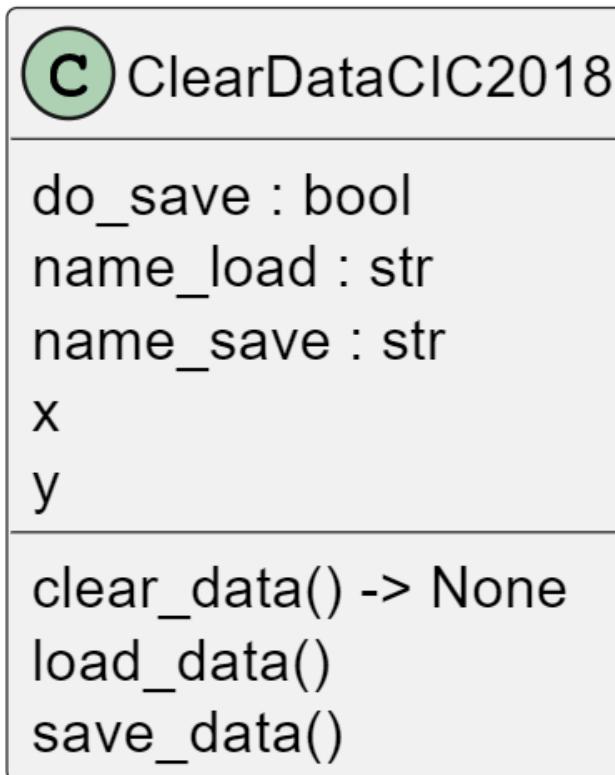


Ilustración 80. Diseño de Clases: *ClearDataCIC2018*.

La clase “ClearDataCIC2017” (Ilustración 81), que hereda de la clase “ClearData”, se utiliza para limpiar los datos del conjunto de datos *CIC 2017*. Posee atributos de instancia incluyendo “df”, que es el DataFrame de pandas que contiene los datos a limpiar, “do_save”, que determina si los datos limpios deben ser guardados, y “name_save” y “name_load”, que especifican los nombres de los archivos para guardar y cargar los datos respectivamente.

Los principales métodos en la clase son “clear_data”, que realiza la limpieza de los datos, eliminando las características no esenciales y duplicadas, transformando la etiqueta “BENIGN” y eliminando elementos malos de los datos; “save_data”, que guarda los datos limpios y sus respectivas etiquetas en archivos .csv; y “load_data”, que carga los datos limpios y las etiquetas correspondientes desde los archivos guardados. En general, esta clase se encarga del preprocesamiento esencial de los datos para el análisis de seguridad de red utilizando el conjunto de datos *CIC 2017*.

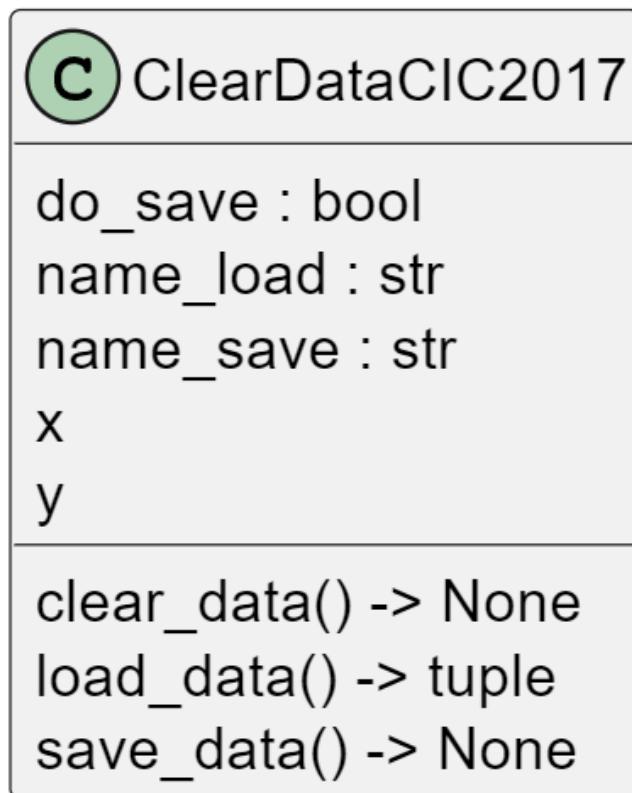


Ilustración 81. Diseño de Clases: ClearDataCIC2017.

8.3.1.1.2.2. Model Train

En el paquete “model_train” tenemos una clase llamada “ApoloTrainer”.

La clase “ApoloTrainer” (Ilustración 82) proporciona métodos para entrenar y probar **Apolo**. Contiene dos atributos de instancia: “us”, un objeto de la clase “UtilsSave” para guardar modelos, y “ul”, un objeto de la clase “UtilsLoad” para cargar modelos. Además, en su constructor inicializa una lista de “brazos” que contienen los modelos de *ML* a entrenar, utilizando un *Multi-Armed Bandit (MAB)*.

El método “train_model” es el encargado de realizar el entrenamiento de los modelos. Toma como entradas los conjuntos de datos de entrenamiento y prueba, así como las etiquetas correspondientes y una *URL* donde se guardará el modelo entrenado. El método entrena el *MAB* con los datos proporcionados y luego guarda el modelo entrenado en la ubicación especificada.

El método “test_model” se utiliza para probar un modelo ya entrenado. Este método carga el modelo desde la *URL* especificada, lo prueba con un conjunto de datos preprocesados y finalmente imprime los resultados de la prueba.

En resumen, la clase “ApoloTrainer” encapsula el proceso de entrenamiento y prueba de modelos de aprendizaje automático, permitiendo un flujo de trabajo eficiente y organizado para estas tareas.



Ilustración 82. Diseño de Clases: ApoloTrainer.

8.3.1.1.2.3. Model Predict

En el paquete “model_predict” tenemos una clase llamada “ApoloPredict”.

La clase “ApoloPredict” (Ilustración 83) proporciona métodos para realizar predicciones basadas en modelos guardados. Contiene dos atributos de instancia: “usl”, que es un objeto de la clase “UtilsLoad” utilizado para cargar modelos, y “data_prep”, que es un objeto de la clase “DataSelector” utilizado para preparar los datos para la predicción.

La función “classify_request” es el método principal de esta clase. Toma como entrada una lista de rutas de archivos y una *URL* donde se guarda el modelo. Primero, los datos se cargan y preparan utilizando la función “load_request” de la clase “DataSelector”. Después, el modelo se carga desde la *URL* especificada con la ayuda de la función “load_model” de la clase “UtilsLoad”. Finalmente, se realiza la predicción utilizando los datos preparados y se devuelve el resultado.

La clase “ApoloPredict” encapsula el proceso de preparación de datos, carga de modelos y realización de predicciones, haciendo que la realización de predicciones en nuevos datos sea un proceso fácil y eficiente.

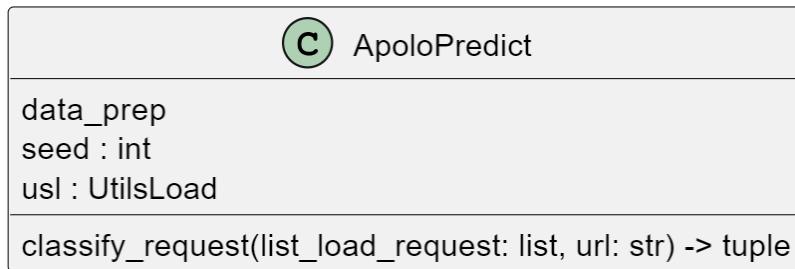


Ilustración 83. Diseño de Clases: *ApoloPredict*.

8.3.1.2. Paquetes *Utils, Services y Storage*

A continuación, se detallarán las clases contenidas en los paquetes “utils”, “services” y “storage”.

8.3.1.2.1. Utils

En el paquete “utils” tenemos tres clases “UtilsSave”, “UtilsLoad” y “DataSelector”.

La clase “UtilsSave” (Ilustración 84) proporciona varios métodos para guardar tanto datos como modelos. No tiene atributos de instancia.

La función “save_model” guarda un modelo en un archivo binario usando la biblioteca *pickle*, el modelo para guardar y el nombre del archivo se proporcionan como argumentos. La función “save_data” guarda un *DataFrame* de *pandas* en un archivo CSV. El *DataFrame* y el nombre del archivo se proporcionan como argumentos.

Estas funciones permiten un almacenamiento eficiente de los modelos y datos generados durante la ejecución de proyectos de ciencia de datos y aprendizaje automático, lo que facilita la reutilización y la replicabilidad de los resultados.

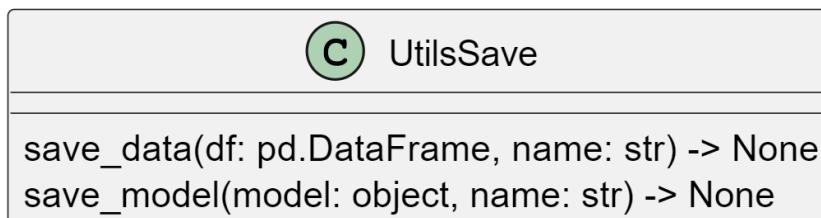


Ilustración 84. Diseño de Clases: *UtilsSave*.

La clase “UtilsLoad” (Ilustración 85) proporciona varios métodos para cargar datos y modelos desde archivos. Tiene un atributo de instancia, “seed”, que se utiliza para asegurar la reproducibilidad durante la carga de datos.

La función “load_data” permite cargar un conjunto de datos desde una o varias rutas de archivo especificadas. Además, esta función ofrece la opción de mezclar aleatoriamente los datos cargados para garantizar una distribución homogénea. Si se proporciona un diccionario JSON, se utiliza la función “load_data_cic” para cargar los datos del JSON y convertirlo en un *DataFrame*.

Por otro lado, la función “load_model” permite cargar un modelo desde un archivo binario utilizando la biblioteca *pickle*, el nombre del archivo se proporciona como argumento.

En general, las funciones proporcionadas por la clase “UtilsLoad” facilitan la carga eficiente de datos y modelos en proyectos de ciencia de datos y aprendizaje automático, permitiendo reanudar el trabajo o realizar análisis posteriores de manera sencilla y eficiente.

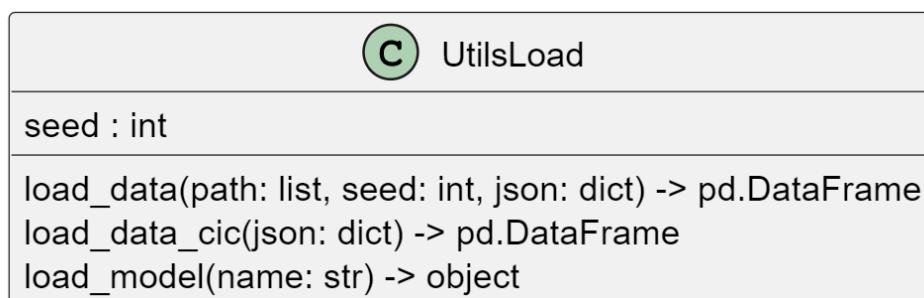


Ilustración 85. Diseño de Clases: *UtilsLoad*.

La clase “DataSelector” (Ilustración 86) proporciona varias funciones para cargar y preprocesar conjuntos de datos. Tiene tres atributos: “seed” que establece una semilla para la aleatoriedad, “usl” que es una instancia de la clase “UtilsLoad”, y “list_load_dataset” que es una lista de rutas a los conjuntos de datos que se cargarán.

El método “load_dataset” carga un conjunto de datos, lo preprocesa y opcionalmente lo guarda. La opción “load_dataset” determina si el conjunto de datos se carga desde una ruta o no. Si es “False”, el conjunto de datos se preprocesa antes de ser cargado.

El método “preprocess_dataset” preprocesa un DataFrame de pandas y devuelve un objeto “transform”. Utiliza el argumento “dataset_type” para determinar qué

método de preprocessamiento utilizar. Si “load” es “True”, el método cargará los datos existentes, de lo contrario, limpiará los datos antes de transformarlos.

El método “preprocess_request” también preprocesa un DataFrame de pandas y devuelve un objeto “transform”. Sin embargo, este método siempre limpiará los datos antes de transformarlos, independientemente del valor de “load”.

Finalmente, el método “load_request” carga una solicitud, la preprocesa y opcionalmente la guarda. A diferencia de “load_dataset”, este método solo puede cargar solicitudes desde una ruta y no tiene la opción de preprocesar los datos antes de cargarlos.

DataSelector	
list_load_dataset : list	
seed : int	
usl : UtilsLoad	
load_dataset(dataset_type: str, name: str, load_dataset: bool, save: bool) -> object	
load_request(dataset_type: str, list_load_request: list, name: str, save: bool) -> object	
preprocess_dataset(df: pd.DataFrame, save: bool, dataset_type: str, seed: int, name_save: str, name_load: str, load: bool) -> transform	
preprocess_request(df: pd.DataFrame, save: bool, dataset_type: str, seed: int, name_save: str, name_load: str) -> transform	

Ilustración 86. Diseño de Clases: DataSelector.

8.3.1.2.2. Services

En el paquete “services” tenemos dos clases “RedisService” y “InfluxDBService”.

La clase “RedisService” (Ilustración 87) proporciona varios métodos para interactuar con el servicio de *Redis*. No tiene atributos de instancia y todas sus funciones interactúan con *Redis* a través de una conexión proporcionada como argumento.

La función “get_redis_connection” establece una conexión con el servidor *Redis* usando el *host*, el puerto y la base de datos proporcionados. La función “close_redis_connection” cierra una conexión existente a *Redis*. La función “get_redis_list” recupera todos los elementos de una lista de *Redis* especificada.

La función “get_redis_list_last_n_elements” devuelve los últimos n elementos de una lista de *Redis* específica, mientras que “get_redis_list_last_n_elements_and_delete_them” recupera los últimos n elementos de una lista de *Redis* y luego los elimina de la lista.

Finalmente, “remove_redis_list_last_n_elements” elimina los últimos n elementos de una lista de *Redis* y “remove_redis_list_all_elements” elimina todos los elementos de una lista de *Redis* especificada.

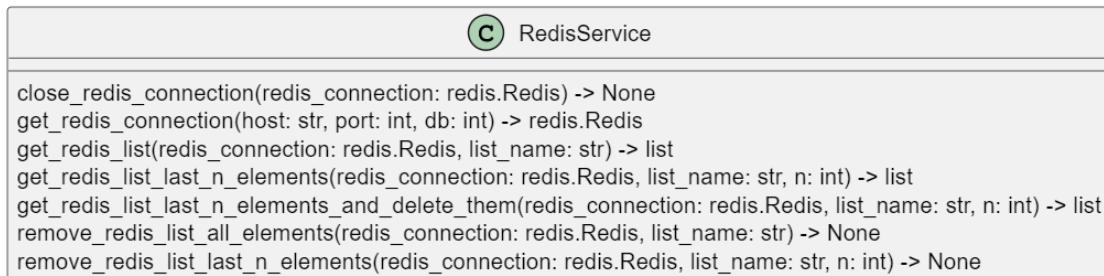


Ilustración 87. Diseño de Clases: RedisService.

La clase “InfluxDBService” (Ilustración 88) proporciona una serie de métodos para interactuar con una base de datos *InfluxDB*. No tiene atributos de instancia y todas sus funciones interactúan con la base de datos a través de una conexión proporcionada como argumento.

La función “get_influxdb_connection” establece una conexión con la base de datos *InfluxDB* utilizando las credenciales proporcionadas (*URL*, *token* y organización) y devuelve un objeto “InfluxDBClient”. Por otro lado, la función “close_influxdb_connection” se utiliza para cerrar una conexión existente con la base de datos *InfluxDB*.

La función “add_influxdb_data” se usa para escribir datos en una base de datos *InfluxDB*. Esta función toma una conexión a *InfluxDB*, un nombre de “bucket”, un nombre de medición, un valor y un diccionario de etiquetas como argumentos. Primero, crea un objeto “Point” con el nombre de la medición y el valor proporcionado, luego añade las etiquetas al objeto “Point”. Finalmente, escribe el objeto “Point” en el “bucket” de *InfluxDB* especificado. Después de escribir los datos, cierra la API de escritura.

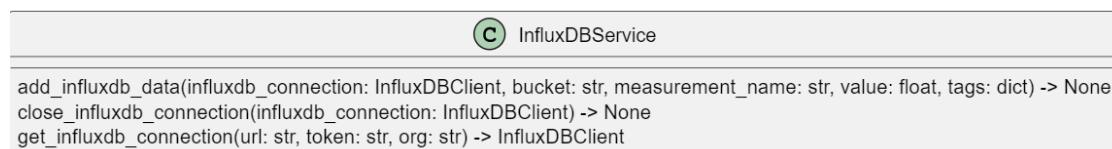


Ilustración 88. Diseño de Clases: InfluxDBService.

8.3.1.2.3. Storage

En el paquete “storage” tenemos dos clases “ScoreManager” y “DataCollector”.

La clase “ScoreManager” (Ilustración 89) se usa para administrar las puntuaciones de las solicitudes. Durante su inicialización, crea una instancia de “InfluxDBService”, que se utiliza para interactuar con la base de datos *InfluxDB*, y una instancia de “ApoloPredict”, que se utiliza para clasificar las solicitudes.

La función “push_data_to_influxdb” de la clase “ScoreManager” se utiliza para enviar datos a la base de datos *InfluxDB*. Esta función toma como entrada el último elemento de la lista de *Redis*, junto con una *URL* y una organización para la base de datos *InfluxDB*. Dentro de la función, primero se establece una conexión con *InfluxDB* y luego se clasifica el último elemento usando la instancia de “ApoloPredict”. Después de esto, se añade el elemento clasificado a la base de datos *InfluxDB*. Al final de la función, se cierra la conexión con *InfluxDB*. Los pasos clave, como la conexión y desconexión de *InfluxDB* y la adición de datos, se acompañan con mensajes de impresión para rastrear el proceso.

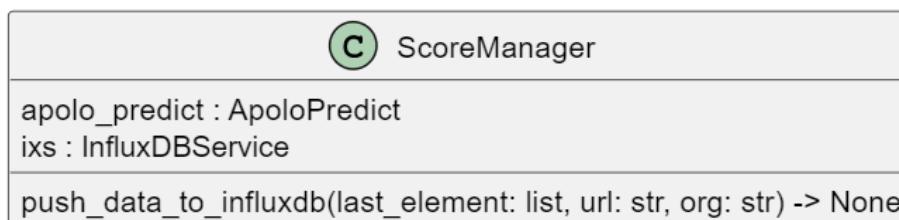


Ilustración 89. Diseño de Clases: ScoreManager.

La clase “DataCollector” (Ilustración 90) es utilizada para recoger datos de una cola de *Redis*. En su inicialización, la clase instancia un objeto de “RedisService”, que se utiliza para interactuar con *Redis*, y también define una lista vacía “last_element” para almacenar el último elemento de la cola de *Redis*.

La función “get_data_from_queue” de esta clase es usada para obtener datos de la cola de *Redis*. La función acepta una conexión a *Redis* y un nombre de lista (con un valor predeterminado de “gatewaylogs”). Dentro de esta función, se hace uso del método “get_redis_list_last_n_elements_and_delete_them” de “RedisService” para obtener el último elemento de la lista de *Redis* y luego eliminarlo de la lista. Esta información es almacenada en la variable “last_element” y luego se imprime.

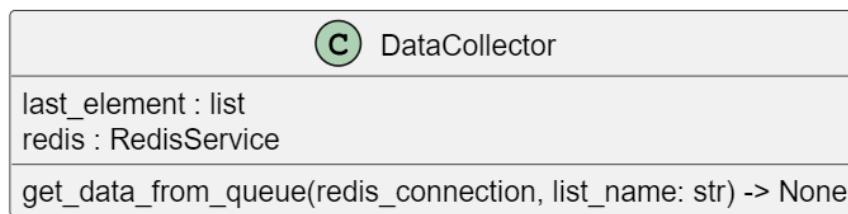


Ilustración 90. Diseño de Clases: DataCollector.

8.3.2. Patrones de Diseño

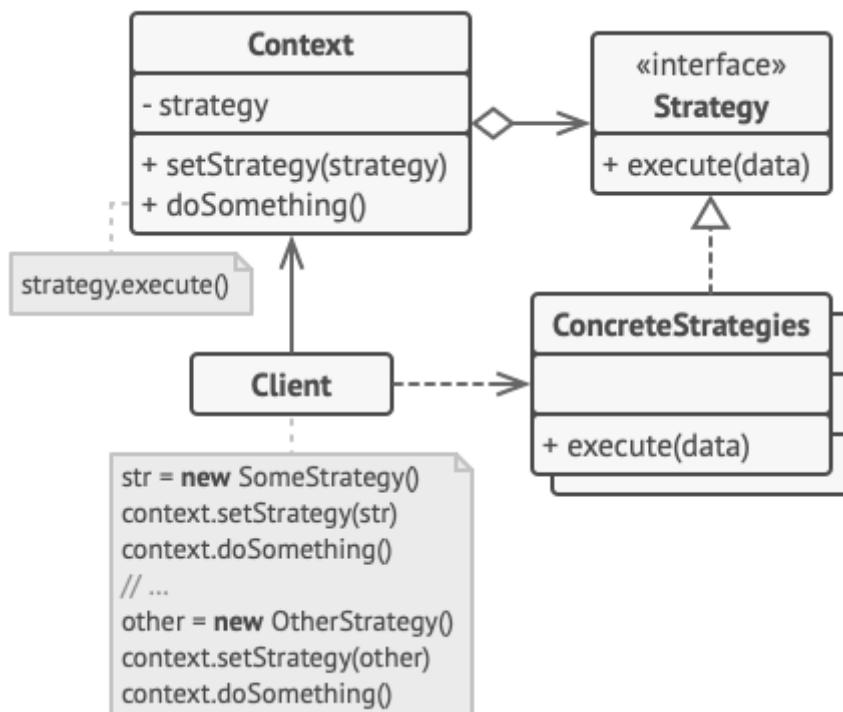


Ilustración 91. Patrón de Diseño: Strategy⁶¹.

El patrón de diseño *Strategy*⁶¹ (Ilustración 91) es un patrón de comportamiento que permite definir una familia de algoritmos, encapsular cada uno en una clase separada y hacer que sus objetos sean intercambiables. Este patrón permite que el algoritmo varíe independientemente del cliente que lo utiliza. En esencia, *Strategy* proporciona una forma de configurar una clase con uno de muchos comportamientos posibles.

⁶¹ <https://refactoring.guru/es/design-patterns/strategy>

En términos de estructura, el patrón *Strategy* implica la creación de una interfaz común para un grupo de clases (estrategias), cada una de las cuales representa un algoritmo posible. Un objeto de contexto mantiene una referencia a un objeto que representa una estrategia y se comunica con este objeto solo a través de la interfaz de estrategia. Los clientes del contexto son responsables de reemplazar la estrategia del contexto con una estrategia adecuada que coincide con la forma en que esperan que el contexto realice su trabajo principal.

En el caso del código presentado, el patrón de diseño *Strategy* se implementa de la siguiente manera.

8.3.2.1. Clase Model y sus Hijas

La clase abstracta “Model” proporciona la interfaz base que todos los modelos de *Machine Learning* deben seguir. Esta clase tiene algunos métodos definidos, como “model_train_test”, “predict”, y “exe”, que son comunes a todos los modelos de *Machine Learning*. Sin embargo, también tiene un método “expecific_model” que es “abstracto” y cada clase hija debe implementar.

El método “expecific_model” es donde se aplica el patrón de diseño *Strategy*. Cada modelo de *Machine Learning*, como *KNeighbors*, *Logistic Regression*, etc., tiene su propia estrategia o implementación para este método. Así, al llamar a “expecific_model”, se ejecuta la estrategia específica del modelo en uso.

Por consiguiente, al usar la clase “Model”, puedes cambiar entre diferentes modelos de *Machine Learning* simplemente eligiendo una clase hija diferente.

8.3.2.2. Clase ClearData y sus Hijas

De forma similar, la clase “ClearData” también implementa el patrón de diseño *Strategy*. “ClearData” es una clase “abstracta” que proporciona una interfaz común para diferentes técnicas de preprocesamiento de datos. Los métodos como “best_features_func”, “drop_bad_elements”, “drop_duplicate_columns”, entre otros; están definidos en la clase “ClearData”.

Sin embargo, la clase “ClearData” también define los métodos “abstractos” “clear_data”, “load_data” y “save_data”, que cada una de sus clases hijas debe implementar.

Cada clase hija de “ClearData” representa una estrategia diferente de preprocesamiento de datos. En consecuencia, al utilizar la clase “ClearData”,

puedes cambiar entre diferentes técnicas de preprocesamiento de datos simplemente eligiendo una clase hija diferente.

8.3.3. Diagrama de Clases

En la Ilustración 92 se puede observar el diagrama de clases en el que aparecen todas las clases descritas y sus relaciones.

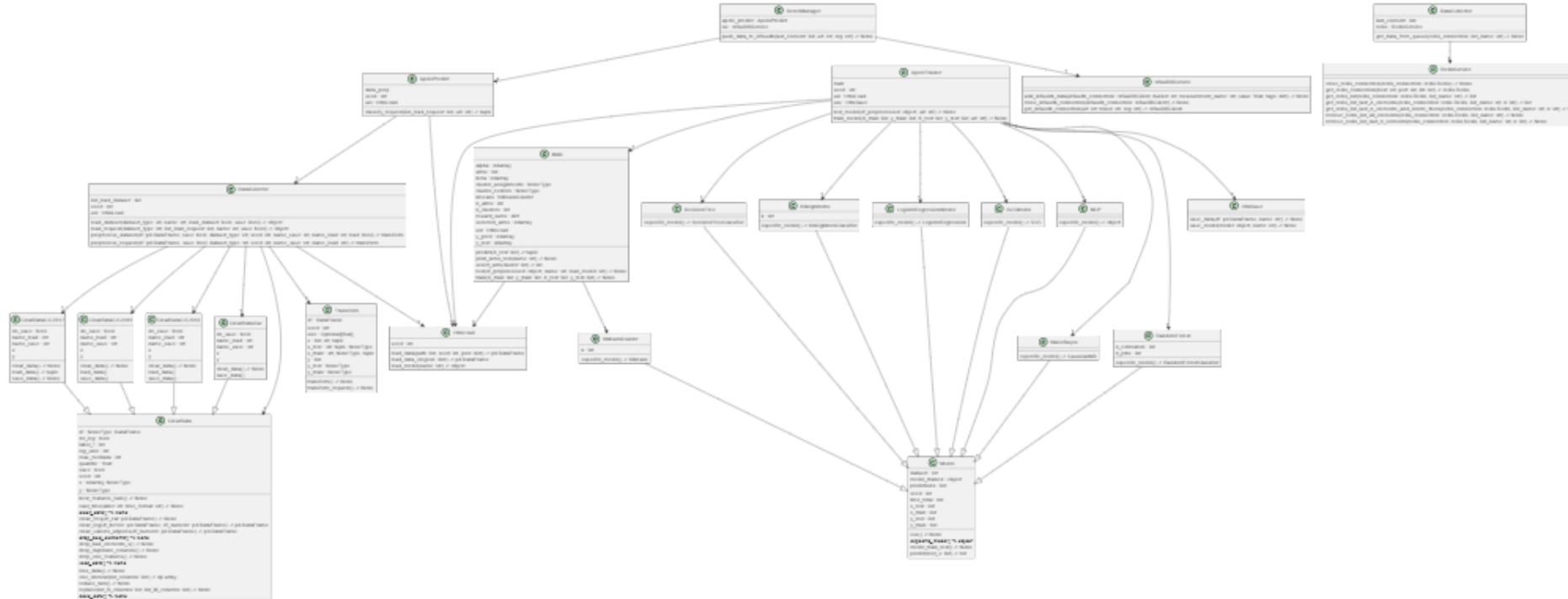


Ilustración 92. Diagrama de Clases.

8.4. Diseño del Sistema de Colas

El diseño del sistema de colas para el sistema se basa en la utilización de *Redis*, un sistema de almacenamiento de datos en memoria. Este sistema se caracteriza por su estructura basada en el almacenamiento de datos clave-valor, que se adapta perfectamente a las necesidades de nuestro proyecto.

8.4.1. Cola FIFO

El sistema de colas de nuestro proyecto sigue el principio *FIFO* (*First In, First Out*)⁶², lo que significa que la primera petición que entra en la cola es la primera que sale, se puede ver un ejemplo en la Ilustración 93. Esto se logra utilizando las operaciones de lista de *Redis*.

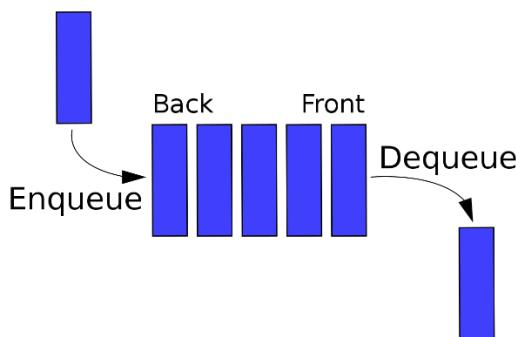


Ilustración 93. FIFO (First In, First Out) ⁶²

Utilizar una cola *FIFO* asegura que las peticiones se procesen en el orden en que llegan, lo que es una característica fundamental para nuestro sistema.

8.4.2. Gestión de Peticiones mediante Clave-Valor

Redis gestiona las peticiones mediante el almacenamiento de datos clave-valor. En nuestro sistema, cada petición se almacena como un valor en una lista de *Redis*. La clave de la lista es el nombre de la cola en nuestro caso tenemos una única clave llamada “*gatewaylogs*”, en esta única clave se almacenan los valores que son las peticiones que están en la cola; esto permite una gestión eficiente de las peticiones, ya que se pueden añadir y eliminar peticiones de la cola simplemente manipulando los valores de la lista de *Redis*.

⁶² https://es.wikipedia.org/wiki/Cola_%28inform%C3%A1tica%29

8.5. Diseño de la Base Datos de Series Temporales

El diseño de la base de datos de series temporales para el proyecto **Apolo** se basa en la utilización de *InfluxDB*, un sistema de gestión de bases de datos de series temporales. Este sistema se caracteriza por su estructura basada en *buckets* y *measurements*, se puede ver esta disposición en la Ilustración 94. Esta estructura se adapta perfectamente a las necesidades de nuestro proyecto.

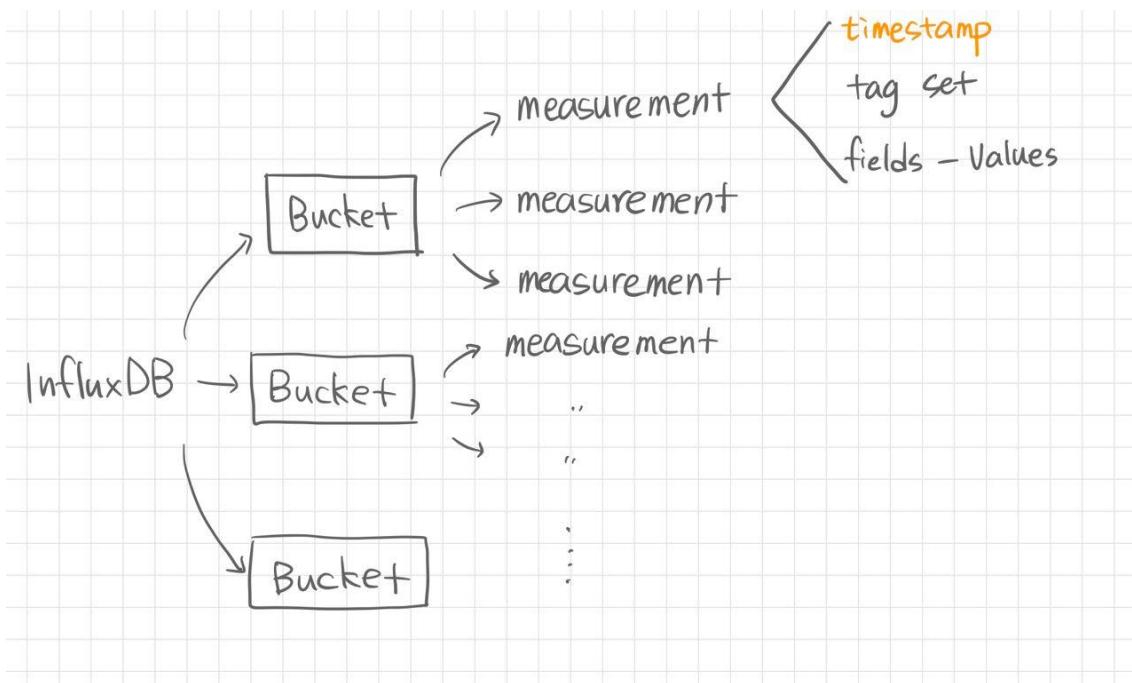


Ilustración 94. Estructuración interna de una Base de Datos InfluxDB⁶³

8.5.1. Buckets

En *InfluxDB*, los *buckets* cumplen una funcionalidad similar a una base de datos relacional tradicional. En el contexto de nuestro sistema, se ha optado por la creación de un único *bucket*. Este *bucket* será el contenedor principal donde se almacenarán todas las medidas o *measurements* que se generen en el sistema.

El uso de un único *bucket* simplifica la gestión de los datos, ya que todos los datos de series temporales se encuentran en un solo lugar. Esto facilita las

⁶³ <https://velog.io/@jee-9/InfluxDB-2-Structure-Data-elements>

operaciones de consulta y análisis de los datos, ya que no es necesario buscar en múltiples buckets.

8.5.2. Measurements

Los *measurements*, o medidas, son el núcleo de *InfluxDB*. Su objetivo es almacenar el valor de una variable a lo largo del tiempo, es decir, cada medida se compone de un par de datos: tiempo y valor. Para el proyecto **Apolo**, se ha definido un único tipo de medida, denominado “scores” o “anomaly_scores”.

Cada medida “scores” o “anomaly_scores” almacena, además de la fecha en la que se tomó el dato (tiempo) y el valor de anomalía generado por **Apolo** (valor), un diccionario de etiquetas. Este diccionario de etiquetas contiene información adicional sobre la petición, lo que permitirá su posterior identificación y análisis.

El uso de un único tipo de medida simplifica la estructura de los datos y facilita su interpretación. Además, al almacenar la información de la petición en un diccionario de etiquetas, se facilita la identificación y el análisis de los datos.

Capítulo 9. Evaluación del Sistema

En este capítulo se describe las métricas, procedimientos y resultados obtenidos de la evaluación del sistema. También, describiremos qué *hardware* se ha utilizado para el desarrollo de las pruebas, las diferentes metodologías explicadas, los resultados de la experimentación de **Apolo** y los resultados obtenidos en nuestra propia arquitectura.

9.1. Hardware utilizado

Para la realización de las pruebas se ha utilizado un equipo con la siguiente configuración:

- Procesador: AMD Ryzen 5 3600 6-core processor x 12.
- Memoria RAM: 16,0 HiB.
- Gráficos: AMD Radeon rx 6700 xt.
- Almacenamiento SSD: 1 TB.
- Sistema Operativo: Ubuntu 22.04.2 LTS.

9.2. Metodología

En este apartado se describirán y explicarán las diferentes métricas usadas para evaluar el sistema, más concretamente **Apolo**. Estas métricas son: Precisión (Accuracy), Tasa de Detección (Recall), Curva ROC y el valor *F1*.

9.2.1. Precisión / Exactitud

La precisión o exactitud (Accuracy) es una medida que se utiliza en estadísticas y *ML* para evaluar el rendimiento de un modelo de clasificación. Esta métrica cuantifica la proporción de predicciones correctas realizadas por el modelo sobre el total de predicciones.

La fórmula para calcular la precisión se puede observar en la Ecuación 1.

$$\text{Accuracy} = (VP + VN) / (VP + FP + VN + FN)$$

Ecuación 1. Formula Accuracy.

Donde:

- Verdaderos Positivos (*VP*) es el número de casos positivos que el modelo ha clasificado correctamente como positivos.
- Verdaderos Negativos (*VN*) es el número de casos negativos que el modelo ha clasificado correctamente como negativos.
- Falsos Positivos (*FP*) es el número de casos negativos que el modelo ha clasificado incorrectamente como positivos.
- Falsos Negativos (*FN*) es el número de casos positivos que el modelo ha clasificado incorrectamente como negativos.

En este contexto, un caso “positivo” es una instancia en la que la condición que el modelo está intentando predecir es verdadera, y un caso “negativo” es una instancia en la que la condición que el modelo está intentando predecir es falsa.

La precisión puede variar de 0 a 1, donde 1 indica que todas las predicciones realizadas por el modelo son correctas, y 0 indica que todas las predicciones realizadas por el modelo son incorrectas.

Sin embargo, es importante tener en cuenta que la precisión por sí sola puede ser una métrica engañosa, especialmente en conjuntos de datos desequilibrados. En estos casos, otras métricas como la tasa de detección (*recall*) o el valor *F1* pueden proporcionar una evaluación más matizada del rendimiento del modelo.

9.2.2. Tasa de Detección

La Tasa de Detección, también conocida como Sensibilidad o *Recall*, es una medida que se utiliza en estadísticas y aprendizaje automático para evaluar la precisión de un modelo de clasificación. Específicamente, el *recall* mide la capacidad del modelo para encontrar todos los casos positivos relevantes dentro de un conjunto de datos.

La fórmula para calcular el *Recall* se puede observar en la Ecuación 2 .

$$\text{Recall} = VP / (VP + FN)$$

Ecuación 2. Formula Recall.

Donde:

- Verdaderos Positivos (*VP*) es el número de positivos que el modelo ha clasificado correctamente como positivos.
- Falsos Negativos (*FN*) es el número de positivos que el modelo ha clasificado incorrectamente como negativos.

En este contexto, un caso “positivo” es una instancia en la que la condición que el modelo está intentando predecir es verdadera. Por ejemplo, si estás usando un modelo de *ML* para detectar anomalías en el tráfico web, un caso “positivo” sería una petición web que sea realmente maligna.

Por lo tanto, el *recall* se calcula tomando el número de Verdaderos Positivos (los casos que son positivos y que el modelo ha clasificado correctamente como positivos) y dividiéndolo por la suma de Verdaderos Positivos y Falsos Negativos (los casos que son positivos pero que el modelo ha clasificado incorrectamente como negativos).

El *recall* puede variar de 0 a 1, donde un recall de 1 significa que el modelo es capaz de detectar todos los casos positivos correctamente, mientras que un recall de 0 significa que el modelo no puede detectar ningún caso positivo correctamente.

9.2.3. Curva ROC

La curva *ROC* (*Receiver Operating Characteristic*) es una representación gráfica que muestra el rendimiento de un modelo de clasificación binaria a medida que varía el umbral de discriminación.

Esta curva traza dos parámetros:

- La Tasa de Verdaderos Positivos (*TVP*) en el eje y.
- La Tasa de Falsos Positivos (*TFP*) en el eje x.

La Tasa de Verdaderos Positivos (*TVP*), también conocida como sensibilidad o *recall* (explicado en el punto 9.2.2. “Tasa de Detección” de este capítulo), se calcula siguiendo la Ecuación 2.

Donde los Verdaderos Positivos (*VP*) es el número de observaciones positivas correctamente clasificadas por el modelo y los Falsos Negativos (*FN*) es el número de observaciones positivas incorrectamente clasificadas como negativas por el modelo.

La Tasa de Falsos Positivos (*TFP*), también conocida como la tasa de alarma o 1 - especificidad, se calcula siguiendo la Ecuación 3.

$$TFP = FP / (FP + VN)$$

Ecuación 3. Formula TFP, Curva ROC.

Donde los Falsos Positivos (*FP*) es el número de observaciones negativas incorrectamente clasificadas como positivas por el modelo y los Verdaderos Negativos (*VN*) es el número de observaciones negativas correctamente clasificadas por el modelo.

La curva *ROC* muestra la relación entre la tasa de detección y la especificidad de un modelo. Un área bajo la curva (*AUC*) cercana a 1 indica que el modelo tiene

una buena capacidad de discriminación, mientras que un área cercana a 0.5 indica que el modelo no es mejor que una clasificación aleatoria.

9.2.4. Valor F1

El valor *F1*, o simplemente *F1-score*, es una medida de rendimiento que se utiliza en estadísticas y aprendizaje automático para evaluar la precisión de un modelo de clasificación binaria cuando las clases son desequilibradas. Este valor es el promedio armónico entre la precisión (*Precision*) y la tasa de detección (*Recall*).

La precisión (*Precision*) es la proporción de Verdaderos Positivos (*VP*) entre el total de casos clasificados como positivos (Verdaderos Positivos + Falsos Positivos), y su cálculo se puede observar en Ecuación 4.

$$\text{Precision} = VP / (VP + FP)$$

Ecuación 4. Formula Precision.

Por otro lado, la Tasa de Detección o *Recall* es la proporción de Verdaderos Positivos (*VP*) entre el total de casos que son realmente positivos (Verdaderos Positivos + Falsos Negativos), y se calcula tal y como se ve en la Ecuación 2.

El valor *F1-score* se calcula como el promedio armónico de la precisión y el *recall*, dicho calculo se encuentra en la Ecuación 5.

$$F1 - score = 2 * (Precision * Recall) / (Precision + Recall)$$

Ecuación 5. Formula F1-score.

El valor *F1-score* varía entre 0 y 1, donde 1 indica el rendimiento perfecto del modelo, y 0 indica que el modelo no tiene rendimiento. Cuando las clases están desequilibradas, el *F1-score* puede ser una mejor medida de rendimiento que la precisión o exactitud (*Accuracy*) ya que toma en cuenta tanto la precisión, como la tasa de detección (*recall*).

9.3. Validación

En esta sección vamos a explicar que técnicas hemos utilizado para validar los resultados obtenidos en la experimentación de **Apolo**.

Cabe mencionar que durante toda la experimentación del modelo se usó la misma semilla para que los resultados puedan ser replicados. Dicha semilla es la 42.

9.3.1. Validación Cruzada

La validación cruzada (*cross-validation*)⁶⁴, es un método esencial para la evaluación de modelos en el campo del *Machine Learning* (ML). Este enfoque nos permite obtener una medición precisa del rendimiento de un modelo y de su capacidad para generalizar en datos no vistos.

En la experimentación de los modelos de *ML* y de **Apolo**, hemos implementado la validación cruzada con un $K = 10$, también conocido como *10-fold cross-validation*. El uso de este método es particularmente apropiado dado el tamaño considerable de nuestro conjunto de datos.

El proceso de validación cruzada *k-fold* implica dividir el conjunto de datos completo en “ k ” subconjuntos (*folds*) de aproximadamente el mismo tamaño. En nuestro caso, con $K = 10$, el conjunto de datos se divide en 10 subconjuntos. A continuación, el modelo se entrena y se valida 10 veces, cada vez utilizando un subconjunto diferente como conjunto de validación y los restantes 9 subconjuntos como conjunto de entrenamiento. Esta explicación teórica se puede ver más clara en la Ilustración 95.

⁶⁴ [https://en.wikipedia.org/wiki/Cross-validation_\(statistics\)](https://en.wikipedia.org/wiki/Cross-validation_(statistics))



Ilustración 95. Validación cruzada ⁶⁴.

Cada una de las 10 iteraciones proporciona su propia medida de rendimiento, y la evaluación final del modelo se realiza promediando las 10 medidas. Esto proporciona una estimación más robusta y menos sesgada del rendimiento del modelo.

La validación cruzada *k-fold* también tiene la ventaja de utilizar todo el conjunto de datos para el entrenamiento y la validación, asegurando que la evaluación del rendimiento del modelo no depende de una partición particularmente afortunada o desafortunada de los datos.

9.4. Experimentación de Apolo

En este apartado explicaremos y daremos los resultados de la experimentación de **Apolo**, hemos decidido separar la experimentación de **Apolo** del resto del sistema, ya que es una parte crucial del mismo, además de los resultados de **Apolo** daremos los resultados de los diferentes modelos por separado, para poder hacer una comparativa del verdadero potencial de **Apolo**.

Hemos utilizado las métricas descritas en el apartado 9.2. “Metodología” y hemos validado los modelos haciendo uso de las técnicas de validación descritas en el apartado 9.3. “Validación”, ambos puntos de este mismo capítulo.

Diseñamos tres escenarios para evaluar el rendimiento de nuestra solución propuesta.

Como este proyecto tiene un enfoque más de investigación, las pruebas más clásicas de los proyectos de desarrollo como pueden ser las unitarias, aceptación, etc.; se han decidido incluir dentro de los diferentes escenarios que proponemos, estos escenarios se encargan de tratar los diferentes casos en los que se puede probar el sistema.

9.4.1. Primer Escenario

En este apartado vamos a definir y dar los resultados correspondientes al primer escenario descrito.

9.4.1.1. Definición del Escenario

En el primer escenario, probamos nuestra solución en tres conjuntos de datos (descritos en el Capítulo 2. “Estado Actual de los Conocimientos Científico-Técnicos”, más concretamente en el punto “Principales Datasets Utilizados en Sistemas de Detección de Intrusos”):

- CIC-IDS-2017.
- CSE-CIC-IDS2018.
- CIC-DDoS-2019.

Dada nuestra limitación en cuanto a la disponibilidad de una máquina potente, optamos por utilizar un subconjunto representativo de los conjuntos de datos en lugar de los datos completos para entrenar y probar nuestros modelos de manera eficiente.

Para el conjunto de datos *CIC-DDoS-2019*, se utilizó el conjunto completo, mientras que para el conjunto de datos *CSE-CIC-IDS2018*, se utilizó el subconjunto correspondiente a la fecha “15-02-2018”. Finalmente, para el conjunto de datos *CIC-IDS-2017*, se seleccionaron los siguientes subconjuntos de datos:

- Viernes en horas laborales de la tarde (*DDoS*).
- Viernes en horas laborales de la tarde (*PortScan*).
- Viernes en horas laborales de la mañana.
- Lunes en horas laborales.
- Jueves en horas laborales de la tarde (*Infiltration*).
- Jueves en horas laborales de la mañana (*WebAttacks*).
- Martes en horas laborales.

9.4.1.2. Resultados Obtenidos

En este escenario, entrenamos los clasificadores seleccionados en los conjuntos de datos *CIC-IDS-2017*, *CSE-CIC-IDS-2018* y *CIC-DDoS2019*. También, hemos entrenado **Apolo** en los mismos conjuntos de datos y con los mismos clasificadores, para que los diferentes resultados obtenidos no sean por diferencias en el preprocesado y entrenamiento.

Para entrenar a **Apolo**, utilizamos el algoritmo *K-Means* [71], [77] para agrupar los datos en 2 clústeres. Para cada clúster, entrenamos los clasificadores seleccionados y actualizamos el algoritmo de *MAB* con los resultados.

Los resultados obtenidos de los experimentos se presentan en las tablas: Tabla 51, Tabla 52 y Tabla 53 y en las ilustraciones: Ilustración 96, Ilustración 97 y la Ilustración 98; respectivamente. Según las tablas, **Apolo** demuestra altas tasas de detección y precisión, comparables a los clasificadores escogidos.

Es importante mencionar que, entre todos los conjuntos de datos, **Apolo** no obtiene las mejores ni las peores puntuaciones. Esto se debe a que **Apolo** selecciona internamente entre los mismos clasificadores. Por lo tanto, la puntuación más alta que **Apolo** puede alcanzar está limitada a la puntuación máxima del mejor clasificador, mientras que nunca puede rendir tan mal como el peor clasificador, ya que tiene otras opciones mejores.

Nuestros resultados demuestran que incluso con la integración de nuevos mecanismos de seguridad, **Apolo** aún puede proporcionar altas puntuaciones de

precisión y tasa de detección en entornos tradicionales de clasificación de tráfico de red (los dataset de la literatura). Por lo tanto, **Apolo** es capaz de preservar la funcionalidad fundamental de un *IDS*.

CIC-IDS-2017	Métricas	Modelos					
		MLP	NB	RF	DT	LR	Apolo
	Precisión	0,9766	0,6694	0,9996	0,9980	0,9516	0,9740
	Tasa de Detención	0,9731	0,8049	0,9996	0,9962	0,8360	0,9420
	F1	0,9760	0,6118	0,9991	0,9959	0,8858	0,7699
	ROC	0,9998	0,8289	1,0000	0,9972	0,9902	0,9420

Tabla 51. Primer escenario: Métricas de los modelos y de Apolo entrenados con CIC-IDS-2017.

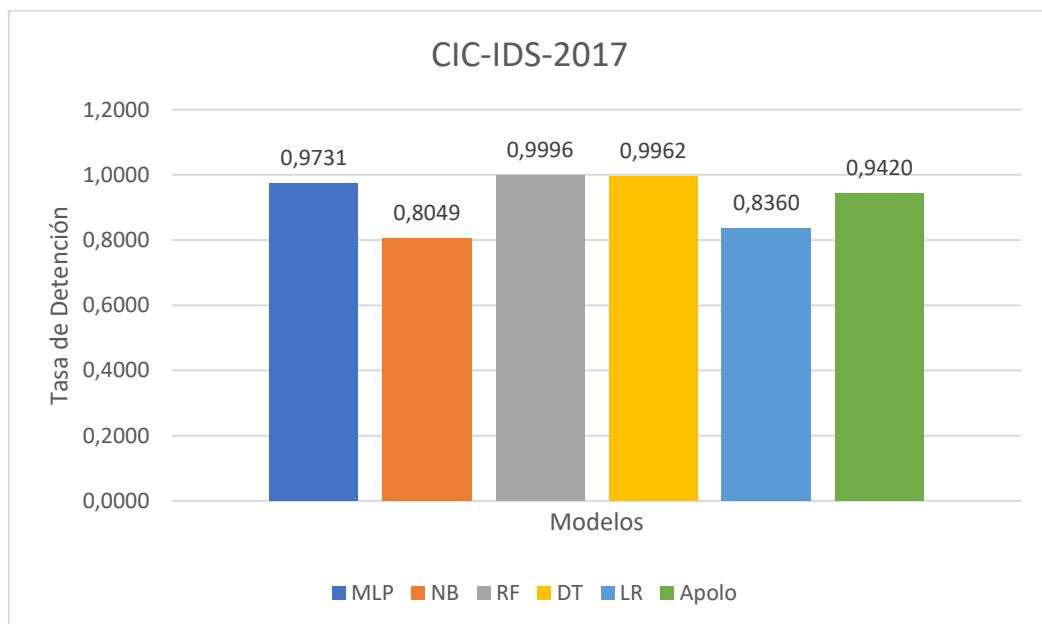


Ilustración 96. Primer escenario: Tasa de Detección (Recall) de los modelos y de Apolo entrenados con CIC-IDS-2017.

CSE-CIC-2018	Métricas	Modelos					
		MLP	NB	RF	DT	LR	Apolo
Precisión	0,9916	0,7280	0,9993	0,9939	0,9593	0,9064	
Tasa de Detención	0,9367	0,8563	0,9967	0,9961	0,6008	0,9419	
F1	0,9545	0,5507	0,9963	0,9968	0,6556	0,7303	
ROC	0,9945	0,8605	0,9993	0,9984	0,9592	0,9419	

Tabla 52. Primer escenario: Métricas de los modelos y de Apolo entrenados con CIC-IDS-2018.

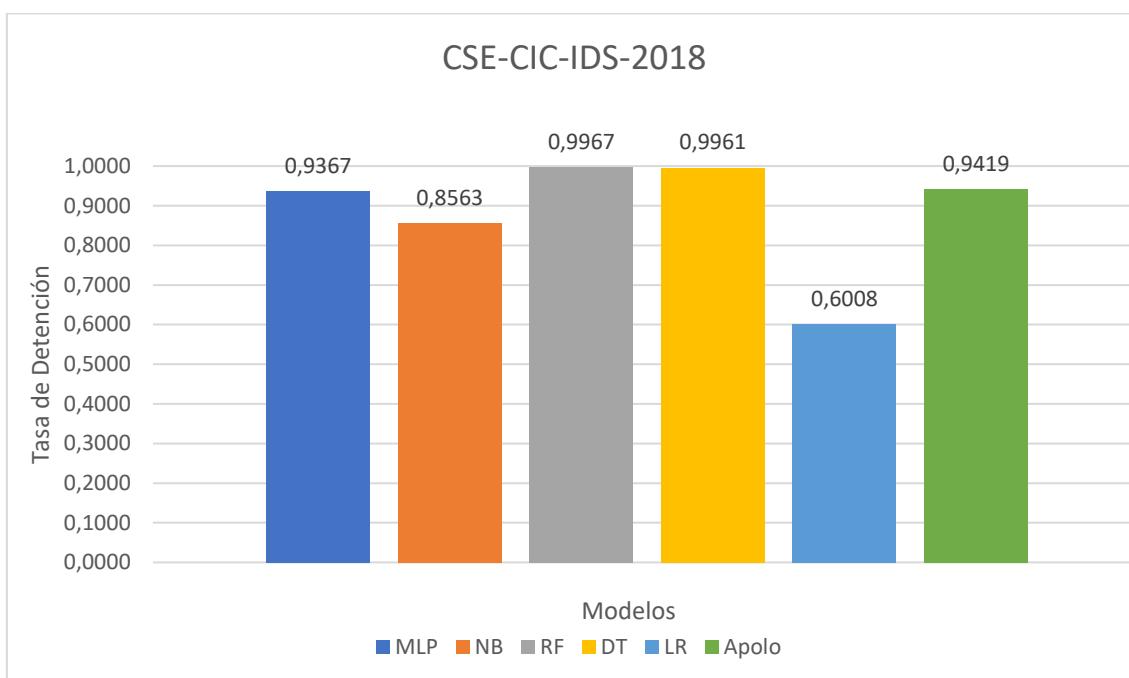


Ilustración 97. Primer escenario: Tasa de Detección (Recall) de los modelos y de Apolo entrenados con CIC-IDS-2018.

CIC-DDoS-2019	Métricas	Modelos					
		MLP	NB	RF	DT	LR	Apolo
Precisión	0,9998	0,9990	0,9999	0,9999	0,9970	0,9996	
Tasa de Detención	0,8985	0,8709	0,9932	0,9999	0,7930	0,9691	
F1	0,9186	0,7148	0,9957	0,9991	0,8541	0,8521	
ROC	0,9817	0,9753	0,9999	0,9999	0,9796	0,9691	

Tabla 53. Primer escenario: Métricas de los modelos y de Apolo entrenados con CIC-DDoS-2019.

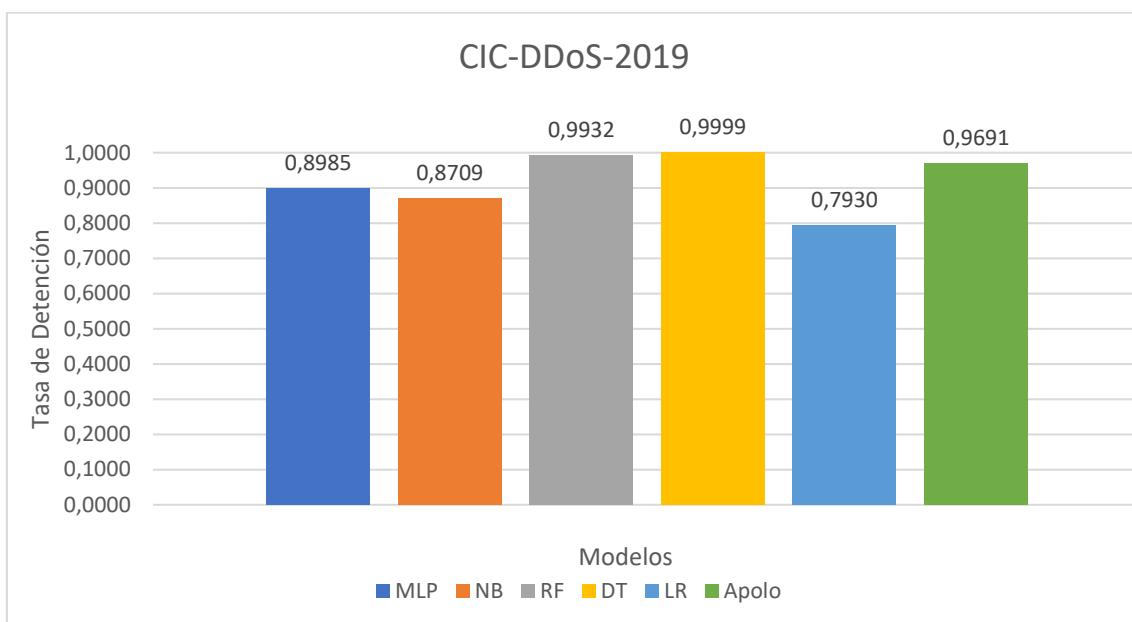


Ilustración 98. Primer escenario: Tasa de Detección (Recall) de los modelos y de Apolo entrenados con CIC-DDoS-2019.

9.4.2. Segundo Escenario

En este apartado vamos a definir el segundo escenario, los ataques que se utilizan en el mismo y dar los resultados correspondientes.

9.4.2.1. Definición del Escenario

En el segundo escenario, utilizamos varios ataques de *AML* de caja gris/negra para evaluar la capacidad de **Apolo** a la hora de defenderse contra ataques de este tipo.

Las puntuaciones de los clasificadores utilizados para comparar nuestra solución pueden ser diferentes a las reportadas en diferentes trabajos relacionados de la literatura debido a varios factores:

- La máquina en la que se realiza el entrenamiento puede diferir de la utilizada en trabajos relacionados. Esto puede afectar la velocidad y eficiencia del proceso de entrenamiento, lo que a su vez puede influir en la precisión final de los clasificadores.
- El preprocessamiento de los datos puede variar entre nuestra solución y los trabajos relacionados. Las técnicas de preprocessamiento pueden impactar significativamente la calidad de los datos y, por ende, el rendimiento de los clasificadores. Por lo tanto, las diferencias en las técnicas de preprocessamiento pueden conducir a diversos niveles de precisión en los clasificadores.

Es importante tener en cuenta estas diferencias al comparar nuestra solución con trabajos relacionados de la literatura y considerar el impacto de estos factores en el rendimiento de los clasificadores.

9.4.2.2. Ataques Utilizados

En este escenario de prueba, utilizamos varios ataques de *AML* de caja gris/negra para evaluar la capacidad de nuestra solución para defenderse contra dichos ataques.

Para simplificar el proceso de evaluación, usamos el conjunto de datos *CIC-IDS-2017* para entrenar a los clasificadores y **Apolo** porque es el conjunto de datos más popular y extendido en la literatura.

Comparamos la precisión y la tasa de detección de los clasificadores y **Apolo** frente a los ataques *AML* de caja gris/negra. Como mencionamos antes, solo probamos con ataques de caja gris/negra porque los ataques de caja blanca no son realistas en escenarios del mundo real.

Los ataques utilizados en este escenario son los siguientes:

- Ataque de optimización de orden cero (ZOO) [48].
- Ataque *HopSkipJump* (HSJA) [78].
- Ataques basados en *W-GAN* [9]

Optamos por estos ataques en particular porque abarcan un amplio espectro de posibles estrategias de evasión de *AML* y se encuentran entre los más extendidos y populares dentro de la literatura.

9.4.2.3. Resultados Obtenidos

Los clasificadores y la implementación de **Apolo** utilizados como objetivos de los ataques son los que se entrenaron en el entorno previo con el conjunto de datos *CIC-IDS-2017*.

Estos resultados comparan las medidas dadas al aplicar los diferentes ataques de *AML* en los diferentes clasificadores y en Apolo. Cabe mencionar que si dejáramos el ataque ejecutando en un tiempo infinito, ninguna defensa podría detenerlo, puesto que estos ataques replican el modelo y se van refinando con cada petición que envían al modelo. Estas mediciones se han tomado todas en el mismo tiempo por lo que las condiciones han sido las mismas para todos los modelos.

9.4.2.3.1. Ataque de Optimización de Orden Cero (ZOO)

Comenzando con el Ataque de Optimización de Orden Cero (ZOO), hemos utilizado la implementación de código abierto proporcionada por ART [53], y hemos creado una clase de “Clasificador” para que **Apolo** pueda ser utilizado como modelo.

El ataque se lanzó con los siguientes parámetros para cada clasificador:

- *Classifier*: la instancia de la clase del “Clasificador” al que se va a atacar.
- *targeted*: *True*.
- *learning_rate*: 0.01.
- *max_iter*: 100.

El ataque se lanzó contra los diferentes modelos y los resultados se muestran en la Tabla 54 y en las ilustraciones: Ilustración 99 (Recall) y la Ilustración 100 (Precisión). Según estos resultados, el ataque fue efectivo en cada escenario, resultando en tasas de detección reducidas en todos los clasificadores. Sin

embargo, aunque las puntuaciones de precisión y tasa de detección de la implementación de **Apolo** también disminuyeron, estas se mantuvieron considerablemente superiores a las de los otros clasificadores, con altas puntuaciones de precisión y tasa de detección.

ZOO	Métricas	Modelos					
		MLP	NB	RF	DT	LR	Apolo
	Precisión	0,7630	0,7160	0,7730	0,7890	0,5210	0,9304
Tasa de Detención	0,5260	0,5500	0,5460	0,5780	0,0420	0,8772	

Tabla 54. Segundo escenario: Tasa de Detección (Recall) y Precisión de los modelos y de Apolo. Utilizando el ataque ZOO.

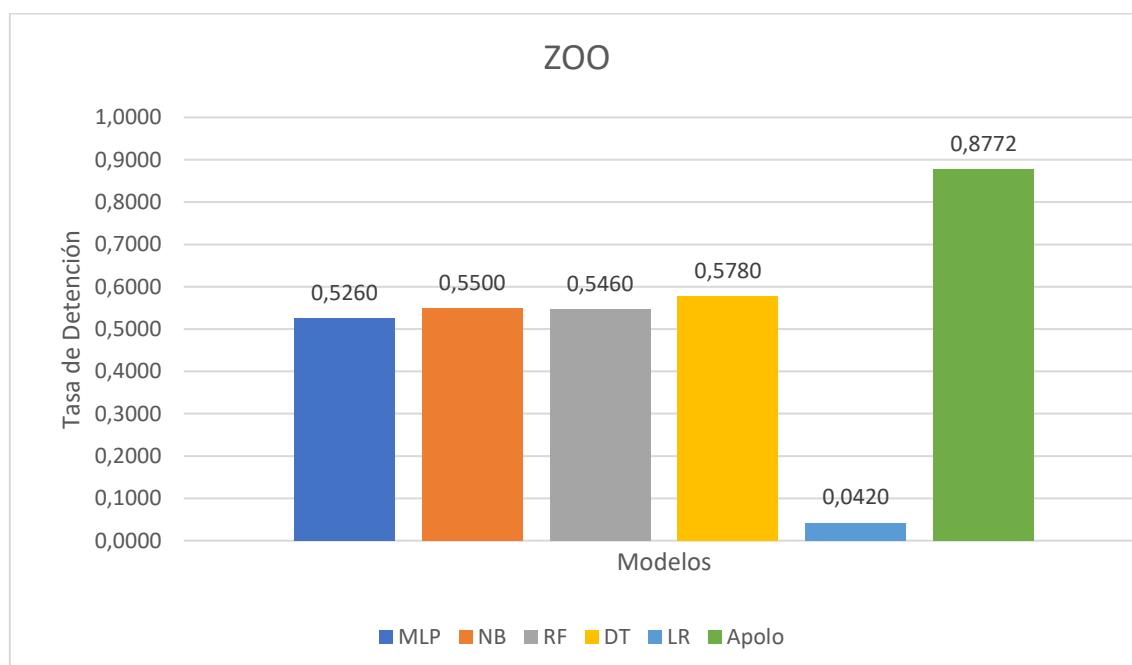


Ilustración 99. Segundo escenario: Tasa de Detección (Recall) de los modelos y de Apolo. Utilizando el ataque ZOO.



Ilustración 100. Segundo escenario: Precisión de los modelos y de Apolo. Utilizando el ataque ZOO.

9.4.2.3.2. Ataque HopSkipJump (HSJA)

Para lanzar el Ataque *HopSkipJump (HSJA)*, hemos utilizado la implementación de código abierto proporcionada por *ART*, y hemos creado un “Clasificador” como en el ataque *ZOO*.

El ataque se lanzó con los siguientes parámetros para cada clasificador:

- *Classifier*: la instancia de la clase del “Clasificador” al que se va a atacar.
- *targeted*: *True*.
- *max_iter*: 100.
- *norm*: *inf*.

Los resultados del ataque se muestran en la Tabla 55 y en las ilustraciones: Ilustración 101 (Recall) y la Ilustración 102 (Precisión). El ataque fue muy efectivo en todos los clasificadores, resultando en tasas de detección reducidas hasta puntuaciones cercanas a cero. La excepción fue la implementación de **Apolo**, que logró mantener una tasa de detección > 0.5 . En la precisión también hubo una disminución de los resultados de los modelos, pero **Apolo** se volvió a posicionar como el mejor entre las opciones.

HopSkipJump	Métricas	Modelos					
		MLP	NB	RF	DT	LR	Apolo
		Precisión	0,5002	0,4301	0,5001	0,5051	0,5900
	Tasa de Detención	0,0000	0,0000	0,0000	0,0100	0,1800	0,5260

Tabla 55. Segundo escenario: Tasa de Detección (Recall) y Precisión de los modelos y de Apolo. Utilizando el ataque HSJA.

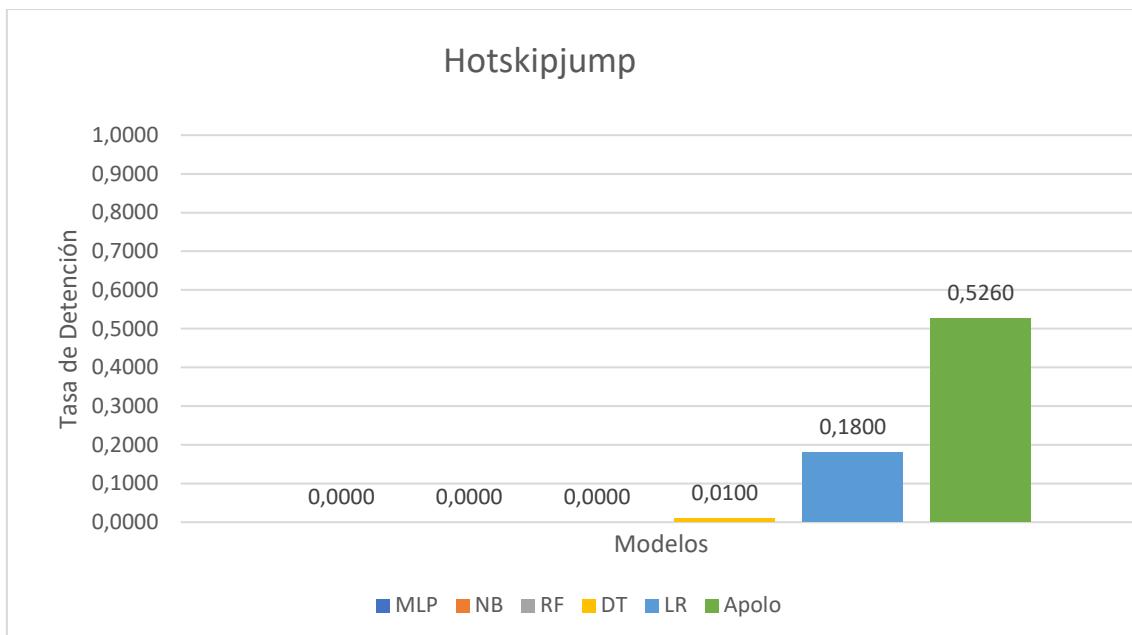


Ilustración 101. Segundo escenario: Tasa de Detección (Recall) de los modelos y de Apolo. Utilizando el ataque HSJA.

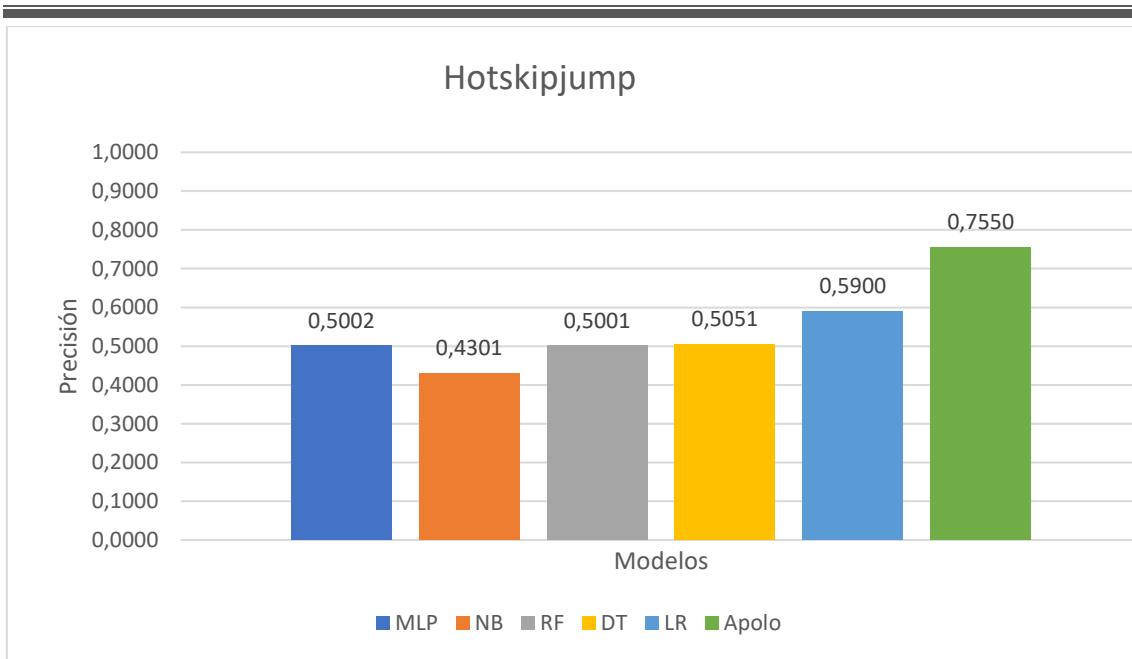


Ilustración 102. Segundo escenario: Precisión de los modelos y de Apolo. Utilizando el ataque HSJA.

9.4.2.3.3. Red Adversaria Generativa de Wasserstein (W-GAN)

Finalmente, el ataque basado en la Red Adversaria Generativa de Wasserstein (W-GAN) se lanzó con una implementación personalizada disponible en el código adjuntado. Esta implementación se basó en el ataque *IDSGAN* [9], adaptado a las necesidades del conjunto de datos seleccionado. Los resultados en la Tabla 56 y en las ilustraciones: Ilustración 103 (Recall) y la Ilustración 104 (Precisión); se obtuvieron después de lanzar el ataque con 100 épocas. El ataque basado en W-GAN fue el ataque más efectivo, reduciendo la tasa de detección a cero en todos los clasificadores. Nuestra solución, por otro lado, logró mantener una tasa de detección > 0.4. Mientras que en precisión **Apolo** no obtiene la mejor solución posible.

W-GAN	Métricas	Modelos					
		MLP	NB	RF	DT	LR	Apolo
	Precisión	0,5002	0,4398	0,5001	0,4280	0,3230	0,4260
	Tasa de Detención	0,0000	0,0000	0,0000	0,0000	0,0000	0,4250

Tabla 56. Segundo escenario: Tasa de Detección (Recall) y Precisión de los modelos y de Apolo. Utilizando el ataque W-GAN.

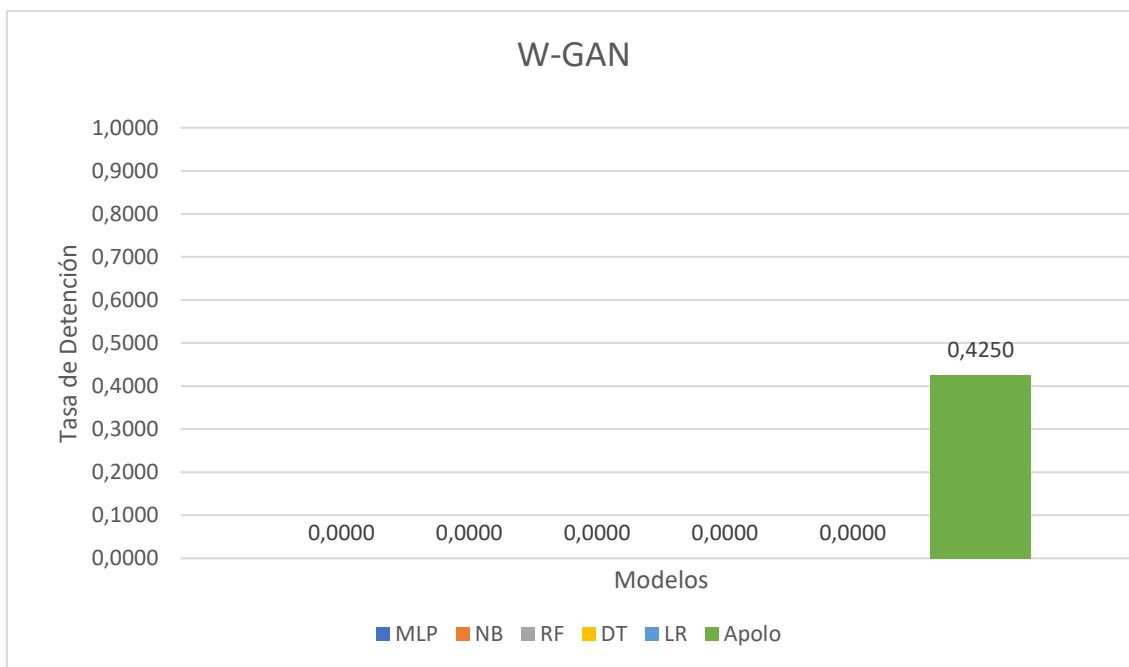


Ilustración 103. Segundo escenario: Tasa de Detección (Recall) de los modelos y de Apolo. Utilizando el ataque W-GAN.

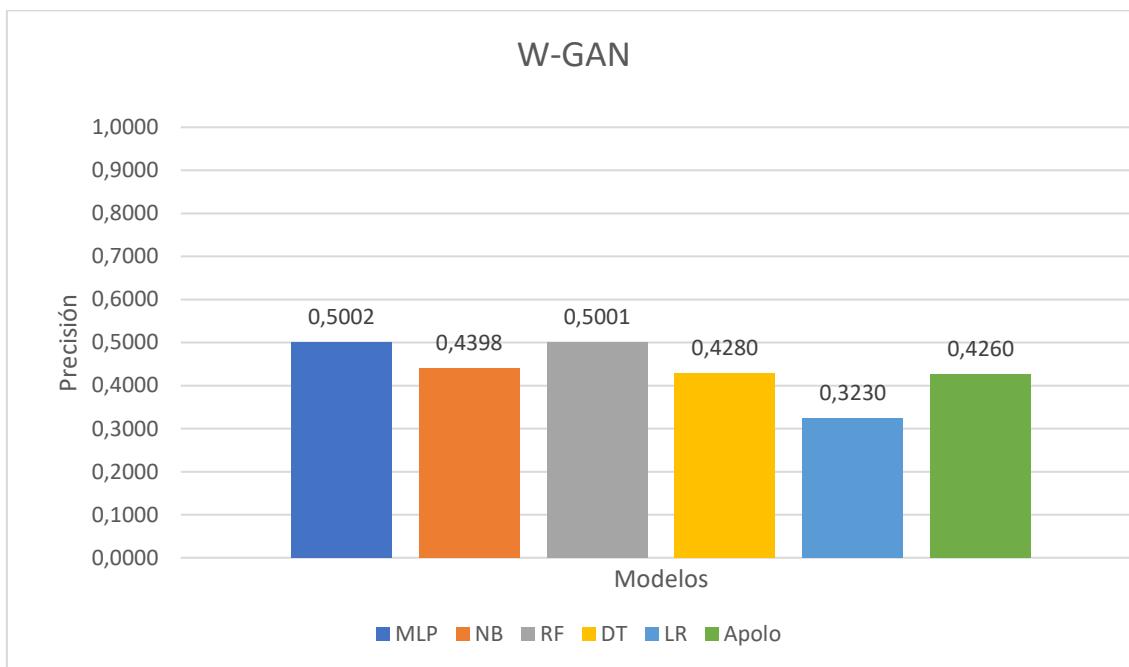


Ilustración 104. Segundo escenario: Precisión de los modelos y de Apolo. Utilizando el ataque W-GAN.

9.4.2.3.4. Conclusión

La mejor precisión y las tasas de detección de **Apolo** en los ataques de *AML* pueden atribuirse a la inclusión de un componente de incertidumbre en su proceso de selección de modelos (*MAB*). Esto hace que sea difícil entrenar un modelo basándose únicamente en sus respuestas.

Los resultados experimentales revelan que nuestra solución muestra una mayor robustez en comparación con los otros clasificadores utilizados individualmente. Sin embargo, también observamos que, aunque nuestra solución reduce efectivamente la efectividad de los ataques, no los anula completamente. Esto significa que aún hay margen para mejorar en términos de aumentar la robustez de la solución para fortalecer aún más su resistencia contra este tipo de ataques.

En particular, si generáramos los ataques con más tiempo, como por ejemplo aumentando el número de iteraciones o épocas, es probable que la efectividad de estos ataques contra nuestra solución aumentaría.

9.4.3. Tercer Escenario

En este apartado se va a definir y dar los resultados correspondientes al tercer escenario descrito.

9.4.3.1. Definición del Escenario

En el tercer escenario, hemos desarrollado un nuevo dataset generado con los datos provenientes de nuestra arquitectura. Para ello, hemos desarrollado dos agentes que simulan el comportamiento de diferentes usuarios:

- **Usuario Benigno:** Este usuario simula el comportamiento de un uso corriente del sistema. Las peticiones generadas con este agente deberían de ser clasificadas como benignas. Para poder simular este flujo de datos benignos, hemos desarrollado un *script* en *Python* utilizando la biblioteca “Selenium” que navega e interacciona con la aplicación de ejemplo. Además, aleatoriamente genera peticiones erróneas que simulan posibles fallos humanos, por ejemplo, errores “404” o peticiones HTTP no permitidas.

- **Usuario Maligno:** Este usuario simula el comportamiento de un posible atacante del sistema. Para este escenario hemos generado ataques de *Denegación de Servicio (DDoS)* de tipo *Slowloris*. Las peticiones generadas por este agente deberían de ser clasificadas como malignas.

El objetivo de este escenario es demostrar que **Apolo** es capaz de adaptarse a diferentes tipos de arquitecturas y topologías de red, ya que, los anteriores escenarios se enfocaban en probar **Apolo** en datasets utilizados en la literatura, mientras en este escenario probamos **Apolo** en nuestra arquitectura.

9.4.3.2. Resultados Obtenidos

En este escenario, entrenamos **Apolo** en el conjunto de datos generado con nuestra arquitectura “Data_Our”. **Apolo** cuenta con todos los modelos explicados en el punto 3.1.3. “Clasificadores” del Capítulo 3. “Apolo”.

En la Tabla 57 se pueden observar los resultados obtenidos al probar el modelo entrenado con el conjunto de datos generado. En primer lugar, la precisión (*accuracy*) es de aproximadamente 0.9544, lo que indica que el modelo es bastante preciso al realizar predicciones. Este valor significa que el modelo predijo correctamente el 95.44% de los casos. Sin embargo, es importante señalar que la precisión por sí sola puede no ser un indicador confiable si los datos están desequilibrados. En nuestro escenario esta métrica no es la más fiable, ya que, el número de datos benignos es mucho mayor que el número de datos malignos, al igual que ocurre en las aplicaciones web reales y en algunos de los datasets utilizados en la literatura.

Por otro lado, el *recall* o tasa de detención es de aproximadamente 0.8783, esto indica que el modelo fue capaz de identificar correctamente el 87.83% de los casos positivos reales. El *F1 Score*, es de 0.8834, lo que sugiere que **Apolo** mantiene un buen equilibrio entre precisión y tasa de detención.

Por último, el *ROC AUC Score* es de 0.8782, esto indica que **Apolo** tiene una alta capacidad para distinguir entre clases positivas y negativas. En general, estos resultados sugieren que **Apolo** ha obtenido buenos resultados y ha rendido bien en nuestra arquitectura.

Data_Our	Métricas	Modelo
		Apolo
	Precisión	0,9544
	Tasa de Detención	0,8783
	F1	0,8834
	ROC	0,8782

Tabla 57. Tercer Escenario: Resultados.

Capítulo 10. Manuales del Sistema

Este capítulo ofrecerá los manuales de instalación, ejecución, usuario y programador del sistema con el objetivo de describir en profundidad los pasos necesarios para utilizar el sistema o para ampliarlo.

Tanto el manual de instalación como el de ejecución están orientados a un equipo con sistema operativo *Linux*, puesto que es el que hemos utilizado para desarrollar el sistema, pero en el manual de Instalación y ejecución está recogido como instalar y ejecutar **Apolo**, en los sistemas operativos más utilizados (Windows 10 y macOS).



10.1. Manual de Instalación y Ejecución

Este apartado contendrá el manual de instalación y ejecución del sistema, estos describirán todo lo necesario para instalar y ejecutar el sistema, asegurando un correcto funcionamiento del mismo.

Para el correcto funcionamiento de Apolo es necesario tener bastante espacio en disco, puesto que los contenedores Docker utilizan una gran cantidad de almacenamiento en disco, recomendamos tener alrededor de 50 GB disponibles.

También, es obligatorio para este manual tener descargado y descomprimido el código de **Apolo** (está adjuntado al documento).

10.1.1. Instalación

Este apartado proporciona instrucciones paso a paso para instalar *Docker* y *Docker Compose*, que son necesarios para la instalación de **Apolo**.

10.1.1.1. Instalación de Docker

En esta sección explicaremos como instalar *Docker* en los sistemas operativos más utilizados.

10.1.1.1.1. Linux

Esta instalación está orientada a los usuarios de *Linux/Ubuntu*:

1. Actualizar paquetes del sistema: Primero, es necesario actualizar los paquetes del sistema. Abra una terminal y ejecute el siguiente comando:

```
sudo apt-get update
```

2. Instalar Docker: A continuación, instale *Docker* con el siguiente comando:

```
sudo apt-get install docker.io
```

3. Iniciar y habilitar *Docker*: Una vez instalado, inicie el servicio *Docker* y habilítelo para que se ejecute al arrancar:

```
sudo systemctl start docker  
sudo systemctl enable docker
```

10.1.1.2. MacOS

Esta instalación está orientada a los usuarios de *macOS*:

1. Descargar *Docker Desktop*: Vaya a la página oficial de *Docker*⁶⁵ y descargue *Docker Desktop* para *Mac*.
2. Instalar *Docker Desktop*: Abra el archivo descargado y siga las instrucciones en pantalla para instalar *Docker Desktop*.

10.1.1.3. Windows 10 Pro/Enterprise

Esta instalación está orientada a los usuarios de *Windows 10 Pro/Enterprise*:

1. Descargar *Docker Desktop*: Vaya a la página oficial de *Docker*⁶⁵ y descargue *Docker Desktop* para *Windows*.
2. Instalar *Docker Desktop*: Abra el archivo descargado y siga las instrucciones en pantalla para instalar *Docker Desktop*. Es posible que necesite habilitar la virtualización en la configuración de la *BIOS* de su sistema.

10.1.1.2. Instalación de Docker Compose

En esta sección explicaremos como instalar *Docker Compose* en los sistemas operativos más utilizados.

10.1.1.2.1. Linux / Ubuntu / macOS

Esta instalación está orientada a los usuarios de *Linux/Ubuntu/macOS*:

1. Descargar *Docker Compose*: Ejecute el siguiente comando en su terminal para descargar la versión estable actual de *Docker Compose*:

```
sudo curl -L "https://github.com/docker/compose/releases/download/1.27.4/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

⁶⁵ <https://www.docker.com/products/docker-desktop>

2. Aplicar permisos ejecutables: A continuación, aplique permisos ejecutables al archivo binario:

```
sudo chmod +x /usr/local/bin/docker-compose
```

10.1.1.2.2. Windows 10 Pro/Enterprise

Esta instalación está orientada a los usuarios de *Windows 10 Pro/Enterprise*.

La instalación de *Docker Desktop* ya incluye *Docker Compose*, por lo que no es necesario instalarlo por separado.

10.1.1.3. Verificación

Una vez que Docker y Docker Compose estén correctamente instalados en su sistema, puede verificar su instalación ejecutando, en su terminal, los comandos:

```
docker -version  
docker-compose --version
```

Si ambas herramientas están correctamente instaladas, estos comandos deberían mostrar las versiones instaladas de Docker y Docker Compose, respectivamente.

10.1.2. Ejecución del Sistema

Descargue el código fuente de la aplicación y abra una terminal para navegar hasta el directorio donde se encuentra el archivo “*docker-compose.yml*”, el cual se encuentra en el directorio *./Apolo/environment*.

Para ejecutar todo el sistema basta ejecutar en la terminal abierta el siguiente comando:

```
docker compose up
```

Alternativamente, también se puede añadir las opciones “*-build*”, la cual sirve para que *Docker Compose* construya las imágenes antes de iniciar los contenedores; y la opción “*-d*”, que es la abreviatura de la opción “*--detach*” (se

pueden usar indistintamente), esta opción para que se ejecuten los contenedores en segundo plano, permitiendo así seguir interactuando con la terminal. Por lo que nos quedaría un comando:

```
docker compose up -build -d
```

Estos comandos iniciarán el despliegue de la aplicación utilizando la configuración definida en el archivo “*docker-compose.yml*”. Docker-Compose se encargará de crear los contenedores necesarios, descargar las imágenes requeridas y establecer las conexiones entre los servicios.

Después de ejecutar el comando, puede verificar que todos los contenedores se hallan desplegado correctamente. Para ello ejecute, en la misma terminal, el siguiente comando:

```
docker ps
```

Esto mostrará una lista de todos los contenedores en ejecución en su sistema. Verifique que todos los contenedores relacionados con **Apolo** estén presentes y en estado “running”. Asegúrese de que no haya contenedores con errores o que estén detenidos.

Alternativamente, si ha instalado *Docker Desktop*, puede abrir dicha aplicación y verificar que todos los contenedores están de color verde y en estado “running”.

10.1.3. Detención del Sistema

Para detener todo el sistema basta con abrir una terminal en el directorio del proyecto *./Apolo/environment* y ejecutar el comando:

```
docker compose down
```

Este comando se encargará de finalizar y limpiar el entorno de la aplicación previamente desplegada. También, detendrá todos los contenedores que están corriendo, eliminará los contenedores junto con las redes definidas en el archivo “*docker-compose.yml*”. Este proceso garantiza que los recursos del sistema se liberan y se mantiene un entorno limpio para futuros despliegos.

Después de ejecutar el comando, puede verificar que todos los contenedores se hayan cerrado correctamente. Para ello ejecute, en la misma terminal, el siguiente comando:

```
docker ps
```

Esto mostrará una lista de todos los contenedores en ejecución en su sistema. Verifique que todos los contenedores relacionados **Apolo** no estén presentes. Asegúrese de que no haya contenedores con errores o que estén en estado “running”.

Alternativamente, si ha instalado *Docker Desktop*, puede abrir dicha aplicación y verificar que todos los contenedores no están levantados y en funcionamiento.

10.2. Manual de Usuario

En esta sección describiremos como seria la función del administrador de la aplicación. Empezaremos con el monitoreo en Kibana y seguiremos con el de InfluxDB.

El usuario utilizado para este ejemplo es: "SergioArroni" y la contraseña "HijosDeSanPedro".

10.2.1. Kibana

A continuación, describiremos paso a paso como se tiene que cargar el archivo de datos a *Kibana* y como se puede monitorizar el tráfico web mediante dashboards.

Tras iniciar sesión nos encontramos con la pantalla de inicio de *Kibana* (Ilustración 105)

Kibana cuenta con un menú desplegable a la izquierda, con múltiples funciones que podemos realizar con *Kibana*, nosotros nos centraremos en la opción de "Dashboards", por lo que clicamos en ella.

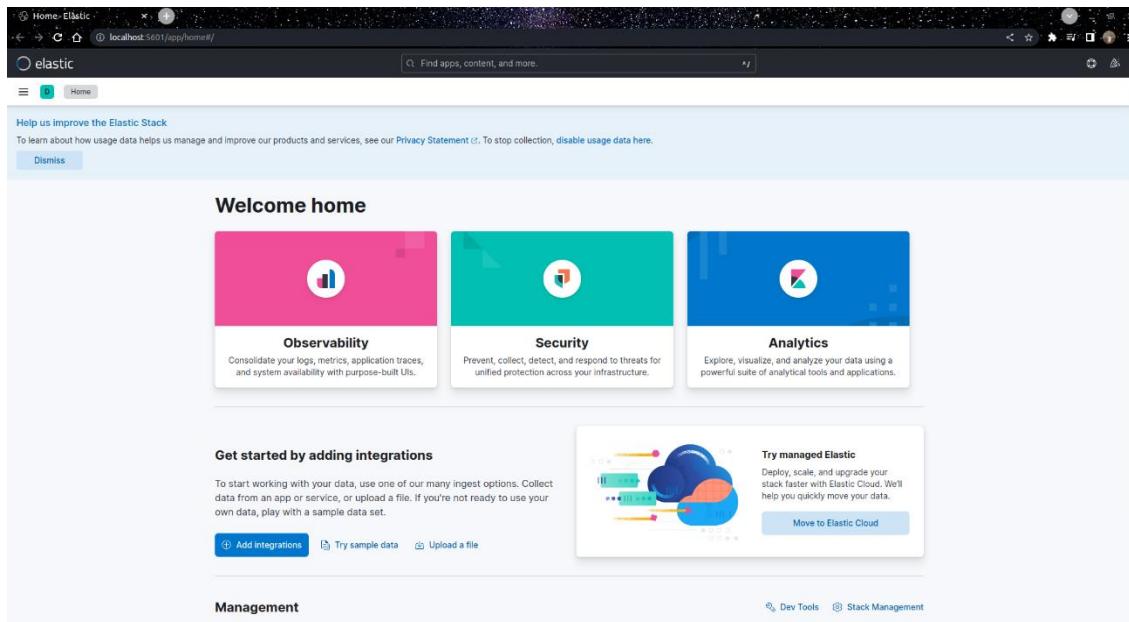


Ilustración 105. Manual de Usuario Kibana: Inicio.

Apolo: Sistema inteligente de monitorización y detección de ataques basados en inteligencia artificial

Una vez clicamos en “Dashboard” nos sale el mensaje de si queremos crear una nueva visualización de datos (Ilustración 106), clicamos en el botón.

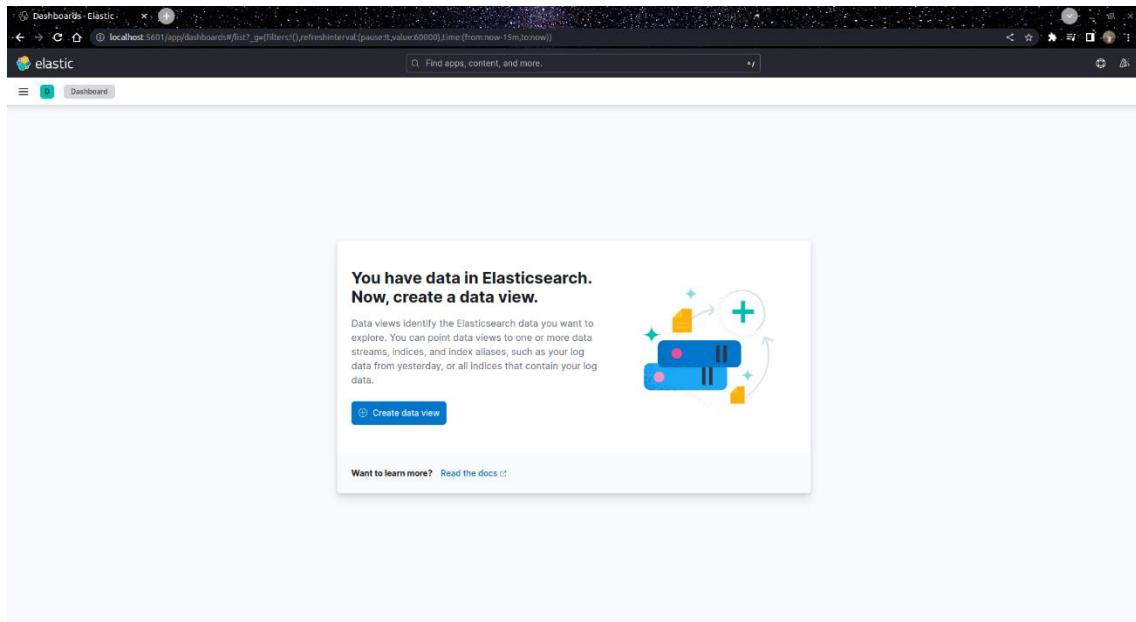


Ilustración 106. Manual de Usuario Kibana: Dashboards.

En esta pestaña (Ilustración 107) podemos darle nombre a la visualización de datos, en nuestro caso “demo” y seleccionamos nuestros datos “weblog-2023.06.09”.

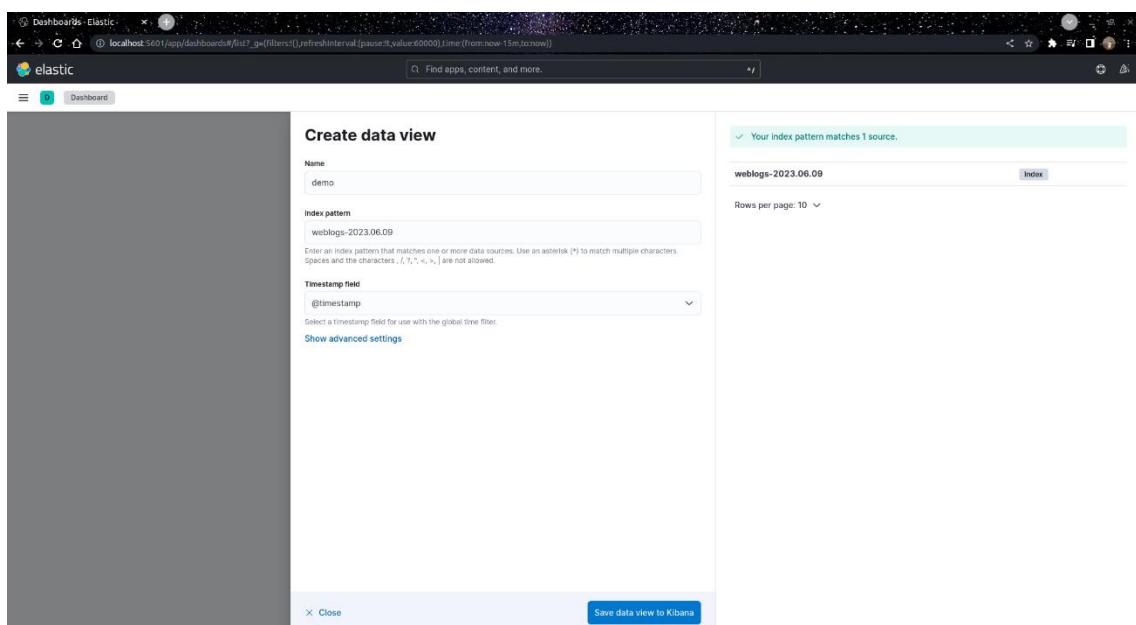


Ilustración 107. Manual de Usuario Kibana: Visualización de Datos.

Una vez creamos la visualización de datos, podemos crear diferentes gráficas, para ello únicamente tenemos que arrastrar el atributo que queramos monitorizar y *Kibana* nos recomendara la tabla a usar, pero podemos seleccionar entre las múltiples opciones que contiene *Kibana*, en la imagen (Ilustración 108) se puede observar un ejemplo con la variable “Flow Duration”, al derecha de la imagen, se puede observar que se puede cambiar como visualizar el dato, en el ejemplo está posicionado en “media”, pero se podría poner en “max” o “min”.

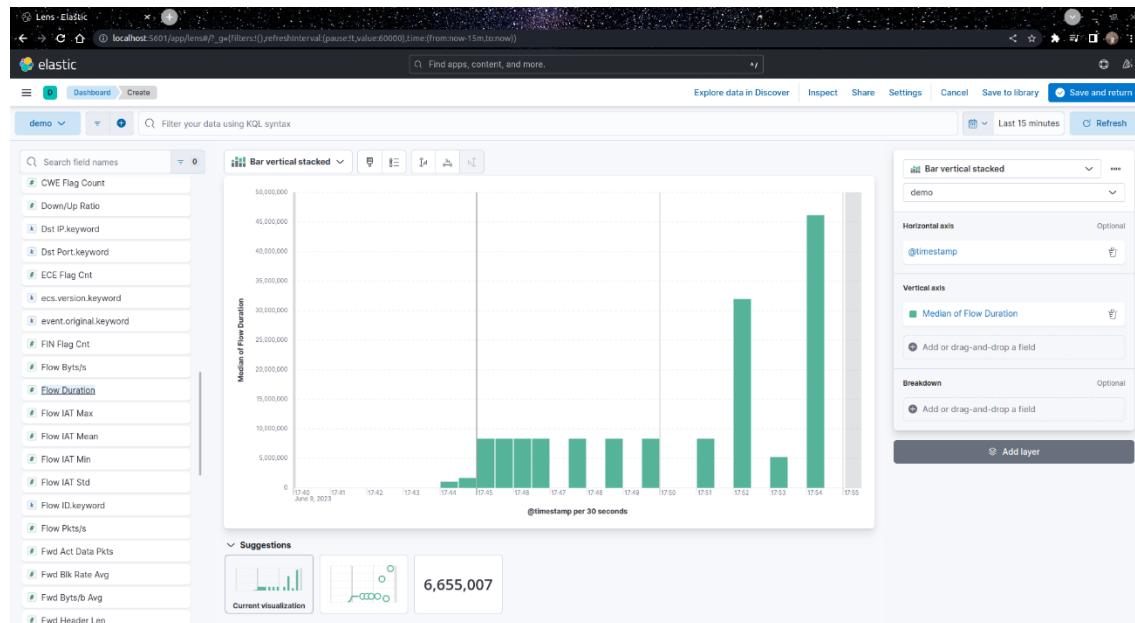


Ilustración 108. Manual de Usuario Kibana: Creación de Gráficas.

Una vez le damos a guardar (boton de arriba a la derecha “Save and return”), nos lleva a un panel en el que se ven todas las graficas y tablas creadas hasta el momento (Ilustración 109).

Apolo: Sistema inteligente de monitorización y detección de ataques basados en inteligencia artificial

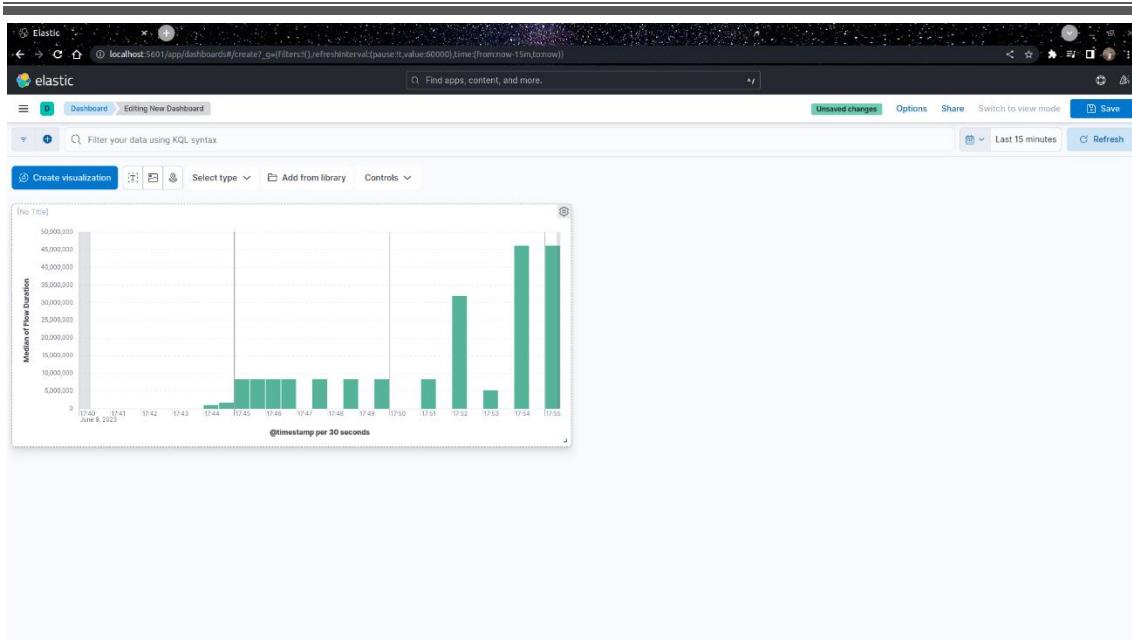


Ilustración 109. Manual de Usuario Kibana: Panel Dashboards.

Como ejemplo, vamos a seleccionar diferentes atributos para mostrar todas las gráficas y opciones que ofrece *Kibana*, en este caso (Ilustración 110), hemos seleccionado, la variable del protocolo, el grafico que nos recomienda *Kibana* es un gráfico en forma de “donut”, en que se observan las diferentes proporciones de los protocolos utilizados.

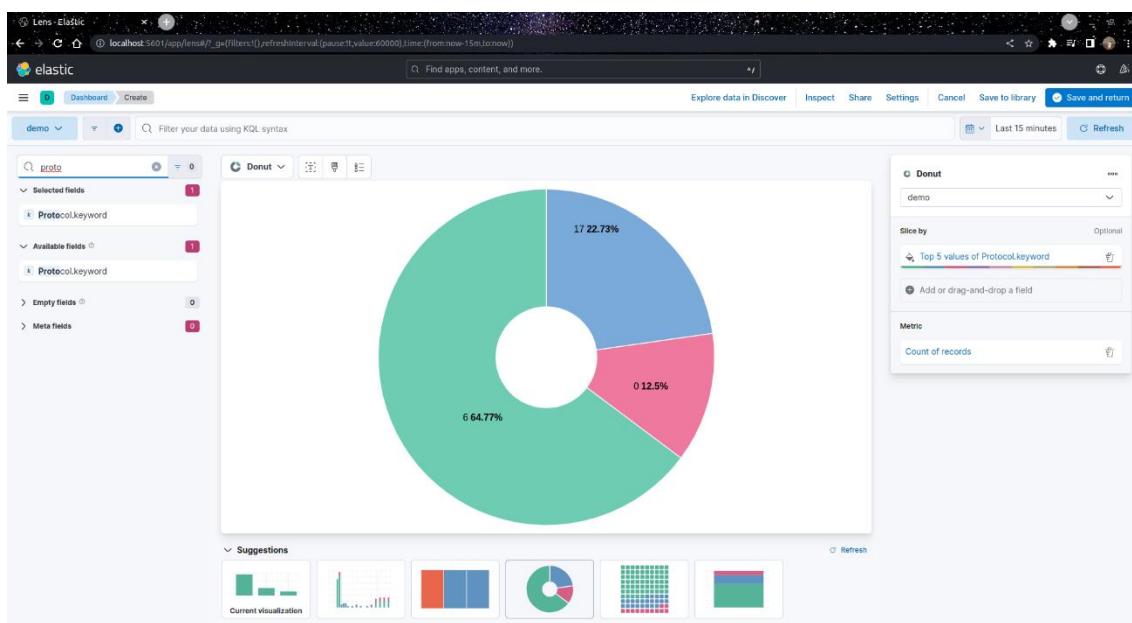


Ilustración 110. Manual de Usuario Kibana: Gráfica Donut.

Otra opción que nos da *Kibana* es la de combinar atributos en una sola gráfica, por ejemplo, en la Ilustración 111 se puede observar un gráfico de barras de la variable “Tot Fwd Pkts” y en la Ilustración 112 se puede observar la misma gráfica, pero con otras barras, dichas barras (en color verde), se corresponden con la variable “Pkt Size Avg”. Las variables se pueden hacer visibles o invisibles ya que conforman diferentes capas del gráfico, esto se puede realizar en la esquina superior derecha del gráfico.

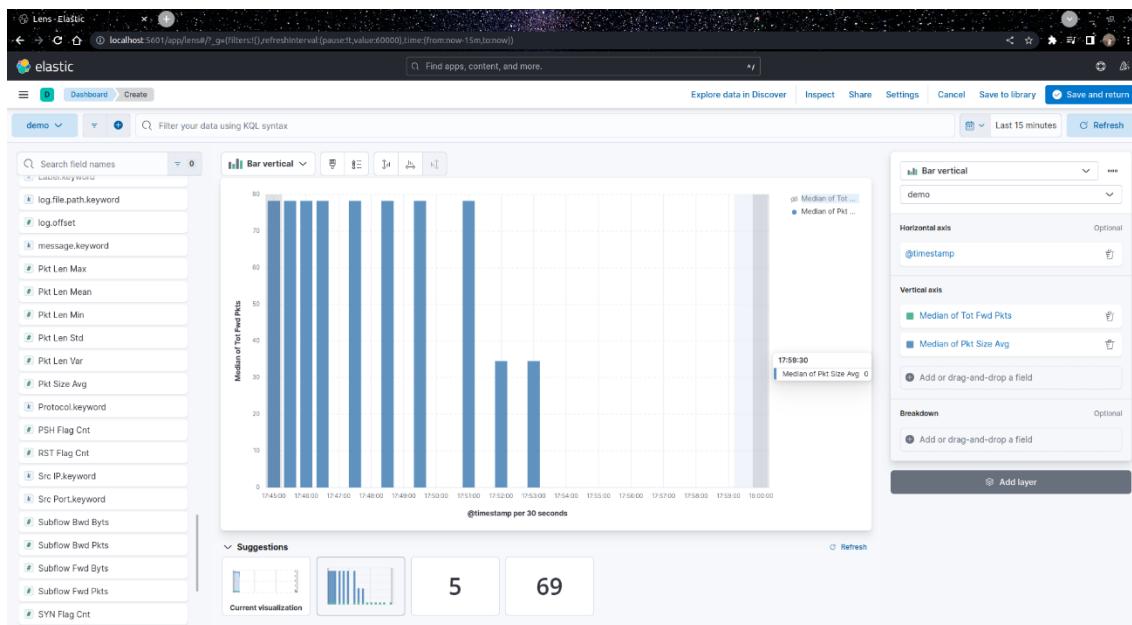


Ilustración 111. Manual de Usuario Kibana: Múltiples Atributos en Gráficas (Parte I).

Apolo: Sistema inteligente de monitorización y detección de ataques basados en inteligencia artificial

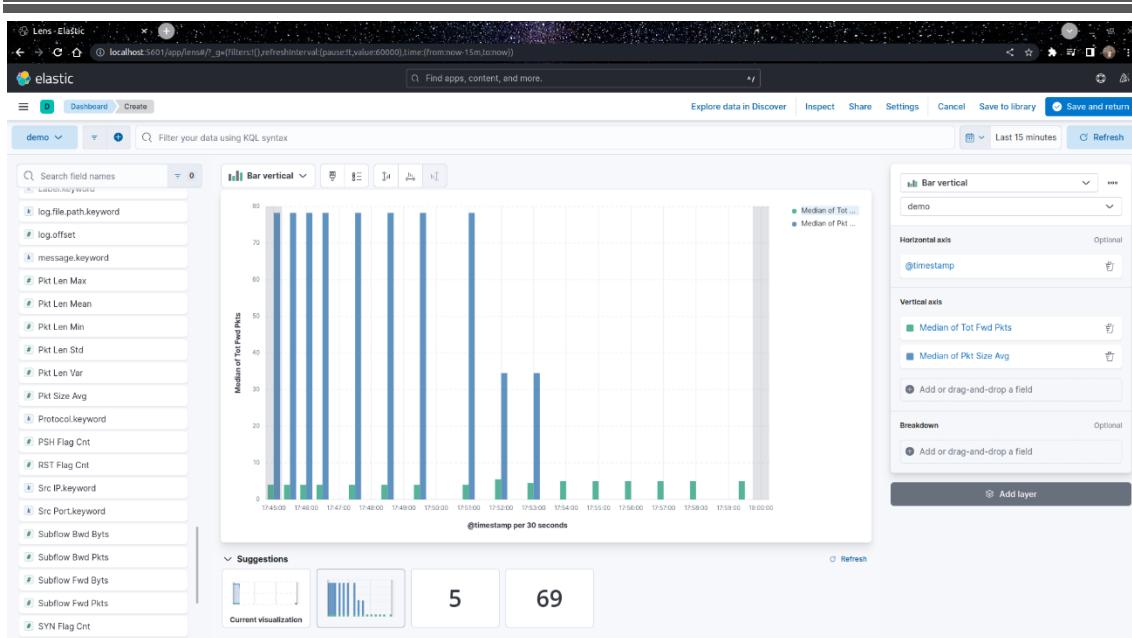


Ilustración 112. Manual de Usuario Kibana: Múltiples Atributos en Gráficas (Parte II).

En la Ilustración 113 se puede observar cómo se muestran las 3 graficas que acabamos de realizar. En la Ilustración 113 se puede observar que por ejemplo el color verde de la gráfica del donut tiene un valor de 47,69% que son un total de 6 peticiones de ese tipo, pero en la Ilustración 114 se puede observar que el mismo tipo de peticiones tiene un 49,21% esto se debe a que el panel se va actualizando cuando se le envían nuevas peticiones o cuando se le da al botón de "Refresh", en la esquina superior derecha.

Apolo: Sistema inteligente de monitorización y detección de ataques basados en inteligencia artificial

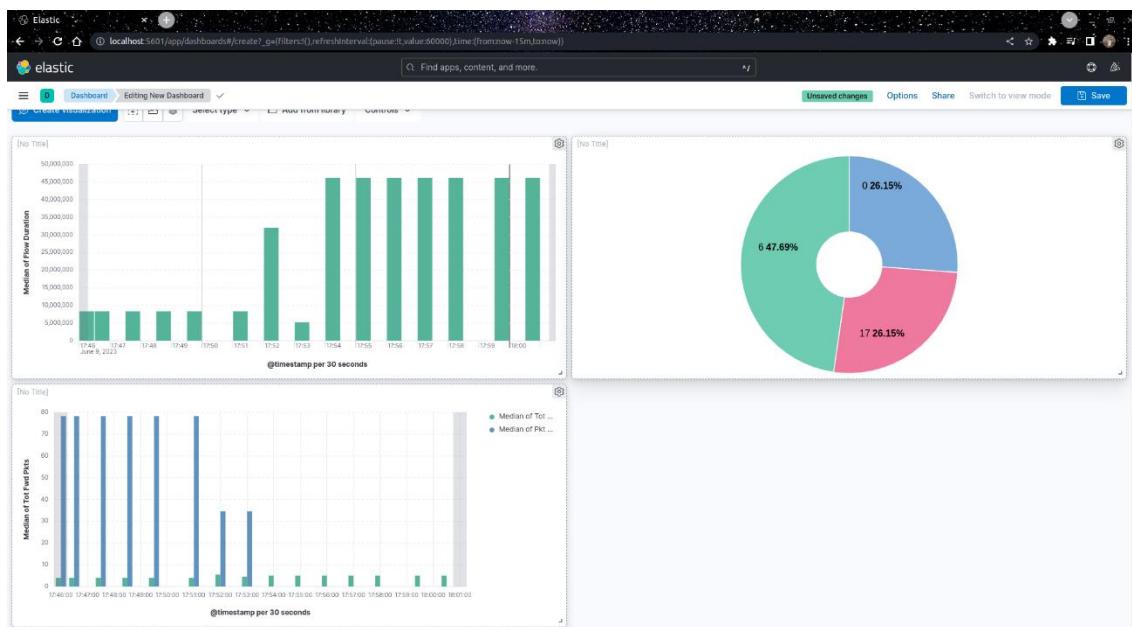


Ilustración 113. Manual de Usuario Kibana: Panel con Gráficas.

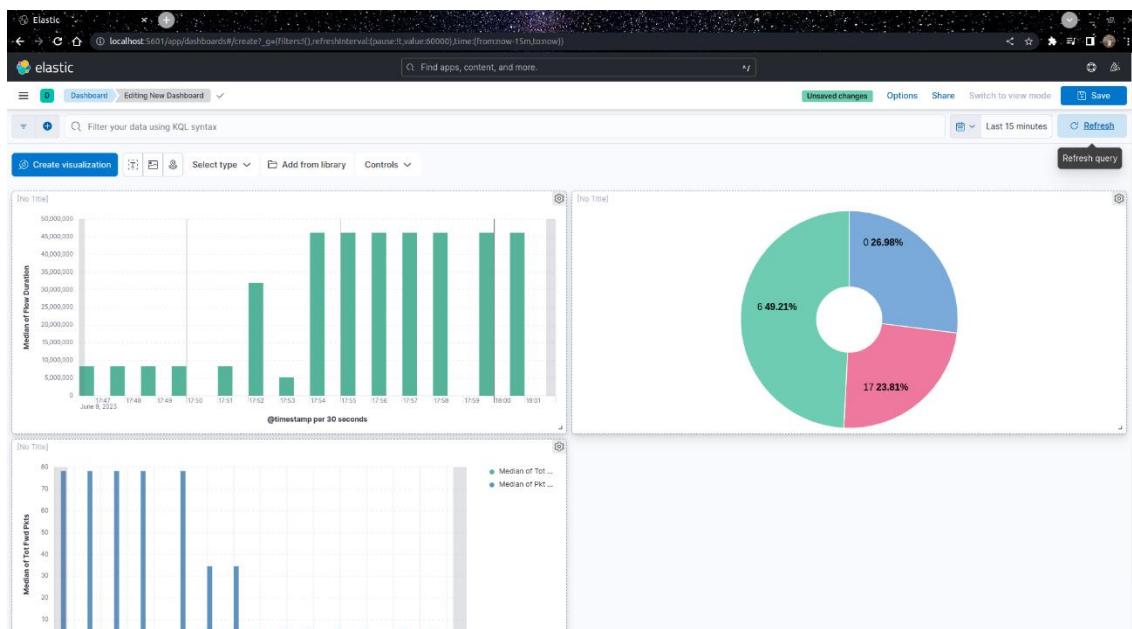


Ilustración 114. Manual de Usuario Kibana: Panel con Gráficas, Actualización.

Una vez estemos satisfechos con la configuración de graficas que hemos realizado, podemos guardar el panel y darle un nombre en nuestro caso “Demo Dashboard”, como se observa en la Ilustración 115.

Apolo: Sistema inteligente de monitorización y detección de ataques basados en inteligencia artificial

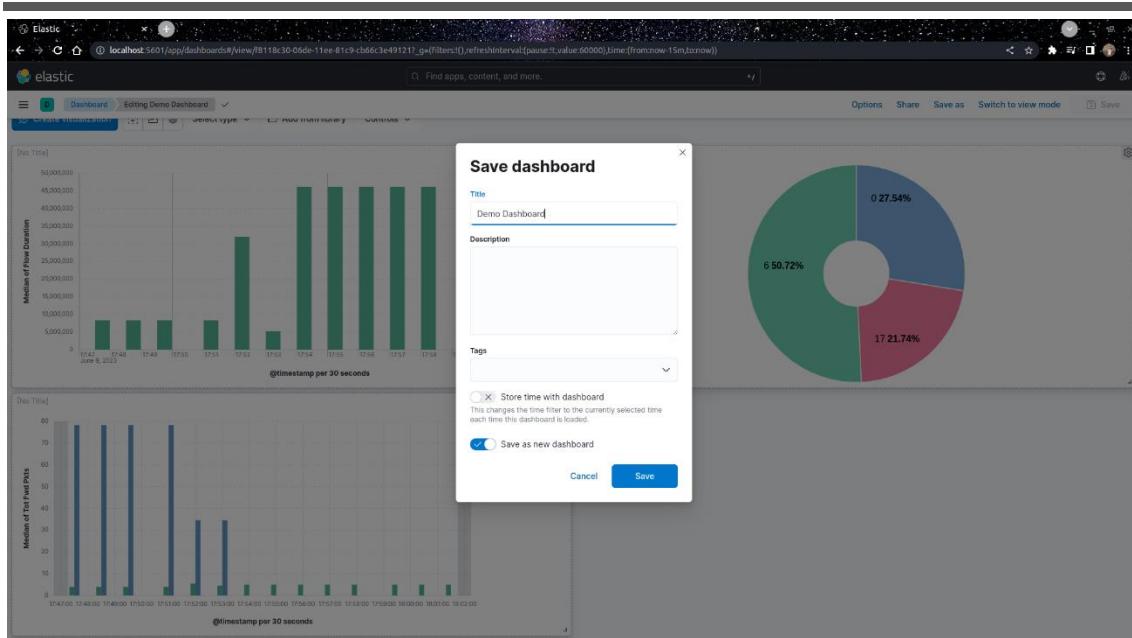


Ilustración 115. Manual de Usuario Kibana: Guardado Panel.

Una vez se ha guardado el panel este es accesible desde la pantalla de inicio de “Dashboards”, como se observa en la Ilustración 116.

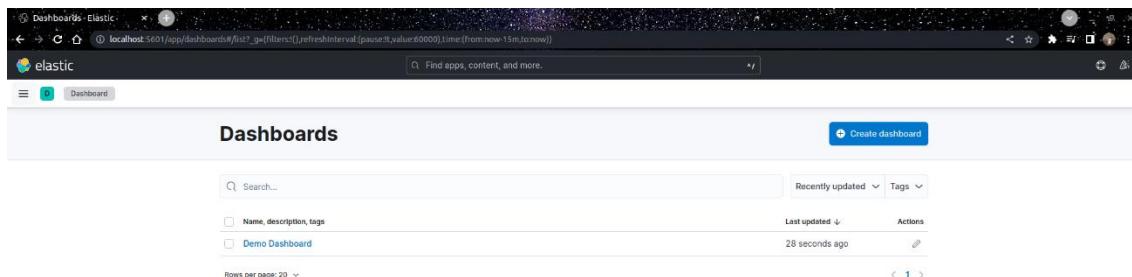


Ilustración 116. Manual de Usuario Kibana: Paneles Guardados.

10.2.2. InfluxDB

A continuación, describiremos paso a paso como se tiene que cargar el archivo de datos a *InfluxDB* y como se puede monitorizar las clasificaciones mediante dashboards.

En la Ilustración 117 se puede observar el inicio de sesión en *InfluxDB*, las credenciales son “SergioArroni” para el *username* y la *password* es “HijosDeSanPedro”.

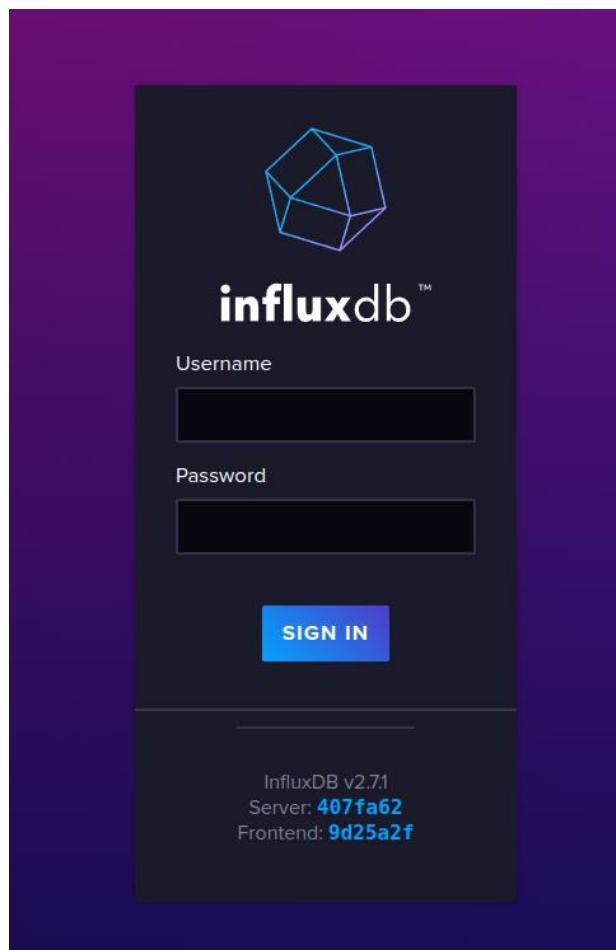


Ilustración 117. Manual de Usuario InfluxDB: Inicia.

En la Ilustración 118 se puede ver el panel de inicio de *InfluxDB*, en el panel lateral izquierdo clicamos en la segunda opción “Load Data”.

Apolo: Sistema inteligente de monitorización y detección de ataques basados en inteligencia artificial

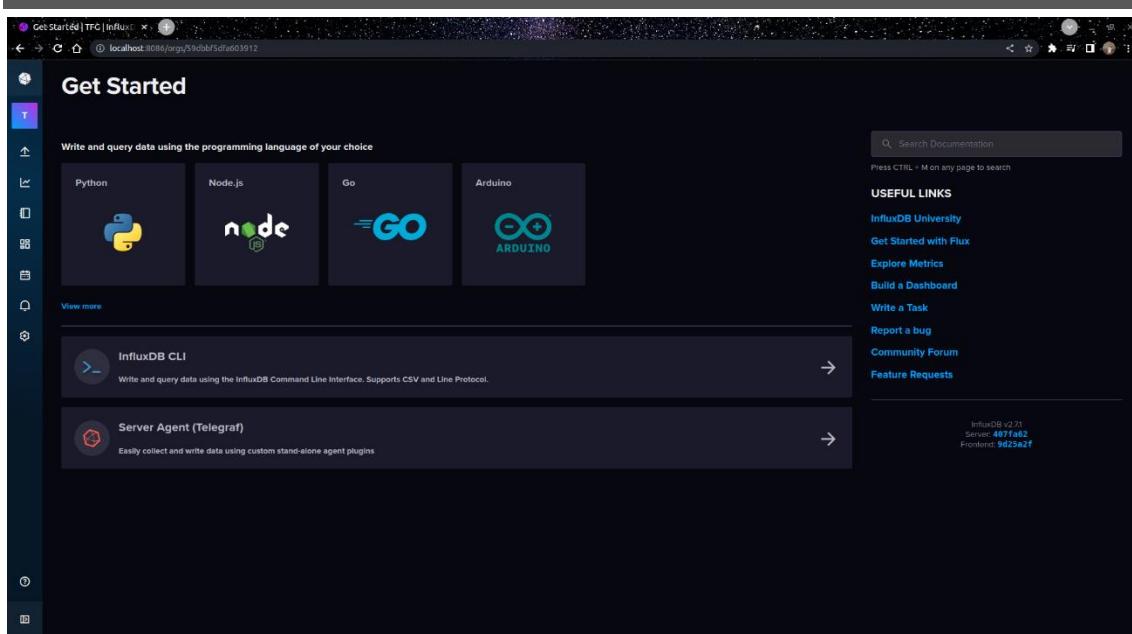


Ilustración 118. Manual de Usuario InfluxDB: Panel Principal.

En esta pestaña (Ilustración 119), podemos observar que el bucket "requests_scores" está correctamente creado.

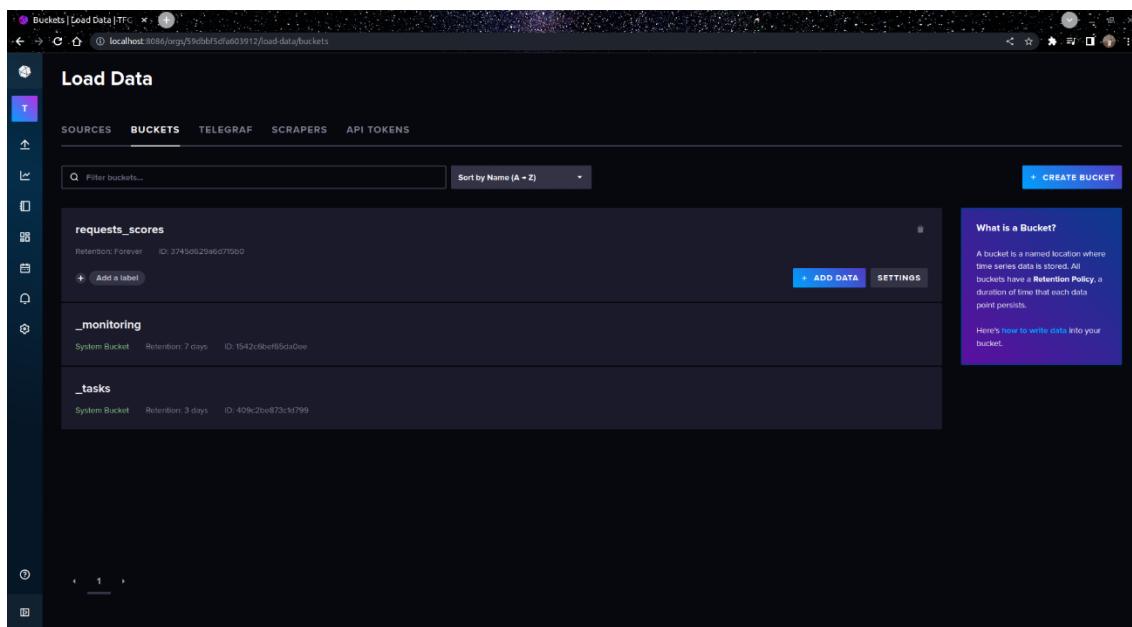


Ilustración 119. Manual de Usuario InfluxDB: Buckets.

Una vez comprobado que el bucket está correctamente creado vamos a la tercera opción del panel lateral "Data Explorer". En este panel (Ilustración 120)

Apolo: Sistema inteligente de monitorización y detección de ataques basados en inteligencia artificial

vemos los datos en el archivo “weblogs”, que son los datos que se van actualizando durante el uso del sistema.

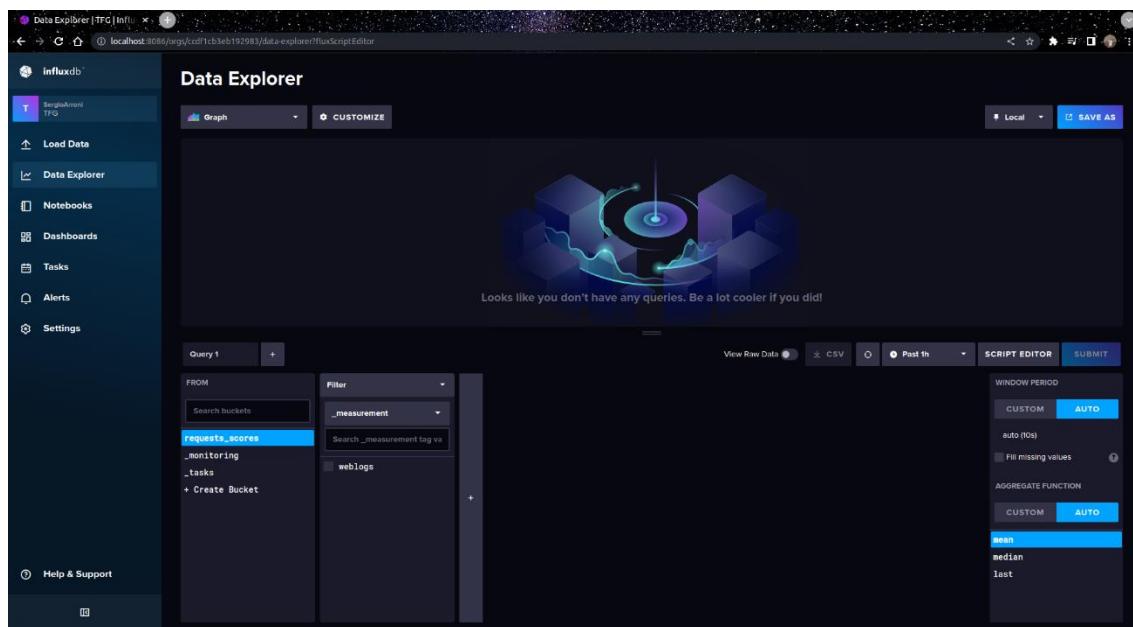


Ilustración 120. Manual de Usuario InfluxDB: Data Explorer.

En la Ilustración 121 se pueden observar los datos que van entrando en el sistema, estos datos se van actualizando en tiempo real. El eje X significa el timestamp y en el eje Y el value de cada petición.

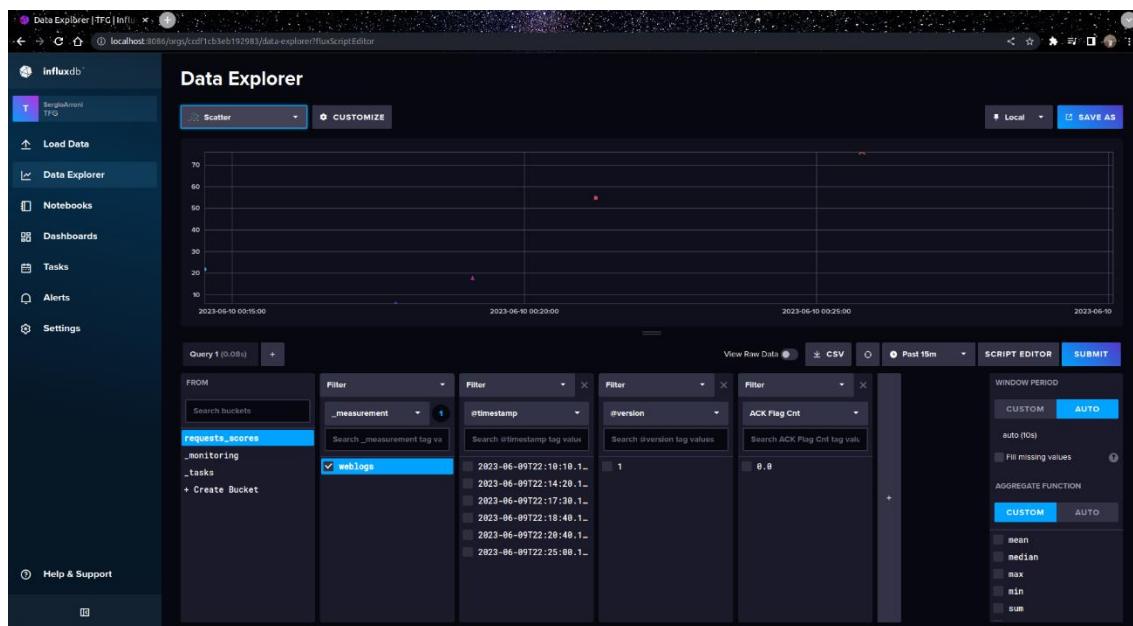


Ilustración 121. Manual de Usuario InfluxDB: Datos Peticiones.

Apolo: Sistema inteligente de monitorización y detección de ataques basados en inteligencia artificial

InfluxDB es muy personalizable, por lo que se puede cambiar la representación de los datos, en la Ilustración 122 se puede ver el mapa de calor de los valores de las peticiones. En la Ilustración 123 se puede ver que atributos alberga una petición que llega a *InfluxDB*. En la Ilustración 124 se puede observar los datos en formato tabla, esta implementación es de las más cómodas de utilizar. Todas estas formas de visualizar los datos se cambian en el desplegable de la esquina superior izquierda.

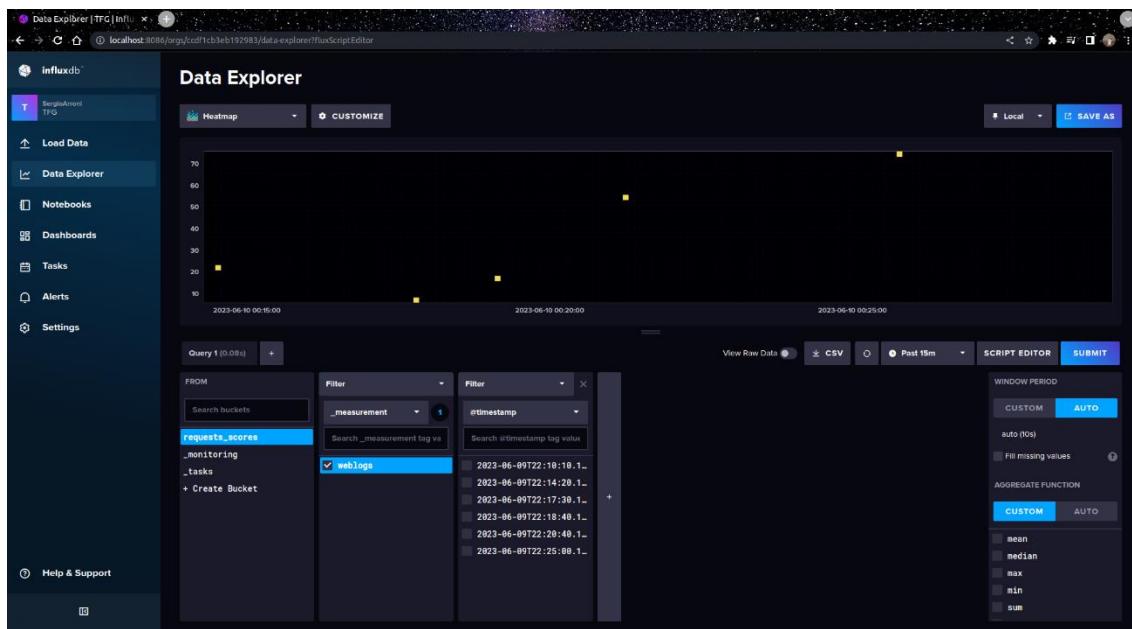


Ilustración 122. Manual de Usuario InfluxDB: Heatmap.

Apolo: Sistema inteligente de monitorización y detección de ataques basados en inteligencia artificial

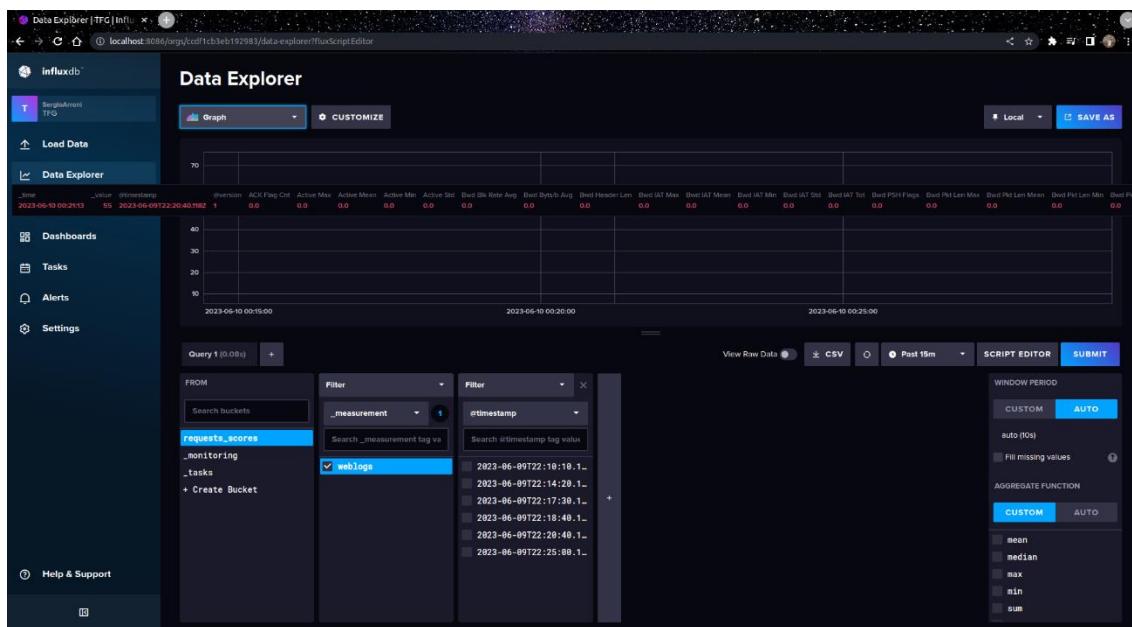


Ilustración 123. Manual de Usuario InfluxDB: Data.

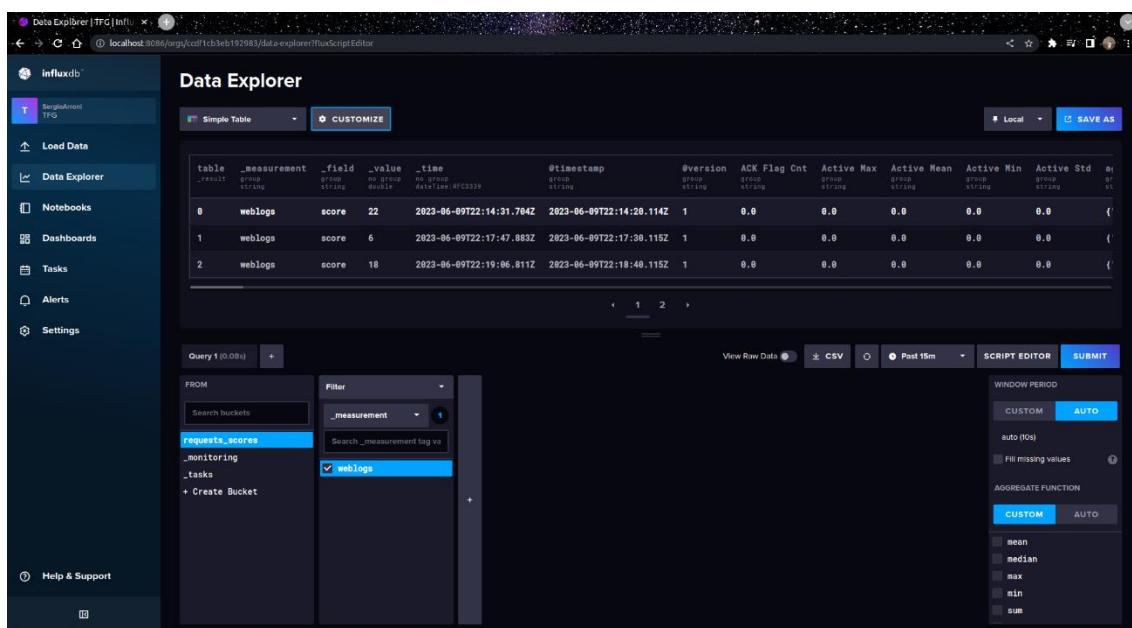


Ilustración 124. Manual de Usuario InfluxDB: Tabla de Datos.

También, cuenta con muchos filtros en la visualización de los datos, tal y como se puede observar en la Ilustración 125.

Apolo: Sistema inteligente de monitorización y detección de ataques basados en inteligencia artificial

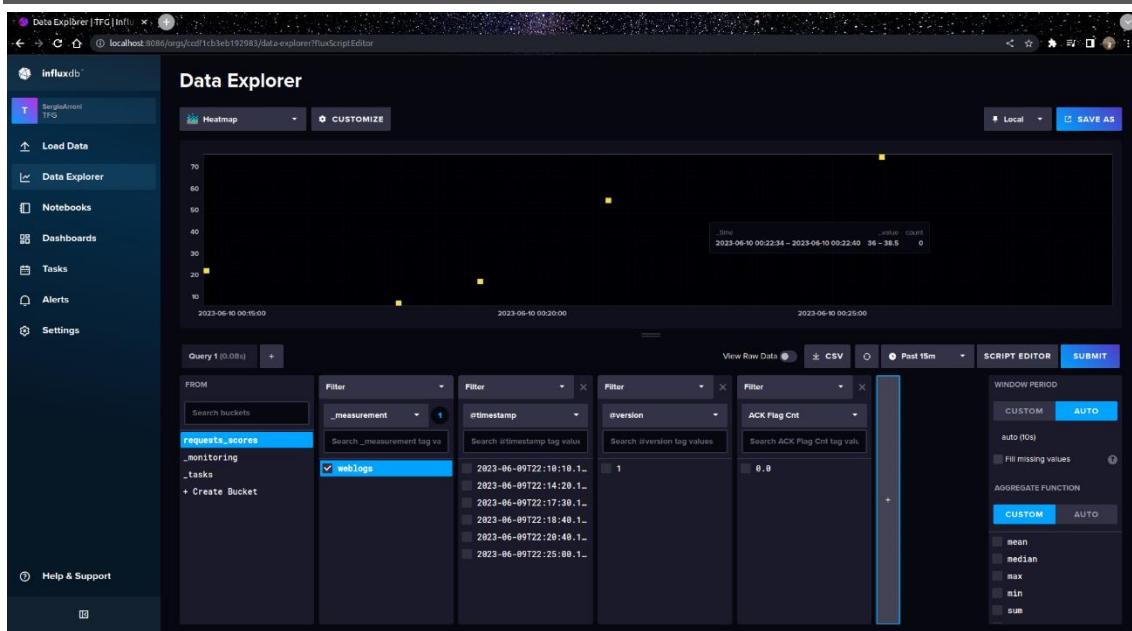


Ilustración 125. Manual de Usuario InfluxDB: Filtros.

Una vez estos conforme con la visualización de los datos, puedes guardar el dashboard creado, para ello daremos al botón “Save as” en la esquina superior derecha y le asignaremos un nombre siguiendo el ejemplo de las imágenes: Ilustración 126 y la Ilustración 127. Una vez se haya guardado el modelo se podrá ver y acceder a él desde el panel de dashboards de *InfluxDB*, tal y como se puede ver en la Ilustración 128.

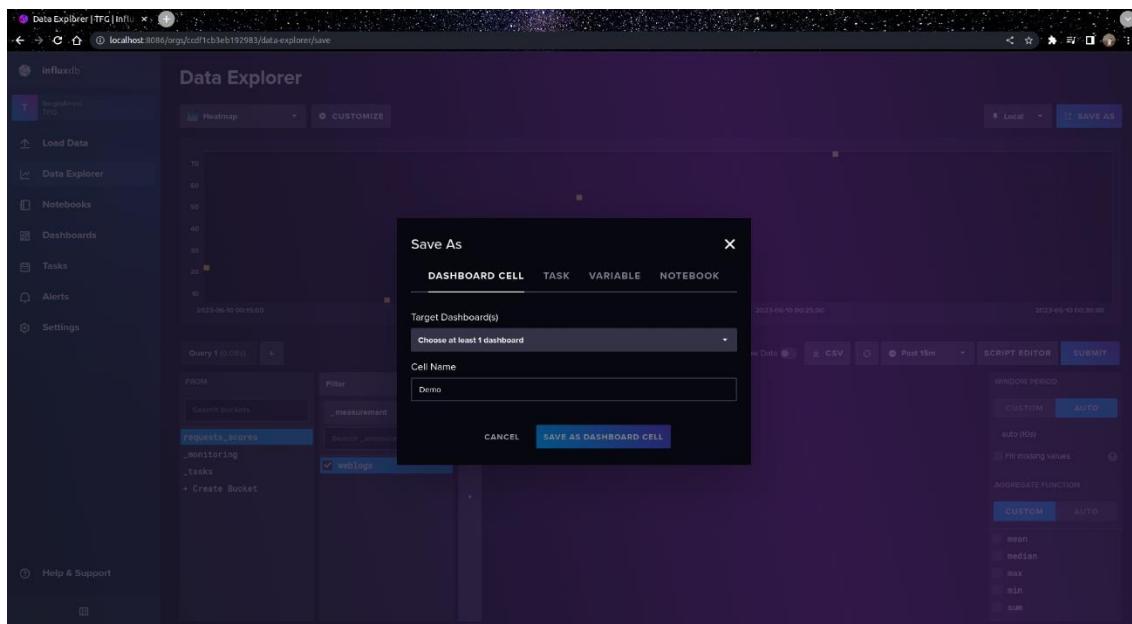


Ilustración 126. Manual de Usuario InfluxDB: Guardado (Parte I).

Apolo: Sistema inteligente de monitorización y detección de ataques basados en inteligencia artificial

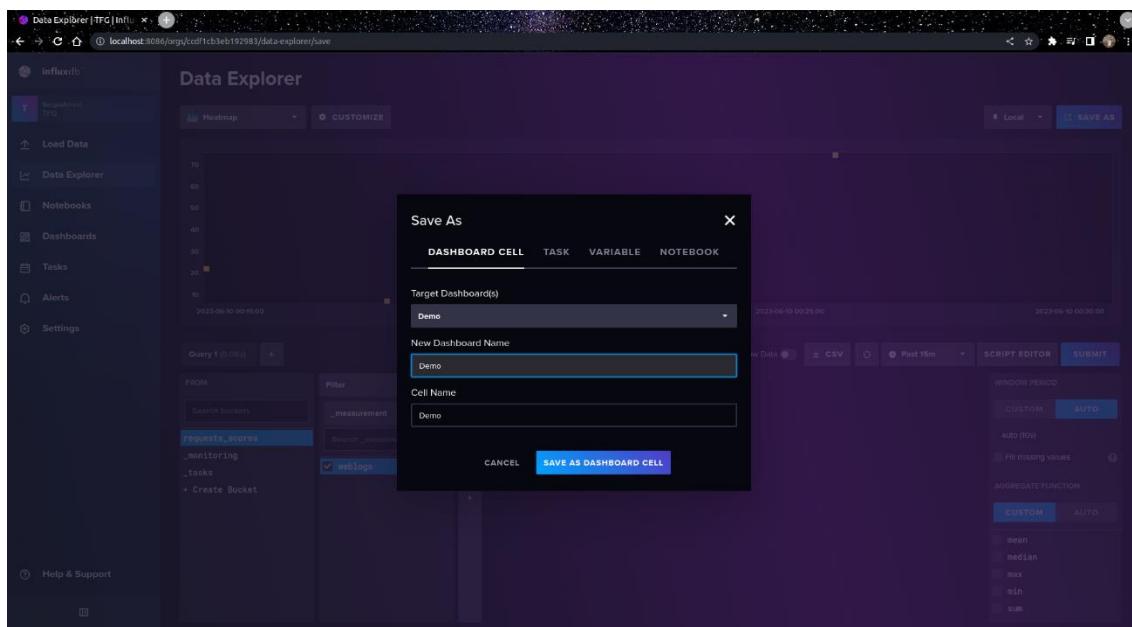


Ilustración 127. Manual de Usuario InfluxDB: Guardado (Parte II).

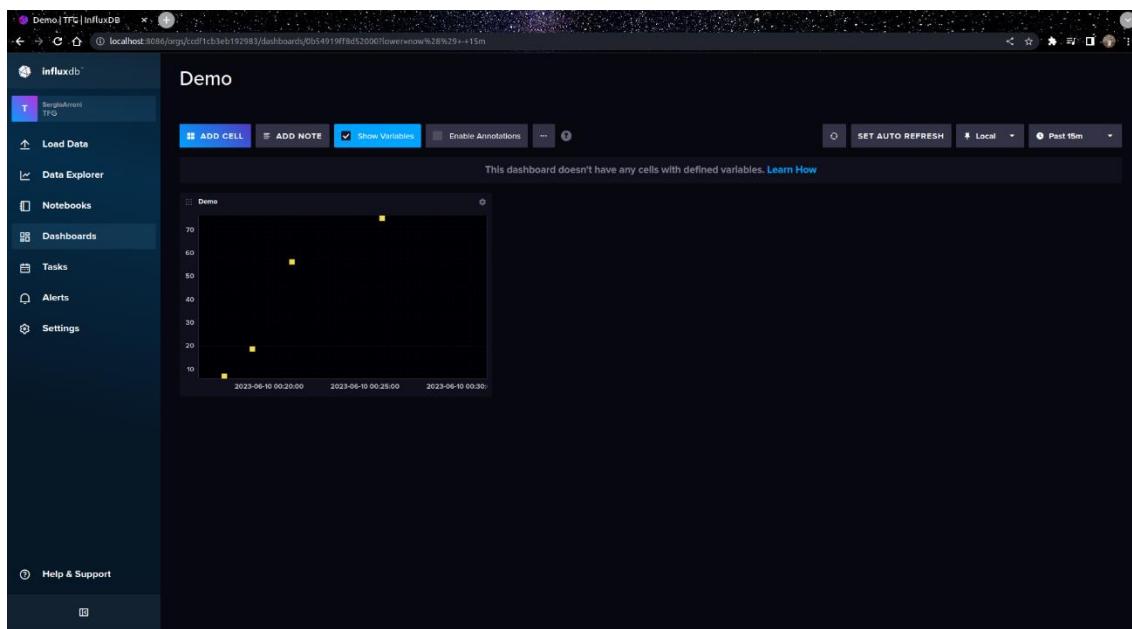


Ilustración 128. Manual de Usuario InfluxDB: Dashboards Guardados.

10.3. Manual del Programador

En este manual explicaremos como se añadiría un nuevo modelo de *ML* y como se añadiría un nuevo preprocesamiento de datos; a los ya existentes de **Apolo**. Estas dos funciones son muy útiles para mejorar y personalizar el modelo a cada arquitectura y caso en el que se implemente el sistema.

10.3.1.1. Añadir un nuevo modelo de aprendizaje automático

Para añadir un nuevo modelo de aprendizaje automático, se debe seguir una serie de pasos sencillos basados en el diseño actual de las clases:

1. Crear el nuevo fichero: Dentro de los paquetes “ids.apolo.layers.models.all_models”, se tiene que crear un nuevo fichero de *Python*, por ejemplo “new_model.py”.
2. Importaciones necesarias: Asegúrate de importar la clase “Model” de “ids.apolo.layers.models” y el modelo de aprendizaje automático específico que deseas implementar de “sklearn”¹¹.
3. Crear una nueva clase: Define una nueva clase que herede de la clase “Model”, por ejemplo “NewModel”.
4. Constructor: En el constructor de tu clase (el método “__init__”), asegúrate de llamar al constructor de la clase padre (“Model”) con los parámetros adecuados. Si tu modelo necesita parámetros específicos, puedes agregarlos aquí también.
5. Métodos: Dentro de esta clase, debes implementar el método “expecific_model” que retorna una instancia del modelo específico de *sklearn* que deseas usar.
6. Nombre del modelo: Implementa el método “__str__” para que retorne el nombre de tu clase. Esto es útil para la identificación del modelo.
7. Añadir el model al “__init__.py”: Una vez el nuevo modelo este creado y configurado correctamente, se tiene que realizar su importación en el fichero “__init__.py” de esa misma carpeta “all_models”. Se tiene que seguir el siguiente estilo:

```
from .new_model import NewModel
```

También, se debe de importar el nuevo modelo en el fichero “apolo_trainer” del paquete “ids.apolo.model_train” y una vez se haya importado añadirlo al parámetro “arms” del “__init__.py”, siguiendo el siguiente estilo:

```
NewModel(seed=self.seed,exe=False).expecific_model(),
```

Hay que tener en cuenta que cada modelo puede requerir diferentes parámetros y métodos de ajuste. Debes referirte a la documentación de *sklearn*¹¹ para tu modelo específico para entender qué parámetros necesita y cómo usarlos correctamente.

10.3.1.2. Añadir una nueva clase de preprocesamiento de datos

El preprocesamiento de datos es una etapa esencial en el *Machine Learning*. Hasta ahora, hemos visto cómo se pueden implementar diferentes modelos de *Machine Learning*, pero antes de alimentar, con datos, a estos modelos, a menudo necesitamos limpiar y transformar los datos. Aquí es donde la clase “ClearData” entra en juego.

La clase “ClearData” es una clase de preprocesamiento que proporciona un conjunto de métodos útiles para limpiar y preprocesar datos. Si deseas añadir una nueva clase de preprocesamiento, puedes seguir un proceso similar al de añadir un nuevo modelo:

1. Crear el nuevo fichero: Dentro de los paquetes “ids.apolo.preprocessing.datasets”, se tiene que crear un nuevo fichero de *Python*, por ejemplo “clear_data_new.py”.
2. Importaciones necesarias: Asegúrate de importar la clase “ClearData” de “apolo.preprocessing” y cualquier otro módulo que necesites para tu preprocesamiento.
3. Crear una nueva clase: Define una nueva clase que herede de la clase “ClearData”, por ejemplo “ClearDataNew”.
4. Constructor: En el constructor de tu clase (el método “__init__”), asegúrate de llamar al constructor de la clase padre (“ClearData”) con los parámetros adecuados.
5. Métodos: Dentro de esta clase, debes implementar el método “clear_data”, “save_data” y “load_data” que son los métodos de la clase padre que espera que sean diferentes para cada implementación.

6. Añadir el preprocesamiento al “`__init__.py`”: Una vez el nuevo modelo este creado y configurado correctamente, se tiene que realizar su importación en el fichero “`__init__.py`” de esa misma carpeta “datasets”. Se tiene que seguir el siguiente estilo:

```
from .clear_data_new import ClearDataNew
```

También, se debe de importar el nuevo modelo en el fichero “`data_selectotr`” del paquete “`utils`” y una vez se haya importado añadirlo a la enumeración “`datasets_types`”, siguiendo el siguiente estilo:

```
"Data_New": Data_New,
```

En general, las clases de preprocesamiento son muy específicas para la tarea en cuestión, por lo que la forma exacta en que implementes tu clase dependerá de tus necesidades específicas. Sin embargo, recuerda seguir las mejores prácticas y mantener tu código organizado y legible para facilitar su mantenimiento y comprensión en el futuro.

Capítulo 11. Conclusiones y Futuro Trabajo

En este capítulo vamos a dar las conclusiones que se han alcanzado tras el desarrollo del proyecto, también daremos el posible futuro trabajo que hemos valorado para el proyecto.



11.1. Conclusiones

En conclusión, en este proyecto se presenta **Apolo**, un nuevo y robusto sistema de defensa contra los ataques de *Adversarial Machine Learning (AML)* en *Sistemas de Detección de Intrusos (IDS)*.

Apolo utiliza varias estrategias para seleccionar, en tiempo real, un clasificador basado en *Machine Learning (ML)* para cada petición de tráfico red. Esta selección está basada en una *clusterización* de las peticiones, con el objetivo de agrupar peticiones de red similares, y el uso de un modelo *Multi-Armed Bandits (MAB)* con muestreo *Thompson*, encargado de la selección del clasificador *ML* que procesará la petición de tráfico red mediante aprendizaje por refuerzo.

Esta selección añade una capa de semi incertidumbre al comportamiento del *IDS*, ya que para una misma petición de tráfico red, no tiene por qué ser siempre el mismo clasificador *ML* el encargado de proporcionar la respuesta. De esta forma, se hace más difícil a los posibles atacantes replicar el comportamiento del *IDS* con el que poder generar tráfico adversario.

Para la correcta evaluación de **Apolo**, se ha realizado una experimentación utilizando varios conjuntos de datos, con estructuras y topologías de red diferentes.

Los resultados muestran que **Apolo** es un sistema de defensa eficaz contra los ataques generados con *AML* en el ámbito de los *IDS*. **Apolo** es capaz de detectar con éxito los ataques, sin comprometer el rendimiento del sistema en datos de tráfico de red y ataques normales (es decir, no generados con *AML*).

Sin embargo, **Apolo** no elimina completamente el riesgo de este tipo de ataques, únicamente lo mitiga, aumentando el tiempo y el esfuerzo requerido por los atacantes para generar tráfico adversario, haciendo que, en la mayoría de los casos, sea inviable la realización del ataque, tanto por los recursos como por el tiempo necesario.

De esta forma, **Apolo** puede ayudar a mejorar la seguridad de los sistemas y entornos donde las decisiones de un algoritmo basado en *Machine Learning* son críticas. Casos como los vehículos autónomos o sistemas médicos, son especialmente críticos y vulnerables frente a este tipo de ataques, por lo que sistemas de defensa como **Apolo** son necesarios.

11.2. Futuro trabajo

Con el objetivo de mejorar el rendimiento y la robustez de **Apolo**, tenemos previsto explorar el uso de otros modelos e implementaciones de *MAB*, como la optimización bayesiana o el *Deep Bayesian Bandits*. Además, también se plantea la exploración de otras técnicas de aprendizaje por refuerzo y comparar su rendimiento con nuestra solución basada en *MAB*.

También tenemos previsto explorar el uso de otros clasificadores y modelos *ML*, como los modelos de previsión de solicitudes (*forecasting*), que se pueden utilizar para predecir el número esperado de solicitudes en la siguiente ventana de tiempo y compararlo con el número real de solicitudes. Esta capa de previsión de solicitudes formaría una nueva estrategia de defensa complementaria a las ya existentes.

Otro enfoque que tenemos previsto es explorar el uso de datos generados mediante técnicas de *AML* en los entrenamientos de los clasificadores. Estos datos podrían servir para reentrenar los modelos y que estos estén más protegidos frente a ataques que utilicen técnicas de *AML*.

Por último, planteamos explorar el uso de otros conjuntos de datos para evaluar el rendimiento de **Apolo** en diferentes entornos de red. Entre estos nuevos conjuntos de datos, tenemos previsto el uso de **Apolo** en redes basadas en los Perímetros Definidos por Software (*SDP*) o en redes empresariales con un uso real.

Capítulo 12. Planificación y Presupuesto del Proyecto (Final)

En este capítulo se muestran la planificación y el presupuesto finales del proyecto. Se trata de evoluciones de la planificación y presupuesto iniciales creados al principio del proyecto que se muestran en el Capítulo 4. “Planificación y Presupuesto del Proyecto (Inicial)”.

12.1. Planificación Final

La planificación se ha ido actualizando a medida que se desarrollaba el proyecto hasta llegar a esta planificación final. No solo se ha actualizado la duración de las diferentes tareas con su duración real, sino que muchas tareas han sido reorganizadas para seguir el orden real de realización de estas.

También se han añadido nuevos recursos y roles al proyecto, esto es debido a que cuando se realizó la planificación inicial se desconocía como realizar una correcta planificación del proyecto.

12.1.1. Planificación

Se ha mantenido la jornada de trabajo de 3 horas diarias, únicamente días laborables. Al haber más tareas y una restructuración del proyecto, las horas totales dedicadas al proyecto han aumentado, en la planificación inicial estaba pensado que el proyecto llevara 342 horas, tras finalizar la planificación final el proyecto ha llevado 384 horas. En la planificación inicial se cometió un desfase de 40 horas con respecto al resultado final. Aunque en el documento *Project* ponga que en total son 441 horas en total, las 57 horas de diferencia son las realizadas por los directores en las reuniones y formación.

Como comentaba anteriormente, la planificación cambió drásticamente a la planteada en el capítulo inicial. Entre los cambios se encuentra la aparición de hitos, la correcta utilización de todas las prestaciones de *Microsoft Project*, la utilización de más recursos, añadir un calendario personalizado para las horas de trabajo y roles y la verificación de que no haya solapamientos.

Se han añadido hitos a la planificación, estos hitos representan los diferentes puntos de la documentación y señalan cuando se pueden dar por finalizados, mediante la realización de las diferentes tareas que se les asigna como predecesores. Estos hitos se pueden observar en la Ilustración 129.

Apolo: Sistema inteligente de monitorización y detección de ataques basados en inteligencia artificial

« Documentación	0 horas lun 02/01/23	mar 30/05/23		No	100%
<i>Capítulo 1. Definición del Proyecto</i>	<i>0 horas vie 13/01/23</i>	<i>vie 13/01/23</i>	<i>45;46;47</i>	<i>Si</i>	<i>100%</i>
<i>Capítulo 2. Estado Actual de los Conocimientos Científico-Técnicos</i>	<i>0 horas mié 08/02/23</i>	<i>mié 08/02/23</i>	<i>51;52;53;54;55</i>	<i>Si</i>	<i>100%</i>
<i>Capítulo 3. Apolo</i>	<i>0 horas vie 03/03/23</i>	<i>vie 03/03/23</i>	<i>58;59;60;62;63,</i>	<i>Si</i>	<i>100%</i>
<i>Capítulo 4. Planificación y Presupuesto del Proyecto (Inicial)</i>	<i>0 horas mié 18/01/23</i>	<i>mié 18/01/23</i>	<i>48;49</i>	<i>Si</i>	<i>100%</i>
<i>Capítulo 5. Análisis</i>	<i>0 horas jue 23/03/23</i>	<i>jue 23/03/23</i>	<i>69;70;71;72;73</i>	<i>Si</i>	<i>100%</i>
<i>Capítulo 6. Definición del Sistema</i>	<i>0 horas lun 10/04/23</i>	<i>lun 10/04/23</i>	<i>75;76;79;80;81,</i>	<i>Si</i>	<i>100%</i>
<i>Capítulo 7. Evaluación de las Alternativas</i>	<i>0 horas lun 10/04/23</i>	<i>lun 10/04/23</i>	<i>77;79;80;81;82,</i>	<i>Si</i>	<i>100%</i>
<i>Capítulo 8. Diseño del Sistema</i>	<i>0 horas vie 14/04/23</i>	<i>vie 14/04/23</i>	<i>88;89</i>	<i>Si</i>	<i>100%</i>
<i>Capítulo 9. Evaluación del Sistema</i>	<i>0 horas jue 25/05/23</i>	<i>jue 25/05/23</i>	<i>100;101;102;10</i>	<i>Si</i>	<i>100%</i>
<i>Capítulo 10. Manuales del Sistema</i>	<i>0 horas lun 01/05/23</i>	<i>lun 01/05/23</i>	<i>98</i>	<i>Si</i>	<i>100%</i>
<i>Capítulo 11. Conclusiones y Futuro Trabajo</i>	<i>0 horas mar 30/05/23</i>	<i>mar 30/05/23</i>	<i>120;121</i>	<i>Si</i>	<i>100%</i>
<i>Capítulo 12. Planificación y Presupuesto del Proyecto (Final)</i>	<i>0 horas dom 28/05/23</i>	<i>dom 28/05/23</i>	<i>115;116;117;11</i>	<i>Si</i>	<i>100%</i>
<i>Capítulo 13. Referencias Bibliográficas</i>	<i>0 horas lun 10/04/23</i>	<i>lun 10/04/23</i>	<i>3;4;5;9</i>	<i>Si</i>	<i>100%</i>
<i>Apéndices</i>	<i>0 horas lun 02/01/23</i>	<i>lun 02/01/23</i>		<i>Si</i>	<i>100%</i>

Ilustración 129. Planificación Final: Hitos.

Los nuevos recursos añadidos han sido los directores del TFG y los nuevos roles que se han añadido son las diferentes facetas que adoptan los recursos actuales a lo largo del desarrollo del proyecto. Estos roles son los más comunes que suele haber en un proyecto, pero para este proyecto al no tocar todos los campos que se tocan en un proyecto común, se ha decidido eliminar algunos roles o redefinirlos, por ejemplo, roles como el de “comercial” no tiene cabida en nuestro proyecto al no estar enfocado en ese punto.

El recurso “Sergio Arroni del Riego”, adopta las facetas de:

- Secretario: Este rol realiza funciones estándares en el proyecto como son la documentación y escritura del documento, asistencia a reuniones; entre otras funciones.
- Investigador: Este rol realiza funciones de investigación de Apolo como son el estudio de los conocimientos científico-técnicos, realización del modelo **Apolo** y la experimentación del mismo.
- Analista: Este rol realiza las funciones de análisis del sistema.
- Arquitecto del Software: Este rol realiza las funciones de definición del sistema y evaluación de las alternativas.

- **Tester:** Este rol realiza las funciones de realizar las pruebas de la arquitectura del sistema.
- **Programador:** Este rol realiza las funciones de codificación del sistema y de formación en las nuevas tecnologías.
- **Diseñador del Software:** Este rol se encarga de la realización del diseño del sistema.

También se han añadido roles al recurso “Directores”:

- **Jefes del Proyecto:** Este rol se encarga de realizar las reuniones de seguimiento y la formación del personal.
- **Director del Proyecto:** Este rol se encarga de sentar las bases del proyecto y de realizar una validación y verificación final del mismo.

Tanto los recursos como los roles se pueden observar en la Ilustración 130.

Nombre del recurso	Tipo	Inici	Capacidad máxima	Tasa estándar	Tasa horas extra	Acumular	Calendario base	Trabajo	Costo	% trabajo completo
Sergio Arroni del Riego (Secretario)	Trabajo	SE	100%	27,53 €/hora	30,00 €/hora	Prorrateo	Apolo	106 horas	2.918,18 €	100%
Sergio Arroni del Riego (Investigador)	Trabajo	SI	100%	32,12 €/hora	34,00 €/hora	Prorrateo	Apolo	135 horas	4.336,20 €	100%
Sergio Arroni del Riego (Analista)	Trabajo	SA	100%	32,12 €/hora	34,00 €/hora	Prorrateo	Apolo	42 horas	1.349,04 €	100%
Sergio Arroni del Riego (Arquitecto Software)	Trabajo	SAS	100%	36,70 €/hora	38,00 €/hora	Prorrateo	Apolo	36 horas	1.321,20 €	100%
Sergio Arroni del Riego (Tester)	Trabajo	ST	100%	29,82 €/hora	32,00 €/hora	Prorrateo	Apolo	8 horas	238,56 €	100%
Sergio Arroni del Riego (Programador)	Trabajo	SP	100%	27,53 €/hora	30,00 €/hora	Prorrateo	Apolo	48 horas	1.321,44 €	100%
Sergio Arroni del Riego (Diseñador del Software)	Trabajo	SDS	100%	29,82 €/hora	32,00 €/hora	Prorrateo	Apolo	9 horas	268,38 €	100%
Directores (Jefe de Proyecto)	Trabajo	DJ	100%	57,35 €/hora	60,00 €/hora	Prorrateo	Apolo	51 horas	2.924,85 €	100%
Directores (Director del Proyecto)	Trabajo	DD	100%	68,82 €/hora	70,00 €/hora	Prorrateo	Apolo	6 horas	412,92 €	100%

Ilustración 130. Planificación Final: Recursos y Roles.

Como se mencionaba anteriormente se han eliminado, creado y redefinido muchas tareas para adecuarlas a la realidad del desarrollo del proyecto. Al estructurar los hitos de la documentación, se orientó las tareas a estos hitos,

pero tal y como estaba previsto en la planificación inicial se siguieron realizando reuniones, tanto iniciales, de conclusión y las semanales.

A continuación, daremos las nuevas tareas de la planificación, como consultar todos sus parámetros se puede consultar el archivo de *Project* adjuntado “planificación_Final.mpp”, el cual esta adjuntado a este documento. Para no saturar el documento de imágenes de la planificación, se adjunta a este documento un archivo de *Project* “Planificación_Final.mpp”, para que se pueda entender mejor la planificación y asignación de recursos de cada tarea. A modo de ejemplo del archivo daremos la siguiente Ilustración 147 para que se pueda observar un apartado de la planificación que hemos realizado, dicha ilustración también muestra el fragmento del diagrama de Gantt correspondiente.

- Tareas (Orden de realización): Ilustración 131, Ilustración 132, Ilustración 133, Ilustración 134, Ilustración 135, la Ilustración 136 e Ilustración 137.
- Diagrama Gantt (Orden de realización): Ilustración 138, Ilustración 139, Ilustración 140, Ilustración 141, Ilustración 142, Ilustración 143, Ilustración 144, Ilustración 145 e Ilustración 146.

EDT	Nombre de tarea	Trabajo	Comienzo	Fin
1	↳ Apolo	448 horas	lun 02/01/23	mié 07/06/23
1.1	↳ Documentación	0 horas	lun 02/01/23	mar 30/05/23
1.2	↳ Reuniones	104 horas	lun 02/01/23	mié 07/06/23
1.3	↳ Introducción del proyecto	39 horas	mar 03/01/23	mié 18/01/23
1.4	↳ Investigación	45 horas	mié 18/01/23	mié 08/02/23
1.5	↳ Apolo	51 horas	mié 08/02/23	vie 03/03/23
1.6	↳ Análisis	42 horas	vie 03/03/23	jue 23/03/23
1.7	↳ Definición del sistema	36 horas	jue 23/03/23	lun 10/04/23
1.8	↳ Formación	10 horas	lun 10/04/23	mié 12/04/23
1.9	↳ Diseño del sistema	9 horas	mar 11/04/23	vie 14/04/23
1.10	↳ Realización de los componentes	34 horas	vie 14/04/23	lun 01/05/23
1.11	↳ Evaluación del sistema	54 horas	lun 01/05/23	jue 25/05/23
1.12	↳ Realizar la planificación y presupuesto del proyecto	9 horas	jue 25/05/23	dom 28/05/23
1.13	↳ Conclusión	2 horas	dom 28/05/23	mar 30/05/23
1.14	↳ Realizar los apéndices	4 horas	mar 30/05/23	mié 31/05/23
1.15	Desplegar el sistema	9 horas	mié 31/05/23	lun 05/06/23

Ilustración 131. Planificación Final: Tareas I

Apolo: Sistema inteligente de monitorización y detección de ataques basados en inteligencia artificial

EDT	Nombre de tarea	Trabajo	Comienzo	Fin
1.2	‣ Reuniones	104 horas	lun 02/01/23	mié 07/06/23
1.2.1	Reuniones iniciales	6 horas	lun 02/01/23	mar 03/01/23
1.2.2	‣ Reuniones semanales con los directores	92 horas	lun 02/01/23	mié 07/06/23
1.2.3	Reunión de clausura del documento	6 horas	mar 06/06/23	mié 07/06/23
1.3	‣ Introducción del proyecto	39 horas	mar 03/01/23	mié 18/01/23
1.3.1	Realización de la introducción del proyecto	9 horas	mar 03/01/23	vie 06/01/23
1.3.2	Realización de la motivación del proyecto	9 horas	vie 06/01/23	mar 10/01/23
1.3.3	Realización del alcance y descripción del proyecto	9 horas	mar 10/01/23	vie 13/01/23
1.3.4	Realizar planificación inicial	6 horas	sáb 14/01/23	dom 15/01/23
1.3.5	Realizar presupuestos inicial	6 horas	lun 16/01/23	mié 18/01/23
1.4	‣ Investigación	45 horas	mié 18/01/23	mié 08/02/23
1.4.1	Investigar el estado de los conocimientos actuales sobre los SNI	3 horas	mié 18/01/23	jue 19/01/23
1.4.2	Investigar el estado de los conocimientos actuales sobre los IDS	12 horas	jue 19/01/23	mié 25/01/23
1.4.3	Investigar el estado de los conocimientos actuales sobre el AML	12 horas	mié 25/01/23	mar 31/01/23
1.4.4	Investigar el estado de los conocimientos actuales sobre MAB	9 horas	mar 31/01/23	vie 03/02/23
1.4.5	Realizacion del estado de los conocimientos científico-técnicos	9 horas	vie 03/02/23	mié 08/02/23

Ilustración 132. Planificación Final: Tareas II

Apolo: Sistema inteligente de monitorización y detección de ataques basados en inteligencia artificial

EDT	Nombre de tarea	Trabajo	Comienzo	Fin
1.5	• Apolo		51 horas mié 08/02/23	vie 03/03/23
1.5.1	• Capa Modelos		24 horas mié 08/02/23	lun 20/02/23
1.5.1.1	Preprocesado de los datasets		9 horas mié 08/02/23	lun 13/02/23
1.5.1.2	Eleccion de los clasificadores		6 horas lun 13/02/23	mié 15/02/23
1.5.1.3	Implementación de los clasificadores		9 horas mié 15/02/23	lun 20/02/23
1.5.2	• Capa MAB		15 horas lun 20/02/23	jue 23/02/23
1.5.2.1	Implementación de MAB a los clasificadores		6 horas lun 20/02/23	mié 22/02/23
1.5.2.2	Añadir el muestreo Thomson		2 horas mié 22/02/23	jue 23/02/23
1.5.2.3	Adaptar algoritmo MAB al código		7 horas jue 23/02/23	lun 27/02/23
1.5.3	• Capa Clusterización		12 horas lun 27/02/23	vie 03/03/23
1.5.3.1	Realización de una clusterización binaria de los clasificadores		6 horas lun 27/02/23	mié 01/03/23
1.5.3.2	Adaptar la clusterización al código		6 horas mié 01/03/23	vie 03/03/23
1.6	• Análisis		42 horas vie 03/03/23	jue 23/03/23
1.6.1	Requisitos del sistema		12 horas vie 03/03/23	jue 09/03/23
1.6.2	Identificación de los componentes de la arquitectura		12 horas jue 09/03/23	mié 15/03/23
1.6.3	Identificación de actores		3 horas mié 15/03/23	jue 16/03/23
1.6.4	Realización del diagrama de contexto		6 horas jue 16/03/23	lun 20/03/23
1.6.5	Realización de los casos de uso y escenarios		9 horas lun 20/03/23	jue 23/03/23

Ilustración 133. Planificación Final: Tareas III

Apolo: Sistema inteligente de monitorización y detección de ataques basados en inteligencia artificial

EDT	Nombre de tarea	Trabajo	Comienzo	Fin
1.7	• Definición del sistema	36 horas	jue 23/03/23	lun 10/04/23
1.7.1	Buscar las tecnologías y herramientas a utilizar en los componentes	9 horas	jue 23/03/23	mar 28/03/23
1.7.2	IDEAR la comunicación entre componentes	3 horas	mar 28/03/23	mié 29/03/23
1.7.3	Realizar un estudio de alternativas de las opciones descartadas	6 horas	mié 29/03/23	vie 31/03/23
1.7.4	• Documentar la definición y alternativas	18 horas	vie 31/03/23	lun 10/04/23
1.7.4.1	Microservicios	2 horas	vie 31/03/23	lun 03/04/23
1.7.4.2	Herramientas utilizadas	4 horas	lun 03/04/23	mar 04/04/23
1.7.4.3	Tecnologías utilizadas	4 horas	mar 04/04/23	mié 05/04/23
1.7.4.4	Lenguajes utilizados	2 horas	jue 06/04/23	jue 06/04/23
1.7.4.5	Estandares y normas utilizados	1 hora	jue 06/04/23	jue 06/04/23
1.7.4.6	Componentes utilizados	5 horas	vie 07/04/23	lun 10/04/23
1.8	• Formación	10 horas	lun 10/04/23	mié 12/04/23
1.8.1	Formar al personal sobre las tecnologías y herramientas utilizadas	5 horas	lun 10/04/23	mié 12/04/23
1.8.2	Recibir formación sobre las tecnologías y herramientas utilizadas	5 horas	lun 10/04/23	mié 12/04/23
a se completó el mié 3.		9 horas	mar 11/04/23	vie 14/04/23
	Diseño de los diagramas del sistema	6 horas	mar 11/04/23	jue 13/04/23
1.9.2	Diseño de las bases de datos	3 horas	jue 13/04/23	vie 14/04/23

Ilustración 134. Planificación Final: Tareas IV

EDT	Nombre de tarea	Trabajo	Comienzo	Fin
1.10	• Realización de los componentes	34 horas	vie 14/04/23	lun 01/05/23
1.10.1	Implementar el firewall y proxy reverso	5 horas	vie 14/04/23	mar 18/04/23
1.10.2	Busqueda de una aplicación de ejemplo	3 horas	mar 18/04/23	mié 19/04/23
1.10.3	Implementar la gestión de peticiones	6 horas	mié 19/04/23	vie 21/04/23
1.10.4	• Implementar la clasificación de las peticiones	15 horas	vie 21/04/23	vie 28/04/23
1.10.4.1	Implementar el sistema de colas	4 horas	vie 21/04/23	lun 24/04/23
1.10.4.2	Implementar Apolo	6 horas	lun 24/04/23	mié 26/04/23
1.10.4.3	Implementar la gestión de anomalías	5 horas	mié 26/04/23	vie 28/04/23
1.10.5	Realizar los manuales del sistema	5 horas	vie 28/04/23	lun 01/05/23

Ilustración 135. Planificación Final: Tareas V

Apolo: Sistema inteligente de monitorización y detección de ataques basados en inteligencia artificial

EDT	Nombre de tarea	Trabajo	Comienzo	Fin
1.11	▪ Evaluación del sistema	47 horas	lun 01/05/23	mar 23/05/23
1.11.1	Elegir las metodologías a utilizar	4 horas	lun 01/05/23	mié 03/05/23
1.11.2	Elegir la validación a utilizar	2 horas	mié 03/05/23	mié 03/05/23
1.11.3	▪ Realizar una experimentación de Apolo en varios escenarios	41 horas	mié 03/05/23	mar 23/05/23
1.11.3.1	▪ Primer escenario	14 horas	mié 03/05/23	mié 10/05/23
1.11.3.1.1	Probar los clasificadores más populares en un entorno tradicional de IDS	8 horas	mié 03/05/23	lun 08/05/23
1.11.3.1.2	Probar Apolo en un entorno tradicional de IDS	6 horas	lun 08/05/23	mié 10/05/23
1.11.3.2	▪ Segundo escenario	19 horas	mié 10/05/23	jue 18/05/23
1.11.3.2.1	Probar los clasificadores más populares frente a ataques AML	10 horas	mié 10/05/23	lun 15/05/23
1.11.3.2.2	Probar Apolo frente a ataques AML	9 horas	lun 15/05/23	jue 18/05/23
1.11.3.3	▪ Tercer Escenario	8 horas	jue 18/05/23	mar 23/05/23
1.11.3.3.1	Generar un dataset	5 horas	jue 18/05/23	lun 22/05/23
1.11.3.3.2	Entrenar y testear Apolo con el dataset generado	3 horas	mar 23/05/23	mar 23/05/23
1.12	▪ Realizar la planificación y presupuesto del proyecto	9 horas	mié 24/05/23	jue 01/06/23
1.12.1	Realizar la planificación	5 horas	mié 24/05/23	vie 26/05/23
1.12.2	Realizar el presupuesto	4 horas	mié 31/05/23	jue 01/06/23

Ilustración 136. Planificación Final: Tareas VI

EDT	Nombre de tarea	Trabajo	Comienzo
1.13	▪ Conclusión	2 horas	dom 28/05/23
1.13.1	Realizar la conclusión del proyecto	1 hora	dom 28/05/23
1.13.2	Realizar el trabajo futuro del proyecto	1 hora	lun 29/05/23
1.14	▪ Realizar los apéndices	4 horas	mar 30/05/23
1.14.1	Realizar un análisis y gestión de los riesgos del proyecto	3 horas	mar 30/05/23
1.14.2	Documentar el apéndice	1 hora	mié 31/05/23
1.15	Desplegar el sistema	9 horas	mié 31/05/23

Ilustración 137. Planificación Final: Tareas VII

Apolo: Sistema inteligente de monitorización y detección de ataques basados en inteligencia artificial

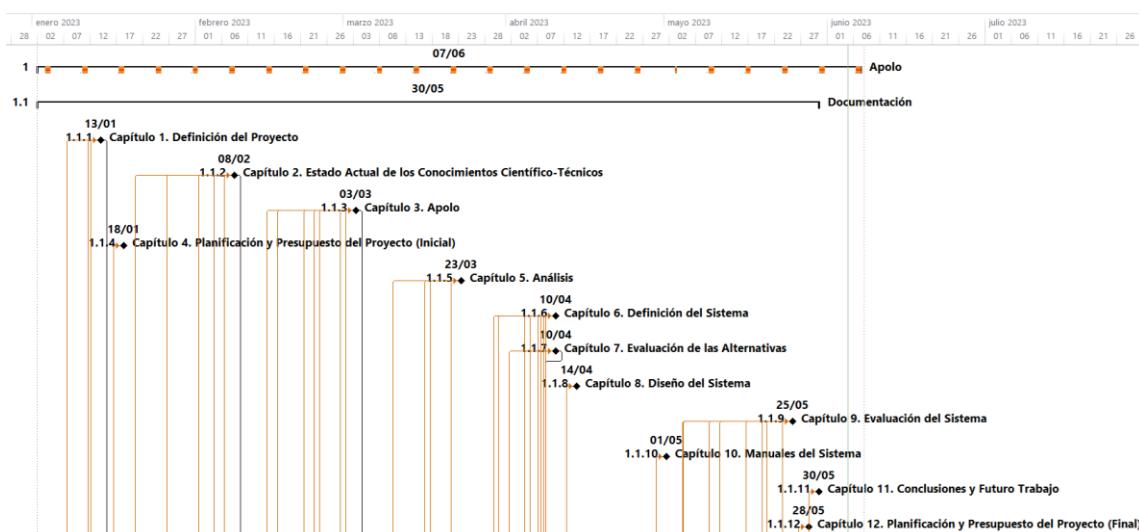


Ilustración 138. Planificación Final: Diagrama de Gantt I

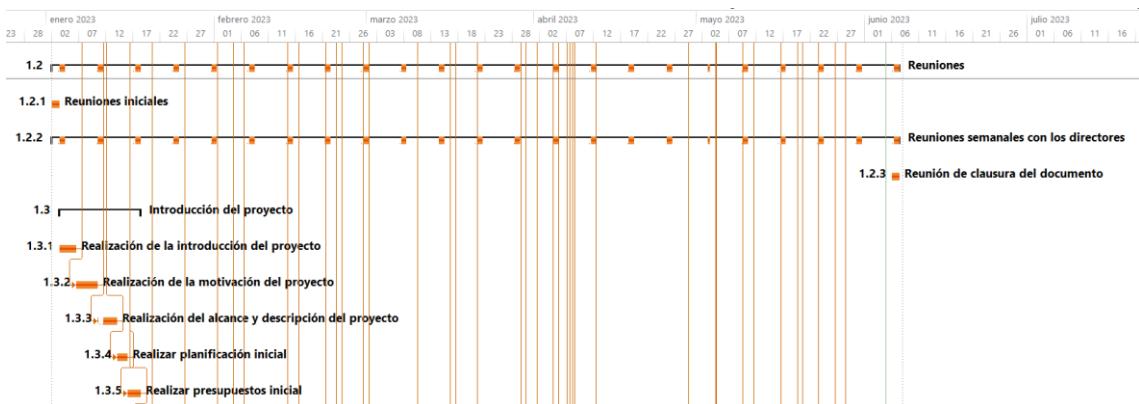


Ilustración 139. Planificación Final: Diagrama de Gantt II

Apolo: Sistema inteligente de monitorización y detección de ataques basados en inteligencia artificial

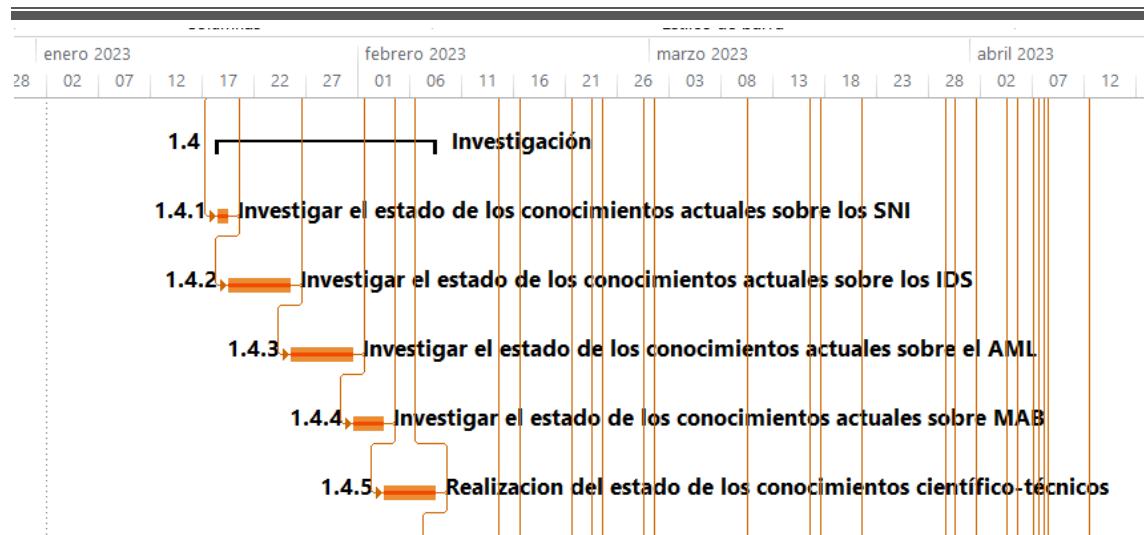


Ilustración 140. Planificación Final: Diagrama de Gantt III

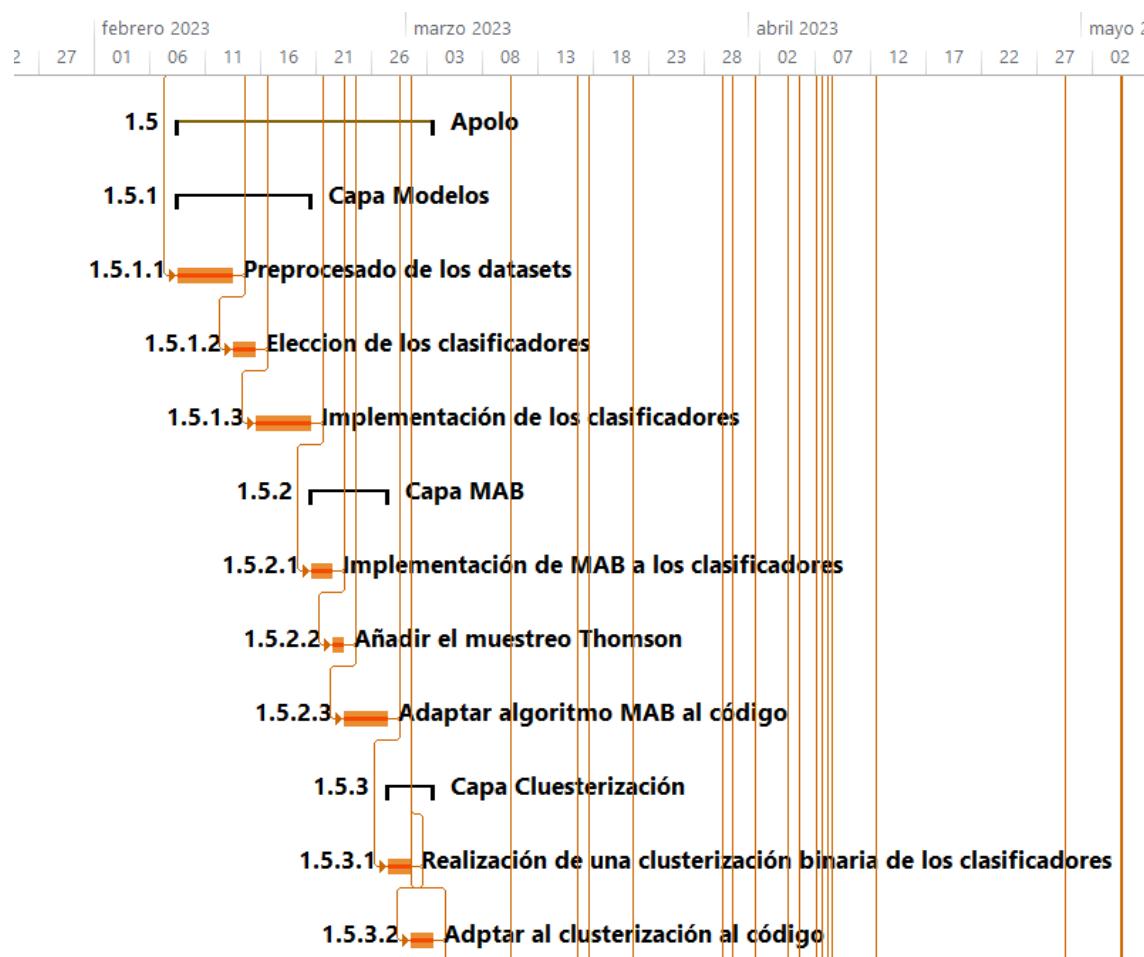


Ilustración 141. Planificación Final: Diagrama de Gantt IV

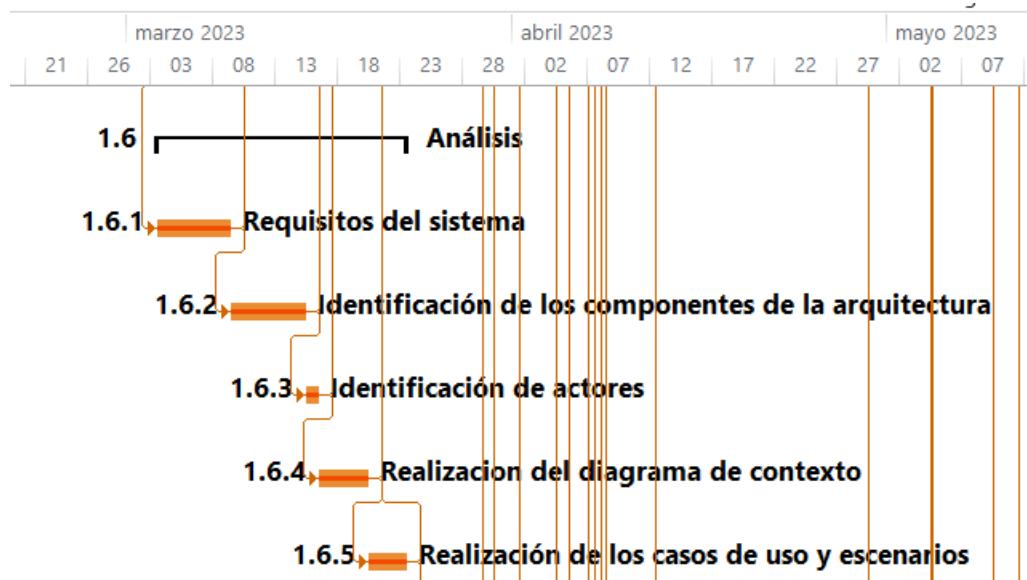


Ilustración 142. Planificación Final: Diagrama de Gantt V

Apolo: Sistema inteligente de monitorización y detección de ataques basados en inteligencia artificial

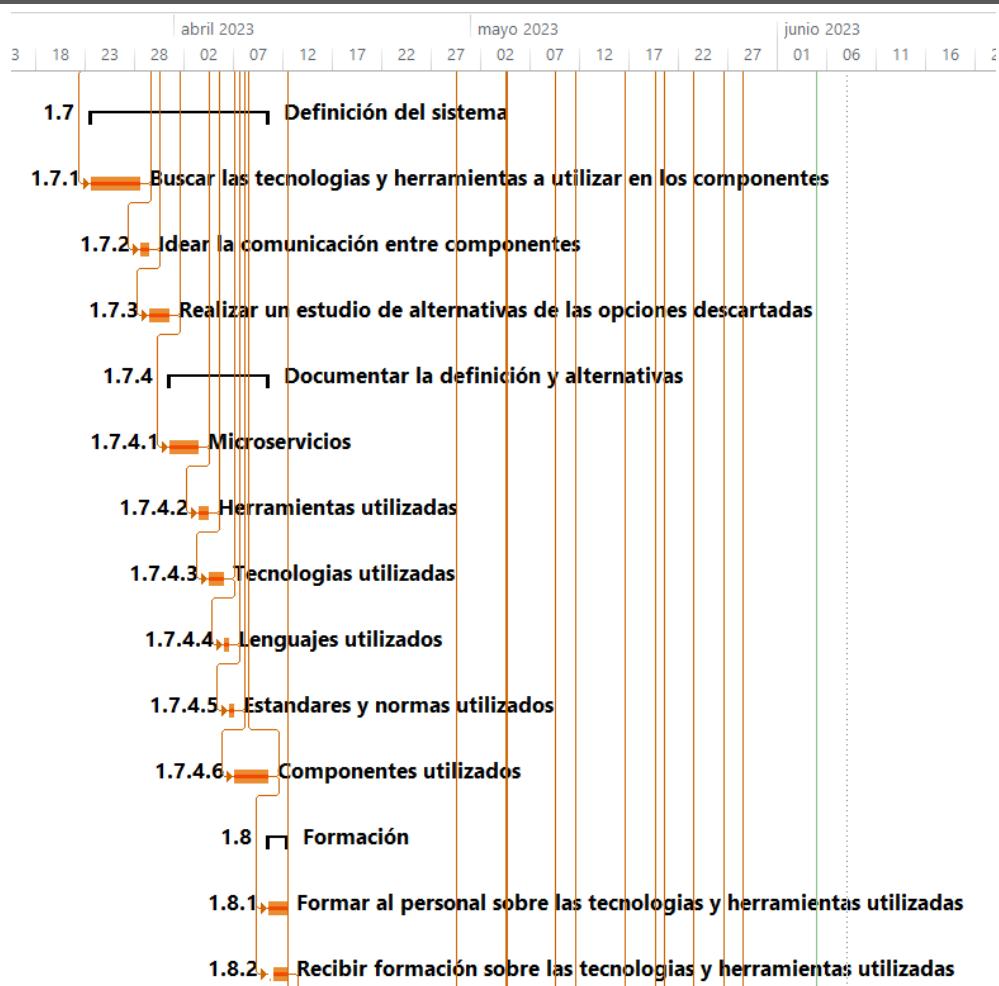


Ilustración 143. Planificación Final: Diagrama de Gantt VI

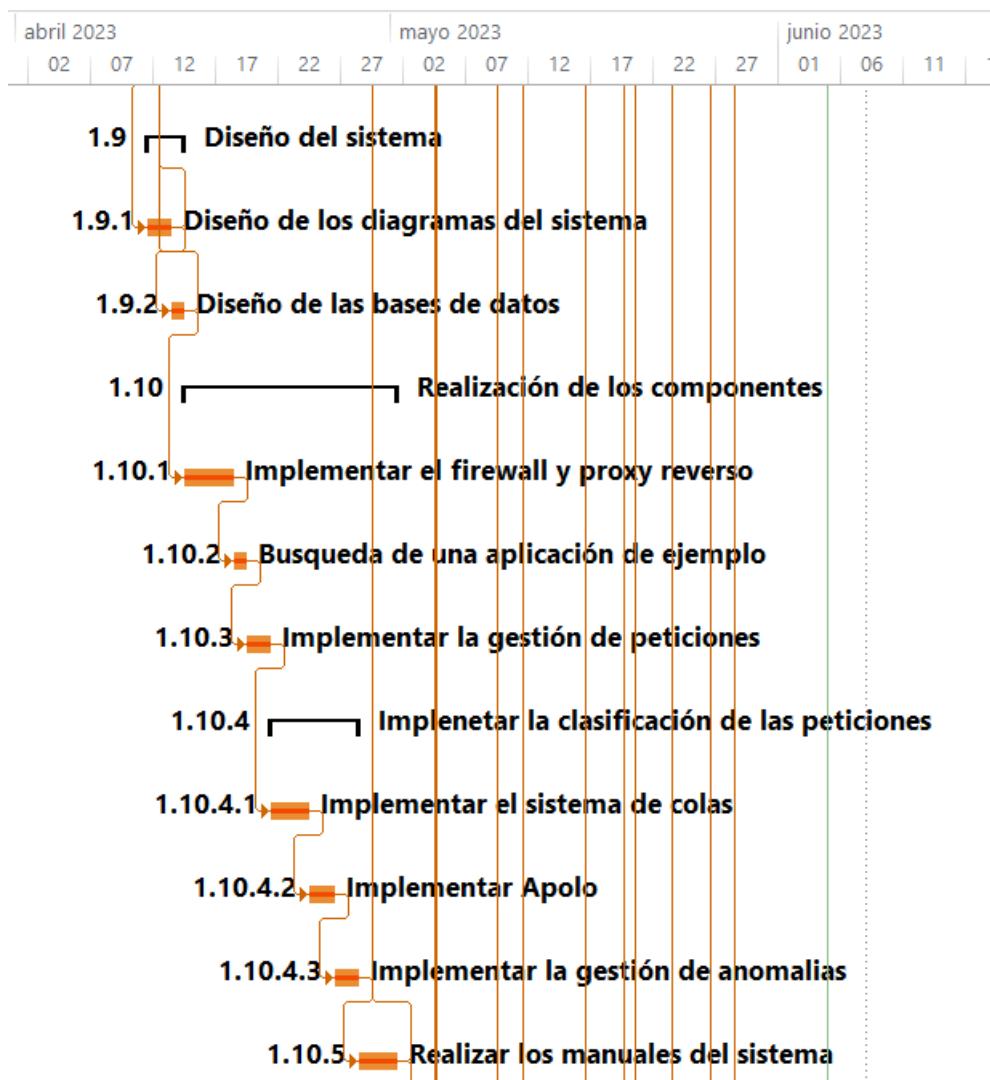


Ilustración 144. Planificación Final: Diagrama de Gantt VII

Apolo: Sistema inteligente de monitorización y detección de ataques basados en inteligencia artificial

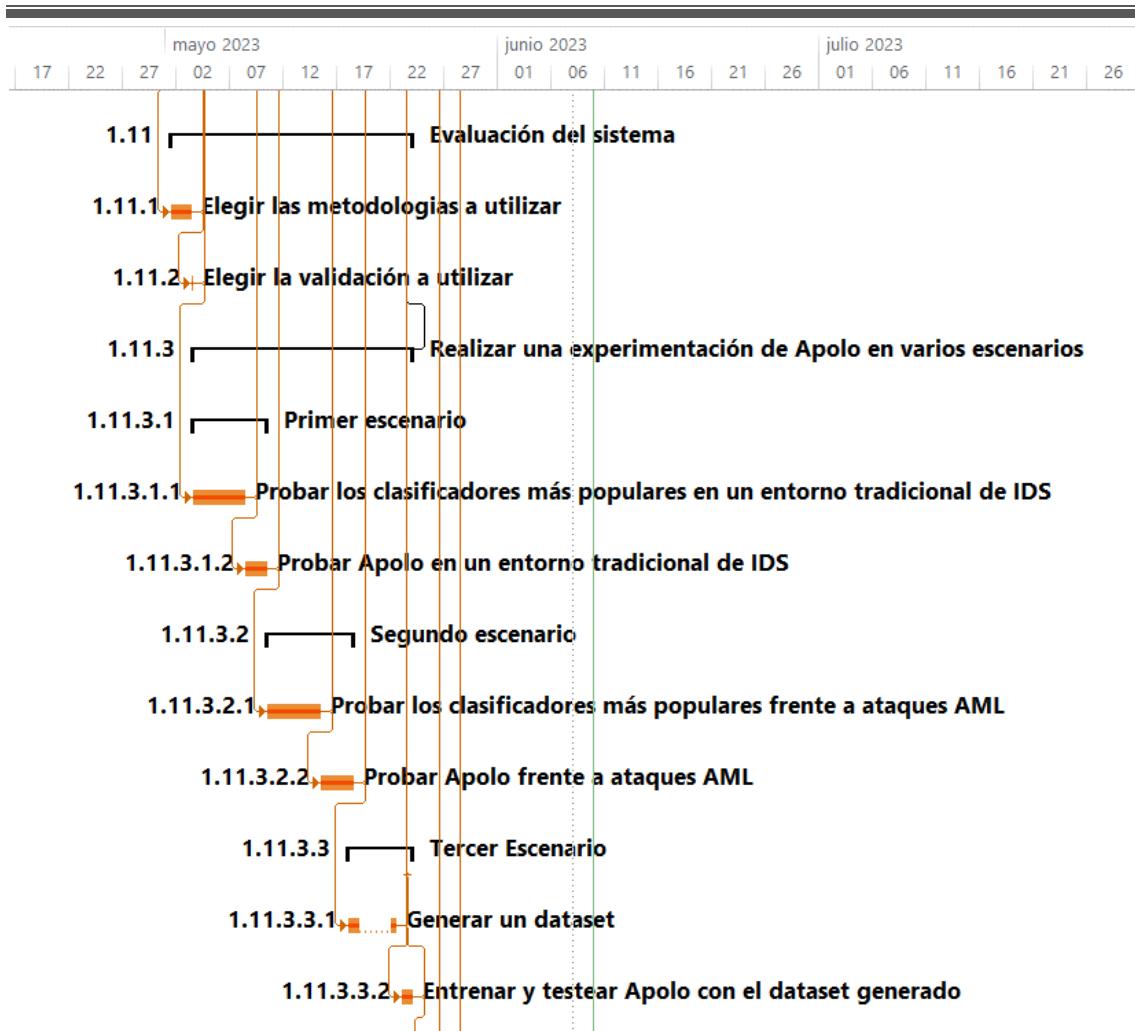


Ilustración 145. Planificación Final: Diagrama de Gantt VIII



Ilustración 146. Planificación Final: Diagrama de Gantt IX



Ilustración 147. Planificación Final: Apolo.

12.1.2. Organigrama

En esta sección, nos adentraremos en la creación y comprensión del organigrama, una representación gráfica esencial de la estructura jerárquica y las relaciones dentro de una organización. Un organigrama es una herramienta vital que refleja la estructura interna de una entidad y proporciona una vista clara

de los roles, responsabilidades y relaciones entre los diferentes niveles de la organización.

Para este organigrama hemos tenido en cuenta los recursos y roles asignados en la planificación. En la Ilustración 148 se puede observar el organigrama de este proyecto. Al implementar un organigrama facilitamos la visualización de la jerarquía y la estructura del equipo.

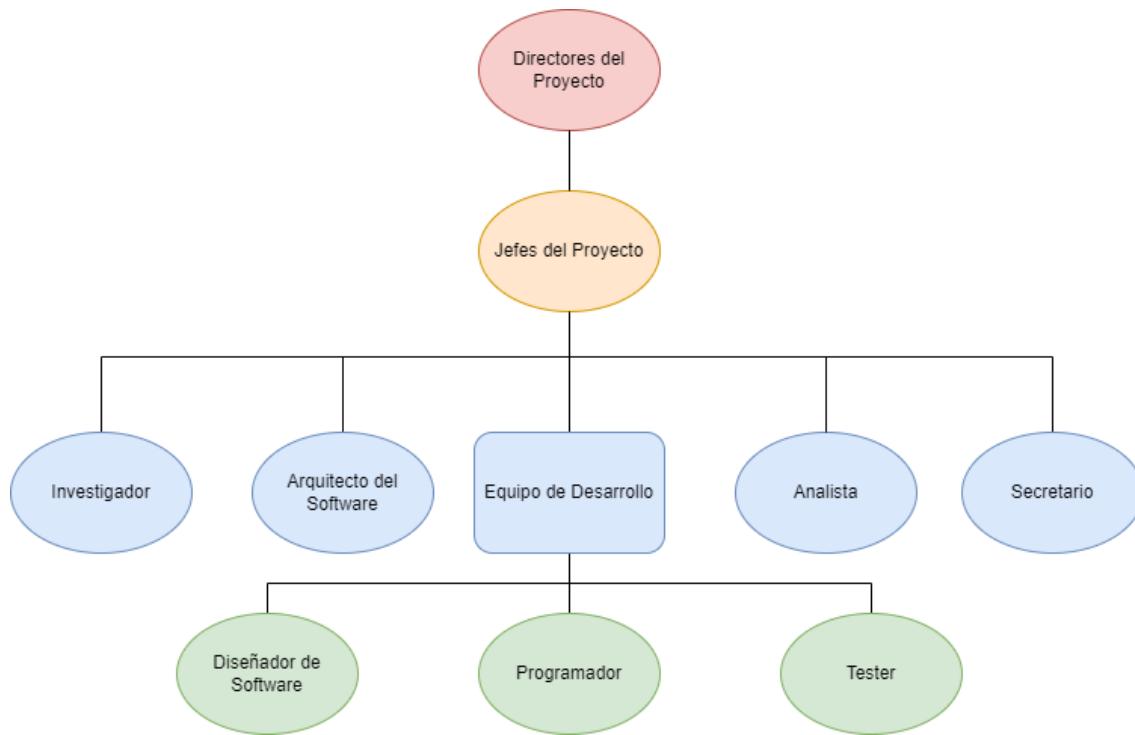


Ilustración 148. Planificación: Organigrama.

12.2. Presupuesto Final

En esta sección daremos el presupuesto final del proyecto, al haber cambiado la planificación del proyecto, el presupuesto ha sufrido grandes cambios. Vamos a seguir la misma estructura seguida en el presupuesto inicial.

12.2.1. Definición de la Empresa

Vamos a empezar definiendo la empresa y sus gastos generales anuales. Estos gastos deben ser tenidos en cuenta para no incurrir en pérdidas y que al final de cada año fiscal se pueda tener un gran margen de beneficio, es por eso por lo que son de vital importancia en la realización del presupuesto.

12.2.1.1. Personal

Para la realización de este proyecto contamos con un personal variado Tabla 58. Reciben un sueldo bruto anual resultado de multiplicar las 1.776 horas de trabajo laborables al año por el sueldo bruto/hora que tienen asignado. No obstante, a ese sueldo bruto anual hay que añadirle el 25% extra en concepto de contingencias comunes para obtener la cantidad total real que le cuesta a la empresa el poder pagarle al empleado dicho sueldo.

Por lo que, a la empresa le cuesta el sueldo del personal un total de 330.780,00 € anualmente, Tabla 59.

Personal	Cantidad
Sergio Arroni del Riego (Secretario)	1
Sergio Arroni del Riego (Investigador)	1
Sergio Arroni del Riego (Analista)	1
Sergio Arroni del Riego (Arquitecto Software)	1
Sergio Arroni del Riego (Tester)	1
Sergio Arroni del Riego (Programador)	1
Sergio Arroni del Riego (Diseñador del Software)	1
Directores (Jefe de Proyecto)	1
Directores (Director del Proyecto)	1

Tabla 58. Personal

Personal	Sueldo Bruto Hora	Sueldo Bruto Año	Coste Salario Hora	Coste Salario Año
Sergio Arroni del Riego (Secretario)	12,00 €	21.312,00 €	15,00 €	26.640,00 €
Sergio Arroni del Riego (Investigador)	14,00 €	24.864,00 €	17,50 €	31.080,00 €
Sergio Arroni del Riego (Analista)	14,00 €	24.864,00 €	17,50 €	31.080,00 €
Sergio Arroni del Riego (Arquitecto Software)	16,00 €	28.416,00 €	20,00 €	35.520,00 €
Sergio Arroni del Riego (Tester)	13,00 €	23.088,00 €	16,25 €	28.860,00 €
Sergio Arroni del Riego (Programador)	12,00 €	21.312,00 €	15,00 €	26.640,00 €
Sergio Arroni del Riego (Diseñador del Software)	13,00 €	23.088,00 €	16,25 €	28.860,00 €
Directores (Jefe de Proyecto)	25,00 €	44.400,00 €	31,25 €	55.500,00 €
Directores (Director del Proyecto)	30,00 €	53.280,00 €	37,50 €	66.600,00 €
TOTAL	149,00 €	264.624,00 €	186,25 €	330.780,00 €

Tabla 59. Precio / hora personal

12.2.1.1.1. Productividad del Personal

En general, el personal no puede mantener un nivel de productividad al 100% durante su jornada laboral diaria de 3 horas. Es probable que solo dedique estrictamente 2 horas y cuarto a generar valor al proyecto, lo que significa que los otros 45 minutos serían pagados, pero no generaría beneficios para la empresa. Por lo tanto, al vender nuestros servicios, debemos tener en cuenta el costo indirecto que representa para nosotros cada empleado que no es productivo al 100%, Tabla 60.

Para calcular esto, se estima un porcentaje de productividad, y se aplica ese porcentaje a su salario bruto anual. De esta manera, se obtiene el coste directo,

que representa el coste real del servicio que presta cada empleado. Por otro lado, la diferencia entre el porcentaje de productividad y el 100% representa el porcentaje de coste indirecto que, aplicado al salario bruto anual, resultaría en el coste total indirecto de cada empleado. La Tabla 61 muestra los diferentes costes asociados con cada empleado.

Personal	Total	Prod (%)	Coste Directo	CI(%)	Coste Indirecto
Sergio Arroni del Riego (Secretario)	26.640,00 €	75,00%	19.980,00 €	25%	6.660,00 €
Sergio Arroni del Riego (Investigador)	31.080,00 €	80,00%	24.864,00 €	20%	6.216,00 €
Sergio Arroni del Riego (Analista)	31.080,00 €	76,00%	23.620,80 €	24%	7.459,20 €
Sergio Arroni del Riego (Arquitecto Software)	35.520,00 €	70,00%	24.864,00 €	30%	10.656,00 €
Sergio Arroni del Riego (Tester)	28.860,00 €	85,00%	24.531,00 €	15%	4.329,00 €
Sergio Arroni del Riego (Programador)	26.640,00 €	83,00%	22.111,20 €	17%	4.528,80 €
Sergio Arroni del Riego (Diseñador del Software)	28.860,00 €	72,00%	20.779,20 €	28%	8.080,80 €
Directores (Jefe de Proyecto)	55.500,00 €	80,00%	44.400,00 €	20%	11.100,00 €
Directores (Director del Proyecto)	66.600,00 €	75,00%	49.950,00 €	25%	16.650,00 €
TOTAL	26.640,00 €		19.980,00 €		6.660,00 €

Tabla 60. Productividad personal.

Personal	Productividad (%)	Horas / Año	Horas productivas / Año (por persona)	Horas productivas (Total Empresa)
Sergio Arroni del Riego (Secretario)	75,00%	1776	1332,00	1332,00
Sergio Arroni del Riego (Investigador)	80,00%	1776	1420,80	1420,80
Sergio Arroni del Riego (Analista)	76,00%	1776	1349,76	1349,76
Sergio Arroni del Riego (Arquitecto Software)	70,00%	1776	1243,20	1243,20
Sergio Arroni del Riego (Tester)	85,00%	1776	1509,60	1509,60
Sergio Arroni del Riego (Programador)	83,00%	1776	1474,08	1474,08
Sergio Arroni del Riego (Diseñador del Software)	72,00%	1776	1278,72	1278,72
Directores (Jefe de Proyecto)	80,00%	1776	1420,80	1420,80
Directores (Director del Proyecto)	75,00%	1776	1332,00	1332,00
TOTAL			12360,96	12360,96

Tabla 61. Horas total productivas personal.

12.2.1.2. Costes Indirectos

No hubo cambios en los costes indirectos generales, consultad el punto 4.2.1.2. "Costes Indirectos" para obtener más información.

12.2.1.3. Amortizaciones

No hubo cambios en los costes indirectos generales, consultad el punto 4.2.1.3. "Amortizaciones" para obtener más información.

12.2.1.4. Precio Por Hora del Personal

A el coste de la Tabla 61 hay que añadirle los costes indirectos que le cuesta a la empresa mantener a ese recurso, el beneficio que esperamos obtener, así como el margen que nos queremos guardar para posibles contingencias. Este nuevo precio por hora, que se puede ver en la Tabla 62 y en la Tabla 63, sería el precio que habría que pasarle al cliente en caso de que nos pida el costo de nuestro recurso. Para la realización de las partidas se utilizará el coste por hora de la Tabla 61.

Personal	Precio / Hora	Horas productivas (Total empresa)	Coste sin beneficios	Precio / Hora (Con beneficios del 25%)
Sergio Arroni del Riego (Secretario)	27,53 €	1332,00	36.667,03 €	137,50 €
Sergio Arroni del Riego (Investigador)	32,12 €	1420,80	45.630,08 €	137,50 €
Sergio Arroni del Riego (Analista)	32,12 €	1349,76	43.348,57 €	137,50 €
Sergio Arroni del Riego (Arquitecto Software)				137,50 €
	36,70 €	1243,20	45.630,08 €	
Sergio Arroni del Riego (Tester)	29,82 €	1509,60	45.018,96 €	137,50 €
Sergio Arroni del Riego (Programador)	27,53 €	1474,08	40.578,18 €	137,50 €
Sergio Arroni del Riego (Diseñador del Software)				137,50 €
	29,82 €	1278,72	38.133,71 €	
Directores (Jefe de Proyecto)	57,35 €	1420,80	81.482,28 €	137,50 €
Directores (Director del Proyecto)	68,82 €	1332,00	91.667,57 €	137,50 €
TOTAL			468.156,44 €	

Tabla 62. Precio / Hora Personal (Parte I)

Personal	Precio / Hora (Con beneficios del 25%)	Coste con beneficio	Precio / Hora (Con margen de facturación 5%)	Facturación
Sergio Arroni del Riego (Secretario)	34,41 €	45.833,78 €	36,13 €	48.125,47 €
Sergio Arroni del Riego (Investigador)	40,14 €	57.037,60 €	42,15 €	59.889,48 €
Sergio Arroni del Riego (Analista)	40,14 €	54.185,72 €	42,15 €	56.895,00 €
Sergio Arroni del Riego (Arquitecto Software)	45,88 €	57.037,60 €	48,17 €	59.889,48 €
Sergio Arroni del Riego (Tester)	37,28 €	56.273,70 €	39,14 €	59.087,38 €
Sergio Arroni del Riego (Programador)	34,41 €	50.722,72 €	36,13 €	53.258,86 €
Sergio Arroni del Riego (Diseñador del Software)	37,28 €	47.667,13 €	39,14 €	50.050,49 €
Directores (Jefe de Proyecto)	71,69 €	101.852,85 €	75,27 €	106.945,49 €
Directores (Director del Proyecto)	86,02 €	114.584,46 €	90,33 €	120.313,68 €
TOTAL		585.195,55 €		614.455,33 €

Tabla 63. Precio / Hora Personal (Parte II)

12.2.1.5. Total

A continuación, vamos a sacar el total de costes indirectos y costes anuales de la empresa.

12.2.1.5.1. Total Costes Indirectos

Primero vamos a calcular el coste indirecto de la empresa, para ello vamos a sumar los Costes Indirectos (CI) generales, las amortizaciones y el coste de la productividad del personal, se puede observar estos costes en la Tabla 64. Como se puede observar hay un incremento considerable de unos 70.000 € aproximadamente, esto es debido al coste de personal.

Motivo	Importe
CI generales	135.601,44 €
Amortizaciones	1.775,00 €
Personal	75.679,80 €
TOTAL CI	213.056,24 €

Tabla 64. Total CI

12.2.1.5.2. Total Anual

Ahora vamos a calcular los costes anuales de la empresa, para ello vamos a añadir un beneficio deseado del 25% y un margen de facturación del 5%. Al final nos sale un total de 614.455,33 € de costes indirectos anuales, se puede ver en la Tabla 65. Esto es un aumento muy grande con respecto al del presupuesto inicial que era de uno 218.185,33€.

Estos gastos son el importe de dinero que tiene que generar la empresa este año para poder obtener beneficios, estos gastos son independientes a los proyectos que se realicen dentro de la empresa y por ende al de este documento.

Gracias a estos costes, podemos saber cuál es el precio / hora, del personal, el cual tenemos que utilizar a la hora de hacer los costes directos para poder contrarrestar sus costes indirectos y cuál es el precio / hora que necesitamos para poder contar con el beneficio y el margen y hemos de pasar al cliente en caso de que nos lo pida.

CONCEPTO	IMPORTE
Total CD	255.100,20 €
Total CI	213.056,24 €
Suma de los costes	468.156,44 €
Beneficio deseado (25%)	117.039,11 €
Coste Total	585.195,55 €
Porcentaje del margen entre Facturación y Coste Total	5,00%
Margen entre Facturación y Coste Total	29.259,78 €
Facturación	614.455,33 €

Tabla 65. Total Anual

12.2.2. Costes Directos

A continuación, vamos con el proyecto per se, en esta sección daremos los costes directos del proyecto **Apolo**. Está dividido en diferentes partidas que representan las partes de alto nivel de la planificación y estas a su vez están divididas en tareas, toda esta información está extraída de la planificación.

12.2.2.1. Partida 1

La primera partida hace referencia a las tareas de “Reuniones”, en ella encontramos tres grupos, las iniciales, las semanales y la de clausura. Nos da un total de 4.488,58 €, se puede ver en la Tabla 66 y en la Tabla 67.

Reuniones					
<i>I1</i>	<i>I2</i>	<i>I3</i>	<i>Descripción</i>	<i>Horas</i>	<i>Unidades</i>
01			Reuniones		
	001		Reunión inicial		
		01	Directores (Director del Proyecto)	3	horas
		02	Sergio Arroni del Riego (Secretario)	3	horas
	002		Reunión de clausura del documento		
		01	Directores (Director del Proyecto)	3	horas
		02	Sergio Arroni del Riego (Secretario)	3	horas
02			Reuniones semanales		
	001		Reunión semanal		
		01	Sergio Arroni del Riego (Secretario)	46	horas
		02	Directores (Jefe de Proyecto)	46	horas

Tabla 66. Partida 1 (Parte I)

Reuniones						
I1	I2	I3	Precio	Subtotal(3)	Subtotal(2)	Total
01						578,10 €
	001				289,05 €	
		01	68,82 €	206,46 €		
		02	27,53 €	82,59 €		
	002				289,05 €	
		01	68,82 €	206,46 €		
		02	27,53 €	82,59 €		
02						3.904,48 €
	001				3.904,48 €	
		01	27,53 €	1.266,38 €		
		02	57,35 €	2.638,10 €		
TOTAL						4.488,58 €

Tabla 67. Partida 1 (Parte II)

12.2.2.2. Partida 2

La segunda partida hace referencia a las tareas de “Introducción del proyecto”. Nos da un total de 1.073,67 €, se puede ver en la Tabla 68 y en la Tabla 69.

Introducción del proyecto						
I1	I2	I3	Descripción	Horas	Unidades	
01			Introducción del proyecto			
	001		Realización de la introducción del proyecto			
		01	Sergio Arroni del Riego (Secretario)	9	horas	
	002		Realización de la motivación del proyecto			
		01	Sergio Arroni del Riego (Secretario)	9	horas	
	003		Realización del alcance y descripción del proyecto			
		01	Sergio Arroni del Riego (Secretario)	9	horas	
	004		Realizar planificación inicial			
		01	Sergio Arroni del Riego (Secretario)	6	horas	
	005		Realizar presupuestos iniciales			
		01	Sergio Arroni del Riego (Secretario)	6	horas	

Tabla 68. Partida 2 (Parte I)

Introducción del proyecto						
I1	I2	I3	Precio	Subtotal(3)	Subtotal(2)	Total
01						1.073,67 €
	001				247,77 €	
		01	27,53 €	247,77 €		
	002				247,77 €	
		01	27,53 €	247,77 €		
	003				247,77 €	
		01	27,53 €	247,77€		
	004				165,18 €	
		01	27,53 €	165,18 €		
	005				165,18 €	
		01	27,53 €	165,18 €		
TOTAL						1.073,67 €

Tabla 69. Partida 2 (Parte II)

12.2.2.3. Partida 3

La tercera partida hace referencia a las tareas de “Investigación”. Nos da un total de 1.445,40 €, se puede ver en la Tabla 70 y en la Tabla 71.

Apolo: Sistema inteligente de monitorización y detección de ataques basados en inteligencia artificial

Investigación					
I1	I2	I3	Descripción	Horas	Unidades
01			Investigación		
	001		Investigar el estado de los conocimientos actuales sobre los SNI		
		01	Sergio Arroni del Riego (Investigador)	3	horas
	002		Investigar el estado de los conocimientos actuales sobre los IDS		
		01	Sergio Arroni del Riego (Investigador)	12	horas
	003		Investigar el estado de los conocimientos actuales sobre el AML		
		01	Sergio Arroni del Riego (Investigador)	12	horas
	004		Investigar el estado de los conocimientos actuales sobre MAB		
		01	Sergio Arroni del Riego (Investigador)	9	horas
	005		Realización del estado de los conocimientos científico-técnicos		
		01	Sergio Arroni del Riego (Investigador)	9	horas

Tabla 70. Partida 3 (Parte I)

Investigación						
I1	I2	I3	Precio	Subtotal(3)	Subtotal(2)	Total
01						1.445,40 €
	001				96,36 €	
		01	32,12 €	96,36 €		
	002				385,44 €	
		01	32,12 €	385,44 €		
	003				385,44 €	
		01	32,12 €	385,44 €		
	004				289,08 €	
		01	32,12 €	289,08 €		
	005				289,08 €	
		01	32,12 €	289,08 €		
TOTAL						1.445,40 €

Tabla 71. Partida 3 (Parte II)

12.2.2.4. Partida 4

La cuarta partida hace referencia a las tareas de “Apolo”. Nos da un total de 1.638,12 €, se puede ver en la Tabla 72 y en la Tabla 73.

Apolo					
<i>I1</i>	<i>I2</i>	<i>I3</i>	Descripción	Horas	Unidades
01			Capa Modelos		
	001		Preprocesado de los datasets		
		01	Sergio Arroni del Riego (Investigador)	9	horas
	002		Elección de los clasificadores		
		01	Sergio Arroni del Riego (Investigador)	6	horas
	003		Implementación de los clasificadores		
		01	Sergio Arroni del Riego (Investigador)	9	horas
02			Introducción del proyecto		
	001		Implementación de MAB a los clasificadores		
		01	Sergio Arroni del Riego (Investigador)	6	horas
	002		Añadir el muestreo Thomson		
		01	Sergio Arroni del Riego (Investigador)	2	horas
	003		Adaptar algoritmo MAB al código		
		01	Sergio Arroni del Riego (Investigador)	7	horas
03			Introducción del proyecto		
	001		Realización de una clusterización binaria de los clasificadores		
		01	Sergio Arroni del Riego (Investigador)	6	horas
	002		Adaptar al clusterización al código		
		01	Sergio Arroni del Riego (Investigador)	6	horas

Tabla 72. Partida 4 (Parte I)

Apolo						
<i>I1</i>	<i>I2</i>	<i>I3</i>	<i>Precio</i>	<i>Subtotal(3)</i>	<i>Subtotal(2)</i>	<i>Total</i>
01						770,88 €
	001				289,08 €	
		01	32,12 €	289,08 €		
	002				192,72 €	
		01	32,12 €	192,72 €		
	003				289,08 €	
		01	32,12 €	289,08 €		
02						481,80 €
	001				192,72 €	
		01	32,12 €	192,72 €		
	002				64,24 €	
		01	32,12 €	64,24 €		
	003				224,84 €	
		01	32,12 €	224,84 €		
03						385,44 €
	001				192,72 €	
		01	32,12 €	192,72 €		
	002				192,72 €	
		01	32,12 €	192,72 €		
TOTAL						1.638,12 €

Tabla 73. Partida 4 (Parte II)

12.2.2.5. Partida 5

Esta partida hace referencia a las tareas de “Análisis”. Nos da un total de 1.349,04 €, se puede ver en la Tabla 74 y en la Tabla 75.

Análisis					
<i>I1</i>	<i>I2</i>	<i>I3</i>	Descripción	Horas	Unidades
01			Análisis		
	001		Requisitos del sistema		
		01	Sergio Arroni del Riego (Analista)	12	horas
	002		Identificación de los componentes de la arquitectura		
		01	Sergio Arroni del Riego (Analista)	12	horas
	003		Identificación de actores		
		01	Sergio Arroni del Riego (Analista)	3	horas
	004		Realización del diagrama de contexto		
		01	Sergio Arroni del Riego (Analista)	6	horas
	005		Realización de los casos de uso y escenarios		
		01	Sergio Arroni del Riego (Analista)	9	horas

Tabla 74. Partida 5 (Parte I)

Análisis					
<i>I1</i>	<i>I2</i>	<i>I3</i>	Precio	Subtotal(3)	Subtotal(2)
01					1.349,04 €
	001			385,44 €	
		01	32,12 €	385,44 €	
	002				385,44 €
		01	32,12 €	385,44 €	
	003				96,36 €
		01	32,12 €	96,36 €	
	004				192,72 €
		01	32,12 €	192,72 €	
	005				289,08 €
		01	32,12 €	289,08 €	
					TOTAL 1.349,04 €

Tabla 75. Partida 5 (Parte II)

12.2.2.6. Partida 6

Esta partida hace referencia a las tareas de “Definición del sistema”. Nos da un total de 1.321,20 €, se puede ver en la Tabla 76 y en la Tabla 77.

Definición del sistema					
<i>I1</i>	<i>I2</i>	<i>I3</i>	<i>Descripción</i>	<i>Horas</i>	<i>Unidades</i>
01			Definición del sistema		
	001		Buscar las tecnologías y herramientas a utilizar en los componentes		
		01	Sergio Arroni del Riego (Arquitecto Software)	9	horas
	002		Idear la comunicación entre componentes		
		01	Sergio Arroni del Riego (Arquitecto Software)	3	horas
	003		Realizar un estudio de alternativas de las opciones descartadas		
		01	Sergio Arroni del Riego (Arquitecto Software)	6	horas
02			Documentar la definición y alternativas		
	001		Microservicios		
		01	Sergio Arroni del Riego (Arquitecto Software)	2	horas
	002		Herramientas utilizadas		
		01	Sergio Arroni del Riego (Arquitecto Software)	4	horas
	003		Tecnologías utilizadas		
		01	Sergio Arroni del Riego (Arquitecto Software)	4	horas
	004		Lenguajes utilizados		
		01	Sergio Arroni del Riego (Arquitecto Software)	2	horas
	005		Estándares y normas utilizados		
		01	Sergio Arroni del Riego (Arquitecto Software)	1	horas
	006		Componentes utilizados		
		01	Sergio Arroni del Riego (Arquitecto Software)	5	horas

Tabla 76. Partida 6 (Parte I)

Definición del sistema						
I1	I2	I3	Precio	Subtotal(3)	Subtotal(2)	Total
01						660,60 €
	001				330,30 €	
		01	36,70 €	330,30 €		
	002				110,10 €	
		01	36,70 €	110,10 €		
	003				220,20 €	
		01	36,70 €	220,20 €		
02						660,60 €
	001				73,40 €	
		01	36,70 €	73,40 €		
	002				146,80 €	
		01	36,70 €	146,01 €		
	003				146,80 €	
		01	36,70 €	146,80 €		
	004				73,40 €	
		01	36,70 €	73,40 €		
	005				36,70 €	
		01	36,70 €	36,70 €		
	006				183,50 €	
		01	36,70 €	183,50 €		
TOTAL						1.321,20 €

Tabla 77. Partida 6 (Parte II)

12.2.2.7. Partida 7

Esta partida hace referencia a las tareas de “Formación”. Nos da un total de 424,40 €, se puede ver en la Tabla 78 y en la Tabla 79.

Formación					
<i>I1</i>	<i>I2</i>	<i>I3</i>	Descripción	Horas	Unidades
01			Formación		
	001		Formar al personal sobre las tecnologías y herramientas utilizadas		
		01	Sergio Arroni del Riego (Programador)	5	horas
		02	Directores (Jefe de Proyecto)	5	horas

Tabla 78. Partida 7 (Parte I)

Formación						
<i>I1</i>	<i>I2</i>	<i>I3</i>	Precio	Subtotal(3)	Subtotal(2)	Total
01						424,40 €
	001				424,40 €	
		01	27,53 €	137,65 €		
		02	57,35 €	286,75 €		
					TOTAL	424,40 €

Tabla 79. Partida 7 (Parte II)

12.2.2.8. Partida 8

Esta partida hace referencia a las tareas de “Diseño del sistema”. Nos da un total de 268,38 € se puede ver en la Tabla 80 y en la Tabla 81.

Diseño del sistema					
<i>I1</i>	<i>I2</i>	<i>I3</i>	Descripción	Horas	Unidades
01			Diseño del sistema		
	001		Diseño de los diagramas del sistema		
		01	Sergio Arroni del Riego (Diseñador del Software)	6	horas
		002	Diseño de las bases de datos		

		01	Sergio Arroni del Riego (Diseñador del Software)	3	horas
--	--	----	--	---	-------

Tabla 80. Partida 8 (Parte I)

Diseño del sistema						
<i>I1</i>	<i>I2</i>	<i>I3</i>	Precio	Subtotal(3)	Subtotal(2)	Total
01						268,38 €
	001				178,92 €	
		01	29,82 €	178,92 €		
	002				89,46 €	
		01	29,82 €	89,46 €		
					TOTAL	268,38 €

Tabla 81. Partida 8 (Parte II)

12.2.2.9. Partida 9

Esta partida hace referencia a las tareas de “Realización de los componentes”. Nos da un total de 936,02 €, se puede ver en la Tabla 82 y en la Tabla 83.

Realización de los componentes						
<i>I1</i>	<i>I2</i>	<i>I3</i>	Descripción	Horas	Unidades	
01			Realización de los componentes			
	001		Implementar el firewall y proxy reverso			
		01	Sergio Arroni del Riego (Programador)	5	horas	
	002		Búsqueda de una aplicación de ejemplo			
		01	Sergio Arroni del Riego (Programador)	3	horas	
	003		Implementar la gestión de peticiones			
		01	Sergio Arroni del Riego (Programador)	6	horas	
	004		Realizar los manuales del sistema			
		01	Sergio Arroni del Riego (Programador)	5	horas	

Apolo: Sistema inteligente de monitorización y detección de ataques basados en inteligencia artificial

02			Implementar la clasificación de las peticiones		
	001		Implementar el sistema de colas		
		01	Sergio Arroni del Riego (Programador)	4	horas
	002		Implementar Apolo		
		01	Sergio Arroni del Riego (Programador)	6	horas
	003		Implementar la gestión de anomalías		
		01	Sergio Arroni del Riego (Programador)	5	horas

Tabla 82. Partida 9 (Parte I)

Realización de los componentes						
I1	I2	I3	Precio	Subtotal(3)	Subtotal(2)	Total
01						523,07 €
	001				137,65 €	
		01	27,53 €	137,65 €		
	002				82,59 €	
		01	27,53 €	82,59 €		
	003				165,18 €	
		01	27,53 €	165,18 €		
	004				137,65 €	
		01	27,53 €	137,65 €		
02						412,95 €
	001				110,12 €	
		01	27,53 €	110,12 €		
	002				165,18 €	
		01	27,53 €	165,18 €		
	003				137,65 €	
		01	27,53 €	137,65 €		
TOTAL						936,02 €

Tabla 83. Partida 9 (Parte II)

12.2.2.10. Partida 10

Esta partida hace referencia a las tareas de “Evaluación del sistema”. Nos da un total de 1.491,24 €, se puede ver en la Tabla 84 y en la Tabla 85.

Evaluación del sistema					
I1	I2	I3	Descripción	Horas	Unidades
01			Evaluación del sistema		
	001		Elegir las metodologías a utilizar		
		01	Sergio Arroni del Riego (Investigador)	4	horas
	002		Elegir la validación a utilizar		
		01	Sergio Arroni del Riego (Investigador)	2	horas
02			Realizar una experimentación de Apolo en varios escenarios: Primer escenario		
	001		Probar los clasificadores más populares en un entorno tradicional de IDS		
		01	Sergio Arroni del Riego (Investigador)	8	horas
	002		Probar Apolo en un entorno tradicional de IDS		
		01	Sergio Arroni del Riego (Investigador)	6	horas
03			Realizar una experimentación de Apolo en varios escenarios: Segundo escenario		
	001		Probar los clasificadores más populares frente a ataques AML		
		01	Sergio Arroni del Riego (Investigador)	10	horas
	002		Probar Apolo frente a ataques AML		
		01	Sergio Arroni del Riego (Investigador)	9	horas
04			Realizar una experimentación de Apolo en varios escenarios: Tercer escenario		
	001		Generar un dataset		
		01	Sergio Arroni del Riego (Tester)	5	horas
	002		Entrenar y testear Apolo con el dataset generado		
		01	Sergio Arroni del Riego (Tester)	3	horas

Tabla 84. Partida 10 (Parte I)

Evaluación del sistema						
I1	I2	I3	Precio	Subtotal(3)	Subtotal(2)	Total
01						192,72 €
	001				128,48 €	
		01	32,12 €	128,48 €		
	002				64,24 €	
		01	32,12 €	64,24 €		
02						449,68 €
	001				256,96 €	
		01	32,12 €	256,96 €		
	002				192,72 €	
		01	32,12 €	192,72 €		
03						610,28 €
	001				321,20 €	
		01	32,12 €	321,20 €		
	002				289,08 €	
		01	32,12 €	289,08 €		
04						238,56 €
	001				149,10 €	
		01	29,82 €	149,10 €		
	002				89,46 €	
		01	29,82 €	89,46 €		
					TOTAL	1.491,24 €

Tabla 85. Partida 10 (Parte II)

12.2.2.11. Partida 11

Esta partida hace referencia a las tareas de “Realizar la planificación y presupuesto del proyecto”. Nos da un total de 247,77 €, se puede ver en la Tabla 86 y en la Tabla 87.

Realizar la planificación y presupuesto del proyecto					
I1	I2	I3	Descripción	Horas	Unidades
01			Realizar la planificación y presupuesto del proyecto		
	001		Realizar la planificación y presupuesto del proyecto		
		01	Sergio Arroni del Riego (Secretario)	5	horas
	002		Realizar el presupuesto		
		01	Sergio Arroni del Riego (Secretario)	4	horas

Tabla 86. Partida 11 (Parte I)

Realizar la planificación y presupuesto del proyecto					
I1	I2	I3	Precio	Subtotal(3)	Subtotal(2)
01					247,77 €
	001				137,65 €
		01	27,53 €	137,65 €	
02					110,12 €
	001		27,53 €	110,12 €	
					TOTAL 247,77 €

Tabla 87. Partida 11 (Parte II)

12.2.2.12. Partida 12

Esta partida hace referencia a las tareas de “Conclusión”. Nos da un total de 55,06 €, se puede ver en la Tabla 88 y en la Tabla 89.

Conclusión					
I1	I2	I3	Descripción	Horas	Unidades
01			Conclusión		
	001		Realizar la conclusión del proyecto		
		01	Sergio Arroni del Riego (Secretario)	1	horas
	002		Realizar el trabajo futuro del proyecto		
		01	Sergio Arroni del Riego (Secretario)	1	horas

Tabla 88. Partida 12 (Parte I)

Conclusión						
I1	I2	I3	Precio	Subtotal(3)	Subtotal(2)	Total
01						55,06 €
	001				27,53 €	
		01	27,53 €	27,53 €		
	002				27,53 €	
		01	27,53 €	27,53 €		
TOTAL						55,06 €

Tabla 89. Partida 12 (Parte II)

12.2.2.13. Partida 13

Esta partida hace referencia a las tareas de “Realizar los apéndices”. Nos da un total de 110,12 €, se puede ver en la Tabla 90 y en la Tabla 91.

Realizar los apéndices						
I1	I2	I3	Descripción		Horas	Unidades
01			Realizar los apéndices			
	001		Realizar un análisis y gestión de los riesgos del proyecto			
		01	Sergio Arroni del Riego (Secretario)	3		horas
	002		Documentar el apéndice			
		01	Sergio Arroni del Riego (Secretario)	1		horas

Tabla 90. Partida 13 (Parte I)

Realizar los apéndices						
I1	I2	I3	Precio	Subtotal(3)	Subtotal(2)	Total
01						110,12 €
	001				82,59 €	
		01	27,53 €	82,59 €		
	002				27,53 €	
		01	27,53 €	27,53 €		
TOTAL						110,12 €

Tabla 91. Partida 13 (Parte II)

12.2.2.14. Partida 14

Esta partida hace referencia a las tareas de “Desplegar el sistema”. Nos da un total de 247,77 €, se puede ver en la Tabla 92 y en la Tabla 93.

Desplegar el sistema				Horas	Unidades
<i>I1</i>	<i>I2</i>	<i>I3</i>	Descripción		
01			Desplegar el sistema		
	001		Desplegar el sistema		
		01	Sergio Arroni del Riego (Programador)		9 horas

Tabla 92. Partida 14 (Parte I)

Desplegar el sistema						
<i>I1</i>	<i>I2</i>	<i>I3</i>	Precio	Subtotal(3)	Subtotal(2)	Total
01						247,77 €
	001				247,77 €	
		01	27,53 €	247,77 €		
				TOTAL		247,77 €

Tabla 93. Partida 14 (Parte II)

12.2.3. Resúmenes

En esta sección daremos los resúmenes del presupuesto, tanto la versión para nuestra empresa (antes de beneficios y margen), como la versión para el cliente (con los beneficios y margen ya incluidos).

12.2.3.1. Costes (Empresa)

Empezamos con el resumen de la empresa, daremos una versión agregada (Tabla 94).

Hemos tenido en cuenta un beneficio del 25% y un margen del 5%, dicho margen es tanto para hacer frente a los más que seguros inconvenientes que ocurrán, como para tener un margen de maniobra en caso de que ocurra algún contratiempo.

Nos da un total de 19.814,51 € a facturar durante el proyecto.

Presupuesto de costes (Agregado)	
Reuniones	4.488,58 €
Introducción del proyecto	1.073,67 €
Investigación	1.445,40 €
Apolo	1.638,12 €
Análisis	1.349,04 €
Definición del sistema	1.321,20 €
Formación	424,40 €
Diseño del sistema	268,38 €
Realización de los componentes	936,02 €
Evaluación del sistema	1.491,24 €
Realizar la planificación y presupuesto del proyecto	247,77 €
Conclusión	55,06 €
Realizar los apéndices	110,12 €
Desplegar el sistema	247,77 €
Coste sin beneficio	15.096,77 €
Margen de beneficio	25%
Beneficio	3.774,19 €
Coste con beneficio	18.870,96 €
Margen de facturación	5%
Margen	943,55 €
Total a Facturar	19.814,51 €

Tabla 94. Presupuesto Empresa

12.2.3.2. Costes (Cliente)

Para finalizar, tenemos el resumen del cliente, para la cual también daremos una versión agregada (Tabla 95). El proyecto tiene un 21% de IVA.

Nos da un total de 23.975,56 € a facturar durante el proyecto.

Resumen del Presupuesto del Cliente	
Reuniones	5.891,26 €
Introducción del proyecto	1.409,19 €
Investigación	1.897,09 €
Apolo	2.150,03 €
Análisis	1.770,62 €
Definición del sistema	1.734,08 €
Formación	557,03 €
Diseño del sistema	352,25 €
Realización de los componentes	1.228,53 €
Evaluación del sistema	1.957,25 €
Realizar la planificación y presupuesto del proyecto	325,20 €
Conclusión	72,27 €
Realizar los apéndices	144,53 €
Desplegar el sistema	325,20 €
Presupuesto estimado antes de impuestos	19.814,51 €
Porcentaje de IVA aplicado	21%
IVA	4.161,05 €
Presupuesto estimado después de impuestos	23.975,56 €

Tabla 95. Presupuesto Cliente

Capítulo 13. Referencias Bibliográficas

- [1] A. S. Ashoor and S. Gore, "Importance of intrusion detection system (IDS)", *Int J Sci Eng Res*, vol. 2, no. 1, pp. 1–4, 2011.
- [2] H.-J. Liao, C.-H. R. Lin, Y.-C. Lin, and K.-Y. Tung, "Intrusion detection system: A comprehensive review", *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 16–24, 2013.
- [3] R. Vinayakumar, K. P. Soman, and P. Poornachandran, "Evaluation of recurrent neural network and its variants for intrusion detection system (IDS)", *International Journal of Information System Modeling and Design (IJISMD)*, vol. 8, no. 3, pp. 43–63, 2017.
- [4] H. Jmila and M. I. Khedher, "Adversarial machine learning for network intrusion detection: A comparative study", *Computer Networks*, vol. 214, p. 109073, 2022.
- [5] S. Qiu, Q. Liu, S. Zhou, and C. Wu, "Review of Artificial Intelligence Adversarial Attack and Defense Technologies", *Applied Sciences*, vol. 9, no. 5, 2019, doi: 10.3390/app9050909.
- [6] Y. He, G. Meng, K. Chen, X. Hu, and J. He, "Towards Security Threats of Deep Learning Systems: A Survey". 2020.
- [7] J. Shroff, R. Walambe, S. K. Singh, and K. Kotecha, "Enhanced security against volumetric DDoS attacks using adversarial machine learning", *Wirel Commun Mob Comput*, vol. 2022, pp. 1–10, 2022.
- [8] H. Liu and B. Lang, "Machine learning and deep learning methods for intrusion detection systems: A survey", *applied sciences*, vol. 9, no. 20, p. 4396, 2019.
- [9] Z. Lin, Y. Shi, and Z. Xue, "Idsgan: Generative adversarial networks for attack generation against intrusion detection", in *Advances in Knowledge*

Discovery and Data Mining: 26th Pacific-Asia Conference, PAKDD 2022, Chengdu, China, May 16–19, 2022, Proceedings, Part III, 2022, pp. 79–91.

- [10] L. Chen, C. Tang, J. He, H. Zhao, X. Lan, and T. Li, "XSS adversarial example attacks based on deep reinforcement learning", *Comput Secur*, vol. 120, p. 102831, Sep. 2022, doi: 10.1016/J.COSE.2022.102831.
- [11] Y. Fang, Y. Li, L. Liu, and C. Huang, "DeepXSS: Cross site scripting detection based on deep learning", *ACM International Conference Proceeding Series*, pp. 47–51, Mar. 2018, doi: 10.1145/3194452.3194469.
- [12] G. Liu, W. Zhang, X. Li, K. Fan, and S. Yu, "VulnerGAN: a backdoor attack through vulnerability amplification against machine learning-based network intrusion detection systems", *Science China Information Sciences*, vol. 65, no. 7, p. 170303, 2022.
- [13] P. Whittle, "Multi-armed bandits and the Gittins index", *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 42, no. 2, pp. 143–149, 1980.
- [14] S. Agrawal, V. Avadhanula, V. Goyal, and A. Zeevi, "Thompson sampling for the mnl-bandit", in *Conference on learning theory*, 2017, pp. 76–78.
- [15] G. W. Milligan and M. C. Cooper, "Methodology review: Clustering methods", *Appl Psychol Meas*, vol. 11, no. 4, pp. 329–354, 1987.
- [16] A. Moubayed, A. Refaey, and A. Shami, "Software-defined perimeter (sdp): State of the art secure solution for modern networks", *IEEE Netw*, vol. 33, no. 5, pp. 226–233, 2019.
- [17] A. H. Lashkari, Y. Zang, G. Owhuo, M. S. I. Mamun, and G. D. Gil, "CICFlowMeter", *GitHub*. [vid. 2021-08-10]. Dostupné z: <https://github.com/ahlashkari/CICFlowMeter/blob/master/ReadMe.txt>, 2017.
- [18] F. Hao, T. V. Lakshman, S. Mukherjee, and H. Song, "Secure Cloud Computing with a Virtualized Network Infrastructure.", in *HotCloud*, 2010.
- [19] M. Kaeo, *Designing network security*. Cisco Press, 2004.
- [20] K. K. Jyothi and B. I. Reddy, "Study on virtual private network (VPN), VPN's protocols and security", *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, vol. 3, no. 5, pp. 919–932, 2018.

- [21] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization.", *ICISSP*, vol. 1, pp. 108–116, 2018.
- [22] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A survey of network-based intrusion detection data sets", *Comput Secur*, vol. 86, pp. 147–167, 2019.
- [23] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization.", *ICISSP*, vol. 1, pp. 108–116, 2018.
- [24] M. Pujari, Y. Pacheco, B. Cherukuri, and W. Sun, "A Comparative Study on the Impact of Adversarial Machine Learning Attacks on Contemporary Intrusion Detection Datasets", *SN Comput Sci*, vol. 3, no. 5, p. 412, 2022.
- [25] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy", in *2019 International Carnahan Conference on Security Technology (ICCST)*, 2019, pp. 1–8.
- [26] M. V Mahoney and P. K. Chan, "An analysis of the 1999 DARPA/Lincoln Laboratory evaluation data for network anomaly detection", in *Recent Advances in Intrusion Detection: 6th International Symposium, RAID 2003, Pittsburgh, PA, USA, September 8-10, 2003. Proceedings* 6, 2003, pp. 220–237.
- [27] M. Tavallaei, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set", in *2009 IEEE symposium on computational intelligence for security and defense applications*, 2009, pp. 1–6.
- [28] J. McHugh, "Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory", *ACM Transactions on Information and System Security (TISSEC)*, vol. 3, no. 4, pp. 262–294, 2000.
- [29] A. M. Al Tobi and I. Duncan, "KDD 1999 generation faults: A review and analysis", *Journal of Cyber Security Technology*, vol. 2, no. 3–4, pp. 164–200, 2018.
- [30] L. Breiman, "Random forests", *Mach Learn*, vol. 45, pp. 5–32, 2001.
- [31] J. R. Quinlan, "Decision trees and decision-making", *IEEE Trans Syst Man Cybren*, vol. 20, no. 2, pp. 339–346, 1990.

- [32] D. Meyer and F. T. Wien, "Support vector machines", *The Interface to libsvm in package e1071*, vol. 28, p. 20, 2015.
- [33] S. Huang and K. Lei, "IGAN-IDS: An imbalanced generative adversarial network towards intrusion detection system in ad-hoc networks", *Ad Hoc Networks*, vol. 105, p. 102177, 2020.
- [34] Z. K. Maseer, R. Yusof, N. Bahaman, S. A. Mostafa, and C. F. M. Foozy, "Benchmarking of machine learning for anomaly based intrusion detection systems in the CICIDS2017 dataset", *IEEE access*, vol. 9, pp. 22351–22370, 2021.
- [35] Z. Wu, H. Zhang, P. Wang, and Z. Sun, "RTIDS: A robust transformer-based approach for intrusion detection system", *IEEE Access*, vol. 10, pp. 64375–64387, 2022.
- [36] A. Rosay, K. Riou, F. Carlier, and P. Leroux, "Multi-layer perceptron for network intrusion detection: From a study on two recent data sets to deployment on automotive processor", *Annals of Telecommunications*, vol. 77, no. 5–6, pp. 371–394, 2022.
- [37] R. Abdulhammed, M. Faezipour, H. Musafer, and A. Abuzneid, "Efficient network intrusion detection using pca-based dimensionality reduction of features", in *2019 International symposium on networks, computers and communications (ISNCC)*, 2019, pp. 1–6.
- [38] O. Faker and E. Dogdu, "Intrusion detection using big data and deep learning techniques", in *Proceedings of the 2019 ACM Southeast conference*, 2019, pp. 86–93.
- [39] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples", *arXiv preprint arXiv:1412.6572*, 2014.
- [40] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2574–2582.
- [41] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks", in *2017 ieee symposium on security and privacy (sp)*, 2017, pp. 39–57.

- [42] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings", in *2016 IEEE European symposium on security and privacy (EuroS&P)*, 2016, pp. 372–387.
- [43] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial machine learning at scale", *arXiv preprint arXiv:1611.01236*, 2016.
- [44] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks", *arXiv preprint arXiv:1706.06083*, 2017.
- [45] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative adversarial networks: An overview", *IEEE Signal Process Mag*, vol. 35, no. 1, pp. 53–65, 2018.
- [46] S. Zhao, J. Li, J. Wang, Z. Zhang, L. Zhu, and Y. Zhang, "attackgan: Adversarial attack against black-box ids using generative adversarial networks", *Procedia Comput Sci*, vol. 187, pp. 128–133, 2021.
- [47] P. T. Duy, N. H. Khoa, A. G.-T. Nguyen, V.-H. Pham, and others, "DIGFuPAS: Deceive IDS with GAN and Function-Preserving on Adversarial Samples in SDN-enabled networks", *Comput Secur*, vol. 109, p. 102367, 2021.
- [48] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh, "Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models", in *Proceedings of the 10th ACM workshop on artificial intelligence and security*, 2017, pp. 15–26.
- [49] W. Brendel, J. Rauber, and M. Bethge, "Decision-based adversarial attacks: Reliable attacks against black-box machine learning models", *arXiv preprint arXiv:1712.04248*, 2017.
- [50] J. Chen and M. I. Jordan, "Boundary attack++: Query-efficient decision-based adversarial attack", *arXiv preprint arXiv:1904.02144*, vol. 2, no. 7, 2019.
- [51] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans", *Adv Neural Inf Process Syst*, vol. 30, 2017.
- [52] M. Usama, M. Asim, S. Latif, J. Qadir, and others, "Generative adversarial networks for launching and thwarting adversarial attacks on network intrusion detection systems", in *2019 15th international wireless*

- communications & mobile computing conference (IWCMC), 2019, pp. 78–83.
- [53] M.-I. Nicolae *et al.*, “Adversarial Robustness Toolbox v1.2.0”, *CoRR*, vol. 1807.01069, 2018, [Online]. Available: <https://arxiv.org/pdf/1807.01069.pdf>
 - [54] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, “Membership inference attacks against machine learning models”, in *2017 IEEE symposium on security and privacy (SP)*, 2017, pp. 3–18.
 - [55] A. Salem, Y. Zhang, M. Humbert, P. Berrang, M. Fritz, and M. Backes, “MI-leaks: Model and data independent membership inference attacks and defenses on machine learning models”, *arXiv preprint arXiv:1806.01246*, 2018.
 - [56] G. Ateniese, L. V Mancini, A. Spognardi, A. Villani, D. Vitali, and G. Felici, “Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers”, *International Journal of Security and Networks*, vol. 10, no. 3, pp. 137–150, 2015.
 - [57] K. Ganju, Q. Wang, W. Yang, C. A. Gunter, and N. Borisov, “Property inference attacks on fully connected neural networks using permutation invariant representations”, in *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, 2018, pp. 619–633.
 - [58] V. Kuleshov and D. Precup, “Algorithms for multi-armed bandit problems”, *arXiv preprint arXiv:1402.6028*, 2014.
 - [59] M. C. Machado, S. Srinivasan, and M. Bowling, “Domain-independent optimistic initialization for reinforcement learning”, *arXiv preprint arXiv:1410.4604*, 2014.
 - [60] A. Carpentier, A. Lazaric, M. Ghavamzadeh, R. Munos, and P. Auer, “Upper-confidence-bound algorithms for active learning in multi-armed bandits”, in *Algorithmic Learning Theory: 22nd International Conference, ALT 2011, Espoo, Finland, October 5-7, 2011. Proceedings* 22, 2011, pp. 189–203.
 - [61] S. Agrawal and N. Goyal, “Analysis of thompson sampling for the multi-armed bandit problem”, in *Conference on learning theory*, 2012, pp. 31–39.

- [62] H. Park and M. K. S. Faradonbeh, "Analysis of Thompson sampling for partially observable contextual multi-armed bandits", *IEEE Control Syst Lett*, vol. 6, pp. 2150–2155, 2021.
- [63] D. Ramos, P. Faria, L. Gomes, P. Campos, and Z. Vale, "Selection of features in reinforcement learning applied to energy consumption forecast in buildings according to different contexts", *Energy Reports*, vol. 8, pp. 423–429, 2022.
- [64] D. Ramos, P. Faria, L. Gomes, P. Campos, and Z. Vale, "A Learning Approach to Improve the Selection of Forecasting Algorithms in an Office Building in Different Contexts", in *Progress in Artificial Intelligence: 21st EPIA Conference on Artificial Intelligence, EPIA 2022, Lisbon, Portugal, August 31–September 2, 2022, Proceedings*, 2022, pp. 271–281.
- [65] R. Wright, "Interpreting black-box machine learning models using partial dependence and individual conditional expectation plots", *Exploring SAS® Enterprise Miner Special Collection*, vol. 2018, 1950.
- [66] I. Rish and others, "An empirical study of the naive Bayes classifier", in *IJCAI 2001 workshop on empirical methods in artificial intelligence*, 2001, pp. 41–46.
- [67] F. Murtagh, "Multilayer perceptrons for classification and regression", *Neurocomputing*, vol. 2, no. 5–6, pp. 183–197, 1991.
- [68] R. E. Wright, "Logistic regression.", 1995.
- [69] C.-W. Hsu, C.-C. Chang, C.-J. Lin, and others, "A practical guide to support vector classification". Taipei, Taiwan, 2003.
- [70] Y. Liao and V. R. Vemuri, "Use of k-nearest neighbor classifier for intrusion detection", *Comput Secur*, vol. 21, no. 5, pp. 439–448, 2002.
- [71] A. Dyulgerov, "Study of clustering methods of Linac outages: K-Means vs GMM", 1900.
- [72] Project Management Institute, *A guide to the Project Management Body of Knowledge (PMBOK guide)*, 6th ed. Newton Square, PA: Project Management Institute, 2017.
- [73] E. Pontes and A. E. Guelfi, "IFS—Intrusion forecasting system based on collaborative architecture", in *2009 Fourth International Conference on Digital Information Management*, 2009, pp. 1–6.

- [74] H. A. Kholidy, A. Erradi, S. Abdelwahed, and F. Baiardi, "A hierarchical, autonomous, and forecasting cloud IDS", in *2013 5th International Conference on Modelling, Identification and Control (ICMIC)*, 2013, pp. 213–220.
- [75] Z. Wu, H. Zhang, P. Wang, and Z. Sun, "RTIDS: A robust transformer-based approach for intrusion detection system", *IEEE Access*, vol. 10, pp. 64375–64387, 2022.
- [76] C. Zhang, X. Costa-Perez, and P. Patras, "Adversarial attacks against deep learning-based network intrusion detection systems and defense mechanisms", *IEEE/ACM Transactions on Networking*, vol. 30, no. 3, pp. 1294–1311, 2022.
- [77] A. Likas, N. Vlassis, and J. J. Verbeek, "The global k-means clustering algorithm", *Pattern Recognit*, vol. 36, no. 2, pp. 451–461, 2003.
- [78] J. Chen, M. I. Jordan, and M. J. Wainwright, "Hopskipjumpattack: A query-efficient decision-based attack", in *2020 ieee symposium on security and privacy (sp)*, 2020, pp. 1277–1294.

Apéndices

Este capítulo ofrecerá el análisis y la gestión de riesgos, el glosario de términos utilizados a lo largo del documento describirá la estructura del archivo adjunto y mostrará un índice alfabético del proyecto.



A.1. Artículos

En esta sección adjuntaremos los artículos enviados a las revistas, explicaremos a qué revistas han sido enviados cada artículo y en qué estado están. Posteriormente adjuntaremos en el contenido entregado los artículos enviados.

A.1.1. Harpe

El artículo titulado "**Harpe: Using Adversarial Machine Learning to dodge Intrusion Detection Systems in Software-Defined Perimeters**" se realizó con el objetivo de demostrar la vulnerabilidad de los *Sistemas de Detección de Intrusos (IDS)* frente a los ataques de *AML*, incluso en arquitecturas de red altamente orientadas a la ciberseguridad.

En este artículo se ha desarrollado un nuevo ataque de caja gris/negra que utiliza una red adversaria generativa de Wasserstein. Esta red es capaz de "aprender" del comportamiento de un *IDS*, imitar su comportamiento y generar tráfico red capaz de engañarlo.

A.1.1.1. Abstract

With the increasing reliance on Software-Defined Perimeters (SDPs) for network security, it is important to understand the limitations and vulnerabilities of this architecture. This paper proposes a framework, Harpe, that generates attacks on SDP-based networks to demonstrate the potential weaknesses of this architecture. Harpe generates malicious network traffic that mimics normal traffic to evade the Intrusion Detection System (IDS) component of SDPs. Utilizing Adversarial Machine Learning techniques, it learns how the IDS operates and finds ways to evade detection. The effectiveness of the proposed framework is tested in two different environments, a traditional network architecture and an SDP-based network. The results show that Harpe is capable of significantly reducing the detection capabilities of the IDS in both environments, highlighting the need for further research and improvement of SDP security mechanisms.

A.1.1.2. Revista

El artículo se ha enviado a la revista “**Journal of Systems Architecture**”⁶⁶ (Ilustración 149). Esta revista se encuentra indexada y sus métricas se pueden obtener a través del *Journal Citation Report* (JCR) online.

La revista se encuentra incluida dentro de dos categorías. A continuación, se detallarán dichas categorías junto con su factor de impacto en estas (extraído mediante la herramienta de la Fundación Española para la Ciencia y la Tecnología (FECYT)⁶⁷ para el año 2021):

- COMPUTER SCIENCE, HARDWARE & ARCHITECTURE:
 - Factor de Impacto: 5,836
 - Cuartil: Q1
- COMPUTER SCIENCE, SOFTWARE ENGINEERING
 - Factor de Impacto: 5,836
 - Cuartil: Q1

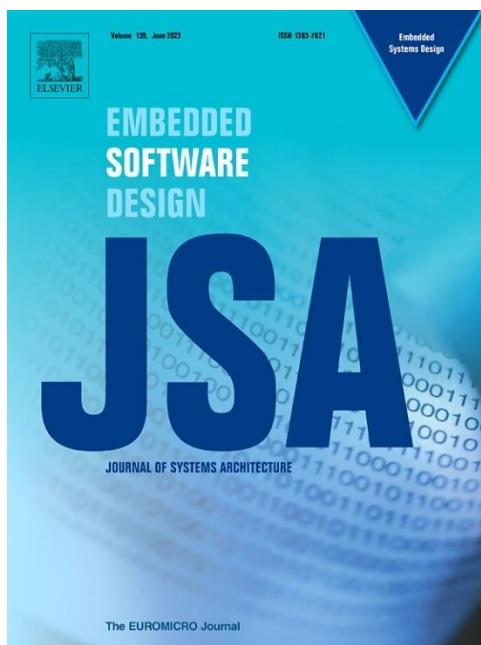


Ilustración 149. Revista del artículo Harpe: *Journal of Systems Architecture*

⁶⁶ <https://www.sciencedirect.com/journal/journal-of-systems-architecture>

⁶⁷ <https://www.recursoscientificos.fecyt.es/factor/>

A.1.1.3. Estado actual

El artículo se encuentra actualmente en el estado “*Under review*”, a la espera de la respuesta por parte de los editores de la revista.

A.1.2. Apollon

El artículo titulado “**Apollon: A Robust Defence System against Adversarial Machine Learning Attacks in Intrusion Detection Systems**” recopila la aportación científica de este proyecto y se compara su rendimiento tanto en conjuntos de datos tradicionales, como contra ataques generados por los más recientes ataques de AML.

A.1.2.1. Abstract

The rise of Adversarial Machine Learning (AML) attacks is presenting a significant challenge to Intrusion Detection Systems (IDS) and their ability to detect threats. To address this issue, we introduce Apollon, a novel defence system that can protect IDS against AML attacks. Apollon utilizes a diverse set of classifiers to identify intrusions and employs Multi-Armed Bandits (MAB) with Thompson sampling to dynamically select the optimal classifier or ensemble of classifiers for each input. This approach enables Apollon to prevent attackers from learning the IDS behaviour and generating adversarial examples that can evade the IDS detection. We evaluate Apollon on several of the most popular and recent datasets, and show that it can successfully detect attacks without compromising its performance on traditional network traffic. Our results suggest that Apollon is a robust defence system against AML attacks in IDS.

A.1.1.2. Revista

El artículo se ha enviado a la revista “**Computers & Security**”⁶⁸ (Ilustración 150). Esta revista se encuentra indexada y sus métricas se pueden obtener a través del *Journal Citation Report (JCR)* online.

La revista se encuentra incluida dentro de dos categorías. A continuación, se detallarán dichas categorías junto con su factor de impacto en estas (extraído

⁶⁸ <https://www.sciencedirect.com/journal/computers-and-security>

mediante la herramienta de la Fundación Española para la Ciencia y la Tecnología (FECYT) ⁶⁷ para el año 2021):

- COMPUTER SCIENCE, HARDWARE & ARCHITECTURE:
 - Factor de Impacto: 5,105
 - Cuartil: Q1



Ilustración 150. Revista del artículo Apollon: Computers & Security

A.1.1.3. Estado actual

El artículo ya ha pasado la primera etapa de revisión, habiendo recibido “**Moderate revisions**”. Tras recibir estas revisiones, se han solucionado los comentarios propuestos por los revisores y se ha enviado el artículo actualizado a la revista.

El artículo se encuentra actualmente en el estado “*Under review*”, a la espera de la respuesta por parte de los editores de la revista.

A.2. Análisis y Gestión de los Riesgos

En este apartado daremos los diferentes riesgos que hemos detectado para nuestro proyecto, en primera instancia, realizaremos un análisis de los mismos, posteriormente vendrá seguido el correspondiente plan de gestión de riesgos.

A.2.1. Metodología

Para el cálculo del impacto se ha seguido la matriz dada por el PMBOK para el cálculo de la estimación del impacto de los riesgos.

A.2.1.1. Conceptos Generales

Con el fin de evitar la ambigüedad, se deben definir tres conceptos utilizados en la Gestión de Riesgos:

- **Riesgo:** es cualquier evento que pueda causar un efecto en el proyecto. El efecto bien podría ser una amenaza o podría ser una oportunidad. La primera de ellas debe ser evitada y la segunda debe ser explotada.
- **Factor de riesgo:** Los factores de riesgo son circunstancias asociadas con el riesgo de que aumenta la posibilidad de conseguir los efectos descritos por el riesgo. Por lo general, los riesgos son imposibles o difíciles de medir o controlar directamente y las decisiones se toman controlando los factores de riesgo con el fin de evitar o minimizar (o potenciar) los efectos de riesgo.
- **Indicadores de Riesgo:** Los indicadores de riesgo son elementos asociados al riesgo y / o sus factores, que se puede medir y sirve como información acerca de la posibilidad de que se produzca el efecto del riesgo.

A.2.1.2. Categorización

A continuación, se definen las diferentes categorías en las que los riesgos puede ser clasificados. Un riesgo puede pertenecer a una o más de las siguientes categorías (solo las subcategorías se utilizan para clasificar riesgos).

1. Técnico

- a. Requisitos
 - b. Tecnología
 - c. Complejidad e Interfaces
 - d. Prestaciones y fiabilidad
 - e. Calidad
- 2.** Externo
- a. Subcontratistas y Proveedores
 - b. Regulación
 - c. Mercado
 - d. Usuario
 - e. Tiempo
- 3.** Organizacional
- a. Dependencias del Proyecto
 - b. Recursos
 - c. Financiación
 - d. Priorización
- 4.** Gestión de Proyecto
- a. Estimación
 - b. Planificación
 - c. Control
 - d. Comunicación

A.2.1.3. Estimación de la Probabilidad e Impacto

El proceso de estimación de la probabilidad e impacto de los riesgos del proyecto implica la evaluación de cada riesgo identificado en términos de su probabilidad de ocurrencia y el grado de impacto que tendría en los objetivos del proyecto si se materializa. Para este propósito, se han establecido dos parámetros fundamentales: la probabilidad y el impacto.

Para la probabilidad, hemos establecido una escala de cinco niveles: Muy Bajo, Bajo, Medio, Alto, Muy Alto. Cada nivel representa la probabilidad de que ocurra un riesgo determinado, siendo “Muy Bajo” para riesgos con muy poca probabilidad de ocurrencia y “Muy Alto” para riesgos con una probabilidad extremadamente alta de ocurrencia.

En cuanto al impacto, también se han definido cinco categorías: Mínimo, Bajo, Medio, Alto, Crítico. Estas categorías representan la gravedad del impacto que un riesgo particular tendría sobre el proyecto si se produjera. “Mínimo” representa un impacto que tendría poco o ningún efecto en los resultados del

proyecto, mientras que “Crítico” indica un impacto que podría poner en peligro el éxito completo del proyecto. Este nivel de gravedad se mide individualmente para cada uno de los siguientes ámbitos: “Presupuesto”, “Planificación”, “Alcance” y “Calidad”, tal y como se ve en la Tabla 96.

Para ayudar en la evaluación de los riesgos, hemos desarrollado una matriz de probabilidad e impacto, presentada en la Tabla 97. En esta matriz, la probabilidad de ocurrencia de un riesgo se cruza con la severidad de su impacto para producir una puntuación de riesgo. El riesgo con la mayor puntuación determinará la prioridad de gestión de riesgos.

Por ejemplo, un riesgo con una probabilidad “Alta” y un impacto “Crítico” tendría una puntuación de riesgo muy alta, lo que indicaría que requiere una atención inmediata y probablemente necesitará un plan de contingencia detallado.

Hemos adoptado también una escala de colores para visualizar y comunicar el nivel de riesgo asociado a cada probabilidad e impacto, como se muestra en la Tabla 98. Los riesgos con una puntuación más alta se destacan en rojo, mientras que los riesgos con una puntuación más baja se muestran en verde. Esta escala de colores ayuda a priorizar los riesgos para su gestión y a identificar rápidamente los riesgos más críticos que requieren la mayor atención.

Esta metodología de estimación de la probabilidad e impacto permite una gestión de riesgos más efectiva, ya que proporciona una representación visual clara de los riesgos y su importancia relativa en el proyecto **Apolo**.

Impacto			
Presupuesto	Planificación	Alcance	Calidad

Tabla 96. Los 4 impactos medibles.

Probabilidad	Muy Alta	0,90	0,05	0,14	0,27	0,50	0,81
	Alta	0,70	0,04	0,11	0,21	0,39	0,63
	Media	0,50	0,03	0,08	0,15	0,28	0,45
	Baja	0,30	0,02	0,05	0,09	0,17	0,27
	Muy Baja	0,10	0,01	0,02	0,03	0,06	0,09
		0,05	0,15	0,30	0,55	0,90	
		Inapreciable	Bajo	Medio	Alto	Crítico	
Impacto							

Tabla 97. Tabla de cálculo de estimación del impacto de los riesgos.

	\geq	<
Verde	0,00	0,05
Amarillo	0,05	0,15
Rojo	0,15	1,00
Rangos de color		

Tabla 98. Rangos de color para el impacto de los riesgos.

A.2.1.4. Estrategias para la Gestión de Riesgos

En el contexto de la gestión de riesgos, la guía del PMBOK destaca cuatro estrategias principales para lidiar con los riesgos. Aquí están las cuatro estrategias:

- Eliminar el riesgo (Evitar en PMBOK): La estrategia de eliminación, o evitación de riesgos, implica cambiar los planes del proyecto de tal manera que se evita completamente el riesgo. Por ejemplo, si existe un riesgo asociado con un determinado proveedor o tecnología, puedes optar por usar un proveedor o tecnología diferente.
- Mitigar el riesgo (Mitigar en PMBOK): La estrategia de mitigación implica reducir la probabilidad o el impacto de un riesgo a un nivel aceptable. Esto podría involucrar el uso de tecnologías probadas en lugar de

experimentales, la capacitación de empleados, la adopción de procesos de mejoras de calidad, entre otros.

- Asumir el riesgo (Aceptar en PMBOK): La estrategia de asunción, o aceptación de riesgos, implica reconocer el riesgo, pero decidir no tomar ninguna acción a menos que ocurra el riesgo. Esta estrategia es a menudo utilizada para riesgos que tienen un impacto bajo o que son poco probables. La aceptación puede ser activa, lo que implica establecer un plan de contingencia, o pasiva, donde no se hacen preparativos específicos.
- Transferir el riesgo (Transferir en PMBOK): La estrategia de transferencia implica hacer que otra parte asuma el riesgo. Esto no elimina el riesgo, pero sí aleja el riesgo del proyecto. Ejemplos de esto incluyen la contratación de seguros, la subcontratación de un área de trabajo, o el uso de garantías y acuerdos de nivel de servicio.

Estas estrategias pueden ser aplicadas a cualquier riesgo que el proyecto pueda enfrentar, y la elección de estrategia dependerá de la naturaleza del riesgo y de la capacidad del personal del proyecto para manejar ese riesgo.

A.2.2. Análisis de Riesgos

En esta sección detallaremos los riesgos encontrados, así como un análisis de cada riesgo, el cual contará con una descripción de este, su impacto estimado y los parámetros que se han tenido en cuenta para este impacto.

A.2.2.1. Ciberataque al sistema

Descripción del Riesgo: Aunque el sistema ha sido diseñado con mecanismos de defensa contra el *Adversarial Machine Learning (AML)* existen muchos otros tipos de ciberataques que podrían afectar seriamente el sistema. Estos pueden incluir, entre otros, ataques de *malware*, *ransomware*, *phishing*, ataques de fuerza bruta, y explotación de vulnerabilidades en el software.

Un ataque exitoso puede comprometer la integridad y confidencialidad de los datos gestionados por el sistema.

El impacto de un ataque no se limita a la interrupción inmediata de las operaciones del sistema.

Apolo: Sistema inteligente de monitorización y detección de ataques basados en inteligencia artificial

Puede dañar la reputación del proyecto y socavar la confianza de los usuarios en la seguridad y fiabilidad del sistema, con consecuencias a largo plazo para la adopción y el uso exitoso del sistema.

Categoría del Riesgo: Prestaciones y Fiabilidad

Probabilidad de Ocurrencia: Media

Impactos Estimados:

- **Impacto al Presupuesto:** Medio - Un ciberataque exitoso puede implicar costos significativos para la corrección de fallas, reparación de daños y posible recuperación de datos. Esto puede tener un impacto medio en el presupuesto del proyecto, dependiendo de la gravedad del ataque.
- **Impacto a la Planificación:** Bajo – Un ciberataque al sistema no conllevaría mucho retraso en el proyecto.
- **Impacto al Alcance:** Crítico - Un ciberataque puede afectar drásticamente la funcionalidad y la seguridad del sistema, alterando el alcance de las prestaciones del proyecto y requiriendo potencialmente una revisión significativa de los objetivos y entregables.
- **Impacto a la Calidad:** Alto - El sistema, si se ve comprometido, podría sufrir una disminución en la calidad de su rendimiento y en la confianza de los usuarios en la seguridad de su información.

Impacto Total del Riesgo: 0.45

Tabla: Tabla 99

Probabilidad	Impacto				Impacto
	Presup.	Planific.	Alcance	Calidad	
Media	Medio	Bajo	Crítico	Alto	0,45

Tabla 99. Análisis de Riesgos: Ciberataque al sistema.

A.2.2.2. Rechazo de los Artículos Realizados

Descripción del Riesgo: El proyecto se ve afectado por la aceptación y publicación de dos artículos científicos enviados a revistas de renombre. Estos artículos no solo proporcionan una base científica para el trabajo del proyecto, sino que también contribuyen a su credibilidad y reconocimiento en la comunidad científica y docente. Un rechazo de estos artículos, en consecuencia, podría debilitar seriamente la base del proyecto.

Este riesgo es amplificado por la incertidumbre inherente al proceso de revisión, el cual puede ser influenciado por una variedad de factores.

Categoría del Riesgo: Calidad

Probabilidad de Ocurrencia: Alta

Impactos Estimados:

- **Impacto al Presupuesto:** Medio - Un rechazo podría generar costos adicionales para la revisión y reenvío de los artículos, estos costes son notables, ya que a veces lleva bastante tiempo.
- **Impacto a la Planificación:** Medio - Los retrasos en la publicación podrían tener un impacto en la cronología del proyecto, puesto que en algunas ocasiones hay que cambiar muchas partes del documento.
- **Impacto al Alcance:** Medio - La no aceptación de los artículos puede disminuir la relevancia y el impacto del proyecto en la comunidad a la que va dirigido.
- **Impacto a la Calidad:** Alto - El rechazo de los artículos tiene un impacto directo en la percepción de la calidad del proyecto. Si los resultados no son considerados válidos por los revisores, puede haber consecuencias para la reputación del proyecto y de los investigadores involucrados.

Impacto Total del Riesgo: 0.39

Tabla: Tabla 100

Probabilidad	Impacto				Impacto
	Presup.	Planific.	Alcance	Calidad	
Alta	Medio	Medio	Medio	Alto	0,39

Tabla 100. Análisis de Riesgos: Rechazo de los Artículos Realizados.

A.2.2.3. Retraso en la Entrega

Descripción del Riesgo: Aunque estamos trabajando con un plan detallado y un equipo competente, siempre existe la posibilidad de problemas inesperados que puedan retrasar el desarrollo y lanzamiento de la aplicación.

Categoría del Riesgo: Planificación

Probabilidad de Ocurrencia: Baja

Impactos Estimados:

- **Impacto al Presupuesto:** Alto - Los retrasos en el desarrollo y el lanzamiento pueden conducir a un aumento de los costos. Esto es debido a la necesidad de tiempo extra de desarrollo, costos laborales adicionales, y costos asociados a la pérdida de oportunidades.
- **Impacto a la Planificación:** Crítico - Un retraso en la entrega tendría un impacto crítico en la planificación del proyecto, al alterar las fechas de entrega y posiblemente afectar a otros hitos y dependencias del proyecto.
- **Impacto al Alcance:** Medio - Dependiendo de la naturaleza y duración del retraso, podría ser necesario reducir o alterar el alcance del proyecto para cumplir con las restricciones de tiempo o presupuesto.
- **Impacto a la Calidad:** Medio - Si el retraso resulta en una compresión del calendario del proyecto, podrían surgir compromisos de calidad para cumplir con los plazos.

Impacto Total del Riesgo: 0.27

Tabla: Tabla 101

Probabilidad	Impacto				Impacto
	Presup.	Planific.	Alcance	Calidad	
Baja	Alto	Crítico	Medio	Medio	0,27

Tabla 101. Análisis de Riesgos: Retraso en el Entrega.

A.2.2.4. Cálculo Erróneo del Presupuesto

Descripción del Riesgo: Aunque se han realizado estimaciones de costos cuidadosas, siempre existe el riesgo de que el presupuesto del proyecto haya sido mal calculado o que surjan costos inesperados. Un presupuesto insuficiente puede llevar a retrasos en la ejecución del proyecto y puede requerir la reducción del alcance del proyecto. A pesar de que la calidad del proyecto es una prioridad, podrían surgir compromisos si se hace necesario reducir costos.

Categoría del Riesgo: Financiación

Probabilidad de Ocurrencia: Baja

Impactos Estimados:

- **Impacto al Presupuesto:** Crítico - Si el presupuesto ha sido mal calculado o si los costos superan las estimaciones originales, esto puede tener un impacto crítico en la capacidad del proyecto para mantenerse dentro del presupuesto previsto.
- **Impacto a la Planificación:** Medio - Un presupuesto insuficiente puede llevar a retrasos en la ejecución del proyecto mientras se buscan fondos adicionales o se hacen ajustes para acomodar los costos adicionales.
- **Impacto al Alcance:** Medio - Si los costos son más altos de lo previsto, puede ser necesario reducir el alcance del proyecto para mantenerse dentro del presupuesto.

- **Impacto a la Calidad:** Bajo - Aunque el cálculo erróneo del presupuesto puede tener un impacto en la calidad si resulta en la necesidad de hacer compromisos para reducir costos, se espera que este impacto sea relativamente bajo dado que la calidad del proyecto es una prioridad clave.

Impacto Total del Riesgo: 0.27

Tabla: Tabla 102

Probabilidad	Impacto				Impacto
	Presup.	Planific.	Alcance	Calidad	
Baja	Crítico	Medio	Medio	Bajo	0,27

Tabla 102. Análisis de Riesgos: Calculo Erróneo del Presupuesto.

A.2.2.5. Falta de Madurez del Personal

Descripción del Riesgo: Aunque nuestro personal es altamente capaz, esta es la primera vez que se enfrentan a un proyecto de este estilo. La falta de experiencia práctica en un proyecto de esta naturaleza puede resultar en ineficiencias, retrasos y una posible disminución en la calidad del trabajo.

Categoría del Riesgo: Recursos

Probabilidad de Ocurrencia: Baja

Impactos Estimados:

- **Impacto al Presupuesto:** Bajo - La falta de madurez del personal puede conducir a ineficiencias que podrían aumentar ligeramente los costos del proyecto, aunque se espera que este impacto sea bajo.
- **Impacto a la Planificación:** Medio - La inexperiencia del personal puede conducir a retrasos en la ejecución del proyecto a medida que el personal se familiariza con las tareas requeridas.
- **Impacto al Alcance:** Medio - Si el personal no puede llevar a cabo tareas a un nivel alto de competencia, es posible que el alcance del proyecto tenga que ser ajustado.

- **Impacto a la Calidad:** Crítico - La falta de madurez del personal es probable que tenga un impacto crítico en la calidad del proyecto, ya que la experiencia y la competencia son factores clave en la entrega de resultados de alta calidad.

Impacto Total del Riesgo: 0.27

Tabla: Tabla 103

Probabilidad	Impacto				Impacto
	Presup.	Planific.	Alcance	Calidad	
Baja	Bajo	Medio	Medio	Crítico	0,27

Tabla 103. Análisis de Riesgos: Falta de Madurez del Personal.

A.2.2.6. Aumento de los Costes Eléctricos

Descripción del Riesgo: Mientras que el proyecto ha tenido en cuenta los costos de electricidad habituales, existe la posibilidad de un aumento inesperado y significativo de estos costos, como sucedió en 2022 durante la invasión rusa de Ucrania. Aunque la probabilidad de tal aumento es baja, la potencial magnitud de su impacto hace que este riesgo sea uno que debemos considerar y prepararnos para manejar, con planes de contingencia que puedan ayudar a mitigar su impacto.

Categoría del Riesgo: Recursos

Probabilidad de Ocurrencia: Baja

Impactos Estimados:

- **Impacto al Presupuesto:** Alto - Un incremento importante en los costos eléctricos tiene un impacto alto en el presupuesto del proyecto.
- **Impacto a la Planificación:** Inapreciable - A menos que el aumento en el costo de la electricidad sea tan grande que requiera una reevaluación de la viabilidad del proyecto, el impacto en la planificación sería inapreciable.

- **Impacto al Alcance:** Bajo - El incremento en el costo de la electricidad tiene un impacto inapreciable en el alcance del proyecto.
- **Impacto a la Calidad:** Inapreciable - A menos que el proyecto dependa de la electricidad para la calidad de los resultados del proyecto, el impacto en la calidad sería inapreciable.

Impacto Total del Riesgo: 0.17

Tabla: Tabla 104

Probabilidad	Impacto				Impacto
	Presup.	Planific.	Alcance	Calidad	
Baja	Alto	Inapreciable	Inapreciable	Inapreciable	0,17

Tabla 104. Análisis de Riesgos: Aumento de los Costes Eléctricos.

A.2.2.7. Problemas con la Aceptación del Cliente

Descripción del Riesgo: Aunque se han hecho todos los esfuerzos para alinear el proyecto con las expectativas y preferencias del cliente, existe el riesgo de que el producto final no sea de su agrado. Para este proyecto se entiende cliente como la entidad de la Universidad de Oviedo.

Categoría del Riesgo: Calidad

Probabilidad de Ocurrencia: Media

Impactos Estimados:

- **Impacto al Presupuesto:** Bajo - Un rechazo del cliente puede conllevar a costos de revisión y corrección, este impacto es considerado bajo dado que no se espera que estas modificaciones sean de gran envergadura.
- **Impacto a la Planificación:** Bajo - Las revisiones o correcciones solicitadas por el cliente tras la entrega del proyecto pueden causar

retrasos, pero no se espera que estos afecten significativamente la planificación general del proyecto.

- **Impacto al Alcance:** Bajo - Es posible que las revisiones solicitadas por el cliente puedan afectar ligeramente el alcance del proyecto, este impacto se considera bajo dado que no se espera que los cambios requeridos sean de gran magnitud.
- **Impacto a la Calidad:** Medio - Un rechazo por parte del cliente puede ser indicativo de que la calidad del proyecto no es adecuada o no cumple con las expectativas, lo que podría requerir esfuerzos adicionales para mejorar la calidad.

Impacto Total del Riesgo: 0.15

Tabla: Tabla 105

Probabilidad	Impacto				Impacto
	Presup.	Planific.	Alcance	Calidad	
Media	Bajo	Bajo	Bajo	Medio	0,15

Tabla 105. Análisis de Riesgos: Problemas con la Aceptación del Cliente.

A.2.2.8. Problemas con la Integración del Sistema

Descripción del Riesgo: El sistema que estamos desarrollando está diseñado para que varios componentes deban integrarse entre sí.

Categoría del Riesgo: Complejidad e Interfaces

Probabilidad de Ocurrencia: Baja

Impactos Estimados:

- **Impacto al Presupuesto:** Medio - La necesidad de solucionar problemas de integración puede implicar costos adicionales no previstos en el presupuesto original, ya sea por la necesidad de contar con expertos, adquirir softwares, o dedicar más tiempo al desarrollo.

- **Impacto a la Planificación:** Medio - Las dificultades en la integración pueden resultar en retrasos en la implementación del proyecto, lo que podría retrasar la finalización del proyecto o extender el cronograma.
- **Impacto al Alcance:** Medio - Los problemas de integración pueden obligar a revisar el alcance del proyecto si ciertas funcionalidades dependen de una integración exitosa.
- **Impacto a la Calidad:** Medio - Los problemas de integración pueden afectar a la calidad final del sistema, debido a que una mala integración puede limitar al sistema.

Impacto Total del Riesgo: 0.09

Tabla: Tabla 106

Probabilidad	Impacto				Impacto
	Presup.	Planific.	Alcance	Calidad	
Baja	Medio	Medio	Medio	Medio	0,09

Tabla 106. Análisis de Riesgos: Problemas con la Integración del Sistema.

A.2.2.9. Problemas con la Escalabilidad

Descripción del Riesgo: La aplicación que estamos desarrollando se espera que maneje un cierto volumen de usuarios y transacciones. Sin embargo, si el número de usuarios o transacciones supera las expectativas, esto podría poner a prueba la capacidad de la aplicación para escalar y manejar la carga adicional.

Categoría del Riesgo: Recursos

Probabilidad de Ocurrencia: Baja

Impactos Estimados:

- **Impacto al Presupuesto:** Medio - Mejorar la escalabilidad del sistema puede requerir una inversión adicional en infraestructura, como servidores

más potentes, o en desarrollo para optimizar el código y hacerlo más eficiente.

- **Impacto a la Planificación:** Bajo - Aunque la necesidad de mejorar la escalabilidad puede llevar a retrasos en el proyecto, se espera que el impacto sea relativamente bajo, ya que estos problemas suelen poder solucionarse sin alterar en gran medida el cronograma general del proyecto.
- **Impacto al Alcance:** Medio - Si se determina que el sistema no puede escalar para manejar un mayor número de usuarios o transacciones, esto puede requerir cambios en el alcance del proyecto, como la adición de nuevas características o funcionalidades para mejorar la escalabilidad.
- **Impacto a la Calidad:** Medio - Los problemas de escalabilidad pueden afectar negativamente la calidad del sistema, ya que un sistema que no escala bien puede experimentar un rendimiento deficiente o tiempos de inactividad cuando se somete a altas cargas.

Impacto Total del Riesgo: 0.09

Tabla: Tabla 107

Probabilidad	Impacto				Impacto
	Presup.	Planific.	Alcance	Calidad	
Baja	Medio	Bajo	Medio	Medio	0,09

Tabla 107. Análisis de Riesgos: Problemas con la Escalabilidad

A.2.2.10. Aparición de Competidores

Descripción del Riesgo: Nuestro proyecto es de naturaleza investigativa, por lo que existe el riesgo inherente de que otro equipo pueda estar trabajando en un sistema similar al nuestro y conseguir publicarlo antes. Esto podría disminuir la relevancia y el valor percibido de nuestro trabajo, ya que puede ser visto como una duplicación de esfuerzos.

Categoría del Riesgo: Calidad

Probabilidad de Ocurrencia: Muy Baja

Impactos Estimados:

- **Impacto al Presupuesto:** Bajo - Si aparece un competidor, es probable que no afecte significativamente nuestro presupuesto, ya que la mayor parte del trabajo y la inversión ya se habrán realizado.
- **Impacto a la Planificación:** Bajo - El surgimiento de competidores podría tener un impacto mínimo en la planificación del proyecto, ya que nuestro cronograma está diseñado independientemente de las actividades de los competidores.
- **Impacto al Alcance:** Medio - En caso de que un competidor lance un sistema similar antes que nosotros, puede ser necesario reevaluar y potencialmente modificar el alcance de nuestro proyecto para diferenciarlo y mantener su relevancia.
- **Impacto a la Calidad:** Alto - La aparición de un sistema similar antes de nuestro lanzamiento puede afectar a la percepción de la calidad y originalidad de nuestro proyecto, disminuyendo su valor.

Impacto Total del Riesgo: 0.06

Tabla: Tabla 108

Probabilidad	Impacto				Impacto
	Presup.	Planific.	Alcance	Calidad	
Muy Baja	Bajo	Bajo	Medio	Alto	0,06

Tabla 108. Análisis de Riesgos: Aparición de Competidores.

A.2.3. Gestión de Riesgos

En este apartado explicaremos que respuesta hemos asignado a cada riesgo y que estrategia seguiremos.

A.2.3.1. Ciberataque al sistema

Aunque Apolo sea una defensa de ciberataques, existe un alto espectro de ataques que no están contemplados en la defensa de **Apolo**.

La gestión de riesgos para un ciberataque al sistema requiere de una respuesta robusta, dada la gravedad que podría implicar un incidente de esta naturaleza para el proyecto. Hemos decidido adoptar la estrategia de transferencia de riesgo, una táctica que puede ser efectiva para enfrentar amenazas externas, especialmente en áreas que requieren de un conocimiento especializado como es el caso de la ciberseguridad.

En respuesta a este riesgo, hemos decidido contratar una consultoría externa especializada en ciberseguridad. Esto permitirá que un equipo con experiencia y especialización en el campo evalúe nuestras defensas actuales, identifique las posibles brechas de seguridad, y proporcione soluciones específicas para reforzar nuestra infraestructura.

La consultoría llevará a cabo auditorías periódicas para evaluar la seguridad del sistema, y se encargará de la implementación de medidas de seguridad necesarias, desde actualizaciones de software y hardware. De esta forma, si se produce un ciberataque, la consultoría será responsable de gestionarlo, lo que minimizará el impacto en el proyecto y permitirá que nuestro equipo se centre en sus tareas principales.

Además, la consultoría también ayudará en la creación de un plan de recuperación ante desastres, que permitirá al proyecto recuperarse rápidamente en caso de un incidente de seguridad. Al transferir esta responsabilidad a una empresa especializada, nos aseguramos de que el riesgo sea manejado de la manera más efectiva posible, lo que reduce el impacto potencial sobre el proyecto.

A.2.3.2. Rechazo de los Artículos Realizados

La gestión de este riesgo, que es la posibilidad de rechazo de los artículos por parte de las revistas, implica una estrategia de aceptación. Asumir este riesgo es una decisión estratégica, dada la naturaleza inherente de la revisión de trabajos de investigación para su publicación. La aprobación o rechazo de un artículo está más allá del control directo del personal del proyecto, y se encuentra sujeto a la revisión y criterio de los revisores y editores.

En respuesta a este riesgo, el personal de **Apolo** está preparado para realizar las modificaciones que sean recomendadas por los revisores de las revistas y enviar de nuevo el artículo para su revisión. Reconocemos que el proceso de revisión es fundamental para mejorar la calidad y la relevancia de nuestro trabajo, y que las críticas y recomendaciones resultantes serán valiosas para la mejora continua de nuestros resultados.

Además, en caso de que un artículo sea rechazado por una revista, el equipo mejorará el trabajo en base a los comentarios recibidos y, en su caso, buscará otra revista.

Aunque el rechazo de un artículo puede resultar en retrasos para la publicación de los resultados del proyecto, asumimos este riesgo como parte del proceso académico y científico.

En definitiva, la estrategia de aceptar este riesgo permite al personal de **Apolo** centrarse en la mejora de la calidad de los artículos y en adaptarse a las circunstancias a medida que se presentan, en lugar de intentar evitar un riesgo que es en gran medida una realidad de la investigación académica.

A.2.3.3. Retraso en la Entrega

El riesgo de un retraso en la entrega del proyecto es un aspecto que hemos considerado durante la planificación y por lo cual hemos optado por la estrategia de aceptación del riesgo. La aceptación de este riesgo no significa que no se tomen medidas para prevenirlo, sino que reconoce que, dada la complejidad y el alcance de nuestro proyecto, pueden surgir imprevistos que nos obliguen a reevaluar nuestro cronograma.

Se trabajará arduamente para cumplir con las fechas establecidas, optimizando las tareas, utilizando recursos de manera eficiente y constante para detectar y resolver cualquier obstáculo lo más rápido posible.

Sin embargo, si a pesar de nuestros esfuerzos llegásemos a exceder la fecha límite de entrega, estamos preparados para asumir las posibles consecuencias.

Por último, asumir el riesgo no significa ignorarlo, sino ser conscientes de su existencia y trabajar en medidas para reducir su impacto. En este caso, el personal de **Apolo** se compromete a aprender de la experiencia y a implementar mejoras para prevenir retrasos en el futuro.

A.2.3.4. Cálculo Erróneo del Presupuesto

En nuestro análisis de riesgos, identificamos la posibilidad de un cálculo erróneo del presupuesto como un riesgo potencial para el proyecto. Este tipo de desviación podría provocar dificultades financieras, retrasos en el proyecto y potencialmente la reducción del alcance o la calidad del proyecto.

Hemos elegido una estrategia de mitigación para este riesgo. Para ello, hemos asignado la responsabilidad de validar y aceptar el presupuesto a los directores del proyecto. Dada su experiencia y competencia en este campo, confiamos en su capacidad para revisar y aprobar un presupuesto preciso y detallado.

Los directores llevarán a cabo una revisión exhaustiva del presupuesto, prestando especial atención a los costos laborales, materiales, gastos operativos y otros costos directos e indirectos asociados al proyecto. También considerarán los posibles costos contingentes que podrían surgir debido a imprevistos durante la ejecución del proyecto.

Adoptar esta estrategia de mitigación nos permite reducir la probabilidad de que se produzcan errores de cálculo en el presupuesto y minimizar el impacto que estos podrían tener en el proyecto. Al hacerlo, aumentamos las posibilidades de que el proyecto se complete dentro del presupuesto previsto, sin comprometer su alcance ni su calidad.

A.2.3.5. Falta de Madurez del Personal

Uno de los riesgos identificados en nuestro análisis es la falta de madurez del personal, específicamente en relación con su inexperiencia en proyectos de esta naturaleza. Esta situación puede dar lugar a errores, ineficiencias, malentendidos y retrasos en la entrega del proyecto.

Hemos seleccionado una estrategia de mitigación para enfrentar este riesgo. Para lograr esto, hemos decidido que los directores del proyecto tomarán un

papel activo en la tutoría y guía de los miembros del equipo menos experimentados.

Estos directores, con su vasta experiencia y conocimientos en proyectos de esta índole, estarán encargados de transmitir su pericia, proporcionar orientación y apoyo a los miembros del personal, y supervisar su trabajo para garantizar que se cumplen los estándares de calidad y eficiencia requeridos.

Al tomar medidas proactivas para apoyar el desarrollo y la madurez de nuestro personal, esperamos disminuir la probabilidad de errores y retrasos, y aumentar la eficiencia y la calidad del trabajo realizado. Así, se reducirá la potencial afectación de este riesgo en la calidad final de **Apolo**.

A.2.3.6. Aumento de los Costes Eléctricos

Nuestro análisis de riesgos ha identificado el aumento potencial de los costes eléctricos como un factor que podría impactar de manera significativa en el presupuesto de nuestro proyecto. Este riesgo puede ser motivado por factores externos fuera de nuestro control, como fluctuaciones del mercado o eventos geopolíticos como la invasión rusa de Ucrania en 2022 que resultó en un incremento importante de estos costes.

Ante esta situación, hemos seleccionado la estrategia de mitigación de riesgos. Nuestra respuesta al riesgo del aumento de costes eléctricos es invertir en equipos con mayor eficiencia energética.

Al hacer una inversión inicial en tecnología de alta eficiencia, podremos reducir nuestro consumo de electricidad a largo plazo. Esto no solo nos ayudará a manejar mejor los posibles incrementos en los costes eléctricos, sino que también se alinea con nuestros objetivos de sostenibilidad y responsabilidad medioambiental. Esta decisión, aunque implica un gasto mayor al inicio, se contempla como una inversión que se traducirá en ahorros futuros.

Mediante estas acciones, pretendemos reducir la probabilidad y el impacto de este riesgo, asegurando la viabilidad financiera y la eficiencia de nuestro proyecto.

A.2.3.7. Problemas con la Aceptación del Cliente

Dado que **Apolo** será presentado al cliente (Universidad de Oviedo) una vez finalizado, existe el riesgo de que no satisfaga completamente sus expectativas o necesidades, lo cual podría llevar a problemas de aceptación. Este riesgo se

acentúa por la falta de un proceso de validación continuo por parte del cliente durante el desarrollo del proyecto, lo que podría dar lugar a discrepancias entre lo que el equipo de proyecto entiende como éxito y lo que el cliente espera del producto final.

Para este riesgo, nuestra estrategia será mitigarlo. El objetivo de esta estrategia es minimizar el impacto potencial en el proyecto, en este caso, limitando cualquier discrepancia entre las expectativas del cliente y el producto final.

La respuesta que hemos planeado para este riesgo es la realización de un exhaustivo proceso de verificación y validación del proyecto antes de su entrega al cliente. Este proceso será llevado a cabo por los directores del proyecto, que son expertos en el campo y tienen la capacidad de evaluar y asegurar que el proyecto cumple con los requisitos y especificaciones acordados.

El proceso de verificación implicará la revisión detallada de cada componente del proyecto para asegurarse de que cumple con los requisitos establecidos. La validación, por otro lado, se centrará en evaluar el producto final para asegurar que es funcional y satisface las necesidades y expectativas del cliente.

En última instancia, si el cliente tiene preocupaciones o no está completamente satisfecho con el producto entregado, estaremos dispuestos y preparados para tomar medidas correctivas y hacer los ajustes necesarios para cumplir con sus necesidades. Con estas medidas, esperamos reducir al mínimo la probabilidad de problemas con la aceptación del cliente.

A.2.3.8. Problemas con la Integración del Sistema

Nuestro análisis de riesgos ha identificado problemas potenciales con la integración del sistema, especialmente dado que la aplicación de **Apolo** debe interactuar internamente con diversos componentes del sistema. Las dificultades de integración pueden dar lugar a fallos en el sistema, retrasos en la planificación y aumentos inesperados en el presupuesto, además de tener un impacto medio en la calidad del proyecto.

Para abordar este riesgo, hemos decidido adoptar la estrategia de mitigación. En este caso, nuestra respuesta específica a los problemas de integración del sistema es proporcionar formación a todo el personal sobre las tecnologías y herramientas relevantes para la integración del sistema.

Esta formación se llevará a cabo a través de cursos donde el personal pueda adquirir y perfeccionar las habilidades necesarias para gestionar y resolver cualquier dificultad que surja durante el proceso de integración. Los cursos estarán diseñados para cubrir no solo el conocimiento teórico, sino también los aspectos prácticos de la integración del sistema.

La expectativa es que, mediante la formación, podamos aumentar la competencia y confianza del personal en la gestión de la integración del sistema, lo que a su vez debería reducir la probabilidad de que surjan problemas en esta área. De esta forma, esperamos mantener la calidad del proyecto, minimizar los retrasos y evitar costes adicionales.

A.2.3.9. Problemas con la Escalabilidad

Apolo se enfrenta al riesgo de tener problemas con la escalabilidad. La escalabilidad es la capacidad de un sistema, red o proceso para manejar una carga creciente de trabajo de manera efectiva. Si la aplicación no puede escalar de manera eficiente para acomodar un aumento en el uso, esto podría llevar a problemas de rendimiento, retrasos y una disminución en la satisfacción del usuario.

Para abordar este riesgo, hemos adoptado la estrategia de mitigación. En este caso, nuestra respuesta específica al riesgo de escalabilidad es utilizar herramientas y tecnologías que sean fácilmente escalables desde el principio.

Esto implica elegir tecnologías y plataformas de desarrollo que estén diseñadas para crecer con el sistema, por ejemplo, utilizando bases de datos escalables, servidores web que se puedan escalar tanto horizontal como verticalmente. También significa diseñar la aplicación desde el principio con la escalabilidad en mente, utilizando patrones de diseño y arquitecturas que permitan un escalado fácil y eficiente.

Además, implementaremos pruebas regulares de rendimiento y de carga para identificar temprano cualquier cuello de botella o problemas de rendimiento, y para validar que nuestras soluciones de escalabilidad están funcionando como se espera. Estos esfuerzos deberían minimizar la posibilidad de que surjan problemas de escalabilidad y reducir el impacto en caso de que lo hagan.

A través de esta estrategia de mitigación, buscamos prevenir el surgimiento de problemas de escalabilidad, asegurando así que nuestro proyecto pueda crecer de manera efectiva y eficiente en respuesta a la demanda del usuario.

A.2.3.10. Aparición de Competidores

Apolo, siendo una labor de investigación y desarrollo, no está exento de la posible aparición de competidores que puedan avanzar en la misma línea de trabajo y publicar resultados similares antes que nosotros. Si esto ocurre, nuestra propuesta puede verse desfavorecida o perder relevancia, lo que afectaría la visibilidad y el impacto de nuestro proyecto.

Dada la naturaleza de este riesgo, hemos optado por una estrategia de asumir el riesgo. Aunque hay ciertos elementos de este riesgo que están fuera de nuestro control, como el trabajo y los avances de otros equipos de investigación, hay acciones que podemos tomar para prepararnos y responder de manera efectiva en caso de que este riesgo se materialice.

Nuestra respuesta a este riesgo es adaptarnos y buscar formas de diferenciarnos en el caso de que se publique un sistema similar al nuestro. Esto puede implicar la revisión y adaptación de nuestro trabajo para destacar nuestros puntos fuertes, diferenciar nuestras propuestas y aportar nuevas soluciones o mejoras respecto a la competencia. Además, si un competidor publica un trabajo similar en una revista específica, buscaremos otras revistas respetadas y relevantes para enviar nuestro trabajo, con el fin de evitar comparaciones directas y garantizar una amplia difusión de nuestras investigaciones.

Es importante tener en cuenta que asumir este riesgo también implica estar dispuestos a aceptar las posibles consecuencias si el riesgo se materializa. Sin embargo, estamos confiados en nuestra capacidad para adaptarnos y reaccionar a estas situaciones, y creemos que nuestra propuesta y nuestro trabajo pueden mantenerse competitivos en el campo de la investigación.

A.3. Glosario y Diccionario de Datos

En este apartado se realizará una descripción breve a algunos términos importantes que se han utilizado durante la memoria.

- **Machine Learning:** Es un subcampo de la Inteligencia Artificial que permite a las máquinas aprender de los datos y mejorar su rendimiento sin la necesidad de programación explícita. Esto se logra a través de algoritmos que pueden modelar y entender patrones complejos.
- **Microservicio:** Es un estilo arquitectónico de software que descompone las aplicaciones en componentes más pequeños e independientes que funcionan juntos para ofrecer funcionalidad completa. Cada microservicio tiene su propio proceso y se comunica a través de mecanismos bien definidos, como APIs REST.
- **Red Neuronal:** Es un modelo computacional inspirado en el funcionamiento del cerebro humano y su red de neuronas. Se utiliza principalmente en el campo del Machine Learning para el reconocimiento de patrones y la toma de decisiones.
- **Firewall:** Es una medida de seguridad de red que monitoriza y controla el tráfico entrante y saliente basándose en políticas de seguridad predeterminadas. Funciona como una barrera entre una red interna de confianza y una red externa no confiable, como Internet.
- **Proxy Reverso:** Es un servidor que se encuentra entre el servidor de origen y los clientes que solicitan recursos. El proxy reverso recibe las solicitudes de los clientes y las redirige al servidor apropiado, puede proporcionar funcionalidades como balanceo de carga, encriptación SSL, y caché de contenido.
- **Preprocesamiento:** Es el proceso de limpieza y transformación de los datos brutos antes de su análisis. Este proceso puede incluir la eliminación de valores atípicos, la imputación de datos faltantes, la normalización de los datos y la codificación de variables categóricas.

- **Contenedor:** Un contenedor es una unidad estándar de software que empaqueta el código y todas sus dependencias para que la aplicación se ejecute de manera rápida y confiable desde un entorno informático a otro.
- **Clúster:** Un *clúster* se refiere a un grupo de servidores o máquinas que trabajan juntas como una sola entidad.
- **Clasificador Binario:** Es un tipo de algoritmo de *Machine Learning* que clasifica los datos en dos grupos o categorías distintas, a menudo denominados positivos y negativos o clase 1 y clase 0.
- **Artículo Científico:** Es un trabajo de investigación original que se presenta a la comunidad científica y, tras un proceso de revisión por la revista, se publica en una revista académica.
- **Dataset:** Es un término en inglés que se traduce como conjunto de datos. En el contexto de la ciencia de datos y el aprendizaje automático, un dataset es una colección de datos estructurados que pueden utilizarse para el análisis y el entrenamiento de modelos.
- **API REST:** *REST* (*Representational State Transfer*) es un estilo de arquitectura de software para servicios web. Una API REST facilita la comunicación entre diferentes partes de un sistema de software a través de *HTTP*, utilizando métodos como *GET*, *POST*, *PUT* y *DELETE* para manipular datos.
- **WebApp:** Es una aplicación de software que se accede a través de la web, generalmente a través de un navegador.
- **Numpy:** Es una biblioteca de *Python* utilizada para cálculos matemáticos y científicos. Proporciona soporte para matrices, junto con una gran colección de funciones matemáticas de alto nivel para operar con estas estructuras.
- **Pandas:** Es una biblioteca de *Python* diseñada para manipulación y análisis de datos. Proporciona estructuras de datos flexibles y permite trabajar con datos de manera eficiente. *Pandas* es particularmente fuerte en el manejo de datos tabulares, como los encontrados en *CSV*.

- **Scikit-learn:** Es una biblioteca de aprendizaje automático de *Python* que proporciona una gama de algoritmos de aprendizaje supervisados y no supervisados a través de una interfaz consistente en *Python*.
- **TensorFlow:** Es una biblioteca de software de código abierto desarrollada por *Google* para cálculos numéricos usando diagramas de flujo de datos.
- **Keras:** Es una biblioteca de *Python* de alto nivel para el aprendizaje profundo. Actúa como una interfaz para la biblioteca *TensorFlow* y permite construir y entrenar modelos de redes neuronales de una manera rápida y fácil con un enfoque en la usabilidad y la modularidad.
- **PyTorch:** Es una biblioteca de aprendizaje profundo de *Python* desarrollada por *Facebook's AI Research lab*.
- **Matplotlib:** Es una biblioteca de visualización de datos en *Python*. Permite crear una amplia gama de gráficos estáticos, animados e interactivos.
- **Seaborn:** Es una biblioteca de visualización de datos en *Python* basada en *Matplotlib*. Proporciona una interfaz de alto nivel para dibujar gráficos estadísticos atractivos e informativos.

A.4. Contenido Entregado en el Archivo adjunto

En este apartado se describe que contenido se ha adjuntado junto con el documento con extensión “.pdf”. Se detallarán que archivos se han adjuntado, los cuales son 2 archivos de *Project*, 4 archivos de *Excel* y el código fuente de **Apolo**.

A.4.1. Project

En los documentos adjuntos se encuentran 2 archivos de *Project* con extensión “.mpp”:

- Planificacion_Inicial: En este archivo se encuentra la planificación inicial realizada para el proyecto.
- Planificación_Final: En este archivo se encuentra la planificación final realizada para el proyecto.

A.4.2. Excel

En los documentos adjuntos se encuentran 4 archivos de *Excel* con extensión “.xlsx”:

- Presupuesto_Inicial: En este archivo se encuentra la planificación inicial realizada para el proyecto.
- Presupuesto_Final: En este archivo se encuentra la planificación final realizada para el proyecto.
- Riesgos_Apolo: En este archivo se encuentran los riesgos realizados para el proyecto.
- Resultados_Apolo: En este archivo se encuentran los resultados obtenidos de la experimentación de **Apolo**.

A.4.3. Código

En este apartado vamos a explicar la estructura de los directorios del código adjuntado. Dicha estructura se puede ver en la Tabla 109.

Directorio	Contenido
./Directorio raíz del Archivo adjunto	Contiene un fichero readme.md explicando toda la organización en ficheros y carpetas.
./TFG	Contiene toda la estructura de directorios del proyecto para desarrollo. En él se encuentran todas las carpetas para desplegar la arquitectura, incluido el paquete "ids" en el que se encuentra todo el código realizado.
./papers	Contiene los dos papers realizados, tanto los archivos <i>LaTex</i> para generar el pdf, como el código e imágenes asociadas.
./TFG/documentation	Contiene toda la documentación asociada al proyecto, incluido el fichero docx de este TFG y el fichero .pdf. También están todos los ficheros adjuntos a este documento y los diagramas realizados.
./TFG/documentation/img	Contiene todas las imágenes utilizadas en la documentación.

Tabla 109. Estructura código.

A.5. Índice Alfabético

A continuación, se muestra el índice alfabético de los términos más relevantes de este documento.

A

adversarios, 46, 52, 53, 54, 55
alertas, 40, 105, 113, 139, 158, 165, 168, 174, 176
algoritmo, 35, 36, 61, 63, 64, 68, 69, 70, 71, 72, 73, 75, 76, 77, 79, 124, 232, 270, 351
algoritmos, 34, 36, 60, 71, 112, 170, 175, 180, 350, 351
amenazas, 32, 33, 36, 44, 46, 50, 113, 136, 168, 343
AML, 34, 35, 36, 37, 40, 52, 63, 64, 112, 159, 235, 236, 242, 270, 271, 332
Apolo, 1, 36, 37, 38, 41, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 75, 76, 77, 79, 91, 101, 102, 106, 111, 112, 113, 115, 118, 121, 124, 133, 137, 138, 141, 157, 158, 169, 220, 221, 223, 225, 229, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 245, 246, 248, 249, 270, 271, 296, 330, 343, 344, 345, 346, 347, 348
ataque, 34, 37, 53, 54, 57, 58, 76, 179, 236, 237, 238, 239, 240, 241, 242, 270, 332
ataques, 1, 32, 33, 34, 35, 40, 46, 47, 48, 49, 52, 53, 54, 55, 56, 58, 63, 64, 75, 76, 109, 110, 133, 158, 159, 164, 235, 236, 242, 270, 271, 332, 343

B

base de datos, 137, 175
benignos, 47, 48, 50, 159

C

ciberseguridad, 35, 343
clasificación, 35, 36, 38, 50, 63, 67, 68, 69, 70, 71, 72, 73, 75, 76, 108, 111, 112, 113, 118, 120, 137, 141, 169, 183, 225, 226, 227, 228, 232
clasificadores, 36, 50, 51, 61, 63, 64, 68, 75, 76, 77, 79, 112, 113, 159, 232, 235, 236, 237, 239, 241, 242, 271

componente, 36, 37, 38, 104, 109, 110, 111, 113, 127, 130, 132, 134, 137, 139, 141, 160, 164, 169, 173, 176, 242, 347
configuración, 71, 128, 130, 133, 136, 140, 144, 147, 148, 161, 162, 163, 166, 167, 168, 170, 171, 173, 174, 175, 176, 177, 224, 247, 249
consumo, 164, 171, 177, 181, 182, 346

D

datos, 34, 38, 41, 46, 47, 48, 50, 52, 53, 57, 61, 64, 67, 68, 69, 70, 71, 72, 75, 83, 102, 103, 106, 108, 109, 110, 111, 112, 117, 122, 133, 134, 135, 136, 137, 138, 139, 140, 144, 145, 147, 148, 150, 151, 158, 159, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 182, 183, 184, 187, 219, 220, 221, 225, 226, 229, 230, 231, 232, 235, 236, 241, 243, 270, 271, 332, 348, 350, 351, 352
Deep Learning, 12, 68
detección, 1, 32, 33, 35, 46, 47, 54, 55, 76, 111, 113, 130, 131, 136, 167, 174, 181, 227, 228, 232, 236, 237, 239, 241, 242
Docker, 128, 130, 131, 141, 145, 147, 148, 149, 153, 154, 178, 179, 180, 181, 194, 246, 247, 248, 249, 250
Docker Compose, 128, 130, 131, 141, 145, 147, 148, 153, 154, 246, 247, 248

F

falsos, 33, 46, 50, 159
firewalls, 33, 44, 46

G

gestión, 38, 42, 108, 111, 116, 117, 118, 119, 122, 123, 125, 130, 132, 133, 134, 136, 137, 139, 140, 145, 147, 148, 158, 161, 163, 164, 167, 168, 171, 172, 173, 174, 176, 179, 180, 219, 220, 323, 328, 330, 331, 343, 347

H

hardware, 32, 41, 44, 102, 161, 168, 184, 343
herramientas, 83, 88, 93, 127, 128, 131, 132, 133, 134, 135, 144, 146, 148, 149, 151, 152, 157, 164, 166, 167, 168, 176, 178, 179, 180, 181, 184, 248, 347, 348

I

IDS, 32, 33, 34, 35, 36, 37, 38, 40, 46, 47, 48, 50, 51, 52, 53, 61, 63, 67, 68, 102, 110, 111, 136, 137, 158, 183, 231, 232, 233, 234, 236, 270
intrusos, 33, 36, 46, 47

L

librería, 69

M

MAB, 36, 60, 61, 63, 64, 68, 69, 70, 71, 72, 75, 76, 79, 111, 112, 113, 121, 124, 158, 232, 242, 270, 271
Machine Learning, 12, 33, 34, 36, 37, 52, 54, 55, 56, 68, 79, 159, 229, 270, 332, 350, 351
maliciosos, 50, 57
mantenimiento, 46, 88, 139, 161, 163, 167, 171, 173, 175, 176, 177
ML, 33, 34, 35, 36, 46, 55, 68, 75, 112, 151, 180, 182, 225, 226, 229, 270, 271
monitoreo, 103, 108, 113, 163, 164, 165, 166, 167, 173, 174, 175, 176
Multi-Armed Bandits, 36, 60, 63, 77, 270

P

positivos, 33, 46, 50, 159, 225, 226, 228, 351

R

recursos, 44, 47, 60, 109, 112, 130, 137, 140, 161, 168, 170, 171, 176, 177, 179, 181, 182, 184, 185, 249, 270, 344, 350

red, 32, 33, 34, 36, 37, 38, 40, 44, 46, 47, 48, 50, 53, 54, 61, 63, 64, 68, 71, 75, 76, 79, 93, 102, 103, 109, 111, 112, 113, 135, 136, 141, 158, 194, 232, 270, 271, 347, 350
rendimiento, 36, 37, 47, 50, 57, 63, 68, 75, 79, 105, 110, 112, 113, 132, 133, 136, 137, 138, 139, 140, 152, 161, 162, 163, 164, 167, 169, 170, 171, 172, 173, 176, 177, 180, 181, 183, 225, 226, 227, 228, 229, 230, 231, 235, 236, 270, 271, 333, 341, 348, 350

S

seguridad, 32, 33, 36, 37, 44, 46, 63, 64, 68, 109, 110, 111, 128, 133, 140, 153, 161, 162, 163, 164, 167, 171, 179, 232, 270, 332, 333, 343, 350
software, 32, 44, 46, 102, 142, 144, 146, 152, 153, 182, 332, 343, 350, 351

T

técnicas, 33, 34, 36, 40, 46, 52, 53, 54, 55, 67, 68, 73, 79, 111, 112, 144, 145, 159, 183, 229, 231, 235, 271
tráfico, 32, 34, 36, 38, 40, 44, 46, 47, 48, 53, 54, 61, 63, 64, 68, 75, 76, 79, 103, 109, 111, 112, 113, 131, 132, 133, 136, 137, 138, 140, 158, 160, 161, 163, 167, 168, 169, 170, 176, 226, 232, 270, 350

V

vulnerabilidad, 32, 33
vulnerables, 46, 270

W

Windows, 48, 148, 245, 247, 248

A.6. Código Fuente

El código fuente puede ser encontrado tanto en la carpeta “Apolo” dentro del fichero adjunto o en mi repositorio de *Github* cuyo enlace es el siguiente:

- <https://github.com/SergioArroni/TFG-Apolo>