



Escuela Técnica Superior de Ingeniería Informática

GRADO EN INGENIERÍA INFORMÁTICA –
TECNOLOGÍAS INFORMÁTICAS

TRABAJO FIN DE GRADO

MODELADO Y PRUEBA DE TPAs EN EL MARCO DE DORA

Realizado por
ANTONIO SABORIDO CAMPOS

Dirigido por
PABLO FERNÁNDEZ MONTES

Departamento
LENGUAJES Y SISTEMAS INFORMÁTICOS

SEGUNDA CONVOCATORIA

Curso 2023/2024

AGRADECIMIENTOS

En primer lugar, me gustaría agradecer a mi tutor, Pablo Fernández Montes, por su orientación, paciencia y valiosos consejos a lo largo de todo el proceso. Su dedicación y compromiso han sido fundamentales para el desarrollo de este proyecto, y sus enseñanzas me han permitido crecer tanto profesional como personalmente.

A mis padres, Antonio y Beatriz. Gracias por su amor incondicional y su apoyo.

A mi hermana Bea, gracias por estar siempre a mi lado.

A mi novia Mirella, gracias por tu paciencia, comprensión y por ser mi refugio durante este tiempo. Tu apoyo emocional y tus ánimos para mantenerme enfocado y motivado.

A mi amigo Guille, gracias por esas tardes de desconexión y risas que tanto necesitaba.

A mis compañeros durante estos años, Alberto, Alejandro, Belén, Calderón y Marcos, gracias por estos años de compañerismo, amistad y aprendizaje compartido. Juntos hemos enfrentado desafíos, celebrado éxitos y, sobre todo, hemos creado recuerdos inolvidables.

Gracias a todos.

ÍNDICE

CAPITULO 1: INTRODUCCIÓN	11
1.1 INTRODUCCIÓN.....	11
1.2 CONTEXTO.....	12
1.2.1. BUENAS PRACTICAS EN EL DESARROLLO INFORMÁTICO.....	12
1.2.2 DÓNDE SE SITÚA EL PROYECTO.....	14
1.3 OBJETIVOS	16
1.3.1. OBJETIVOS TÉCNICOS	16
1.3.2 OBJETIVOS ACADEMICOS	17
CAPITULO 2: ESTRUCTURA DEL DOCUMENTO	18
CAPITULO 3: METODOLOGÍA	19
3.1 MÉTODOLOGÍA ELEGIDA	19
3.2 HERRAMIENTAS DE GESTIÓN UTILIZADAS.....	23
3.2.1 GITHUB	23
3.2.2 TOGGLTRACK	24
3.2.3 GOOGLE CALENDAR	25
3.2.4 GOOGLE CHAT	26
3.3 DEFINICIÓN DE SPRINTS	28
3.4 DESGLOSE DE SPRINTS	30
3.5 PLANIFICACIÓN DE COSTES	35
3.5.1 COSTES DE PERSONAL	35
3.5.2 COSTES MATERIALES	36
3.5.3 CONCLUSION COSTES	37
CAPITULO 4: ANTECEDENTES	39
4.1 AUDITORIA DE BUENAS PRÁCTICAS DE EQUIPO (TPAs).....	39
4.1.1 BLUEJAY	39
4.1.2 TPAs	44
4.2 VISUALIZACIÓN DE TPAs	49
4.2.1 VISUALIZACIONES BASE DE GRAFANA	49
4.2.2 COMPARATIVA GRÁFICOS DISPONIBLES EN GRAFANA	50
4.2.3 ESTADÍSTICAS GENERALES DE VISUALIZACIÓN	65
4.3 MANUAL CREACIÓN DE WIDGET PERSONALIZADO EN GRAFANA ...	69
4.3.1 PREPARACIÓN DEL ENTORNO	69
4.3.2 INICIANDO GRAFANA.....	70

4.3.3 PERSONALIZACIÓN DEL PLUGIN.....	72
4.4 TPAS PARA DEVOPS	76
4.4.1 DORA.....	76
4.4.2 TPA DE DEVOPS.....	86
CAPITULO 5: ANÁLISIS	91
5.1 OBJETIVO DEL SISTEMA	91
5.2 ARQUITECTURA Y DISEÑO FUNCIONAL	93
5.2.1 FRONTEND.....	93
5.2.2 BACKEND.....	95
5.2.3 FLUJO DE INFORMACIÓN.....	96
CAPITULO 6: IMPLEMENTACIÓN	98
6.1. ENTORNO DE DESARROLLO	98
6.1.1 WEBSTORM	99
6.1.2 NODEJS y NPM.....	102
6.1.3 DOCKER.....	105
6.1.4 POSTMAN	107
6.1.5 GIT Y GITHUB DESKTOP	109
6.1.6 TOGGL TRACK	113
6.2 DISEÑO DE LA APLICACIÓN.....	114
6.2.1 BACKEND	114
6.2.2 FRONTEND	114
6.3 STACK TECNOLÓGICO.....	115
6.3.1 ANGULAR.....	115
6.3.2 NODE	115
6.3.3. SISTEMA DE DESPLIEGUE (DOCKER)	115
6.4 DETALLES DE LA IMPLEMENTACIÓN	116
6.4.1 TRADUCCIONES	116
6.4.2 GESTIÓN DE ARCHIVOS LOCALES	119
6.4.3 GESTIÓN DE VARIABLES DE ENTORNO.....	122
6.4.4 EJECUCIÓN DE CASOS DE PRUEBA	124
6.4.5 EJECUCIÓN EN MODO APLICACIÓN.....	129
6.4.6 BIBLIOTECAS UTILIZADAS	132
CAPITULO 7: MANUALES	133
7.1 MANUAL DE DESPLIEGUE	133
7.1.1 MODO DESARROLLO.....	134
7.1.2 MODO PRODUCCIÓN (DOCKER).....	135
7.1.3 MODO APLICACIÓN (ELECTRON)	135

7.2 MANUAL DE USO	136
7.2.1 PÁGINA DE INICIO	136
7.2.2 PÁGINA TESTEAR MÉTRICAS	137
7.2.3 TESTEO MANUAL.....	144
7.2.3. PÁGINA DE TESTING AUTOMÁTICO	149
7.2.4 PÁGINA TESTEO DE TPAs.....	154
7.2.5. PÁGINA DE CONFIGURACIÓN.....	157
7.3 GUIA DE REFERENCIA DE TPA TESTER.YAML.....	161
7.3.1 VARIABLES A USAR	167
7.4 TUTORIAL CICLO DE CREACIÓN DE UN TPA.....	170
CAPITULO 8: CONCLUSIONES	182
8.1 ANÁLISIS DE PROBLEMAS.....	182
8.2 TRABAJO FUTURO	184
8.3 CONCLUSIONES PERSONALES	185
CAPITULO 9: BIBLIOGRAFÍA Y ANEXOS	186
9.1 BIBLIOGRAFIA	186
9.2 ANEXOS	188
9.2.1 SOPORTE DE GRÁFICAS EN GRAFANA.....	188
9.2.2 SOPORTE DE GRAFICAS POR VISUAL VOCABULARY	190
9.2.3 SOPORTE DE GRÁFICAS POR FUNCIONES DATAVIZ PROJECT ...	191
9.2.4 SOPORTE DE GRÁFICAS POR FAMILIAS DATAVIZ PROJECT	193
9.2.5 EJEMPLO TPA	195
9.2.6 EJEMPLO TPA CON FORMATO BLUEJAY	199
9.2.7 FUNCIÓN STEPHANDLERS.....	203

ÍNDICE DE ILUSTRACIONES

ILUSTRACIÓN 1. ESTRUCTURA DE GOVERNIFY	12
ILUSTRACIÓN 2. VISUALIZACIÓN DE GRAFANA CON LOS DATOS PROPORCIONADOS POR BLUEJAY.....	14
ILUSTRACIÓN 3. PROCESO SCRUM Y SUS MIEMBROS.	19
ILUSTRACIÓN 4. EVENTOS DENTRO DE UN SPRINT EN SCRUM.....	20
ILUSTRACIÓN 5. ARTEFACTOS DEL PROCESO SCRUM	21
ILUSTRACIÓN 6. MILESTONES CREADAS PARA EL DESARROLLO DE TP-TESTER.	23
ILUSTRACIÓN 7. ISSUES REALIZADAS EN LA TERCERA MILESTONE.	24
ILUSTRACIÓN 8. EJEMPLO DE USO DEL TEMPORIZADOR, JUNTO A LA TAREA ASIGNADA.	24
ILUSTRACIÓN 9. RESUMEN DE TIEMPOS EN TOGGL.....	25
ILUSTRACIÓN 10. EJEMPLO DE REUNIÓN EN GOOGLE CALENDAR	26
ILUSTRACIÓN 11. WORKSPACE UTILIZADO PARA EL PROYECTO	27
ILUSTRACIÓN 12. CARGA DE TRABAJO POR SPRINTS.	29
ILUSTRACIÓN 13. DIAGRAMA DE FUNCIONAMIENTO DEL MICROSERVICIO RENDER	39
ILUSTRACIÓN 14. FUNCIONAMIENTO DE ASSETS MANAGER	40
ILUSTRACIÓN 15. EJEMPLO DE VISUALIZACIÓN DE GRAFANA.....	41
ILUSTRACIÓN 16. VISUALIZACIÓN DE MICROSERVICIO DE CREACIÓN DE TPAs.....	42
ILUSTRACIÓN 17. ESTRUCTURA DE UN ACUERDO DE EQUIPO	44
ILUSTRACIÓN 18. SCATTERPLOT CON EL PLUGIN “SCATTER”	50
ILUSTRACIÓN 19. SCATTERPLOT CONECTADO CON EL PLUGIN “SCATTER”	51
ILUSTRACIÓN 20. VISUALIZACIÓN PLUGIN “BUBBLE CHART”.....	51
ILUSTRACIÓN 21. VISUALIZACIÓN “HEATMAP” DE GRAFANA.....	51
ILUSTRACIÓN 22. VISUALIZACIÓN PLUGIN “CAL-HEATMAP”	52
ILUSTRACIÓN 23. VISUALIZACIÓN PLUGIN “MULTISTAT”.....	53
ILUSTRACIÓN 24. VISUALIZACIÓN PLUGIN “SLOPE GRAPH PANEL” EN GRAFANA	53
ILUSTRACIÓN 25. VISUALIZACIÓN PLUGIN “BUMP CHART PANEL” EN GRAFANA.....	54
ILUSTRACIÓN 26. VISUALIZACIÓN HISTOGRAMA EN GRAFANA.....	54
ILUSTRACIÓN 27. VISUALIZACIÓN DOT STRIP PLOT EN GRAFANA	55
ILUSTRACIÓN 28. VISUALIZACIÓN BARCODE PLOT EN GRAFANA	55
ILUSTRACIÓN 29. VISUALIZACIÓN DE POLÍGONOS DE FRECUENCIA EN GRAFANA.....	56
ILUSTRACIÓN 30. VISUALIZACIÓN PLUGIN “CDF - CUMULATIVE DISTRIBUTION FUNCTION” EN GRAFANA	56
ILUSTRACIÓN 31. VISUALIZACIÓN “TIME SERIES” EN GRAFANA	57
ILUSTRACIÓN 32. VISUALIZACIÓN “CANDLESTICK” EN GRAFANA	57
ILUSTRACIÓN 33. VISUALIZACIÓN “TIME SERIES” EN GRAFANA.....	59
ILUSTRACIÓN 34. VISUALIZACIÓN “BAR CHART” GRAFANA	59
ILUSTRACIÓN 35. VISUALIZACIÓN PLUGIN RADAR GRAPH	60
ILUSTRACIÓN 36. VISUALIZACIÓN “STATE TIMELINE” DE GRAFANA	60
ILUSTRACIÓN 37. VISUALIZACIÓN PLUGIN “PLOTLY”.....	61
ILUSTRACIÓN 38. VISUALIZACIÓN PIE CHART EN GRAFANA.....	61
ILUSTRACIÓN 39. AJUSTE PARA CAMBIAR UN GRÁFICO A TIPO DONUT	62
ILUSTRACIÓN 40. VISUALIZACIÓN “DONUT” EN GRAFANA	62
ILUSTRACIÓN 41. VISUALIZACIÓN TREEMAP EN GRAFANA.....	62
ILUSTRACIÓN 42. VISUALIZACIÓN STAT EN GRAFANA.....	63
ILUSTRACIÓN 43. VISUALIZACIÓN GEOMAP EN GRAFANA	63

ILUSTRACIÓN 44. VISUALIZACIÓN PLUGIN “SANKEY PANEL”	64
ILUSTRACIÓN 45. VISUALIZACIÓN PLUGIN WATERFALL PANEL	64
ILUSTRACIÓN 46. VISUALIZACIÓN PLUGIN ESNET CHORD	65
ILUSTRACIÓN 47. VISUALIZACIÓN NODE GRAPH EN GRAFANA.....	65
ILUSTRACIÓN 48. CREACIÓN DE PLUGIN PERSONALIZADO GRAFANA POR CONSOLA.....	69
ILUSTRACIÓN 49. OPCIONES SELECCIONADAS AL CREAR PLUGIN GRAFANA	70
ILUSTRACIÓN 50. PLUGIN CREADO CON ÉXITO	70
ILUSTRACIÓN 51. ARRANCAR GRAFANA	71
ILUSTRACIÓN 52. VISUALIZACIÓN DEL PLUGIN I	71
ILUSTRACIÓN 53. VISUALIZACIÓN DEL PLUGIN II	72
ILUSTRACIÓN 54. VISUALIZACIÓN DEL PLUGIN III.....	72
ILUSTRACIÓN 55. SCRIPTS DISPONIBLES	75
ILUSTRACIÓN 56. VISUALIZACIÓN DE GRÁFICO EN GRAFANA.....	75
ILUSTRACIÓN 57. ESTRUCTURA DEL PROYECTO ANGULAR	95
ILUSTRACIÓN 58. VISTA DE LAS OPERACIONES DE LA API DESDE SWAGGER.....	96
ILUSTRACIÓN 59. HERRAMIENTA GIT DENTRO DE WEBSTORM.	100
ILUSTRACIÓN 60. EJEMPLO DE COLECCIONES DENTRO DE POSTMAN	107
ILUSTRACIÓN 61. CREACIÓN DE RAMAS DESDE PÁGINA DE GITHUB.	110
ILUSTRACIÓN 62. EJEMPLO DE ABRIR LA NUEVA RAMA EN GITHUB DESKTOP.....	111
ILUSTRACIÓN 63. CAMBIO DE RAMA EN LOCAL CON GITHUB DESKTOP.....	111
ILUSTRACIÓN 64. SELECCIÓN DE IDIOMAÇ.....	118
ILUSTRACIÓN 65. ESTRUCTURA DE CARPETAS DE ARCHIVOS GUARDADOS.....	119
ILUSTRACIÓN 66. ENDPOINTS USADOS PARA EL MANEJO DE FICHEROS	120
ILUSTRACIÓN 67. ADVERTENCIA DE ARCHIVO MODIFICADO.	121
ILUSTRACIÓN 68. CAMBIO DE VARIABLES DE ENTORNO DE MANERA GRÁFICA.	123
ILUSTRACIÓN 69. APPLICACIÓN EJECUTADA CON ELECTRON	131
ILUSTRACIÓN 70. PÁGINA INCIAL TP-TESTER.....	136
ILUSTRACIÓN 71. NAVEGACIÓN A LA PANTALLA DE MÉTRICAS	137
ILUSTRACIÓN 72. VISUALIZACIÓN DE MÉTRICAS GUARDADAS	137
ILUSTRACIÓN 73. SECCIÓN PARA CREAR UNA NUEVA MÉTRICA	138
ILUSTRACIÓN 74. POP UP TUTORIAL DE CÓMO CREAR UNA MÉTRICA.....	139
ILUSTRACIÓN 75. PRUEBA DE MÉTRICA Y SU RESULTADO	140
ILUSTRACIÓN 76. ACCESO A PÁGINA DE VISUALIZACIÓN DE MÉTRICAS	140
ILUSTRACIÓN 77. PÁGINA DE EJECUCIÓN O DE EDICIÓN DE MÉTRICAS.	141
ILUSTRACIÓN 78. DIFERENCIA PÁGINA DE METRICA DE TPA CON MÉTRICAS INDIVIDUALES.....	143
ILUSTRACIÓN 79. MENSAJE DE MÉTRICA NO EDITADA	143
ILUSTRACIÓN 80. ACCESO A PÁGINA DE TESTEO MANUAL.....	144
ILUSTRACIÓN 81. PANTALLA DE INICIO AL ENTRAR EN EL TESTING MANUAL	144
ILUSTRACIÓN 82. VISUALIZACIÓN DE REPOSITORIOS DISPONIBLES CON EL TOKEN	145
ILUSTRACIÓN 83. PANTALLA DE DESCARGA DE REPOSITORIO	145
ILUSTRACIÓN 84. PÁGINA DE ACCIONES SOBRE REPOSITORIOS CLONADOS	146
ILUSTRACIÓN 85. PÁGINA DE RAMAS EN EL TESTING MANUAL	146
ILUSTRACIÓN 86. ACCIONES EN LA PÁGINA DE PULL REQUESTS.....	147
ILUSTRACIÓN 87. ACCIONES GENERALES EN EL REPOSITORIO.....	148
ILUSTRACIÓN 88. ACCESO A PÁGINA DE "AUTOMATED TESTING"	149
ILUSTRACIÓN 89. SCRIPTS YAMLs GUARDADOS	149
ILUSTRACIÓN 90. ZONA DE EJECUCION EN EL TEST YAML	150

ILUSTRACIÓN 91. "STEPS" DISPONIBLE EN LA PÁGINA DE TESTING AUTOMÁTICO	151
ILUSTRACIÓN 92. VISUALIZACIÓN DE LA EJECUCIÓN	152
ILUSTRACIÓN 93. RESULTADOS DE LAS PRUEBAS	152
ILUSTRACIÓN 94. PÁGINA DE EDICIÓN/EJECUCIÓN DE SCRIPTS YAMLS.....	153
ILUSTRACIÓN 95. ACCESO PANTALLA DE TPA	154
ILUSTRACIÓN 96. TABLA CON TODOS LOS TPAs YA CARGADOS.....	154
ILUSTRACIÓN 97. ÁREA DE TEXTO PARA CREACIÓN DE TPA	155
ILUSTRACIÓN 98. SELECCIÓN DE MODO DE EDICIÓN DE TPAs	155
ILUSTRACIÓN 99. VISUALIZACIÓN DE PANTALLA DE EDICIÓN DE TPA COMPLETO.	156
ILUSTRACIÓN 100. ACCESO A PANTALLA DE CONFIGURACIÓN	157
ILUSTRACIÓN 101. VISUALIZACIÓN DE CONTENEDORES DOCKERS ACTIVOS.....	158
ILUSTRACIÓN 102. SECCIÓN DE ACTUALIZACIÓN DE TOKEN GITHUB	158
ILUSTRACIÓN 103. POP UP DE AYUDA PARA OBTENER TOKEN DE GITHUB	159
ILUSTRACIÓN 104. CONSTANTES CONFIGURABLES EN LA APLICACIÓN	159
ILUSTRACIÓN 105. PUNTO DE ACCESO A LA DOCUMENTACIÓN DE LA API PROPIA	159
ILUSTRACIÓN 106. CICLO DE VIDA DE CREACIÓN DE UN TPA.	170
ILUSTRACIÓN 107. VISTA DE SECCIÓN PARA CREAR UNA NUEVA MÉTRICA.....	172
ILUSTRACIÓN 108. EJECUCIÓN DE QUERY EN GITHUB GQL	174
ILUSTRACIÓN 109. PASO PARA OBTENER LA COMPUTACIÓN DE BLUEJAY.....	176
ILUSTRACIÓN 110. RESULTADOS DE LA MÉTRICA.	176
ILUSTRACIÓN 111. MÉTRICA GUARDADA EN SECCIÓN DE MÉTRICAS INDIVIDUALES.	177
ILUSTRACIÓN 112. DISTRIBUCIÓN DE LA PÁGINA DE AUTOMATED TESTING	177
ILUSTRACIÓN 113. RESULTADOS DE LA EJECUCIÓN DE LA MÉTRICA.....	180
ILUSTRACIÓN 114. ENTRADA A PANTALLA DE EDICIÓN DE TPA.	181
ILUSTRACIÓN 115. ACTUALIZACIÓN DE TPA.	181

ÍNDICE DE TABLAS

TABLA 1. PLANIFICACIÓN DE SPRINTS.....	28
TABLA 2. DESGLOSE DE TAREAS SPRINT 1	30
TABLA 3. DESGLOSE DE TAREAS SPRINT 2	30
TABLA 4. DESGLOSE DE TAREAS SPRINT 3	31
TABLA 5. DESGLOSE DE TAREAS SPRINT 4	31
TABLA 6. DESGLOSE DE TAREAS SPRINT 5	31
TABLA 7. DESGLOSE DE TAREAS SPRINT 6	32
TABLA 8. DESGLOSE DE TAREAS SPRINT 7	32
TABLA 9. DESGLOSE DE TAREAS SPRINT 8	32
TABLA 10. DESGLOSE DE TAREAS SPRINT 9	33
TABLA 11. DESGLOSE DE TAREAS SPRINT 10	33
TABLA 12. DESGLOSE DE TAREAS SPRINT 11	33
TABLA 13. RESUMEN DE TIEMPOS Y PLANIFICACIÓN DEL PROYECTO.....	34
TABLA 14. COSTE TOTAL DEL PROYECTO	38
TABLA 15. SOPORTE DE PLUGINS EN GRAFANA	66
TABLA 16. CANTIDAD DE VISUALIZACIONES DE "VISUAL VOCABULARY" SOPORTADAS	66
TABLA 17. CANTIDAD DE VISUALIZACIONES DE "DATA VIZ PROJECT" SOPORTADAS.	67
TABLA 18. VISUALIZACIONES SOPORTADAS POR FAMILIAS	67
TABLA 19. VISUALIZACIONES SOPORTADAS POR TIPO DE ENTRADA DE DATOS	68

ÍNDICE DE CÓDIGOS

CÓDIGO 1. ESTRUCTURA BÁSICA DE UN TPA	45
CÓDIGO 2. INFRAESTRUCTURA DE BLUEJAY DENTRO DE UN TPA	45
CÓDIGO 3. SECCIÓN DE SCOPES DENTRO DE UN TPA	46
CÓDIGO 4. EJEMPLO PARA UN COLECTOR DE EVENTOS EN UN ACUERDO DE PRÁCTICAS DE EQUIPO	46
CÓDIGO 5. EJEMPLO DE CONFIGURACIÓN DE UN TABLERO SIMPLE	47
CÓDIGO 6. EJEMPLO DE UNA MÉTRICA EN UN ACUERDO DE PRÁCTICAS DE EQUIPO UTILIZANDO EL COLECTOR DE EVENTOS	47
CÓDIGO 7. EJEMPLO DE GARANTÍA UTILIZANDO LA MÉTRICA DEL EJEMPLO ANTERIOR.....	48
CÓDIGO 8. LLAMADA A LA API	73
CÓDIGO 9. TRATAMIENTO DE LOS DATOS DE LA LLAMADA A LA API.....	73
CÓDIGO 10. CREACIÓN DE GRÁFICA USANDO HIGHCHART.....	74
CÓDIGO 11. MANEJO DE LLAMADAS VACÍAS.	74
CÓDIGO 12. CONFIGURACIÓN DE LAS TRADUCCIONES EN EL APP MODULE.....	116
CÓDIGO 13. USO DE LAS TRADUCCIONES EN ARCHIVOS HTML	117
CÓDIGO 14. ENDPOINT PARA EL CALCULO DE HASHES.	121
CÓDIGO 15. CONTENIDO DE ARCHIVO DE VARIABLES DE ENTORNO (CONFIG.JS)	122
CÓDIGO 16. FUNCIÓN QUE EXPORTA LAS VARIABLES DE ENTORNO A TYPESCRIPT.....	122
CÓDIGO 17. ENDPOINT DE ACTUALIZACIÓN DE VARIABLES DE ENTORNO.....	123
CÓDIGO 18. FUNCIÓN EXECUTEYAML.....	125
CÓDIGO 19. CONTENIDO DE MAIN.JS DE ELECTRON	130
CÓDIGO 20. SCRIPT DE EJECUCIÓN ELECTRON	131
CÓDIGO 21. SCRIPTS DISPONIBLES EN LA APLICACIÓN	133
CÓDIGO 22. CONTENIDO DE ARCHIVO .ENV PARA LEVANTAR MICROSERVICIOS DE BLUEJAY.....	134
CÓDIGO 23. USO DE ACTUAL TIME EN TEST.YML	168
CÓDIGO 24. ESPECIFICACIÓN DE EVIDENCIA EN LOS QUE BUSCAR LOS RESULTADOS	168
CÓDIGO 25. USO DE MINEXPECTEDVALUE EN TESTING.....	168
CÓDIGO 26. SCRIPT 1: CREACIÓN DE FICHERO EN EL REPOSITORIO.....	179
CÓDIGO 27. SCRIPT 2: BORRADO DE ARCHIVO SUBIDO EN SCRIPT 1.....	180
CÓDIGO 28. SECCIÓN DE TESTING DENTRO DEL SCRIPT 1	180

CAPITULO 1: INTRODUCCIÓN

1.1 INTRODUCCIÓN

Este capítulo da inicio al Trabajo de Fin de Grado titulado "Modelado y Prueba de TPAs en el Marco de DORA". En él, se describirá la estructura del documento y se presentará el proyecto, realizado bajo la supervisión de Pablo Fernández Montes del Departamento de Lenguajes y Sistemas Informáticos (LSI), dentro del Grado en Ingeniería Informática – Tecnologías Informáticas con especialización en Sistemas Informáticos.

1.2 CONTEXTO

1.2.1. BUENAS PRACTICAS EN EL DESARROLLO INFORMÁTICO

En el desarrollo informático, las buenas prácticas son esenciales para un equipo de desarrolladores por varias razones. Facilitan la legibilidad y el mantenimiento del código mediante el uso de nombres de variables descriptivos, comentarios claros y estructuras coherentes. Además, la implementación de patrones de diseño y principios sólidos ayuda a evitar errores comunes y crear sistemas más robustos. Estas prácticas también optimizan el uso de recursos del sistema, mejorando el rendimiento de las aplicaciones, y promueven la seguridad mediante la validación de entradas y la codificación segura.

En un entorno colaborativo, las buenas prácticas permiten una integración fluida del trabajo de varios desarrolladores y aseguran una buena documentación y comunicación. Finalmente, favorecen la escalabilidad y flexibilidad del sistema, así como la reutilización de código, y contribuyen a la reducción de costos y tiempos de desarrollo, garantizando la entrega de productos de alta calidad y que cumplen con las expectativas de los usuarios.

Para facilitar la implementación y auditoría de estas buenas prácticas en equipos, el grupo de Desarrollo de la Universidad de Sevilla desarrolla un marco de gobernanza de servicios diseñado para mejorar la operación de servicios mediante la provisión de capacidades de auditoría automatizadas. Este marco, llamado **Governify**, permite la creación de arquitecturas de microservicios personalizadas para adaptarse a diversos dominios. [1]

Este marco ha sido aplicado en escenarios reales tanto en la industria como en el ámbito educativo, donde ha servido a investigadores y profesionales en la gobernanza de servicios como una herramienta analítica visual y un banco de pruebas para experimentos. Governify ha demostrado su capacidad para recopilar información sobre riesgos potenciales asociados con el incumplimiento y para diseñar y monitorear las mejores prácticas en forma de acuerdos.

Governify está diseñado para mejorar la gobernanza de servicios a través del soporte de auditoría automatizada. Los componentes integrados pueden combinarse para formar arquitecturas adaptables adecuadas para diversos escenarios. Las plataformas desarrolladas con Governify recopilan evidencias de diferentes fuentes utilizando APIs, calculan métricas, evalúan los objetivos esperados basados en estas métricas y presentan el estado de cumplimiento a través de paneles visuales.

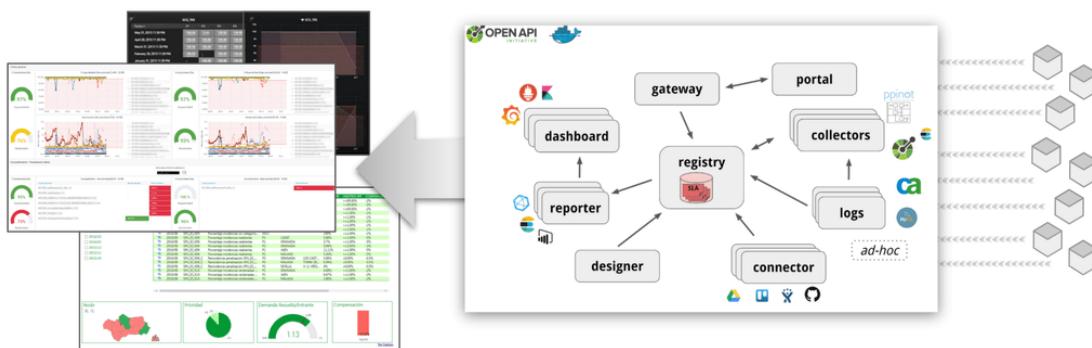


Ilustración 1. Estructura de Governify

Este marco se basa en un modelo de acuerdo unificado llamado **iAgree**, que ofrece un enfoque consistente para la modelación de métricas y objetivos a través de diversos dominios, desde SLA (Acuerdos de Nivel de Servicio) RESTful hasta mesas de soporte IT dirigidas por humanos, permitiendo una plataforma de gobernanza cohesiva para dirigir la estrategia organizacional y experimentar con el nivel de servicio (o desempeño) entregado por infraestructuras y equipos. [2]

Governify proporciona una arquitectura flexible utilizando microservicios RESTful como componentes con implementación en contenedores. Este diseño permite el desarrollo de plataformas de auditoría específicas de dominio utilizando configuraciones de microservicios personalizadas.

Aplicaciones destacadas de Governify:

- Auditoría de Equipos Ágiles (**Bluejay**):
Governify ayuda a construir plataformas de auditoría para equipos ágiles estableciendo Acuerdos de Práctica de Equipo. Integra métricas de varias herramientas de desarrollo para generar paneles personalizados, asegurando que el equipo adhiera a un conjunto de buenas prácticas.
- Auditoría de Infraestructuras (**Falcon**):
Governify proporciona una plataforma de gestión de SLA que recopila métricas a través de exportadores Prometheus para procesarlas y generar paneles estratégicos para cada infraestructura.
- Auditoría de Catálogo de APIs (**Galibo**):
Governify ayuda en la gobernanza de APIs, integrándose con SLA4OAI y sistemas de monitoreo existentes como ElasticSearch para proporcionar paneles.
- Auditoría de Equipos de Soporte (**Gauss**):
Governify traduce los SLA de soporte en paneles visuales, ofreciendo información sobre el desempeño de los equipos de soporte.

Governify actúa como una infraestructura fundamental para investigar prácticas de gobernanza de servicios. Su diseño facilita la ejecución de experimentos estructurados en el dominio de la gobernanza, permitiendo la exploración y análisis de varios modelos y estrategias de gobernanza. En particular, permite investigaciones sobre la efectividad y relevancia de diferentes métricas de gobernanza, ayudando a determinar qué métricas proporcionan las ideas más precisas y útiles.

1.2.2 DÓNDE SE SITÚA EL PROYECTO

Este proyecto se enmarca en las herramientas preexistentes, de Governify, en concreto de Bluejay. Bluejay es fruto de la colaboración entre el grupo ISA (Ingeniería del Software Aplicada) de la Universidad de Sevilla y la Universidad de California, Berkeley (UCB) [3].

Bluejay es una plataforma extensible y de código abierto diseñada para auditar prácticas de equipo (TPs) en equipos de desarrollo ágil, recolectando datos de múltiples herramientas mediante sus APIs y presentando estos datos en tableros de control. Su objetivo principal es facilitar a los instructores de ingeniería de software la auditoría de TPs, utilizando una metodología similar a la de los acuerdos de nivel de servicio (SLA) [4].

Para permitir las extensiones y nuevas herramientas, Bluejay se basa en una arquitectura de microservicios que permiten su activación o desactivación según las necesidades del sistema. El objetivo de este proyecto era dar soporte a un nuevo TPA (Acuerdo de Prácticas de Equipo) que implemente las buenas prácticas de DORA (DevOps Research and Assessment), las cuales son esenciales para la mejora continua y la eficiencia en el desarrollo y operaciones de software. Dada la complejidad del TPA desarrollado, se decidió focalizar en la construcción de un nuevo servicio para realizar pruebas sobre este TPA, que no solo servirá para el TPA de DORA, sino también para todos los demás.

Toda la arquitectura de Bluejay, se verá con más detalle posteriormente en 4.1.1 BLUEJAY, pero uno de los microservicios más interesante de los comentados es Dashboard. Ya que al proporcionar una interfaz gráfica basada en Grafana permite la visualización del seguimiento de las métricas establecidas de una forma sencilla y fácil comprender para todos los miembros del equipo.



Ilustración 2. Visualización de Grafana con los datos proporcionados por Bluejay.

El principal objetivo de este proyecto será la creación de un microservicio nuevo para Bluejay que permita la creación y comprobación de los acuerdos de equipo a través de una interfaz gráfica. Además de ello, se creará una herramienta que permita la comprobación automática del funcionamiento de estas métricas.

Además, este proyecto también se lleva a cabo en colaboración con la Universidad de Castilla-La Mancha (UCLM), concretamente con la profesora Elena Navarro, con la intención de crear un conjunto de buenas prácticas de equipo (TPA) para llevarlas a la práctica en asignaturas de desarrollo de aplicaciones en equipo.

Para la elaboración de este acuerdo de prácticas de equipo, se llevó a cabo varias reuniones con la alumna Celia Fernández. En ellas establecimos unas bases para este

documento como la duración de los sprints y algunas definiciones generales como que entendíamos por rama activa. Tras establecer estas bases, elaboramos un documento de 24 acuerdos de equipo (TP), que posteriormente crearemos usando la herramienta que se creará en este proyecto para la creación de TPAs.

Tras establecer estas bases, Celia creó 12 TPs (acuerdos de equipos), y yo otros 12 TPs. Posteriormente esto se elaboró un documento conjunto que contiene los 24 acuerdos de equipo (TP) y todos estos acuerdos fueron revisados por ambas partes en una reunión final formando así un TPA. Este TPA, será implementado posteriormente utilizando la herramienta que se desarrollará en este proyecto para la creación y Testing de TPAs.

1.3 OBJETIVOS

1.3.1. OBJETIVOS TÉCNICOS

El objetivo de este proyecto es proporcionar al usuario una interfaz intuitiva que le permita incluir de manera más rápida y eficiente nueva funcionalidad en el sistema, ofreciendo un portal capaz de probar los acuerdos descritos en el lenguaje formal, así como extraer dicha funcionalidad de cara a implementarlo en otros proyectos completamente ajenos.

Para ello, el primer objetivo será una herramienta que permita probar métricas de forma individual. Una vez todas las métricas que compongan el acuerdo de buenas prácticas de equipo (TPA). También tener una herramienta o página que permita actualizar, eliminar o subir un nuevo TPA de una forma gráfica e intuitiva a Bluejay.

Una vez esté esta nueva herramienta, el objetivo será la confección de forma formal de un Team Practice Agreement (TPA). Para una vez estén establecidas sus métricas y garantías poder llevarlas a la práctica usando la herramienta TP-Tester y poder comprobar así su correcto funcionamiento.

Además, otro objetivo de este documento es la investigación de las gráficas disponibles actualmente en el dashboard que utiliza Bluejay, que actualmente es una integración de Grafana, y la posibilidad de añadir nuevas visualizaciones extras que no se encuentren añadidas actualmente.

En relación con todo lo anterior podemos identificar una serie de objetivos que servirán de punto de referencia durante todo este proyecto:

1. Creación de una herramienta que sirva como extensión a Bluejay, que permita crear, editar y eliminar métricas. Esta herramienta también deberá permitir la creación, actualización y eliminación de los TPA (Team Practice Agreement) que se encuentren en ese momento cargados en Bluejay. Y además de ello, deberá tener alguna funcionalidad que permita crear scripts de pruebas para la comprobación del funcionamiento de las métricas automáticamente.
2. La elaboración de un documento formal que describa una serie de acuerdos de equipo que posteriormente vayan a ser convertidas en un TPA.
3. La investigación de las visualizaciones gráficas que estén disponibles en Grafana en ese momento e investigar la posibilidad de añadir nuevas personalizadas.

1.3.2 OBJETIVOS ACADEMICOS

La elaboración del presente Trabajo de Fin de Grado tiene como objetivo fomentar el aprendizaje y desarrollo personal del alumno. Por ello, se han establecido una serie de objetivos adicionales a los puramente técnicos:

- Colaborar con un grupo de investigación para observar una faceta distinta de la universidad.
- Aplicar los conocimientos adquiridos durante la carrera en un caso real, aprendiendo nuevas tecnologías actualmente en auge en las empresas, como son las herramientas basadas en microservicios, mejorando así en los ámbitos de investigación, planificación, organización y aprendizaje.
- Ver un ejemplo real como desplegar una nueva herramienta en un proyecto bastante más grande y con más microservicios.

CAPITULO 2: ESTRUCTURA DEL DOCUMENTO

El presente documento está dividido en 9 capítulos. A continuación, se detalla el contenido de cada parte:

1. **Introducción:** El primer capítulo proporciona el contexto del proyecto y establece los objetivos principales que se pretenden alcanzar.
2. **Estructura del documento:** Describe la organización y los contenidos de cada capítulo del documento.
3. **Metodología:** En el tercer capítulo se detalla el enfoque metodológico del proyecto, incluyendo las herramientas utilizadas, la planificación y el estudio de costos a lo largo de las diferentes fases del proyecto.
4. **Antecedentes:** En el cuarto capítulo se centra en el análisis de todas las herramientas disponibles previamente como los TPAs y la herramienta actual Bluejay. Además de un estudio de las Gráficas disponible actualmente en Grafana dentro de todas las gráficas reconocidas por estudios como el Visual Vocabulary y la posibilidad de añadir aquellas que no se encuentren disponibles a través de plugins.
5. **Diseño:** El cuarto capítulo presenta la arquitectura de la solución propuesta, destacando los aspectos técnicos relevantes para la integración de TP-Tester a todo el ecosistema ya creado de Bluejay.
6. **Implementación:** En este capítulo se detalla el desarrollo realizado para cumplir con los requisitos establecidos, explicando cómo y dónde se ha integrado cada componente en el proyecto.
7. **Manuales:** El penúltimo capítulo incluye dos guías:
 - a. Una guía paso a paso para la instalación, puesta en marcha y despliegue de la aplicación.
 - b. Un manual de usuario para la navegación a través de todas las páginas de TP-Tester.
 - c. Un manual de desarrollador explicando la creación de nuevas métricas de Bluejay
 - d. Otro manual de desarrollador explicando la creación de scripts yaml, para realizar tests sobre las métricas creadas previamente.
8. **Conclusiones:** En el último capítulo se analiza el proceso de desarrollo del proyecto, considerando las desviaciones, dificultades y problemas encontrados, y se ofrece una evaluación final de los resultados obtenidos junto con las conclusiones personales sobre la realización del proyecto.
9. **Bibliografía y anexos:** Ofrecen soporte adicional al contenido del documento, proporcionando referencias y materiales complementarios.

CAPITULO 3: METODOLOGÍA

En este capítulo se lleva a cabo la planificación y el cálculo de los costos del proyecto. La planificación es crucial para asegurar que el proyecto alcance sus objetivos. Este proceso es muy complejo, ya que implica prever cómo se desarrollará el proyecto desde el comienzo hasta su conclusión. Para lograr esto, es necesario considerar una gran cantidad de variables, como los recursos económicos, humanos, técnicos, el tiempo, entre otros.

3.1 MÉTODOLOGÍA ELEGIDA

En el desarrollo de este proyecto se ha elegido la metodología ágil Scrum para gestionar y planificar el trabajo. Scrum es un marco de trabajo que facilita la colaboración entre equipos para producir software de manera iterativa e incremental. Esta metodología es especialmente útil para proyectos complejos donde los requisitos pueden cambiar con frecuencia.

El enfoque de Scrum permite dividir el trabajo en sprints, que son ciclos de trabajo cortos y definidos, usualmente de dos a cuatro semanas, al final de los cuales se entrega un incremento del producto potencialmente funcional.

Equipo Scrum

El equipo Scrum está compuesto por diferentes roles que, en conjunto, son auto-organizados y multidisciplinarios. Esto significa que no tienen ningún tipo de dirección externa y las decisiones de organización son propias. Los roles principales dentro de un equipo Scrum son:

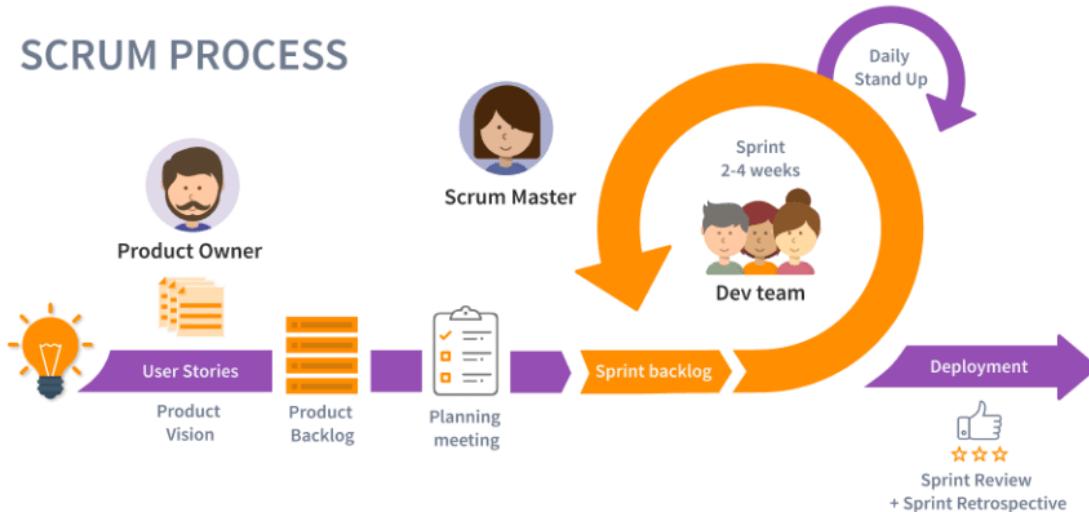


Ilustración 3. Proceso Scrum y sus miembros.

- **Product Owner:** El *Product Owner* es responsable de maximizar el valor del producto y del trabajo del equipo de desarrollo. Esto implica definir y priorizar el backlog del producto, asegurando que el equipo trabaje en las tareas más importantes y valiosas primero. El Product Owner actúa como el enlace entre el

equipo de desarrollo y las partes interesadas, traduciendo las necesidades del negocio en requisitos técnicos y asegurando que el producto final cumpla con las expectativas del cliente.

- **Scrum Master:** El *Scrum Master* es un facilitador y líder de servicio para el equipo Scrum. Su principal responsabilidad es asegurarse de que el equipo siga las prácticas y principios de Scrum. El Scrum Master elimina obstáculos que puedan impedir el progreso del equipo, facilita las reuniones y fomenta un ambiente de colaboración y mejora continua. Aunque el Scrum Master no tiene autoridad directa sobre el equipo de desarrollo, su papel es crucial para guiar y apoyar al equipo en la adopción y práctica de Scrum.
- **Equipo de Desarrollo:** El equipo de desarrollo está compuesto por profesionales que trabajan en la creación del incremento del producto al final de cada sprint. Estos equipos son multidisciplinarios, es decir, incluyen todas las habilidades necesarias para entregar un producto funcional sin depender de personas externas al equipo. Además, son auto-organizados, lo que significa que deciden internamente cómo llevar a cabo el trabajo asignado, distribuyendo las tareas según sus propias dinámicas y competencias.

Eventos Scrum

En Scrum, los eventos están diseñados para aportar regularidad en el desarrollo y reducir la cantidad de reuniones extra. La duración de los eventos es previamente fijada y no se puede modificar conforme avanzamos en él. Los eventos clave en Scrum incluyen:

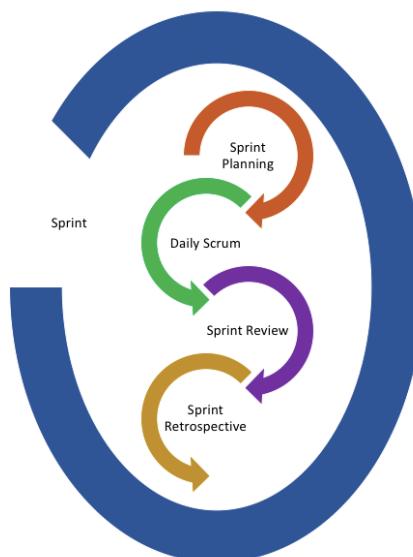


Ilustración 4. Eventos dentro de un Sprint en Scrum.

- **Sprint Planning:** Esta es la reunión de planificación que inicia el sprint. Durante el sprint planning, el equipo define el objetivo del sprint y selecciona los ítems del backlog del producto que se comprometen a completar durante el sprint. Se crea el sprint backlog, que es la lista de tareas necesarias para alcanzar el objetivo del sprint. La duración de esta reunión varía, pero generalmente no debe exceder las ocho horas para un sprint de un mes.

- **Daily Scrum:** También conocida como daily stand-up, es una reunión diaria de corta duración (máximo 15 minutos) donde el equipo sincroniza sus actividades y crea un plan para las próximas 24 horas. Cada miembro del equipo responde a tres preguntas: ¿Qué hice ayer? ¿Qué voy a hacer hoy? ¿Hay algún impedimento en mi camino? Esta reunión mejora la comunicación, elimina otras reuniones y fomenta la toma rápida de decisiones.
- **Sprint Review:** Al final de cada sprint, el equipo y las partes interesadas se reúnen para revisar el incremento del producto desarrollado durante el sprint. El objetivo es inspeccionar el trabajo realizado y adaptar el backlog del producto según sea necesario. Es una oportunidad para obtener feedback y asegurarse de que el desarrollo del producto está alineado con las expectativas y necesidades del cliente. La duración de esta reunión es de un máximo de cuatro horas para un sprint de un mes.
- **Sprint Retrospective:** Esta es la última reunión del sprint, donde el equipo reflexiona sobre el proceso de trabajo durante el sprint y discute qué fue bien, qué no fue tan bien y cómo pueden mejorar en el siguiente sprint. El objetivo es encontrar maneras de trabajar de manera más efectiva y eficiente. La retrospectiva se realiza después del sprint review y antes de la siguiente planificación del sprint, y su duración es de un máximo de tres horas para un sprint de un mes.

Artefactos Scrum

Scrum también define varios artefactos que proporcionan visibilidad y transparencia del trabajo que se está realizando. Estos incluyen:

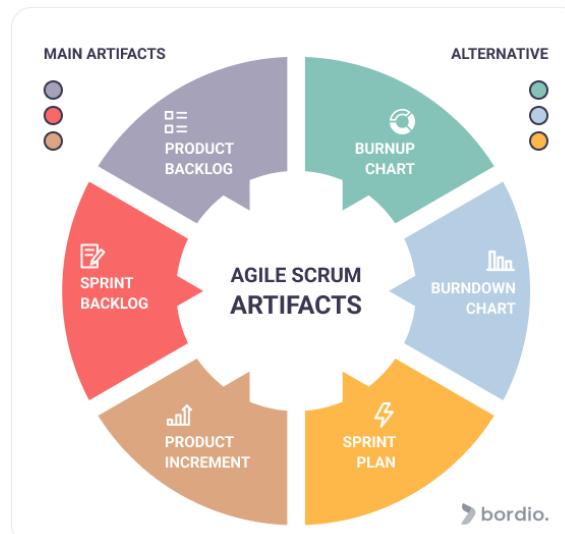


Ilustración 5. Artefactos del proceso scrum

- **Product Backlog:** Es una lista priorizada de todo el trabajo que se necesita hacer en el producto. El Product Owner es responsable de mantener y priorizar el backlog del producto, asegurando que esté alineado con los objetivos y necesidades del negocio.

- **Sprint Backlog:** Es la lista de tareas seleccionadas del backlog del producto que el equipo se compromete a completar durante el sprint. El sprint backlog también incluye un plan para entregar el incremento del producto y cumplir con el objetivo del sprint.
- **Incremento:** Es el resultado del trabajo completado durante el sprint y todos los incrementos anteriores. El incremento debe estar en un estado utilizable y cumplir con la definición de "hecho" (definition of done) acordada por el equipo.
- **Sprint Plan:** Es un documento preparado durante la reunión de planificación del sprint. Viene en varios formatos y longitudes, pero siempre define la carga de trabajo para el sprint: el objetivo principal del sprint, lo que debe hacerse y cómo se ejecutará. Al igual que los gráficos, el plan puede ser utilizado por el equipo de desarrollo como punto de referencia durante las reuniones de revisión del sprint o retrospectivas del sprint.
- **Burndown Chart:** Es un gráfico que muestra el trabajo restante en el sprint backlog frente al tiempo. Ayuda a visualizar el progreso del sprint y a predecir si el equipo podrá completar su trabajo en el tiempo restante. Un burndown chart eficaz permite al equipo y a las partes interesadas ver rápidamente si están en camino de cumplir con el objetivo del sprint.
- **Burnup Chart:** Es similar al burndown chart, pero muestra el trabajo completado en lugar del trabajo restante. Este gráfico puede ser útil para visualizar el progreso hacia la finalización de un proyecto o la entrega de un producto, mostrando tanto el trabajo completado como el trabajo total requerido.

3.2 HERRAMIENTAS DE GESTIÓN UTILIZADAS

En el desarrollo de este proyecto, se han utilizado varias herramientas de gestión para facilitar la organización, el seguimiento del tiempo y la comunicación entre los miembros del equipo.

Estas herramientas han sido esenciales para mantener la coordinación y asegurar el cumplimiento de los plazos. A continuación, se detallan las herramientas principales empleadas: GitHub, TogglTrack, Google Calendar y Google Chat.

3.2.1 GITHUB

GitHub es una plataforma de desarrollo colaborativo que permite gestionar el código fuente y controlar las versiones del proyecto. Es ampliamente utilizada en la industria del software debido a su capacidad para facilitar la colaboración entre desarrolladores distribuidos geográficamente. En este proyecto, GitHub ha sido fundamental para mantener un control riguroso sobre el código, gestionar las versiones y asegurar que todos los cambios realizados sean rastreables y revertibles en caso necesario.

En particular, se han utilizado las Milestones de GitHub para planificar y gestionar el desarrollo de la aplicación. Las Milestones son herramientas de planificación que permiten agrupar issues (tareas o problemas) y pull requests (solicitudes de extracción) en objetivos específicos, definiendo plazos y facilitando el seguimiento del progreso.



Ilustración 6. Milestones creadas para el desarrollo de TP-Tester.

Cada Milestone representa un objetivo importante o una fase del proyecto, y su uso ha permitido una mejor organización y priorización de las tareas. Las Milestones facilitan la visualización del estado del proyecto, identificando rápidamente qué tareas están pendientes, en progreso o completadas, y asegurando que el equipo se mantenga enfocado en alcanzar los objetivos definidos.

Otra funcionalidad muy interesante de estas milestone, que aun habiendo terminado ya, siguen teniendo relacionados las issues que se realizó en cada milestone.

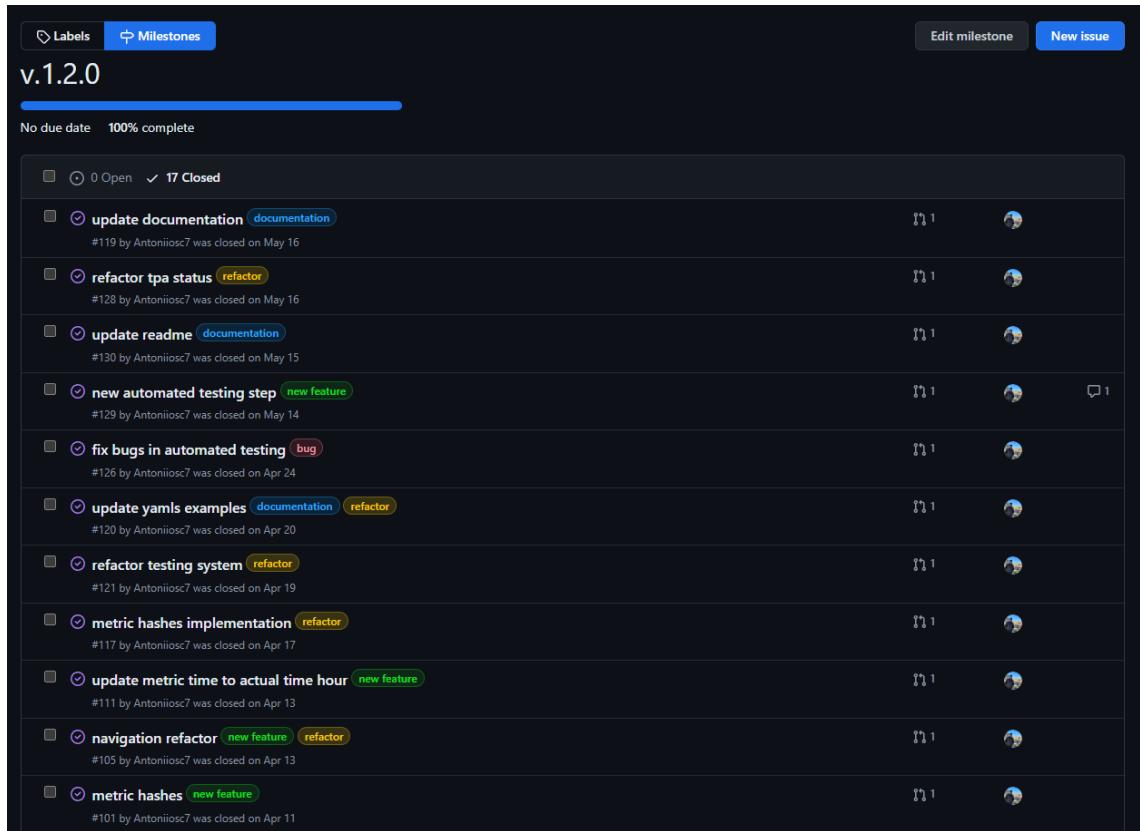


Ilustración 7. Issues realizadas en la tercera milestone.

Y además cada Issue, va relacionada a una pull request, lo que también permite ver que código fue modificado en esa issue y como y cuando fue implementada.

3.2.2 TOGGLTRACK

TogglTrack es una herramienta de seguimiento del tiempo que permite registrar y analizar cómo se emplea el tiempo en diferentes tareas y proyectos. Esta aplicación es especialmente útil para obtener una visión clara de la productividad y la distribución del tiempo a lo largo del proyecto. En este proyecto, TogglTrack ha sido utilizado para guardar tiempos y registrar la dedicación a cada tarea específica, proporcionando datos precisos sobre el esfuerzo invertido en cada fase del desarrollo.

El funcionamiento de TogglTrack es sencillo pero eficaz: los usuarios inician un temporizador cuando comienzan una tarea y lo detienen al finalizar.

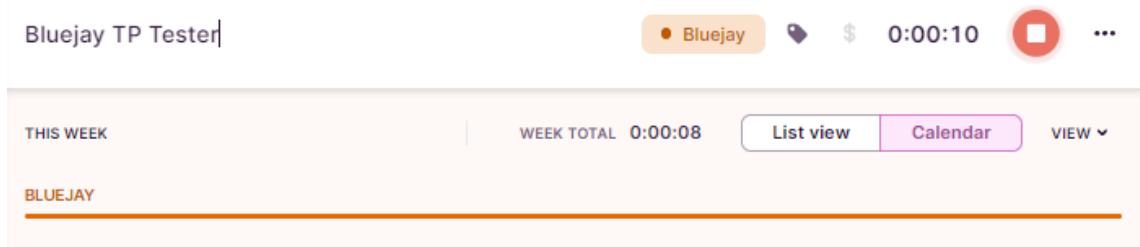


Ilustración 8. Ejemplo de uso del temporizador, junto a la tarea asignada.

Esto permite generar informes detallados sobre el uso del tiempo, identificar posibles áreas de mejora y asegurar una gestión eficiente del tiempo. La herramienta también permite categorizar las tareas y proyectos, lo que facilita el análisis posterior de los datos recogidos. La información obtenida ha sido crucial para evaluar la eficiencia del equipo y ajustar las estimaciones de tiempo en función de los datos reales.

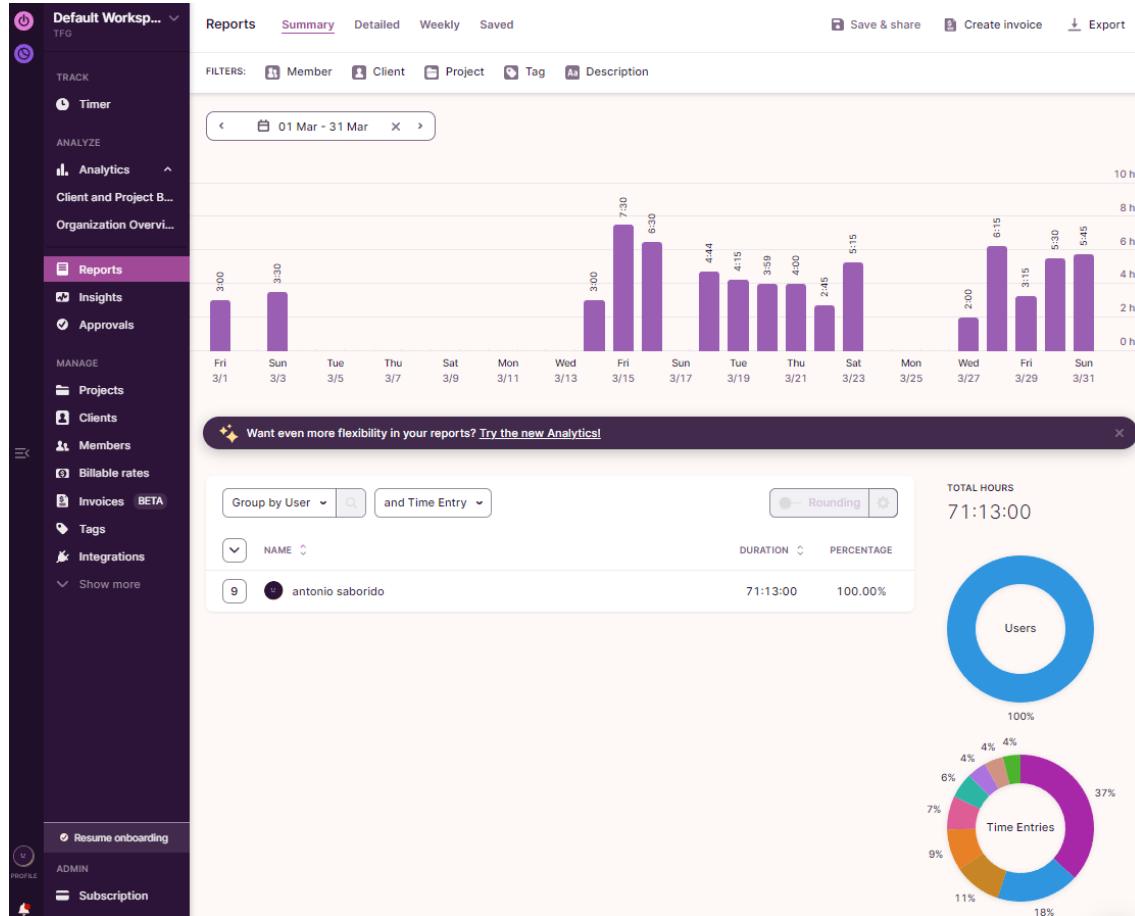


Ilustración 9. Resumen de tiempos en Toggl

Como se puede ver en Ilustración 9, otra de las funciones muy útiles de Toggle, es que puedes ver resúmenes de los meses anteriores, permitiéndote ver las horas empleadas por día y a que tarea estaban asignadas (en la parte inferior derecha, “Time Entries”).

Esto permite valorar el nivel de esfuerzo que ha llevado cada tarea y calcular los esfuerzos que han sido necesarios.

3.2.3 GOOGLE CALENDAR

Google Calendar es una herramienta de gestión del tiempo y planificación de eventos que permite programar reuniones, establecer recordatorios y compartir calendarios con otros miembros del equipo. En este proyecto, Google Calendar ha sido utilizado para coordinar reuniones y asegurar que todos los participantes estuvieran al tanto de las sesiones de planificación, revisiones de sprint y otros eventos importantes.

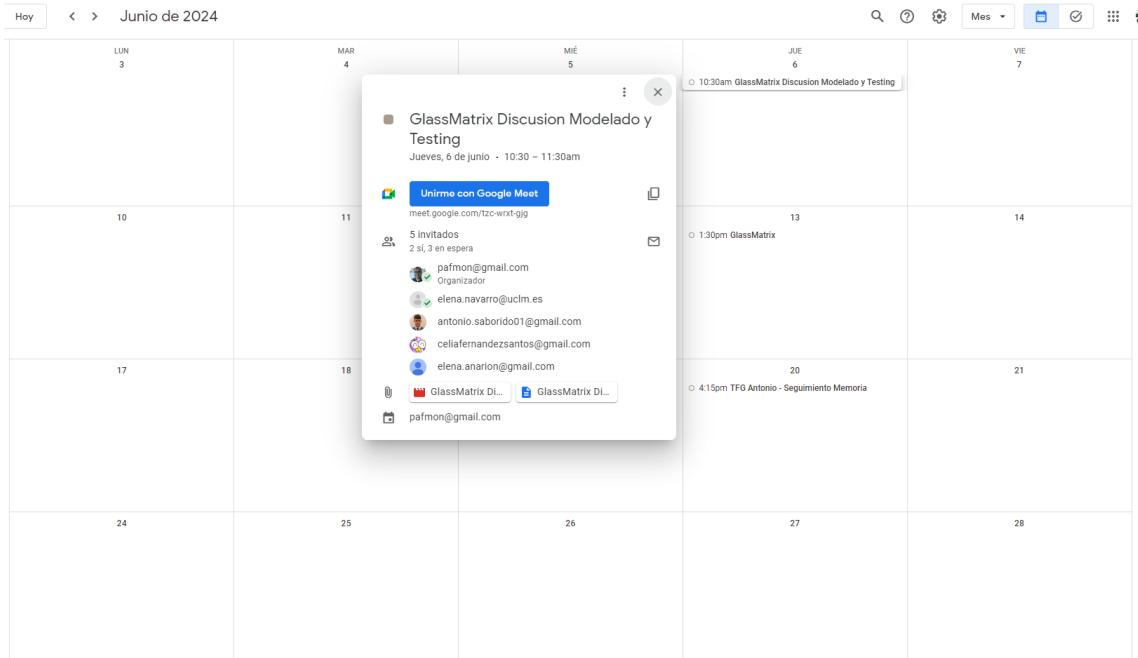


Ilustración 10. Ejemplo de reunión en Google Calendar

El funcionamiento de Google Calendar es intuitivo: los usuarios pueden crear eventos, invitar a otros participantes y recibir notificaciones antes de que los eventos ocurran.

Esta herramienta facilita la gestión del tiempo y asegura que todos los miembros del equipo estén sincronizados en cuanto a los horarios y compromisos. Además, Google Calendar permite visualizar la disponibilidad de los demás miembros, lo que facilita la programación de reuniones en momentos convenientes para todos.

Otra funcionalidad muy interesante de Google Calendar, es que permite volver a ver las reuniones grabadas gracias a la conexión con Google Meet. Por lo que permite revisar algunas reuniones en caso de dudas o malentendidos.

Las reuniones programadas y registradas en Google Calendar han sido esenciales para mantener una comunicación fluida y una organización efectiva del equipo.

3.2.4 GOOGLE CHAT

Google Chat es una plataforma de mensajería instantánea que facilita la comunicación y colaboración en tiempo real entre los miembros del equipo. En este proyecto, Google Chat ha sido utilizado para consultar dudas, compartir información rápidamente y mantener una comunicación constante y efectiva durante todo el proceso de desarrollo.

Google Chat permite la creación de salas de chat para equipos y proyectos específicos, así como la comunicación directa entre miembros individuales. La herramienta soporta la integración con otras aplicaciones de Google Workspace, lo que mejora la eficiencia y facilita el acceso a documentos y recursos compartidos.

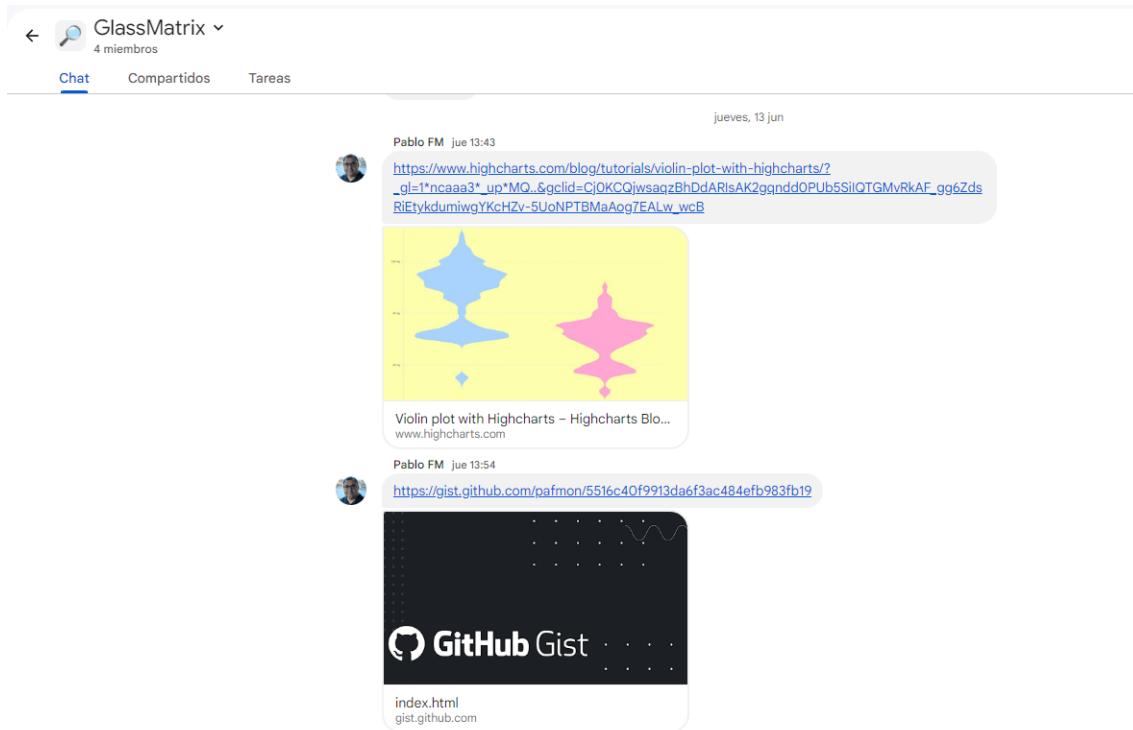


Ilustración 11. Workspace utilizado para el proyecto

Como se ve en la Ilustración 11, Google Chat ha permitido compartir enlaces interesantes para el desarrollo del proyecto. Además, aprovechando la conexión entre aplicaciones de Google, cualquier documento que se mande por el chat del workspace, se convertía visible para el resto de miembros agilizando así el proceso de revisión de documentos.

La capacidad de enviar mensajes instantáneos y realizar videollamadas ha sido crucial para resolver dudas y problemas rápidamente, asegurando que el flujo de trabajo no se vea interrumpido por falta de comunicación. La utilización de Google Chat ha permitido un alto nivel de colaboración y ha contribuido significativamente al éxito del proyecto.

3.3 DEFINICIÓN DE SPRINTS

La gestión del proyecto se ha estructurado en 11 sprints cuya duración ha ido variando desde las 2 semanas hasta las 4. Al final de cada sprint se concertaba una reunión entre autor y supervisor para definir el nuevo Sprint, con nuevas tareas y estimaciones. Además, también se hacía una revisión al sprint anterior para la revisión de lo que se hubiera desarrollado o posibles dudas que hubieran surgido.

Aquí se adjunta un desglose de cada uno de los sprints junto a su duración, esfuerzo en horas y carga de trabajo.

Sprints	Inicio - Fin	Núm días	Esfuerzo	Carga de trabajo
Sprint 1	21/09/23 a 19/10/23	28	24	5.30%
Sprint 2	20/10/23 a 20/11/23	31	39	8.61%
Sprint 3	21/11/23 a 18/12/23	27	40	8.83%
Sprint 4	19/12/23 a 21/01/24	33	29	6.41%
Sprint 5	22/01/24 a 06/02/24	15	26	5.73%
Sprint 6	07/02/24 a 04/03/24	26	60	13.24%
Sprint 7	05/03/24 a 01/04/24	27	40	8.83%
Sprint 8	02/04/24 a 24/04/24	22	55	12.14%
Sprint 9	25/04/24 a 07/05/24	12	50	11.04%
Sprint 10	08/05/24 a 04/06/24	27	50	11.04%
Sprint 11	05/06/24 a 20/06/24	15	40	8.83%

Tabla 1. Planificación de Sprints.

Tras la generación de sprints, en la gráfica, Ilustración 12, resultante de los esfuerzos por Sprint, se puede observar perfectamente la evolución de los mismos, con una carga mucho menor al principio, y un incremento de la misma a lo largo del tiempo con una posible reducción en los sprints 4 y 5, probablemente provocada por las fechas navideñas.

Carga de trabajo por sprints

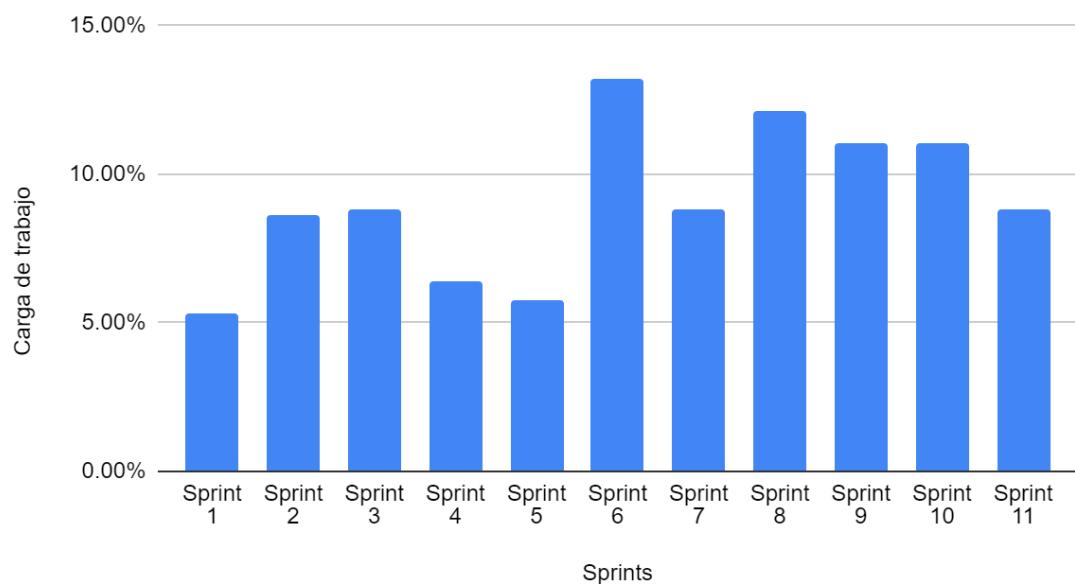


Ilustración 12. Carga de trabajo por Sprints.

3.4 DESGLOSE DE SPRINTS

En esta sección se lleva a cabo el desglose de las tareas que han sido realizadas en cada uno de los sprints que hemos definido anteriormente.

Sprint 1

El objetivo principal de este sprint inicial fue la investigación inicial de los frameworks y tecnologías que se podrían utilizar en el proyecto, así como un repaso a Node.js y un estudio de Angular y Grafana.

Sprint 1 (21/09/23 a 19/10/23)		
Tareas	Definición de tareas	Horas
1	Investigación frameworks a usar	3
2	Repasar Nodejs	5
3	Estudio Angular 13.3.0	6
4	Investigar sobre grafana	10

Tabla 2. Desglose de tareas Sprint 1.

Sprint 2

En este sprint, se profundiza en el ecosistema Bluejay ya que objetivo del proyecto es realizar una extensión de este

Otra parte clave es comprender Grafana, ya que Bluejay utiliza las visualizaciones propias que se encuentran incluida en Grafana. Y en relación a esto, es clave saber cuáles son los diferentes tipos de visualizaciones que ofrece Grafana por defecto.

También se realizó un pequeño estudio y pruebas con Docker-Compose ya que será útil para cuando se quiera desplegar el proyecto en un futuro.

Sprint 2 (20/10/23 a 20/11/23)		
Tareas	Definición de tareas	Horas
1	Investigación ecosistema Bluejay	15
2	Investigación y comprensión de Grafana	15
3	Estudio tipos de visualizaciones	4
4	Estudio y pruebas Docker-Compose	5

Tabla 3. Desglose de tareas Sprint 2.

Sprint 3

El objetivo de este sprint es el estudio de metodologías ágiles y en relación a esto, comprender el informe Dora, para comprender el modo adecuado de implantar estas metodologías ágiles.

Además de esto, gracias a la comprensión de Grafana alcanzada en el Sprint anterior, se desarrolla y se documenta la creación de un plugin personalizado en Grafana.

Sprint 3 (21/11/23 a 18/12/23)		
Tareas	Definición de tareas	Horas
1	Estudio de metodologías agiles	10
2	Investigación Informe Dora resumen	15
3	Desarrollo plugin personalizado en Grafana	15

Tabla 4. Desglose de tareas Sprint 3.

Sprint 4

Durante este sprint, se investigan y documentan las buenas prácticas en el desarrollo de software y los tipos de visualizaciones necesarios para el proyecto.

Sprint 4 (19/12/23 a 21/01/24)		
Tareas	Definición de tareas	Horas
1	Investigación sobre buenas prácticas en desarrollo software	5
2	Investigar y documentar tipos de visualizaciones	24

Tabla 5. Desglose de tareas Sprint 4.

Sprint 5

Este sprint marca el comienzo de la creación del proyecto Angular, incluyendo el diseño de la estructura de la página.

También se probaron endpoints de las APIs de bluejay mediante Postman. Estas funcionalidades serán utilizadas en el futuro por el proyecto Angular.

Sprint 5 (22/01/24 a 06/02/24)		
Tareas	Definición de tareas	Horas
1	Comienzo creación proyecto Angular	5
2	Diseño estructura de la página	15
3	Pruebas de funcionalidades con Postman	6

Tabla 6. Desglose de tareas Sprint 5.

Sprint 6

El objetivo principal de este sprint es la creación y documentación de un acuerdo de equipo (TPA) siguiendo, así como el desarrollo de la interfaz Angular y una API propia.

Sprint 6 (07/02/24 a 04/03/24)		
Tareas	Definición de tareas	Horas
1	Creación y documentación TPA	15
2	Desarrollo interfaz Angular	15
3	Desarrollo API propia	30

Tabla 7. Desglose de tareas Sprint 6.

Sprint 7

Durante este sprint, se enfoca en la documentación y definición del documento TPA, así como el desarrollo de la página de TPAs y métricas.

Sprint 7 (05/03/24 a 01/04/24)		
Tareas	Definición de tareas	Horas
1	Documentación y definición documento TPA	5
2	Desarrollo de página de TPAs	15
3	Desarrollo de página de métricas	20

Tabla 8. Desglose de tareas Sprint 7.

Sprint 8

El objetivo de este sprint es completar la implementación de las visualizaciones necesarias y realizar pruebas exhaustivas de las funcionalidades del proyecto.

Sprint 8 (02/04/24 a 24/04/24)		
Tareas	Definición de tareas	Horas
1	Revisión documentación de definición de TPAs	5
2	Pruebas sobre la página de métricas	20
3	Pruebas sobre la página de TPAs	30

Tabla 9. Desglose de tareas Sprint 8.

Sprint 9

En este sprint se finaliza el desarrollo del proyecto, corrigiendo errores y realizando los últimos ajustes y optimizaciones.

Sprint 9 (25/04/24 a 07/05/24)		
Tareas	Definición de tareas	Horas
1	Desarrollo página de testeo manual de TPAs	10
2	Desarrollo página de testeo automático	40

Tabla 10. Desglose de tareas Sprint 9.

Sprint 10

Este sprint se dedica a la preparación para la presentación final del proyecto. Para ello se realizan pruebas en las últimas herramientas desarrolladas y se comienza con el proceso de documentación tanto para el repositorio público de Github en el que se encontrará el proyecto desarrollado como para el documento que se está leyendo.

Sprint 10 (08/05/24 a 04/06/24)		
Tareas	Definición de tareas	Horas
1	Pruebas sobre la página de testeo automático	5
2	Creación de TPA con las herramientas desarrolladas	20
3	Documentar proyecto en Github	5
4	Documentar memoria	20

Tabla 11. Desglose de tareas Sprint 10.

Sprint 11

En el último sprint, se lleva a cabo la presentación del proyecto y se entrega toda la documentación.

Sprint 11 (05/06/24 a 20/06/24)		
Tareas	Definición de tareas	Horas
1	Documentar memoria	40

Tabla 12. Desglose de tareas Sprint 11.

Resumen temporal del proyecto

En la siguiente tabla se muestra el resumen temporal de los sprints que acabamos de desglosar.

Resumen del proyecto	
Fecha de inicio	21/09/2023
Fecha de fin	20/06/2024
Periodicidad de las revisiones	2-4 Semanas
Carga de trabajo semanal	12 horas
Horas totales previstas	300 horas
Horas finales	453 horas

Tabla 13. Resumen de tiempos y planificación del proyecto

En la Tabla 13, se puede observar que las horas totales previstas fueron 300, en cambio, finalmente han sido 453 horas, esta desviación es bastante significativa ya que es del 51%. Por ello en capítulo 8, de conclusiones, se detallan las causas de esta variación en la planificación.

3.5 PLANIFICACIÓN DE COSTES

Teniendo en cuenta la planificación por Sprints desglosada en el capítulo anterior, procedemos a realizar un estudio de los costes del proyecto. De este modo, se conseguirá formalizar un presupuesto inicial.

En este caso para elaborar el presupuesto inicial se han tenido en cuenta tanto costes de personal como costes materiales.

A continuación, se desglosan los presupuestos obtenidos tanto para costes de personal como para costes materiales.

3.5.1 COSTES DE PERSONAL

Programador. El sueldo para un Analista Programador se ha establecido en base al siguiente convenio:

XVIII Convenio colectivo estatal de empresas de consultoría y estudios de mercado y de la opinión pública, publicado en el BOE número 177, de 26 de julio de 2023. [6]

El sueldo anual para un Analista Programador es de 21.057,94 euros. También se establece en el convenio que el máximo de horas anuales será de 1800 horas, por tanto, se concluye que el salario por hora (€/h) para el rol de Analista Programador es:

$$\frac{21.057,94\text{€}}{1800 \text{ horas}} = 11,69\text{€/h}$$

A este coste se le debe añadir el coste de la Seguridad Social a cargo de la empresa, el cual se calcula aplicando un porcentaje sobre la base de cotización de cada trabajador. La seguridad social cubre los siguientes riesgos:

- Contingencias comunes: 23,6%
- Contingencias profesionales: 6,7%
- Desempleo: 0,1%
- Fondo de garantía salarial: 1,1%
- Formación profesional: 0,6%

Por lo tanto, el porcentaje total de la seguridad social es del 32,10%. El coste total por hora, incluyendo la seguridad social, es:

$$11,69\text{€/h} \times 1,3210 = 15,45\text{€/h}$$

Dado que el desarrollo de este proyecto requiere un total de 453 horas, el coste de personal total sería:

$$15,45 \text{ €/h} \times 453 \text{ horas} = 7.000,74 \text{ €}$$

3.5.2 COSTES MATERIALES

Para calcular el coste de los materiales se ha considerado tanto el gasto en software como en hardware.

3.5.2.1. HARDWARE

PRODUCTO	COSTE
Memoria RAM (8GB x 2)	129.99€
Disco SSD (240GB)	75.00€
Disco HDD (1TB)	60.00€
Pantalla 27''	169.05€
Procesador Intel I-7	270.00€
Tarjeta gráfica	232.99€
Placa base	130.34€
Fuente de alimentación	92.00€
Disipador	23.99€
Caja	69.99€
TOTAL	1.253,35 €

3.5.2.2 SOFTWARE

Concepto	Coste
Microsoft Windows 11 Pro	109,99€
Web Storm	159.00€/año
Github Pro	100 €/año
Microsoft Office	69,00 €/año
Google Drive	0€
Google Calendar	0€
Toggle Track	0€
TOTAL	437.99€

3.5.3 CONCLUSION COSTES

Cualquier activo fijo, ya sea tangible o intangible, que se integra en la estructura operativa de una empresa, sufre una depreciación o pérdida de valor con el tiempo. Esta pérdida de valor puede deberse a dos razones: el deterioro del activo por su uso o la obsolescencia tecnológica debido a la aparición de nuevos equipos o métodos que hacen obsoleto el activo.

Utilizando el método de amortización lineal, se distribuye de manera uniforme la amortización del activo a lo largo de su vida útil. Los conceptos manejados en este método son los siguientes:

- Valor residual: es el valor neto que se obtendría al vender los activos en el mercado vigente en la fecha de valuación, una vez finalizada su vida útil, operativa o tecnológica.
- Valor de reemplazo: es el valor de compra del bien equivalente a la fecha de relevamiento.
- Vida útil: es el tiempo en años durante el cual el bien puede ser utilizado normalmente, con un mantenimiento adecuado y en buenas condiciones operativas y tecnológicas.

La fórmula para calcular la amortización de un bien, utilizando este método, es la siguiente:

$$\text{Amortización} = (\text{Valor de reemplazo} - \text{Valor residual}) / \text{Vida útil}$$

Para los activos fijos considerados en los gastos hardware y el coste de licencia de Windows 11, se estima una vida útil de 6 años. Por lo tanto, procedemos a calcular los costes derivados de las amortizaciones:

- Vida útil = 6 años * 12 meses * 4 semanas * 7 días * 5 horas = 10.080h
- Valor residual 200€

Valor de reemplazo de hardware y Windows: 1.253,35€ + 109,99€ - 200€ = 1.163,34€

Por tanto, la amortización será: 1.163,34€ / 10.080h = 0,1154€/h.

La amortización del software será:

- Vida útil = 12 meses * 4 semanas * 7 días * 5 horas = 1.680h

Por tanto, la amortización será: 328,00€ / 1.680h = 0,1952€/h

La amortización total por hora será 0,1154€/h + 0,1952€/h = 0,3106€/h

Estimando que los activos, tanto Software como Hardware, serán usados en el desarrollo de este proyecto durante un total de 453 horas, el coste total de materiales sería el siguiente:

- $0.3105\text{€}/\text{h} * 453 = 140,65\text{€}$

Tras el estudio de los costes de personal y los costes materiales, acogidos al método de amortización lineal, el presupuesto total para este proyecto:

Coste total del proyecto	
Coste de personal	7.000,74
Coste material	140,65€
TOTAL	7.141,39

Tabla 14. Coste total del proyecto

CAPITULO 4: ANTECEDENTES

4.1 AUDITORIA DE BUENAS PRÁCTICAS DE EQUIPO (TPAs)

4.1.1 BLUEJAY

Bluejay es una plataforma extensible y de código abierto diseñada para auditar prácticas de equipo (Team Practices, TPs) en equipos ágiles de desarrollo de software, utilizando datos de múltiples herramientas de desarrollo. Su propósito principal es facilitar a los instructores y equipos la evaluación y el seguimiento de las prácticas ágiles a lo largo del tiempo, mejorando así la adherencia a estas prácticas y optimizando los procesos de desarrollo. [4]

Introducción y Motivación

Agile se ha convertido en una metodología estándar en la industria del software y en entornos educativos debido a su flexibilidad y eficacia. Sin embargo, auditar si los equipos siguen adecuadamente las prácticas ágiles puede ser complicado y consume mucho tiempo, especialmente cuando se utilizan múltiples herramientas. Bluejay aborda este problema permitiendo la auditoría de prácticas que involucran varias herramientas mediante la integración de sus APIs.

Implementación Técnica

Bluejay utiliza una arquitectura de microservicios basada en el ecosistema de Governify. A continuación, se explica para qué sirve cada microservicio:

Render

Render, es el servicio encargado de renderizar las vistas que se muestran en el navegador del cliente. Por lo tanto, es un servicio genérico cuya única tarea es renderizar archivos HTML, CSS y JS desde fuentes externas.

El componente de renderizado sigue el patrón de diseño MVC y los modelos, vistas y controladores se recuperan del gestor de activos. Actualmente, el servicio es capaz de renderizar vistas y controladores de Angular.js.

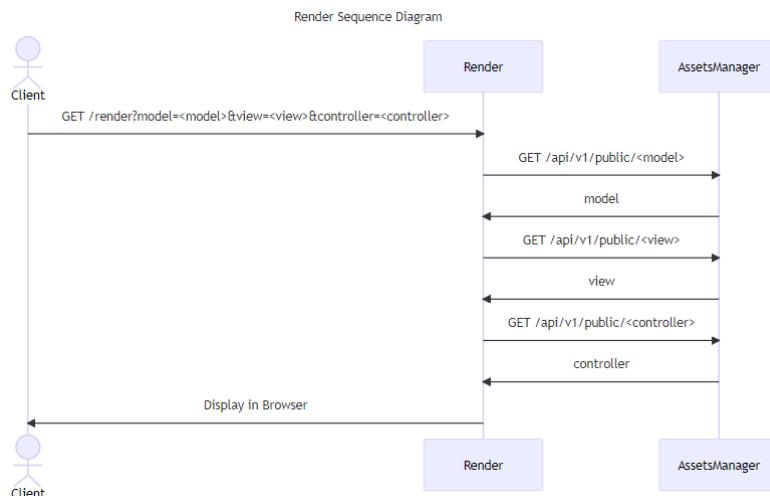


Ilustración 13. Diagrama de funcionamiento del microservicio render

Assets Manager

El gestor de *assets* es el componente central para servir todos los archivos que necesitan ser accedidos a través de la red. Un archivo que debe ser accedido por múltiples componentes (por ejemplo, `infrastructure.yaml` accedido por componentes que usan Governify Commons) puede colocarse en los *assets*.

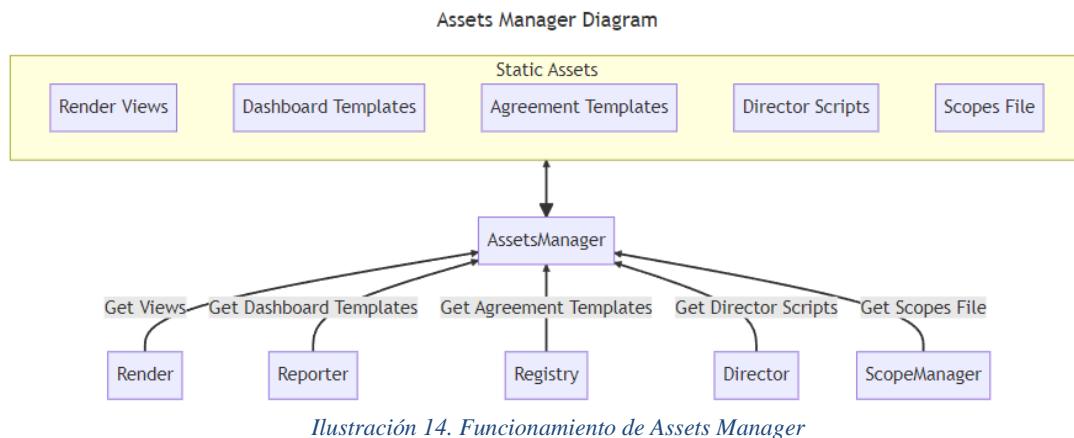


Ilustración 14. Funcionamiento de Assets Manager

Reporter

Como su nombre indica, el servicio *Reporter* es el encargado de reportar los resultados (estados) a los paneles para su visualización. Para ello, recibe los estados del servicio de *Registry* y los almacena en una base de datos de series temporales (InfluxDB), que es utilizada por el servicio de paneles como fuente de datos.

Los paneles en los *dashboards* (conjuntos de visualizaciones de Grafana) son generados por este microservicio en formato JSON, y luego son renderizados por los servicios de tablero.

Registry:

Es el servicio encargado de almacenar y gestionar la información relacionada con los Acuerdos de Nivel de Servicio (SLAs) registrados en la plataforma. La información se almacena en dos colecciones en una base de datos MongoDB. El servicio de *Registry* gestiona los estados que resultan del cálculo de métricas y garantías, y los propios acuerdos.

Dashboard:

Los paneles son una forma de visualizar datos de manera gráfica. Governify proporciona un servicio *dashboard* que permite crear y gestionar estos paneles. Este servicio está construido sobre Grafana para renderizar los paneles JSON a partir de los acuerdos, mientras coloca los puntos de InfluxDB almacenados por *Reporter* en esos paneles.

Al igual que el servicio Render, este es un componente genérico cuyo único propósito es renderizar paneles definidos en otros lugares. Por lo tanto, esta página pondrá más énfasis en la creación de paneles y la configuración del servicio.



Ilustración 15. Ejemplo de visualización de Grafana

Director:

El microservicio de *Director* es un servicio independiente y sencillo que ejecuta un conjunto de tareas periódicamente según la configuración. Las tareas se definen en un archivo JS y la configuración del intervalo se define en JSON; ambos archivos deben tener el mismo nombre y almacenarse en la carpeta /public/director del gestor de activos.

Una vez creados los archivos JS y JSON en el gestor de activos, cuando se crea la tarea en el director, este solicitará el script al gestor de activos en cada ejecución. Una vez que una tarea se crea en el director, puede estar en ejecución o detenida.

Scopes Manager:

El gestor de alcances es un servicio que permite gestionar diferentes alcances al auditar equipos ágiles. Se utiliza principalmente en la Infraestructura Bluejay debido a la necesidad de gestionar acuerdos para diferentes equipos dentro de una misma clase u organización.

Al usar el *Scope Manager*, se crea una plantilla de acuerdo para una clase u organización. Luego, diferentes equipos dentro de esa clase u organización se registran en el gestor de alcances para obtener su propio acuerdo basado en la plantilla. Esta interacción se muestra en el siguiente diagrama:

TPA Designer:

El TPA Manager, como dice su traducción es un diseñador de TPAs, una aplicación web que permite al usuario crear y editar Acuerdos de Prácticas de Equipo (TPAs) para el marco de trabajo Governify.

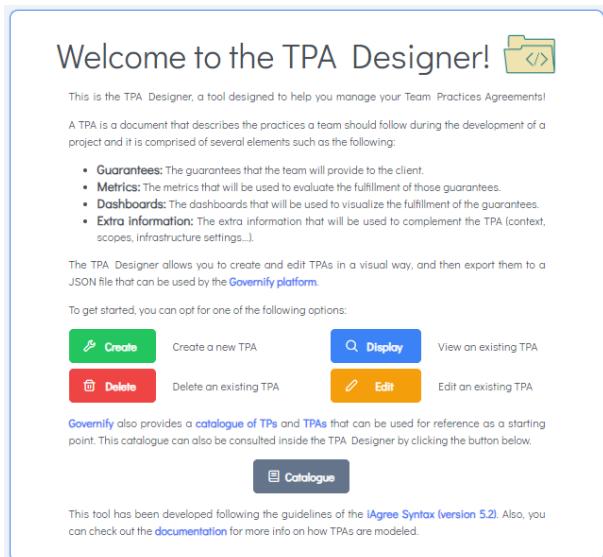


Ilustración 16. Visualización de microservicio de creación de TPAs

Collector events:

Los servicios de *Collectors* son los encargados de recopilar los datos necesarios para calcular las métricas y garantías definidas en los SLAs. Los *Collectors* son los que realmente calculan los valores y evidencias para las garantías, y luego los envían al microservicio de *Registry* para ser almacenados.

Personalización y Extensibilidad

Una de las principales fortalezas de Bluejay es su capacidad de personalización y extensibilidad:

- Adición de Nuevas Herramientas: Bluejay puede integrarse con cualquier herramienta que disponga de una API REST, permitiendo la adición de nuevas fuentes de datos.
- Definición de Nuevas Métricas: Los usuarios pueden definir nuevas métricas y patrones para adaptarse a las necesidades específicas de sus equipos.
- Documentación y Recursos: La plataforma proporciona una extensa documentación y ejemplos para ayudar a los usuarios a extender y personalizar el sistema según sus necesidades.

Impacto y Aplicaciones

Bluejay ha sido probado en entornos académicos, mostrando resultados positivos en la adherencia a las prácticas ágiles. Los estudios de caso han demostrado que la disponibilidad de TPAs y el uso de Bluejay mejoran el comportamiento de los equipos en cuanto a la entrega de tareas y el seguimiento de historias de usuario.

En resumen, Bluejay es una herramienta poderosa y flexible que facilita la auditoría y el seguimiento de prácticas ágiles en equipos de desarrollo de software. Al integrar datos de múltiples herramientas y proporcionar una visualización clara del cumplimiento de las prácticas, Bluejay ayuda a los equipos a mejorar continuamente sus

procesos y adherirse a las mejores prácticas definidas. Su capacidad de personalización y extensibilidad asegura que pueda adaptarse a una variedad de entornos y necesidades específicas.

4.1.2 TPAs

Un "*Team Practice Agreement*" (TPA) es un acuerdo que define un conjunto de prácticas que un equipo debe seguir para cumplir con ciertos objetivos y métricas, ayudando a auditar y monitorear el cumplimiento de dichas prácticas. En el contexto del marco Bluejay, que se utiliza para equipos ágiles en la ingeniería de software, los TPAs se modelan, monitorean y auditán a través de una serie de componentes y procesos.

Modelado de TPAs

El proceso comienza con la creación de los TPAs por parte de los desarrolladores, instructores o profesores, quienes definen las prácticas del equipo (TPs) y los objetivos asociados a cada una.

Cada TPA incluye múltiples TPs, que son prácticas específicas del equipo medidas por métricas concretas. Estas métricas se calculan utilizando la información recopilada de diferentes herramientas a través de patrones de métricas.

Estos patrones ayudan a estandarizar la recopilación y evaluación de datos desde distintas herramientas como Pivotal Tracker (PT) y GitHub, permitiendo medir aspectos como la cobertura de pruebas o el número de historias de usuario activas.

Desglose estructura de TPAs

Los TPAs se definen como documentos JSON o YAML que contienen toda la información necesaria para modelar, medir y auditar uno o más procesos o información relacionada con servicios en línea. El objetivo de este archivo es proporcionar suficiente información para recuperar datos de los servicios y medir que las garantías especificadas en el acuerdo se estén cumpliendo.

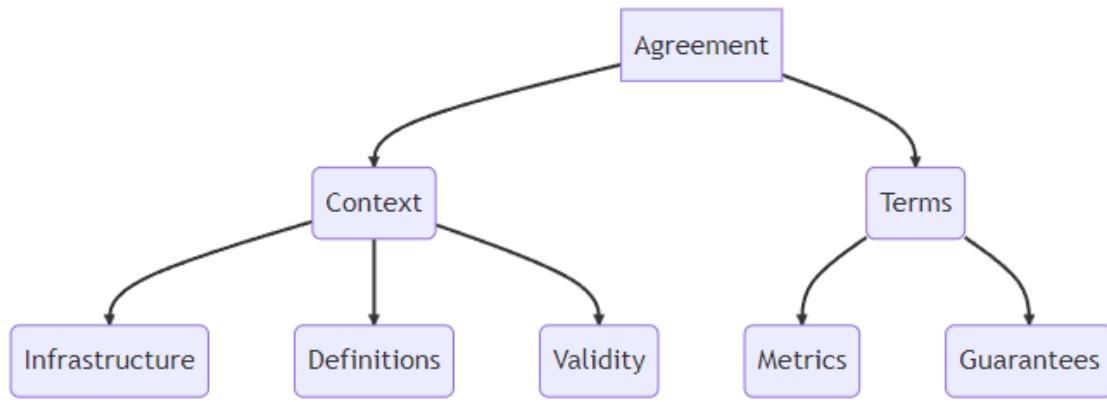


Ilustración 17. Estructura de un Acuerdo de Equipo

Como se puede ver en la Ilustración 17. Estructura de un Acuerdo de Equipo, los acuerdos de equipo, tienen dos grandes zonas o bloques. La primera sería la zona de contexto en el que trataríamos la Infraestructura, las definiciones y la validez del acuerdo. Y la segunda sería el bloque de los términos, que incluirá las métricas y las garantías.

Por tanto, traduciendo esto a un formato JSON o YAML, podríamos decir que un TPA tendrá la siguiente estructura:

```
1. {
2.   "id": "Plantilla de Acuerdo de Bluejay",
3.   "version": "1.0.0",
4.   "type": "agreement",
5.   "context": {
6.     "validity": {
7.       "initial": "2020-01-01",
8.       "timeZone": "America/Los_Angeles"
9.     },
10.    "infrastructure": {},
11.    "definitions": {
12.      "schemas": {},
13.      "scopes": {},
14.      "collectors": {},
15.      "dashboards": {}
16.    }
17.  },
18.  "terms": {
19.    "metrics": {},
20.    "guarantees": []
21.  }
22. }
```

Código 1. Estructura básica de un TPA.

Se puede observar que el contexto, contiene la información relacionada con los servicios de los cuales se están extrayendo los datos y la información adicional sobre el acuerdo, como la fecha de inicio/fin de validez, la zona horaria, etc.

Después, dentro del contexto, se debe especificar la validez del acuerdo con una fecha inicial y la zona horaria donde se aplica. En Código 1 se puede observar que la validez es desde el 1 de enero de 2020 y que utilizará la zona horaria de America, concretamente la de Los Angeles.

Infraestructura deberá contener la información sobre qué servicios deben usarse para calcular las métricas. Para recuperar los datos y comunicarse con los diferentes microservicios del ecosistema Governify o Bluejay, se deben especificar las URL donde se desplegará cada componente. Un ejemplo de esto podría ser el siguiente:

```
1. {
2.   "infrastructure": {
3.     "registry": "http://localhost:8081/api/v6",
4.     "reporter": "http://reporter.bluejay.governify.io/api/v4",
5.     "render": "https://ui.bluejay.governify.io/render?model=https://registry.bluejay.governify.io/api/v6/agreements/tpa-1010101010&view=/renders/tpa/default.html&ctrl=/renders/tpa/default.js",
6.     "dashboard": "http://dashboard.bluejay.governify.io",
7.     "scopeManager": "https://sm.bluejay.governify.io/api/v1"
8.   }
9. }
```

Código 2. Infraestructura de Bluejay dentro de un TPA.

Y por último en la última zona de contesto, es la zona de definiciones, que estará compuesta por “Scopes”, “Collectors” y Dashboards”.

El alcance (scope). El alcance define para qué servicio/persona/proceso aplica la métrica. Por ejemplo, si queremos medir una empresa, podemos tener tres alcances:

- Proyecto
- Equipo
- Miembro

```
1. {
2.   "scopes": {
3.     "development": {
4.       "project": {
5.         "name": "Proyecto",
6.         "description": "Proyecto",
7.         "type": "string",
8.         "default": "1010101010"
9.       },
10.      "class": {
11.        "name": "Clase",
12.        "description": "Agrupa algunos proyectos",
13.        "type": "string",
14.        "default": "2020202020"
15.      }
16.    }
17.  }
18. }
```

Código 3. Sección de Scopes dentro de un TPA.

En la sección de collectors, se deben especificar los servicios utilizados para recolectar información, con el fin de poder calcular los valores de las métricas.

```
1. {
2.   "computers": {
3.     "eventcomputer": {
4.       "url": "http://event.collector.bluejay.governify.io",
5.       "endpoint": "/computations",
6.       "apiVersion": "2",
7.       "config": {
8.         "scopeManager":
9.           "http://sm.bluejay.governify.io/api/v1/scopes/development"
10.      }
11.    }
12.  }
```

Código 4. Ejemplo para un colector de eventos en un Acuerdo de Prácticas de Equipo

Y la última sección dentro de la zona de contexto será la de dashboards. Aquí especificaremos como deberá pintar Grafana todos los datos que hayamos recolectado. Debe especificar al menos un tablero (dashboard).

```
1. {
2.   "dashboards": {
3.     "team-dashboard": {
4.       "default": true,
5.       "overlay": "/blocks/overlay.js",
6.       "base": "/blocks/base.json",
7.       "modifier": "/blocks/modifier.js",
8.       "config": {
9.         "blocks": [
10.           {
11.             "type": "text",
12.             "text": "Este es el dashboard de tu equipo."
13.           }
14.         ]
15.       }
16.     }
17.   }
18. }
```

```

11.          "type": "time-graph",
12.          "guarantee": "NUMBER_MASTER_PR_MERGE_WEEKLY_OVER_1_OR_EQUAL",
13.          "config": {}
14.      }
15.    ]
16.  }
17.}
18.
19.}

```

Código 5. Ejemplo de configuración de un tablero simple.

Por último, entraremos en el bloque de términos, donde definiremos las métricas y las garantías que tendrá este acuerdo de equipo.

Cada **métrica** en el acuerdo contiene toda la información necesaria para recuperar datos específicos de los servicios y obtener un valor final (Booleano, Entero, Cadena, etc.) a partir de estos datos. La configuración de la métrica depende del colector utilizado.

```

1. {
2.   "PERCENTAGE_OPENPR_FINISHEDSTORIES_BIND": {
3.     "collector": {
4.       "$ref": "#/context/definitions/collectors/eventcollector"
5.     },
6.     "measure": {
7.       "computing": "actual",
8.       "element": {
9.         "percentage": {
10.           "related": {
11.             "github": {
12.               "allPR": {
13.                 "head": {
14.                   "ref": "#CONTAINS(primary_resources.0.id)"
15.                 }
16.               }
17.             },
18.             "window": 86400
19.           }
20.         }
21.       },
22.       "event": {
23.         "pivotal": {
24.           "activity": {
25.             "highlight": "finished"
26.           }
27.         }
28.       },
29.       "scope": {
30.         "$ref": "#/context/definitions/scopes/development"
31.       }
32.     }
33.   }
34. }

```

Código 6. Ejemplo de una métrica en un Acuerdo de Prácticas de Equipo utilizando el colector de eventos.

Las **garantías** especifican la calidad o el rendimiento que deben tener las métricas. El objetivo es una fórmula que puede componerse a partir de una o más métricas. El objeto "window" se refiere al tiempo de cálculo, donde el período se utiliza para cómo se dividen los períodos de tiempo.

```

1. {
2.   "id": "75_PERCENT_NEWBRANCH_STARTEDSTORIES_WITHIN_A_DAY_BIND",
3.   "description": "Al menos el 75% de las historias iniciadas deben coincidir con la creación de una rama dentro de un día.",
4.   "scope": {

```

```

5.      "$ref": "#/context/definitions/scopes/development"
6.    },
7.    "of": [
8.      {
9.        "scope": {
10.          "project": "1010101010"
11.        },
12.        "objective": "PERCENTAGE_NEWBRANCH_STARTEDSTORIES_BIND >= 75",
13.        "with": {
14.          "PERCENTAGE_NEWBRANCH_STARTEDSTORIES_BIND": {}
15.        },
16.        "window": {
17.          "type": "static",
18.          "period": "daily",
19.          "initial": "2018-01-01"
20.        }
21.      }
22.    ]
23.  }

```

Código 7. Ejemplo de garantía utilizando la métrica del ejemplo anterior

Se puede ver el ejemplo de un TPA completo y funcional en el anexo 9.2.5 EJEMPLO TPA.

Monitoreo de TPAs

Una vez que los TPAs están definidos, Bluejay comienza a monitorearlos mediante la recopilación de datos de las herramientas externas especificadas. Esta información se procesa para calcular el cumplimiento de los TPAs, presentándose en un tablero gráfico accesible tanto para instructores como para los equipos. Bluejay usa componentes específicos llamados "colectores" para cada herramienta externa, encargándose de recolectar y calcular las métricas necesarias.

Auditoría de TPAs

Los instructores pueden utilizar las visualizaciones generadas por Bluejay para auditar el desempeño del equipo y de sus miembros individuales. Los gráficos y tablas resultantes muestran los valores obtenidos en función del tiempo y permiten identificar si se cumplen los objetivos establecidos. Este proceso facilita la detección de desviaciones y la identificación de áreas que requieren mejoras.

Personalización y Extensión

El sistema Bluejay está diseñado para ser extensible y personalizable. Los TPAs se pueden adaptar para incluir nuevas herramientas externas y métricas adicionales, permitiendo a los instructores crear TPAs específicos para las necesidades de su equipo. Además, Bluejay ofrece un catálogo de ejemplos de TPAs y métricas que los usuarios pueden utilizar como punto de partida.

En resumen, un TPA es un acuerdo que estructura y define las prácticas de equipo en un contexto ágil, permitiendo su monitoreo y auditoría sistemática a través de la plataforma Bluejay.

4.2 VISUALIZACIÓN DE TPAS

Las visualizaciones del dashboard que utiliza Bluejay se hacen en grafana, para ello podríamos utilizar las visualizaciones básicas que incluye Grafana o utilizar visualizaciones externas.

4.2.1 VISUALIZACIONES BASE DE GRAFANA

Grafana ofrece una variedad de visualizaciones por defecto. Podemos dividir estas en 4 grupos, gráficos y diagramas, estadísticas y números, widgets y varios. A continuación, detallo que visualizaciones ofrecen cada widget.

4.2.1.1 GRÁFICOS Y DIAGRAMAS

- **Time series:** La serie temporal es la visualización principal y predeterminada del gráfico.
- **State timeline:** Línea de tiempo de estado para cambios de estado a lo largo del tiempo.
- **Status history:** Historial de estado para estados periódicos a lo largo del tiempo.
- **Bar chart:** El gráfico de barras muestra datos categóricos.
- **Histogram:** El histograma calcula y muestra la distribución de valores en un gráfico de barras.
- **Heatmap:** El mapa de calor visualiza datos en dos dimensiones, utilizando típicamente para la magnitud de un fenómeno.
- **Pie chart:** El gráfico circular se utiliza típicamente cuando la proporción es importante.
- **Candlestick:** El gráfico de velas se utiliza típicamente para datos financieros donde el enfoque es el movimiento de precios/datos.

4.2.1.2 ESTADÍSTICAS Y NÚMEROS

- **Stat:** Estadísticas para estadísticas importantes y una línea de tendencia opcional.
- **Bar gauge:** La barra de medición es una barra de medición horizontal o vertical.

4.2.1.1 WIDGETS

- **Dashboard list:** Muestra los paneles de control.
- **Alert list:** La lista de alertas puede enumerar alertas.
- **Text:** Puede mostrar markdown y HTML.
- **News:** Puede mostrar fuentes RSS.

4.2.1.4 VARIOS

- **Table:** La tabla es la visualización principal y única para tablas.
- **Logs:** Los registros son la visualización principal para registros.
- **Node graph:** El gráfico de nodos es para gráficos dirigidos o redes.
- **Traces:** Las trazas son la visualización principal para trazas.
- **Flame Graph:** El gráfico de llamas es la visualización principal para perfiles.

4.2.2 COMPARATIVA GRÁFICOS DISPONIBLES EN GRAFANA

Grafana es una herramienta ampliamente utilizada para la visualización y monitoreo de datos gracias a que ofrece una amplia gama de opciones para la representación de datos. Pero como hemos visto en el punto anterior no ofrece gran cantidad de visualizaciones por defecto, por ello a continuación, voy a analizar qué tipos de gráficos son soportados directamente por grafana, cuales de ellos necesitan plugin externos que podemos instalar desde grafana y para qué tipo de gráficos será necesario crear un plugin a medida.

Para analizar todos los tipos de visualizaciones vamos a dividir estas en 9 grandes grupos basándose en un artículo del Financial Times llamado Visual Vocabulary [7].

4.2.2.1 CORRELACIÓN

Muestra la relación entre dos o más variables.

Columna + línea temporal

Una buena manera de mostrar la relación entre la cantidad (columnas) y una tasa (línea). Esta visualización no está disponible en Grafana y será necesario crear un plugin a medida.

Scatterplot

En castellano llamado gráfico de dispersión, es la forma de mostrar la relación entre dos variables continuas donde cada una utiliza su propio eje. Una variante de este gráfico sería el gráfico de dispersión conectado que se utiliza para mostrar cómo ha cambiado la relación entre dos variables a lo largo del tiempo.

Podríamos llegar a crear una gráfica de dispersión utilizando la visualización “Time series” y en la sección “graph styles” señalar únicamente puntos, pero es más recomendable el plugin externo “scatter” que proporciona tanto la visualización scatterplot como el scatterplot conectado de una manera más rápida y sencilla.

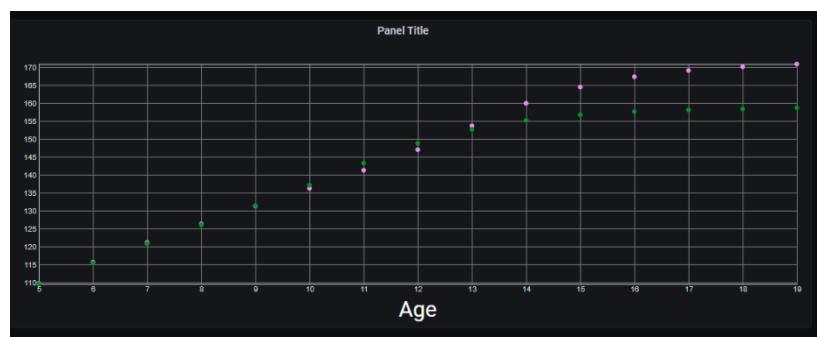


Ilustración 18. Scatterplot con el plugin “Scatter”

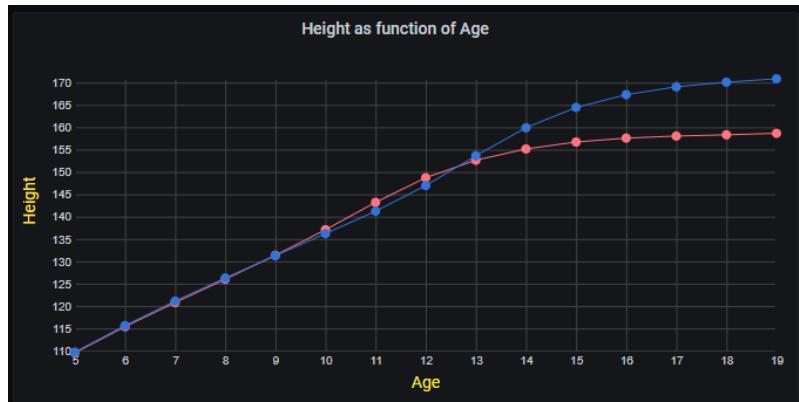


Ilustración 19. Scatterplot conectado con el plugin “Scatter”

Burbujas

Similar a un gráfico de dispersión, pero agrega detalle adicional al dimensionar los círculos de acuerdo con una variable.

Grafana no incluye ninguna visualización de este tipo pero si podemos encontrar un plugin llamado “Bubble Chart” que sí nos permite hacer estas visualizaciones. Aunque este plugin no está firmado tiene más de 8 millones de descargas.

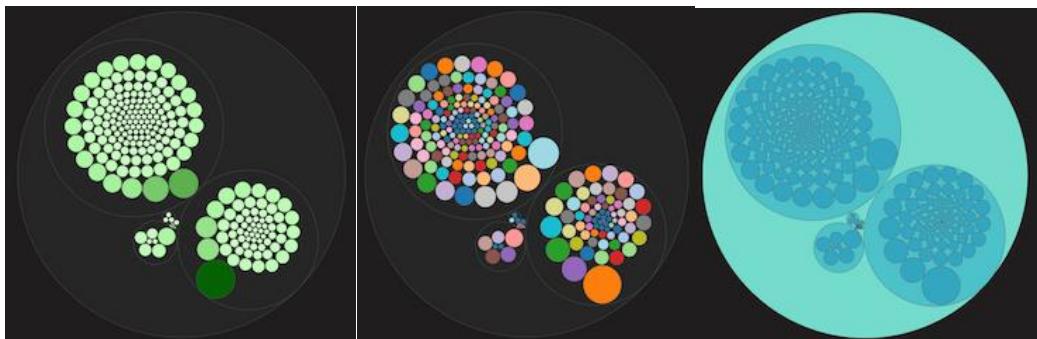


Ilustración 20. Visualización plugin “Bubble Chart”

Mapa de calor XY

Una buena manera de mostrar los patrones entre 2 categorías de datos, menos efectiva para mostrar diferencias precisas en las cantidades. Grafana trae por defecto la visualización “Heatmap”, pero esta funciona como un histograma a lo largo del tiempo. El eje y se mantiene constante.

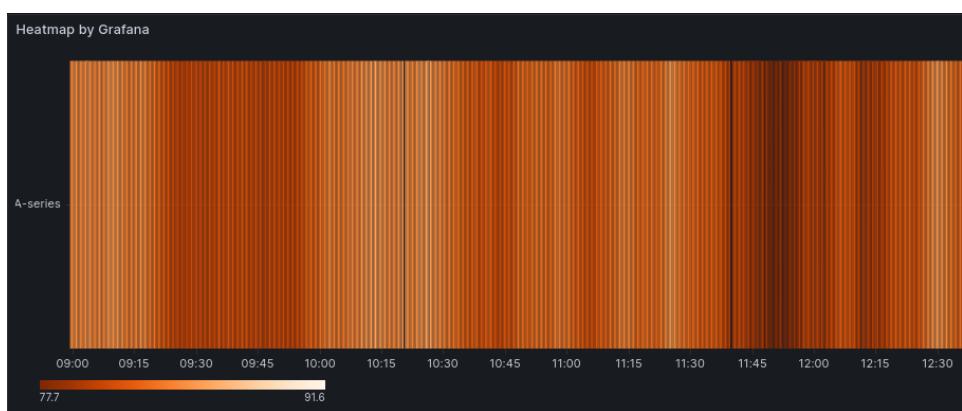


Ilustración 21. Visualización “Heatmap” de Grafana.

Pero en los plugins disponibles, podemos encontrar “Cal-heatmap” que este si permite un mapa de calor que cambia los ejes x e y. Esto se puede ver en la siguiente imagen.

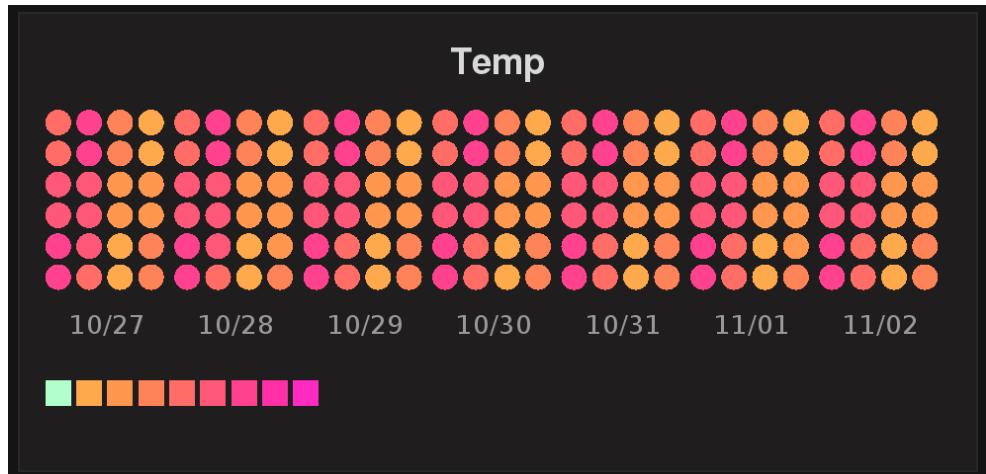


Ilustración 22. Visualización plugin “Cal-Heatmap”

4.2.2.2 RANKING

Se usan cuando la posición de un elemento en una lista ordenada es más importante que su valor absoluto o relativo. A parte de las visualizaciones que se muestran a continuación, habría que crear un plugin a medida para:

- **Ordered proportional symbol:** Este tipo de gráfico se utiliza cuando existen variaciones significativas entre los valores y/o cuando no es tan importante identificar diferencias entre los datos.

Barras y columnas ordenadas

Ordenar columnas y barras se podría llegar hacer con las visualizaciones por defecto de grafana, pero resulta mucho más fácil y rápido de realizar mediante el plugin externo “Multistat”. Así quedarían con este plugin:

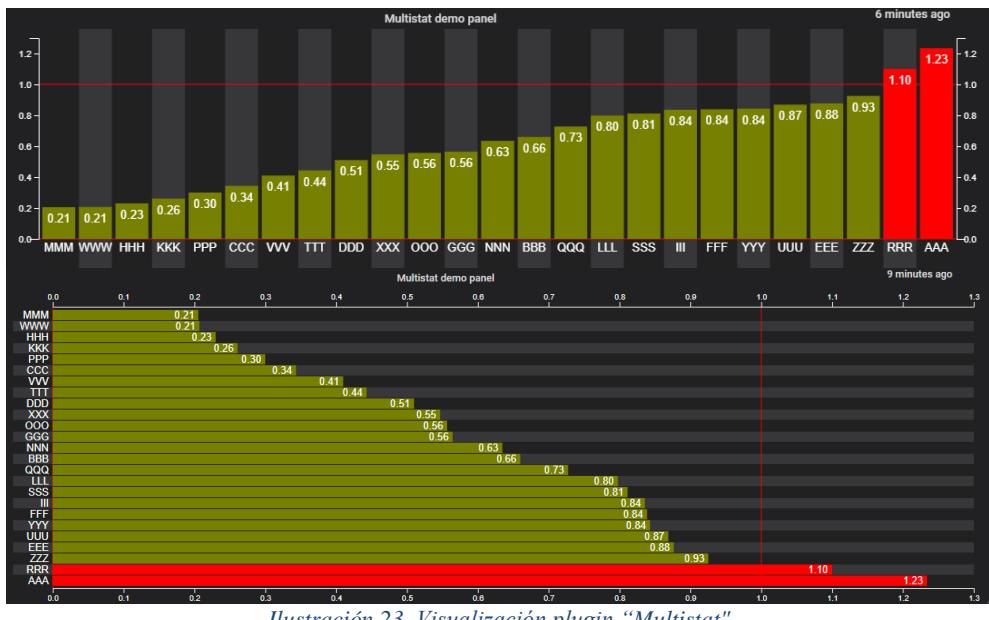


Ilustración 23. Visualización plugin “Multistat”

Slope

Perfecto para mostrar cómo han cambiado las clasificaciones con el tiempo o varían entre categorías. No viene de forma nativa en grafana por lo que requiere el plugin “Slope Graph Panel” para su visualización.



Ilustración 24. Visualización plugin “Slope Graph Panel” en Grafana

Bump

Efectivo para mostrar cambios en las clasificaciones a lo largo de múltiples fechas. Para conjuntos de datos grandes, considera agrupar las líneas utilizando colores. No está directamente soportado por grafana por lo que es necesario el plugin externo “Bump Chart Panel”.

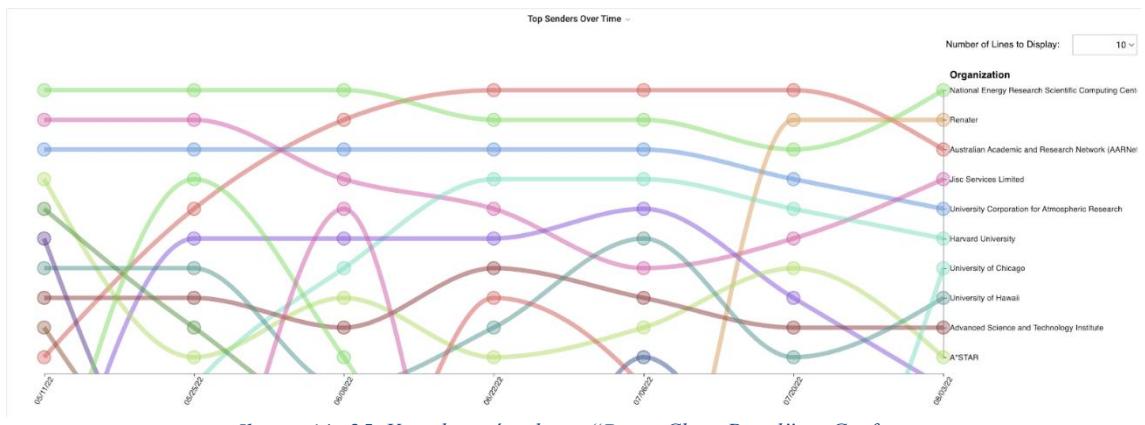


Ilustración 25. Visualización plugin “Bump Chart Panel” en Grafana

4.2.2.3 DISTRIBUCIÓN

Muestran los valores en un conjunto de datos y con qué frecuencia ocurren. A demás de las visualizaciones que se muestran a continuación, hay 3 visualizaciones que no están disponibles y habría que crear un plugin a medida, estas son:

- **Diagrama de caja:** En inglés Boxplot, resume múltiples distribuciones mostrando la mediana y el rango de los datos.
- **Gráfico de violín:** Similar a un diagrama de caja (boxplot), pero más efectivo con distribuciones complejas (datos que no pueden resumirse con un promedio simple).
- **Pirámide de población:** Una forma estándar de mostrar la distribución de edades y género en una población; realmente consiste en dos histogramas enfrentados.

Histograma

El histograma calcula la distribución de valores y la presenta en forma de gráfico de barras. El eje Y la altura de cada barra representan la cantidad de valores que caen en cada intervalo, mientras que el eje X representa el rango de valores.

Esta visualización se encuentra directamente soportado por grafana y no es necesario ningún plugin externo.



Ilustración 26. Visualización histograma en Grafana

Dot plot

Es útil para mostrar valores individuales en una distribución, pero puede ser un problema cuando hay demasiados puntos con el mismo valor.

Tanto el gráfico “dot plot” como el gráfico “dot strip plot” se pueden obtener configurando la visualización “time series” que viene por defecto en grafana.



Ilustración 27. Visualización Dot strip plot en Grafana

Barcode plot

Similar a los gráficos de puntos (dot strip plots), son buenos para mostrar todos los datos en una tabla y funcionan mejor cuando se destacan valores individuales.

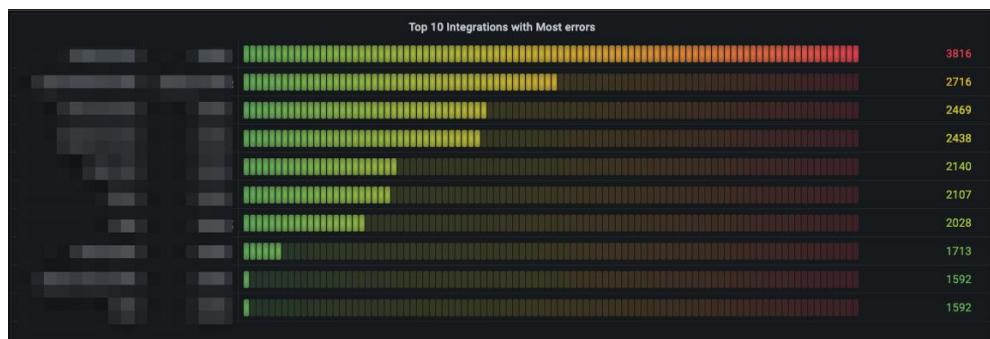


Ilustración 28. Visualización Barcode plot en Grafana

Polígonos de frecuencia

Para mostrar múltiples distribuciones de datos. Similar a un gráfico de líneas regular. Se puede implementar con la visualización por defecto de grafana “time series”.



Ilustración 29. Visualización de polígonos de frecuencia en Grafana

Cumulative curve

Una buena forma de mostrar cuán desigual es una distribución: el eje Y es siempre la frecuencia acumulada, y el eje X es siempre una medida. Para esta visualización es necesario el plugin “CDF - Cumulative Distribution Function”.

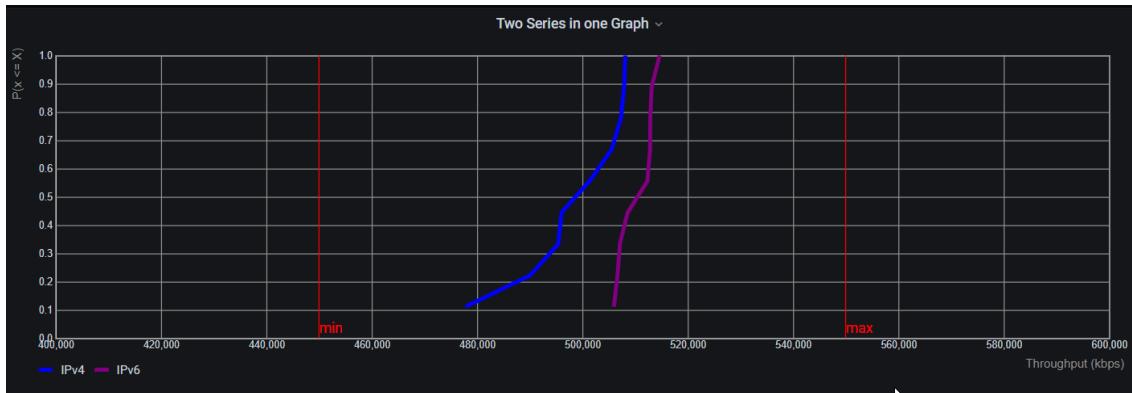


Ilustración 30. Visualización plugin “CDF - Cumulative Distribution Function” en Grafana

4.2.2.4 CAMBIANTES EN EL TIEMPO

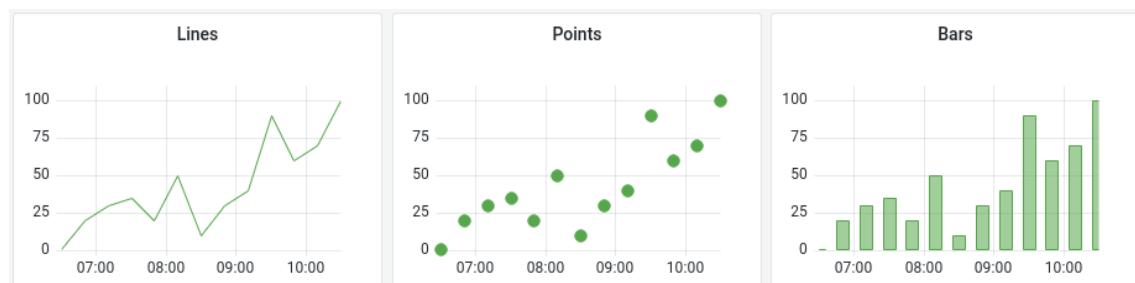
Ponen énfasis en las tendencias cambiantes. Estas pueden ser movimientos breves (intra-día) o series extendidas que abarcan décadas o siglos. Dentro de estas visualizaciones también se podría volver a incluir el mapa de calor y el scatterplot conectado que se encontraban en correlación. Además de estas dos visualizaciones, y del resto que veremos a continuación, las siguientes no se pueden crear en grafana ni con ningún plugin por lo que es necesario crear un plugin a medida para poder visualizarlas.

- **Columna + Línea temporal:** Una buena manera de mostrar la relación a lo largo del tiempo entre una cantidad (columnas) y una tasa (línea).
- **Gráfico de área:** Son buenos para mostrar cambios en el total, pero ver cambios en los componentes puede ser muy difícil.

- **Gráfico de abanico** (Fan chart): Se usan para mostrar la incertidumbre en proyecciones futuras, generalmente esta incertidumbre aumenta a medida que la proyección se extiende más en el futuro.
- **Línea de tiempo de Priestley** (Priestley timeline): Excelente cuando la fecha y la duración son elementos clave.
- **Línea temporal circular** (Circle timeline): Bueno para mostrar valores discretos de diferentes tamaños en múltiples categorías.
- **Línea del tiempo vertical** (Vertical timeline): Presenta el tiempo en el eje Y. Es bueno para mostrar series temporales detalladas que funcionan especialmente bien al desplazarse en dispositivos móviles.
- **Sismograma**: Otra alternativa a la línea de tiempo circular para mostrar series en las que existen grandes variaciones en los datos.
- **Streamgraph**: Un tipo de gráfico de área; es usado cuando se quieren ver cambios en las proporciones a lo largo del tiempo.

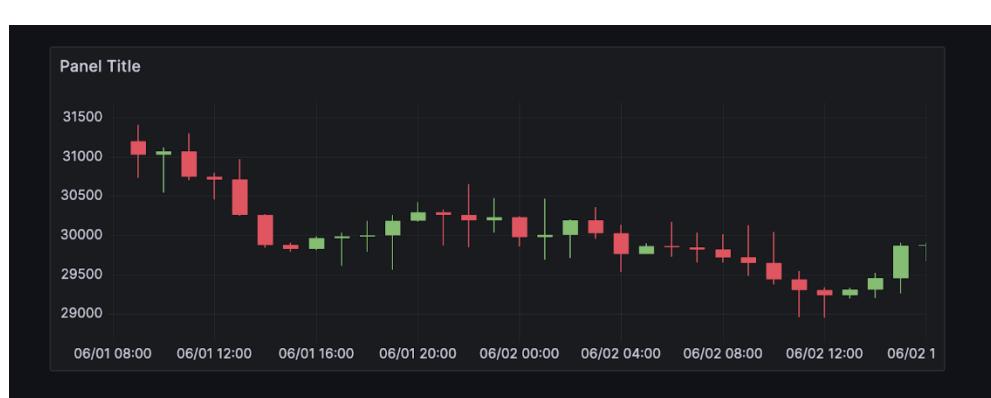
Línea

La forma estándar de mostrar una serie temporal cambiante. Viene integrada en grafana dentro de “Time series”.



Candlestick

Por lo general, se centran en la actividad diaria, estos gráficos muestran los puntos de apertura/cierre y los puntos más altos/bajos de cada día. Este gráfico viene ya incluido en grafana en la visualización con su mismo nombre, “candlestick”.



4.2.2.5 DESVIACIÓN

Este tipo de visualizaciones resalta las variaciones (+/-) desde un punto de referencia fijo. Normalmente, el punto de referencia es cero, pero también puede ser un objetivo o un promedio a largo plazo. En este grupo podemos encontrar las visualizaciones Diverging bar, Diverging stacked bar, Surplus/deficit filled line y Spine, pero ninguna de ellas viene integrada en grafana por lo que necesitan de un plugin personalizado.

- **Diverging bar:** Un gráfico de barras estándar sencillo que puede manejar tanto valores negativos como positivos.
- **Diverging stacked bar:** Permite más opciones que la anterior, por ejemplo, en esta visualización se podrían mostrar resultados como, de acuerdo, neutro o en desacuerdo.
- **Spine:** Divide un único valor en dos componentes contrapuestas
- **Surplus/deficit filled line:** El área sombreada de estos gráficos permite mostrar un equilibrio, ya sea contra una línea base o entre dos series.

4.2.2.6 MAGNITUDES

Muestran comparaciones de tamaño. Estas pueden ser relativas (simplemente poder ver lo más grande/más grande) o absolutas (necesidad de ver diferencias finas). En el caso de las magnitudes también encontramos algunas visualizaciones que requieren de un plugin personalizado:

- **Isotipo:** Es un pictograma y se recomienda usar sólo con números enteros.
- **Símbolos agrupados:** Una alternativa a los gráficos de barras o columnas cuando es útil poder contar datos o resaltar elementos individuales.
- **Marimekko:** Una buena manera de mostrar el tamaño y la proporción de los datos al mismo tiempo, siempre y cuando los datos no sean demasiado complicados.
- **Lollipop:** Los gráficos de chupachups (lollipop charts) llaman más la atención sobre el valor de los datos que las barras o columnas estándar.

Columnas y Barras

Son la forma estándar de comparar el tamaño de alguna magnitud. Esta visualización viene por defecto en Grafana mediante la visualización “time series”.

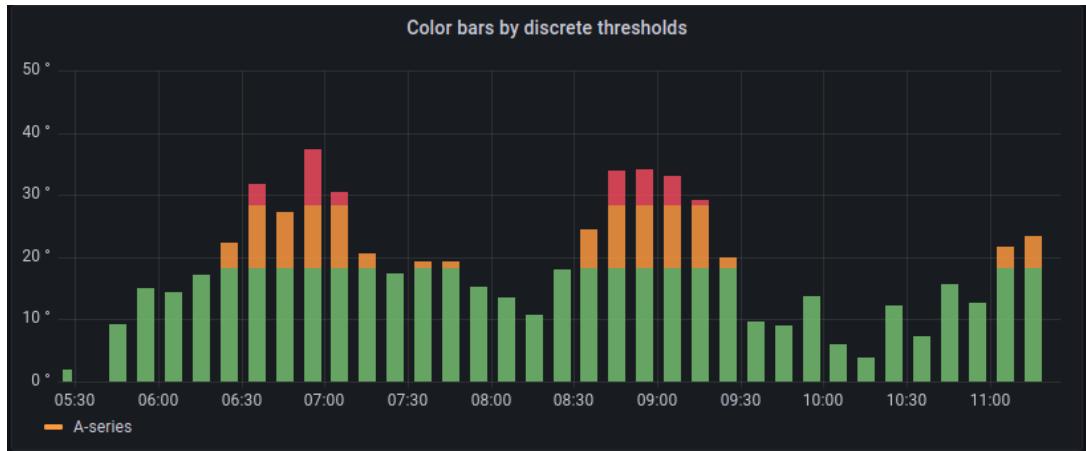


Ilustración 33. Visualización “Time series” en Grafana

Columnas y Barras emparejadas

Estas visualizaciones vienen por defecto en grafana en “Bar chart”.



Ilustración 34. Visualización “Bar chart” Grafana

Radar

Una forma eficiente de espacio para mostrar el valor de múltiples variables. No está directamente soportado en grafana por lo que es necesario el plugin “Radar Graph”.

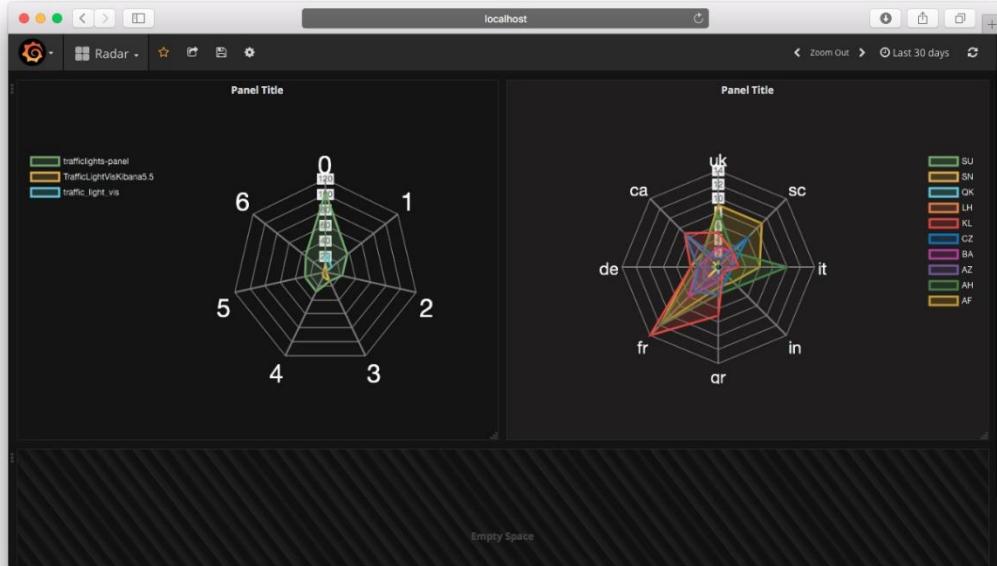


Ilustración 35. Visualización plugin Radar Graph

Bullet

Buenos para mostrar una medición en el contexto de un objetivo o rango de rendimiento. Viene ya implementada en Grafana en la visualización “State timeline”.

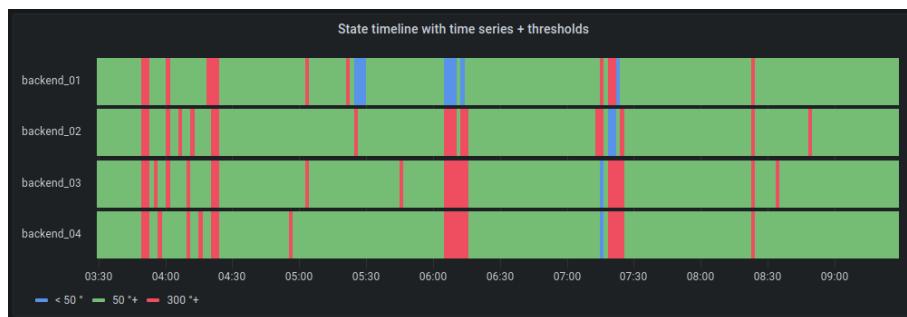


Ilustración 36. Visualización “State timeline” de Grafana

4.2.2.7 PARTES DE UN TODO

Muestran cómo una entidad puede descomponerse en sus componentes. Aparte de las visualizaciones que se muestran a continuación también podría formar parte de este grupo el gráfico en cascada, pero este se encuentra en los gráficos de flujos. En este caso requerirán de un plugin externo:

- **Voronoi:** Una forma de convertir puntos en áreas: cualquier punto dentro de cada área está más cerca del punto central que de cualquier otro centroide.

- **Arco:** Un hemiciclo, a menudo utilizado para visualizar la composición parlamentaria por número de escaños.
- **Diagrama de Venn:** Generalmente solo se utiliza para representación esquemática.

Stacked column/bar

Una forma sencilla de mostrar relaciones de "partes de un todo", pero puede ser difícil de leer cuando hay más de unos pocos componentes. No se puede implementar directamente, es necesario el plugin "plotly".



Ilustración 37. Visualización plugin "plotly"

Pie

El gráfico de tarta o pie es una forma de mostrar los datos que forman parte de algo más grande. Esta visualización viene por defecto en grafana.

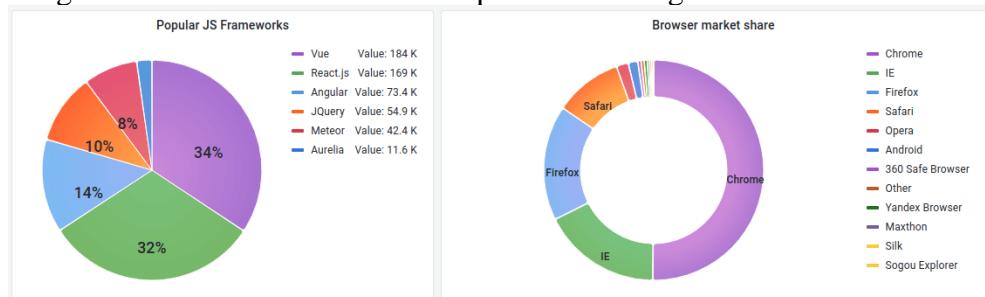


Ilustración 38. Visualización Pie chart en Grafana

Donut

Similar a un gráfico circular, pero el centro puede ser una buena manera de crear espacio para incluir más información sobre los datos, como el total. Este gráfico también se puede crear usando la visualización Pie chart y dentro de esta visualización cambiando el tipo a Donut.

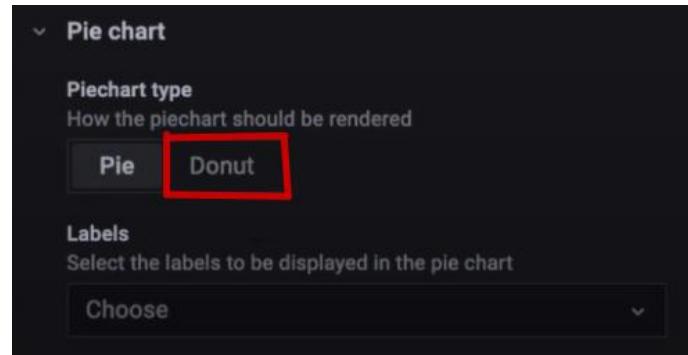


Ilustración 39. Ajuste para cambiar un gráfico a tipo Donut

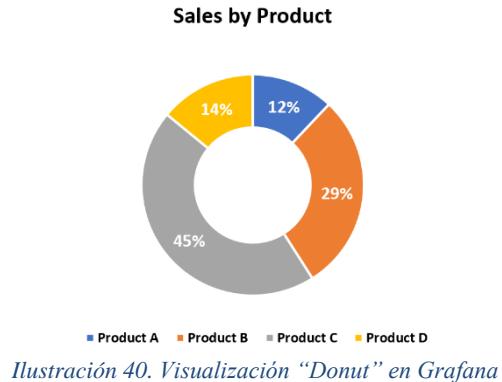


Ilustración 40. Visualización “Donut” en Grafana

Treemap

Se utiliza para representar relaciones jerárquicas de "partes de un todo", pero puede ser difícil de leer cuando hay muchos segmentos pequeños. Viene ya implementada por defecto en grafana en la visualización “Treemap”.



Ilustración 41. Visualización Treemap en Grafana

Gridplot

Son útiles para mostrar información porcentual y funcionan mejor cuando se utilizan con números enteros, además funcionan bien en una disposición de "pequeños

múltiples" (varios gráficos pequeños y similares). Esto viene por defecto en grafana en la visualización “Stat”.

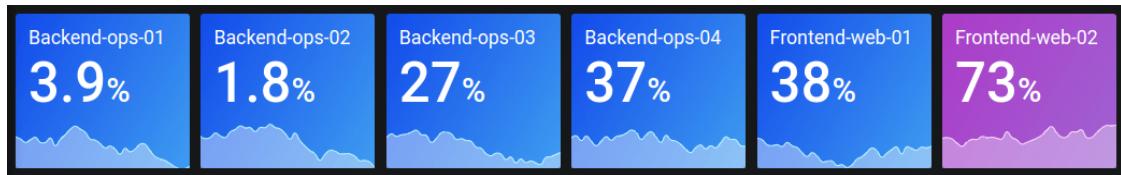


Ilustración 42. Visualización Stat en Grafana

4.2.2.8 ESPACIALES

Se utilizan mapas de localización cuando las ubicaciones precisas o los patrones geográficos en los datos son más importantes para el lector que cualquier otro aspecto.

- Basic choropleth
- Proportional symbol
- Flow map
- Contour map
- Equalized cartogram
- Scaled cartogram
- Dot density
- Heat map

Todos estos gráficos se encuentran incluido en grafana por defecto dentro de la visualización “Geomap”.



Ilustración 43. Visualización Geomap en Grafana

4.2.2.9 GRÁFICOS DE FLUJO

Son un tipo de representación gráfica que se utiliza para visualizar el flujo o movimiento de objetos, información o procesos a través de un sistema o una secuencia de etapas.

Sankey

Un diagrama de Sankey, también conocido como diagrama de flujo de energía o diagrama de flujo de materia, es una representación gráfica que muestra las relaciones entre varias entidades o categorías a través de flechas de diferentes grosos. Para la visualización de este gráfico es necesario el plugin externo “Sankey Panel”.

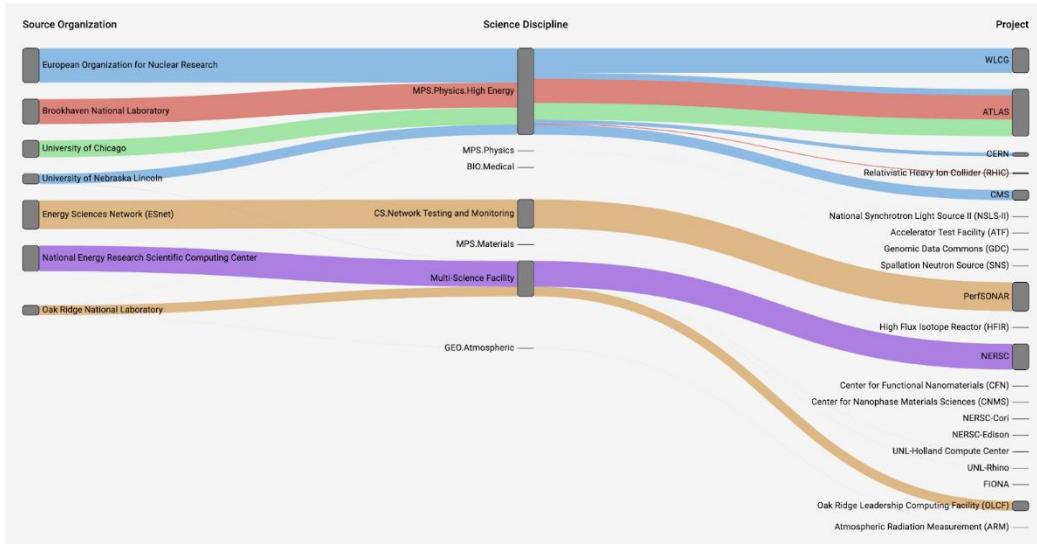


Ilustración 44. Visualización plugin “Sankey Panel”

Waterfall

Un gráfico de cascada, o "waterfall" en inglés, es un tipo de gráfico utilizado para representar el flujo de valores a lo largo de una secuencia de etapas o categorías. Esta visualización se puede obtener mediante el plugin “waterfall panel” ya que no está directamente soportada por grafana.

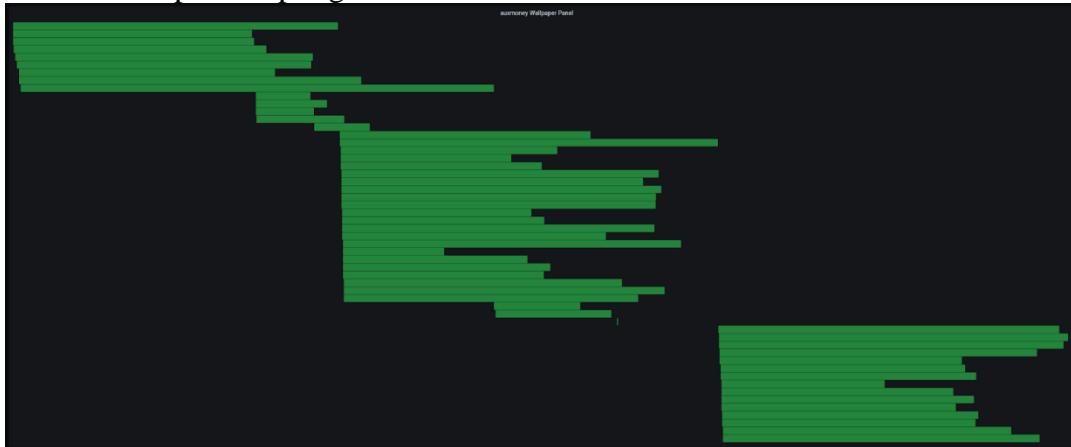


Ilustración 45. Visualización plugin Waterfall panel

Chor

Un gráfico de acordes, o "chord", es una representación visual que muestra las relaciones y conexiones entre entidades o categorías. Para esta visualización es necesario el plugin “ESnet Chord”.

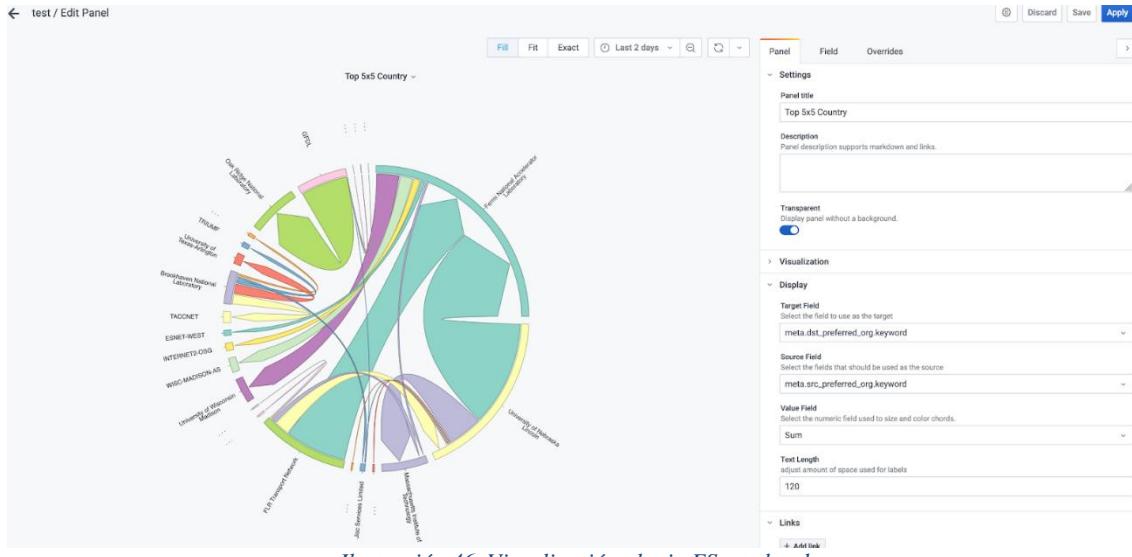


Ilustración 46. Visualización plugin ESnet chord

Network

Un gráfico de red, o "network", es una representación gráfica de un conjunto de nodos (puntos) interconectados por enlaces (líneas). Esto en grafana lo podemos obtener por defecto con la visualización "Node graph".

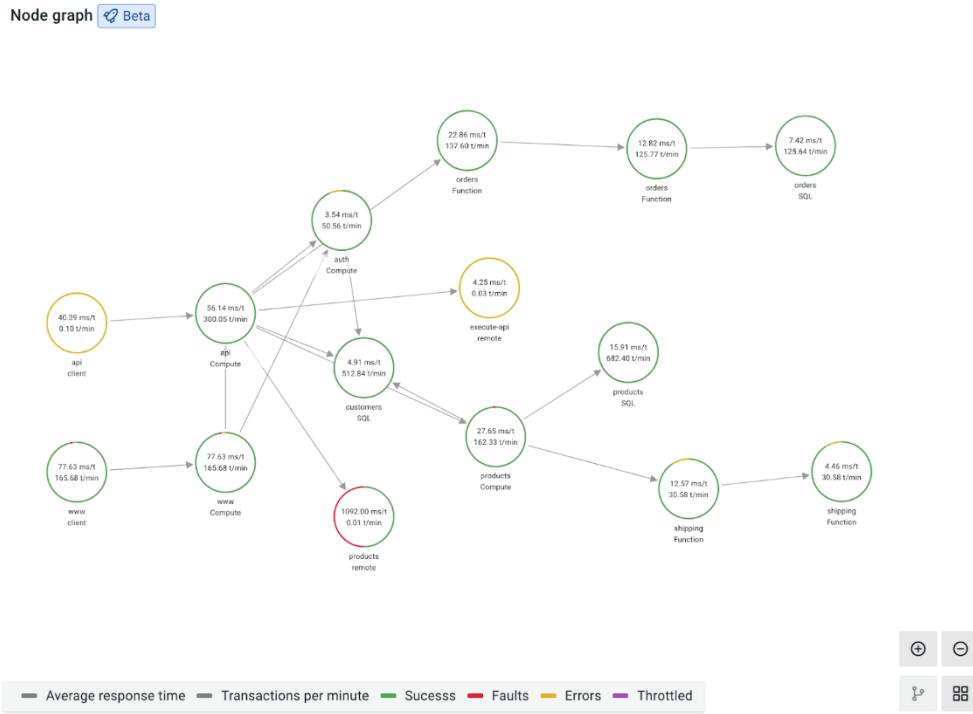


Ilustración 47. Visualización Node graph en Grafana

4.2.3 ESTADÍSTICAS GENERALES DE VISUALIZACIÓN

De los 60 tipos de visualizaciones más comunes recogidos por *Visualization Universe* [7], *Data Viz Project* [8] y *Visual Vocabulary* [9], grafana soporta de forma

nativa 20 de ellas, lo que es equivalente a 1 de cada 3 visualizaciones más usadas. No suficiente con ello, Grafana cuenta también con plugins ya creado que permiten la visualización de 17 más, es decir entre las que trae grafana de forma nativa y las que tienen ya su propio plugin dan soporte al 61'67% de las visualizaciones más usadas.

Grafana Support			
	native	plugin	need custom plugin
visualizations	20	17	23
%	33.33%	28.33%	38.33%

Tabla 15. Soporte de Plugins en grafana

En cuanto a las visualizaciones que necesitan un plugin personalizado, tampoco sería una tarea demasiado complicada ya que podríamos crearlo usando algunas librerías como Plotly o Highcharts que dan soporte prácticamente a la totalidad de visualizaciones sin necesidad de tener que programarlas totalmente a mano.

A continuación, vamos a ver las familias en las que organizan tanto Dataviz Project como Visual Vocabulary las visualizaciones y cuáles son las más comunes.

En cuanto al estudio del Visual Vocabulary, es destacable que ningún gráfico de tipo Desviación se encuentra entre los más comunes, esto también va acorde a Grafana ya que como vimos anteriormente no soporta ningún tipo de visualización de esta familia ni de forma nativa ni con algún plugin ya creado.

Visual Vocabulary									
Deviation	Correlation	Ranking	Distribution	Change-over-Time	Magnitude	Part-to-whole	Spatial	Flow	
0	3	4	6	9	8	11	6	5	
0%	6%	8%	12%	17%	15%	21%	12%	10%	

Tabla 16. Cantidad de visualizaciones de "Visual Vocabulary" soportadas

Como se puede ver, las más comunes serían las que sirven para visualizar una parte del total (un ejemplo sería el gráfico de tarta), aquellas que representan magnitudes (gráfico de barras o de columnas por ejemplo) y las que muestran las variaciones a lo largo del tiempo (como un gráfico de tipo líneas).

Hablando más sobre el estudio *Data Viz Project*, este tiene 3 formas de catalogar las visualizaciones. La primera de ella es según la **función** que realiza. pueden ser:

Function						
Comparison	Concept-visualization	Correlation	Distribution	Geolocation	Part-to-whole	Trend-over-time
32	1	12	20	5	8	16

34%	1%	13%	21%	5%	9%	17%
-----	----	-----	-----	----	----	-----

Tabla 17. Cantidad de visualizaciones de "Data Viz Project" soportadas.

- **Comparison (Comparación):** Este tipo de visualización lo utilizan el 34% de las veces. Indica que la comparación entre diferentes conjuntos de datos o elementos es una parte esencial de este estudio.
- **Correlation (Correlación):** Representa el 13% de las visualizaciones. Es utilizado en aquellos gráficos que muestran las relaciones entre variables.
- **Distribution (Distribución):** Estas representan el 21%. Se utilizan para mostrar cómo se distribuyen los datos en un conjunto.
- **Geolocation (Geolocalización):** Representa el 5% de las visualizaciones. Se utilizan para mostrar datos en un contexto espacial, como mapas. Además todas estas visualizaciones son soportadas de forma nativa por Grafana.
- **Part-to-whole (Parte de un todo):** Estas visualizaciones representan el 9%. Son útiles para desglosar elementos de un conjunto mayor.
- **Trend-over-time (Tendencia a lo largo del tiempo):** Este tipo de visualización es el más común, representando el 17%. Se centran en representar cómo cambian ciertos datos a lo largo de un período de tiempo.
- **Concept-visualization (Visualización de conceptos):** Es el menos común, con solo un 1%. Tienen como objetivo principal comunicar conceptos abstractos o ideas complejas en lugar de datos concretos o números.

La segunda clasificación que realiza *Data Viz Project* es según a la **familia** de visualizaciones que pertenece.

Family				
Charts	Diagram	Geospatial	Plot	Table
31	9	5	11	1
54%	16%	9%	19%	2%

Tabla 18. Visualizaciones soportadas por familias

- **Chart (Gráficos):** Es la más común con un 54%, estas representan la mayoría de las visualizaciones más populares. Los gráficos son una forma efectiva de mostrar datos, incluyen gráficos de barras, gráficos de líneas, y otros tipos de gráficos que permiten comparar datos y patrones de manera fácil y sencilla.
- **Diagram (Diagramas):** Con un 16%, los diagramas también tienen una presencia significativa. Son representaciones visuales que se utilizan para ilustrar relaciones o diferentes elementos o conceptos.
- **Geospatial (Geoespacial):** Representando el 9% del total, las visualizaciones geoespaciales están diseñadas para mostrar información en un contexto geográfico o de ubicación. Va relacionado con las visualizaciones que realizan funciones de geolocalización..

- **Plot (Trazas):** Con un 19%, las representaciones basadas en gráficos de dispersión o gráficos de trazas se utilizan para mostrar la relación entre dos variables y resaltar patrones, tendencias o correlaciones en los datos.
- **Table (Tabla):** Las visualizaciones en forma de tablas son menos comunes, representando solo un 2%. Las tablas se utilizan para presentar datos en un formato tabular, lo que facilita la comparación precisa de valores.

La última clasificación que realiza *Data Viz Project* es según la **entrada de datos** que reciben para realizar la visualización. El estudio distingue entre 35 tipos de datos de entrada, pero en prácticamente la mitad de las visualizaciones más comunes no viene especificada cual es la entrada o no tienen entrada. Pero entre las que sí se conoce su entrada, la más frecuente es la entrada de tipo 5, en la que se recibe una entrada con 3 cadenas, es decir 3 nombres y unos valores asignados a estos nombres. Este tipo de entrada se produce en 5 de las 60 visualizaciones lo que supone un 13'89% de las veces.

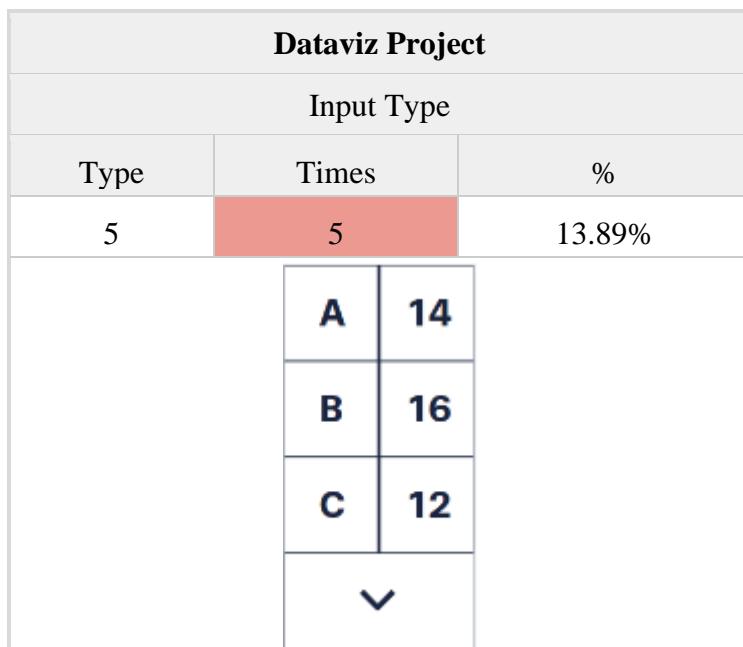


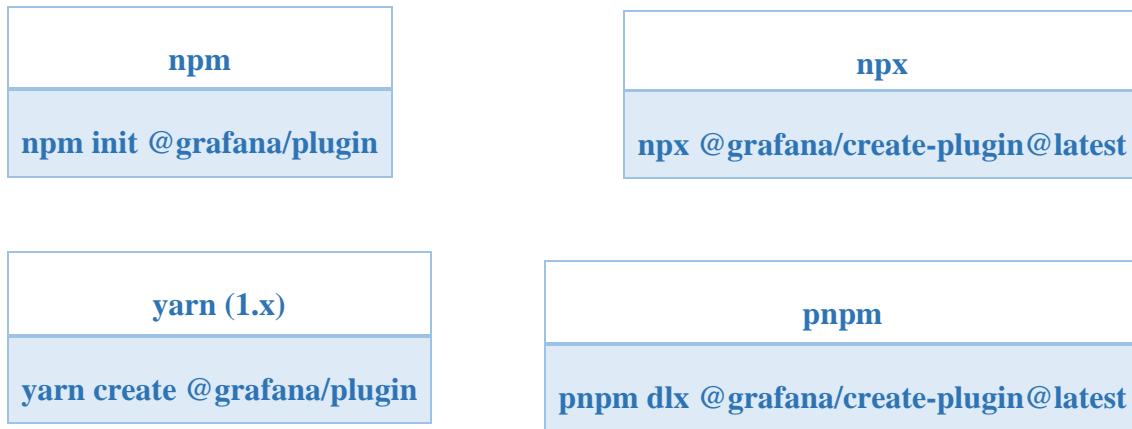
Tabla 19. Visualizaciones soportadas por tipo de entrada de datos

4.3 MANUAL CREACIÓN DE WIDGET PERSONALIZADO EN GRAFANA

Un widget o plugin de Grafana es una extensión personalizada que amplía las capacidades de Grafana al agregar nuevas visualizaciones, paneles, fuentes de datos o integraciones con sistemas externos. Con los plugins, podemos satisfacer unas necesidades más específicas de visualización o análisis de datos.

4.3.1 PREPARACIÓN DEL ENTORNO

En primer lugar, deberemos estar en la carpeta donde queramos tener el plugin. Una vez en ella podremos usar NPM, PNPM o Yarn1 como gestor de paquetes. Según cual prefiramos deberemos ejecutar uno de los siguientes comandos:



Una vez ejecutemos cualquiera de estos comandos en primer lugar nos preguntará cuál será el nombre de nuestro plugin, el nombre de la organización del plugin y una pequeña descripción de lo que va a tratar.

```
user@saborido-house:~/plugin$ npx @grafana/create-plugin@latest
? What is going to be the name of your plugin? Visualizacion1
? What is the organization name of your plugin? TFG
? How would you describe your plugin? Plugin que aportará visualizaciones extras
para grafana
? What type of plugin would you like?
  app
  datasource
> panel
  scenesapp
```

Ilustración 48. Creación de plugin personalizado Grafana por consola

Cuando hayamos completado estos campos, la consola nos preguntará que tipo de plugin va a ser, aquí hay 4 opciones:

- **App:** Estos plugins extienden la funcionalidad de la interfaz del usuario de Grafana, permite añadir páginas, barras laterales o opciones de menú personalizadas.
- **Datasource:** Permite conectar Grafana con fuentes de datos externas como podrían ser bases de datos y permite crear paneles que muestren datos de esa base de datos.
- **Panel:** Los paneles permiten crear visualizaciones personalizadas como gráficos, tablas, mapas, etc.

- **Scenesapp:** Este tipo de plugin permite crear aplicaciones dentro de grafana que a su vez pueden incluir paneles o otros elementos interactivos por lo que pueden crear una experiencia de usuario muy personalizada.

En mi caso, quiero crear un plugin que muestre una visualización de mi API por lo que crearé un plugin de tipo panel.

Una vez se elija el tipo de plugin a desarrollar, la consola nos preguntará si queremos añadir Github CI y el flujo de trabajo al realizar una actualización en github. Al aceptar esto, cuando hagamos una actualización ejecutará un conjunto de tests y comprobaciones para comprobar el correcto funcionamiento de Grafana. También preguntará si queremos que en este flujo se compruebe la compatibilidad de la API de Grafana y también lo añadimos.

```
? Do you want to add Github CI and Release workflows? Yes
? Do you want to add a Github workflow for automatically checking "Grafana API compatibility" on PRs? Yes
```

Ilustración 49. Opciones seleccionadas al crear plugin grafana

Una vez hayamos añadido ambos flujos, se descargará e instalará todo lo necesario y una vez completado deberemos seguir los pasos indicados para comenzar nuestro proyecto de Grafana.

```
Run the following commands to get started:
* cd ./tfg-visualizacion1-panel
* npm install to install frontend dependencies.
* npm run dev to build (and watch) the plugin frontend code.
* docker-compose up to start a grafana development server. Restart this command after each time you run mage to run your new backend code.
  * Open http://localhost:3000 in your browser to create a dashboard to begin developing your plugin.

Note: We strongly recommend creating a new Git repository by running git init in ./tfg-visualizacion1-panel before continuing.

  * Learn more about Grafana Plugin Development at https://grafana.com/developers/plugin-tools
```

Ilustración 50. Plugin creado con éxito

4.3.2 INICIANDO GRAFANA

Una vez hayamos completado y preparado nuestro entorno de Grafana, siguiendo los cuatro comandos que nos dice la consola se iniciará Grafana. [10]

1. *cd ./tfg-visualizacion1-panel*
2. *npm install*
3. *npm run dev*

```

user@saborido-house:~/plugin/tfg-visualizacion1-panel$ npm run dev
> visualizacion1@1.0.0 dev
> webpack -w -c ./config/webpack/webpack.config.ts --env development

<i> [LiveReloadPlugin] Live Reload listening on port 35729
assets by path *.md 2.55 KiB
  asset README.md 2.5 KiB [compared for emit] [from: README.md] [copied]
  asset CHANGELOG.md 52 bytes [compared for emit] [from: ../CHANGELOG.md] [copied]
asset module.js 16.3 KiB [compared for emit] (name: module)
asset LICENSE 11.1 KiB [compared for emit] [from: ../LICENSE] [copied]
asset img/logo.svg 1.55 KiB [compared for emit] [from: img/logo.svg] [copied]
asset plugin.json 632 bytes [emitted] [from: plugin.json] [copied]
cached modules 2.69 KiB (javascript) 937 bytes (runtime) [cached] 10 modules
webpack 5.88.2 compiled successfully in 893 ms
Type-checking in progress...
assets by status 30.6 KiB [cached] 5 assets
asset img/logo.svg 1.55 KiB [emitted] [from: img/logo.svg] [copied]
cached modules 2.69 KiB (javascript) 937 bytes (runtime) [cached] 10 modules
webpack 5.88.2 compiled successfully in 2887 ms
Type-checking in progress...
No errors found.

```

Ilustración 51. Arrancar Grafana

Aquí podremos ver que está todo correctamente y no hay ningún error

4. docker-compose up

De esta forma ejecutaremos nuestro Grafana de forma dockerizada y estará accesible en el puerto localhost:3000.

Cuando accedemos a Grafana, podremos comprobar que nuestro plugin está dentro, aunque de momento no tenga ninguna funcionalidad. Para verlo deberemos ir a “Add” y posteriormente hacer click en “Visualization”.

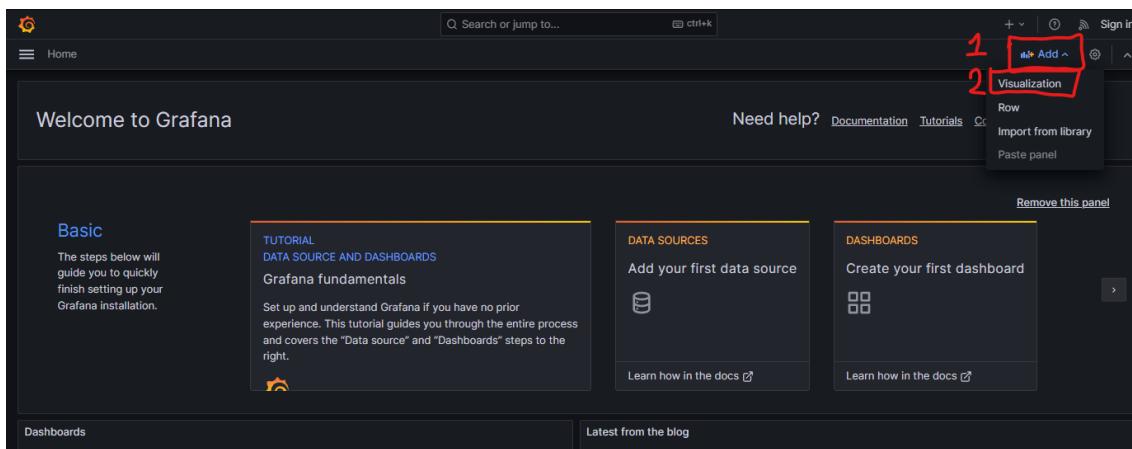


Ilustración 52. Visualización del plugin I

Una vez hecho esto, se nos abrirá una ventana para editar el nuevo panel que acabamos de crear, por defecto la visualización será time series, pero si hacemos click en “Time series”:

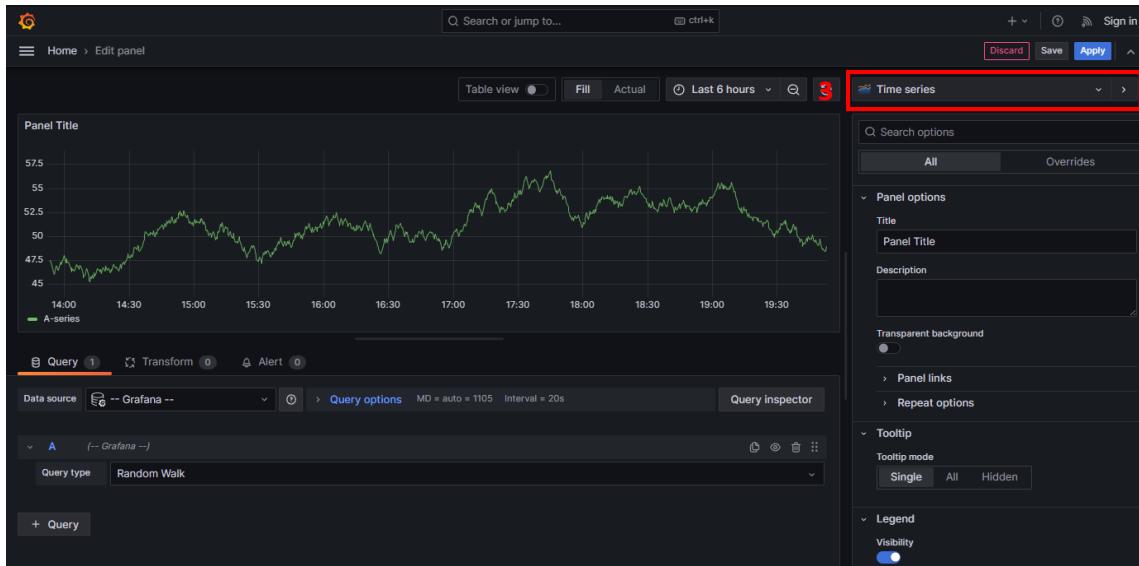


Ilustración 53. Visualización del plugin II

Ahora se abrirán todas las visualizaciones que vienen por defecto en Grafana y también deberá estar el nombre que le acabamos de dar al plugin.

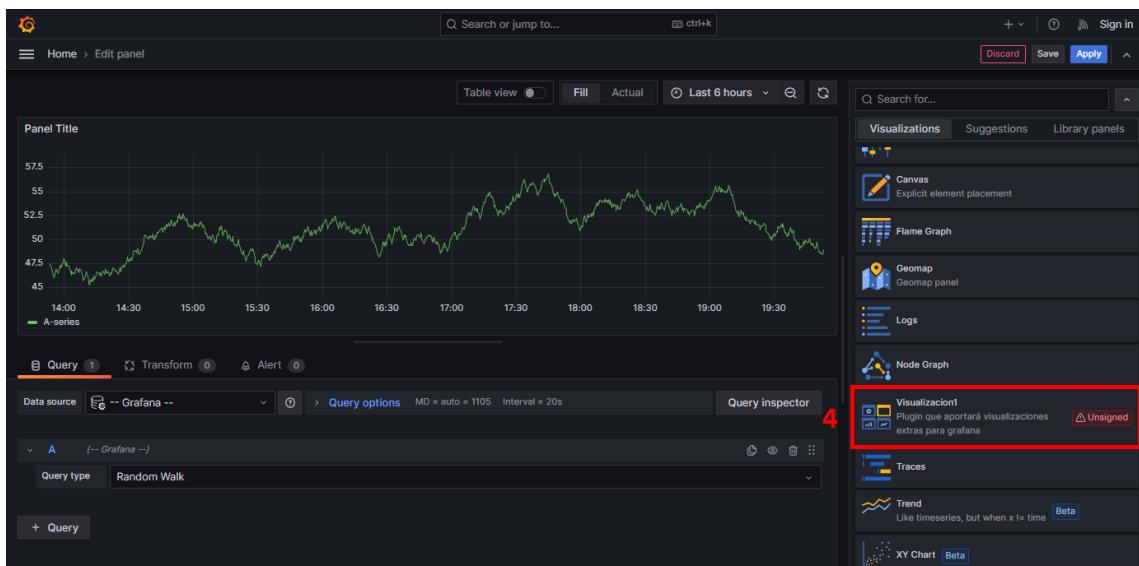


Ilustración 54. Visualización del plugin III

4.3.3 PERSONALIZACIÓN DEL PLUGIN

En este caso me centraré en la personalización del plugin utilizando Highcharts. Highchart es una librería disponible para Angular, JavaScript React y Vue entre otros que ofrece una forma fácil y sencilla de realizar visualizaciones y que sean compatibles tanto con plataformas webs o móviles.

Por ello, primero voy a instalar en el proyecto tanto la librería de highchart como axios para poder obtener los datos de la api para después visualizarla, para ello:

- **`npm install highcharts --save`**
- **`npm install axios`**

En segundo lugar deberemos abrir la carpeta `tfg-visualizacion1-panel` en nuestro editor de código. El archivo del plugin estará en `src -> components -> "SimplePanel.tsx"`.

Por defecto este archivo tiene la función `getStyles` que podemos usarla para estilos con css a nuestro plugin y después la parte más importante la podemos encontrar en el componente `SimplePanel`.

Dentro de `SimplePanel`, lo primero que hay que hacer es cargar los datos que quiero visualizar, para ello, utilizaré `useEffect` que se encarga de realizar operaciones secundarias dentro de React y dentro del `useEffect` cargaré los datos de mi api usando axios y lo guardaré dentro de la variable `setTennisData`.

```
useEffect(() => {
  axios
    .get('http://antoniosaborido.es/api/v2/tennis')
    .then((response) => {
      setTennisData(response.data);
    })
    .catch((error) => {
      console.error('Error al cargar los datos:', error);
    });
}, []);
```

Código 8. Llamada a la api

Después, como en mi caso voy a hacer un diagrama de tipo *pie* necesito saber el total de elementos que hay y cuantos elementos hay de cada tipo y tener guardado esto en alguna variable.

```
const [tennisData, setTennisData] = useState<any>([]);
//Número total de elementos:
const totalGrandSlam = tennisData.reduce((total, item) => total + item.most_grand_slam, 0);
//Elementos de cada tipo.
const seriesData = tennisData.map((item) => ({
  name: item.country,
  y: (item.most_grand_slam / totalGrandSlam) * 100,
}));
```

Código 9. Tratamiento de los datos de la llamada a la api

Una vez que tengo todos los datos que necesito representar, procederé a guardar en una constante como tiene que ser la representación, este código será igual en cualquier front-end que utilice highchart y en él establezco que su tipo será *pie*, le doy un título a la visualización y configuro el formato de las etiquetas que se mostrarán, por último en la sección `series`, digo que quiero representar que en este caso será la variable que creé anteriormente llamada `seriesData`.

```

const loadHighcharts = useCallback(() => {
  if (typeof Highcharts !== 'undefined') {
    // @ts-ignore
    Highcharts.chart('container', {
      chart: {
        type: 'pie',
      },
      title: {
        text: 'Distribución de Most Grand Slam por país',
      },
      plotOptions: {
        pie: {
          allowPointSelect: true,
          cursor: 'pointer',
          dataLabels: {
            enabled: true,
            format: '<b>{point.name}</b>: {point.percentage:.1f} %',
          },
        },
      },
      series: [
        {
          name: 'Most Grand Slam',
          colorByPoint: true,
          data: seriesData,
        },
      ],
    });
  }
}, [seriesData]);

```

Código 10. Creación de gráfica usando Highchart

Una vez hecho esto, ya solo quedaría crear un nuevo *useEffect* al igual que el que utilicé con axios pero en esta ocasión para que llame a esta nueva función que acabo de crear. He añadido un if para que si la petición axios no tiene ningún dato, que directamente no cargue ningún gráfico.

```

useEffect(() => {
  if (seriesData.length > 0) {
    loadHighcharts();
  }
}, [seriesData, loadHighcharts]);

```

Código 11. Manejo de llamadas vacías.

Todo este código está disponible en un de Github [11]. Por último, para ver su funcionamiento habrá que volver a lanzar grafana, esto se puede hacer Mediante comandos como anteriormente o con los script que ya vienen configurados.

```

    NPM SCRIPTS
    package.json
        build webpack -c ./config/webpack/webpack.config.ts --env production
        dev webpack -w -c ./config/webpack/webpack.config.ts --env development
        test jest --watch --onlyChanged
        test:ci jest --passWithNoTests --maxWorkers 4
        typecheck tsc --noEmit
        lint eslint --cache --ignore-path ./gitignore --ext js.jsx.ts.tsx .
        lintfix npm run lint -- --fix
        e2e npm exec cypress install && npm exec grafana-e2e run
        e2e:update npm exec cypress install && npm exec grafana-e2e run --update-screenshots
        server docker-compose up --build
        sign npx --yes @grafana/sign-plugin@latest

```

Ilustración 55. Scripts disponibles

Una vez ejecutado el script “server”, volvemos a entrar a localhost:3000 y siguiendo todos los pasos del apartado Iniciando Grafana, podemos observar que ahora nos muestra la visualización de la API tenis.

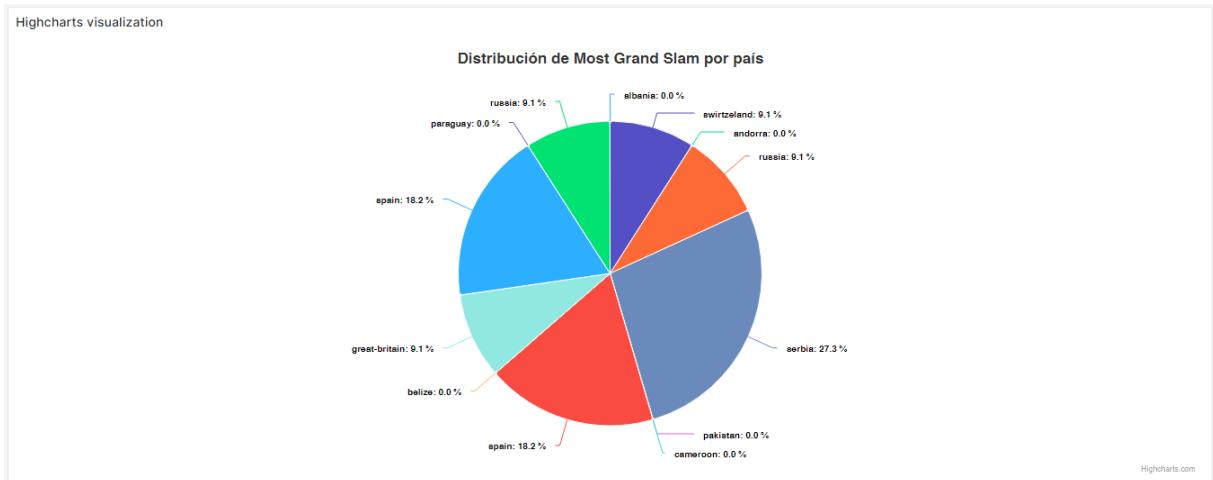


Ilustración 56. Visualización de gráfico en Grafana

4.4 TPAS PARA DEVOPS

4.4.1 DORA

El informe **DORA (DevOps Research and Assessment)** [12], elaborado por Google, es una referencia fundamental en el ámbito de la entrega de software y las prácticas DevOps. Este informe, basado en investigaciones extensivas y análisis de datos de cientos de organizaciones, proporciona un marco detallado para mejorar el rendimiento de los equipos de desarrollo y operaciones. Su objetivo principal es identificar las prácticas y capacidades técnicas que permiten a las empresas entregar software de manera rápida, fiable y segura.

El DORA se utiliza ampliamente en la industria tecnológica para guiar a las organizaciones en la implementación de estrategias de **DevOps** efectivas. Ayuda a las empresas a medir su desempeño, identificar áreas de mejora y adoptar prácticas que han demostrado ser exitosas en otros contextos. A lo largo del informe, se exploran diversas capacidades técnicas clave que son esenciales para la implementación exitosa de DevOps, abarcando desde la infraestructura en la nube hasta la seguridad y la automatización. Estas prácticas permiten a los equipos de desarrollo y operaciones colaborar de manera más eficiente, respondiendo de forma ágil a las necesidades del mercado y asegurando una entrega de software de alta calidad.

Como se verá a continuación, un mantenimiento de código efectivo y la Entrega Continua (Continuous Delivery) son fundamentales para una integración y despliegue sin interrupciones. Al facilitar la reutilización y la estabilidad de las dependencias del código, las organizaciones pueden implementar cambios rápidamente y con menos riesgo de errores. La automatización de pruebas y despliegues se complementa con la Integración Continua (Continuous Integration), que coordina actualizaciones de código de múltiples desarrolladores, asegurando que cada cambio sea compatible y mantenga la calidad del software.

Además, las Pruebas Continuas (Continuous Testing) y la Gestión de Cambios en la Base de Datos son prácticas que refuerzan la estabilidad y calidad del software. Las pruebas continuas permiten una retroalimentación constante y la validación de la funcionalidad del sistema, mientras que la gestión de cambios en la base de datos garantiza que estos se implementen de manera controlada y visible para todos los equipos involucrados. La Automatización de la Implementación reduce errores manuales y mejora la eficiencia, permitiendo despliegues rápidos y seguros.

El informe también destaca la importancia de empoderar a los equipos para elegir sus herramientas, promoviendo la responsabilidad y la adaptabilidad en las decisiones tecnológicas. Una arquitectura de acoplamiento flexible permite a los equipos trabajar de manera independiente, probando e implementando cambios sin afectar a otros sistemas. Monitoreo y Observabilidad son esenciales para comprender y depurar el estado de los sistemas en tiempo real, facilitando una respuesta proactiva a los problemas. Finalmente, el enfoque de cambiar hacia la izquierda en seguridad integra prácticas de seguridad desde las primeras etapas del desarrollo, reduciendo el tiempo y esfuerzo necesarios para solucionar problemas, y mejorando la eficiencia general del proceso de desarrollo.

4.4.1.2 INFRAESTRUCTURA EN LA NUBE

La infraestructura en la nube no es simplemente adquirir un buen proveedor de espacio en la nube, según el Instituto Nacional de Estándares y Tecnología de Estados Unidos (**NIST**) define cinco características esenciales de la computación en la nube: Autoservicio bajo demanda, acceso a redes amplias, agregación de recursos, elasticidad rápida y servicio medido.

Tan solo el 30% de las personas que utilizan las tecnologías en la nube cumplen las características mencionadas. Aquellos que sí las utilizan tienen mejor rendimiento y estabilidad aparte de estimar los costos de su infraestructura de una forma mucho más precisa.

Para implementar estas características muchas organizaciones han utilizado el paradigma de **infraestructura como código (IaC)**. La infraestructura como código permite gestionar los cambios de manera efectiva y aplicar controles de seguridad de la información. Ejemplo de esto es Google, que tiene implementado la segregación de tareas requiriendo que todos los cambios en la configuración especificada en el control de versiones sean aprobados por otro grupo de personas. Por todo esto la IaC requiere un gran cambio y esfuerzo para que sea implementada correctamente.

Los mayores obstáculos para cumplir las características NIST son la falta de alineación y colaboración entre los departamentos que tienen que trabajar para implementarlo, y una inversión insuficiente en los procesos, la parte técnica y el cambio organizativo necesario.

Aunque habrá muchos más posibles problemas a superar como por ejemplo el rediseño de procesos y arquitectura que permita a los equipos realizar despliegues y cambios de configuración de manera autónoma. Conseguir que los desarrolladores comprendan cómo construir aplicaciones nativas en la nube, integraciones entre sistemas nativos en la nube y sistemas no basados en la nube, etc.

Por tanto, superar estos obstáculos será un gran cambio que requerirá una inversión y colaboración continua en todos los niveles de la organización.

4.4.1.2 MANTENIMIENTO DEL CODIGO

El mantenimiento del código es un aspecto fundamental para el éxito de la entrega continua. Si el código tiene un buen mantenimiento, cumplirá estas tres características principales:

1. Será fácil para el equipo reutilizar el código de otras personas y encontrar ejemplos en el código.
2. Será fácil para el equipo añadir nuevas dependencias o migrarlas a una nueva versión de la dependencia.
3. Las dependencias serán estables y rara vez afectarán al código.

Para implementar un buen mantenimiento del código es importante abordar estos puntos:

- **Manejo del código fuente:** Permitir a todo el mundo encontrar, reutilizar y proponer cambios en cualquier parte del código base.
 - a. Esto propiciará una entrega más rápida ya que los equipos tendrán más conocimientos del código para poder optimizarlo.
 - b. También proporcionará niveles más altos de estabilidad y disponibilidad ya que en caso de incidente es esencial encontrar y proponer cambios en cualquier parte del código.
 - c. Por último, también dará mayor calidad al código. La refactorización del código para mejorar su calidad interna a menudo implica hacer cambios en múltiples partes del código. Si esto es difícil, disminuye la probabilidad de que las personas realicen refactorizaciones.
- **Dependencias.** Facilitar que los equipos agreguen y actualicen dependencias, y garantizar que sean estables y rara vez afectan el código.
 - a. Mejora la seguridad: a medida que envejecen las dependencias, es más probable que se descubran vulnerabilidades en ellas.
 - b. Entrega más rápida: Cuando se tienen mecanismos para garantizar que las dependencias sean estables y rara vez afecten el código, el implementar librerías externas permite pasar más tiempo codificando y menos tiempo en mantenimiento.

Hay dos modelos comúnmente utilizados para incluir dependencias en su software: “**vendoring**” y los **manifiestos declarativos**. En el modelo “vendoring”, se verifica en el control de versiones el código fuente o el binario de cada dependencia junto con la aplicación. Alternativamente, la mayoría de las plataformas modernas tienen una herramienta de gestión de dependencias que administra dependencias especificadas en archivos de manifiesto declarativos verificados en el control de versiones.

Entre algunos de los errores comunes destacan el tener múltiples repositorios de control de versiones o repositorios de control de versiones con configuraciones de acceso restrictivas. Las organizaciones deben tener idealmente una **única plataforma de control de versiones** en la que se almacene todo su código.

Las organizaciones suelen restringir quién puede realizar cambios en el control de versiones. Esto lleva al segundo error: **la falta de herramientas y procesos** para que las personas realicen cambios en partes del código base a las que no tienen acceso de escritura.

Para medir la mantenibilidad del código es interesa observar algunos parámetros como por ejemplo el tiempo de espera promedio para realizar un cambio en una parte del código a la que no tenga acceso de escritura. Otro puede ser el porcentaje de código reutilizado o el porcentaje de aplicaciones que no están usando la última versión estable de las librerías que utilizan.

4.4.1.3 ENTREGA CONTINUA

La Entrega Continua o “Continuous Delivery” en inglés, es la capacidad de realizar cambios de todo tipo de manera rápida, segura y sostenible. Los equipos que la

llevan a cabo pueden lanzar versiones de software de manera rápida y segura sin afectar a los usuarios.

Esto proporciona beneficios significativos. Según el informe DORA, tener éxito en la entrega continua proporciona beneficios en el rendimiento de la entrega de software, lleva a mayores niveles de calidad, produce menos agotamiento y mayor satisfacción laboral, reduce el daño que puedan causar los despliegues y produce mayores niveles de seguridad psicológica, así como una cultura organizativa más orientada a la misión del producto.

Para ayudar a los equipos a obtener mayor capacidad de producción y lanzamientos con menor riesgo son útiles las siguientes prácticas:

- Automatización de pruebas y del despliegue
- Desarrollo Trunk-Based
- Empoderar a los equipos
- Integración continua (CI) y pruebas continuas
- Control de versiones
- Gestión de datos de pruebas y gestión de cambios en la base de datos
- Monitoreo y observabilidad integral

Para valorar el funcionamiento de la entrega continua es recomendable observar algunas métricas claves como son que haya tiempos cortos de ejecución para cambios regulares y de emergencia, baja tasa de fallos en los cambios o tiempos cortos para restaurar el servicio en caso de interrupciones o degradaciones del servicio.

4.4.1.4 INTEGRACIÓN CONTINUA

Cuando un gran número de desarrolladores trabaja en sistemas relacionados, coordinar las actualizaciones de código es un problema difícil, y los cambios de diferentes desarrolladores pueden ser incompatibles.

La práctica de la Integración Continua (CI) o “Continuous Integration” en inglés, se creó para abordar estos problemas. Usando la CI se obtiene feedback constantemente lo que produce un software de mayor calidad a la vez que reduce el coste del desarrollo y mantenimiento al aumentar la productividad de los equipos.

Para llevar a cabo esto es necesario lo siguiente:

- **Un proceso de compilación automatizado.** El primer paso en la CI es tener un script automatizado que cree paquetes que se puedan implementar en cualquier entorno. Estos deben ser autoritativos y utilizados por todos los procesos descendentes. Se debe ejecutar su proceso de compilación con éxito al menos una vez al día.
- **Un conjunto de pruebas automatizadas.** Hay que asegurarse de que las pruebas sean confiables. De esta manera, cuando fallen, se sabrá que hay un problema real, y cuando pasen, tendrá la confianza de que no hay problemas graves en el sistema. Todas las funcionalidades deben ser cubiertas por las pruebas. Esas pruebas deben ejecutarse rápidamente para proporcionar feedback.

- **Un sistema de CI que ejecute la compilación y las pruebas automatizadas en cada confirmación.** El sistema también debe hacer que el estado sea visible para el equipo.

Algunos de los problemas comunes son que no esté todo el código en el mismo repositorio, que no se automatice el proceso de compilación o tener pruebas que tardan demasiado en ejecutarse al no desencadenar en pruebas rápidas en cada cambio.

4.4.1.5 PRUEBAS CONTINUAS (CONTINUOUS TESTING)

Es clave a la hora de incorporar nuevo software obtener feedback de manera rápida. Anteriormente era habitual realizar las pruebas de forma manual pero este enfoque tiene las siguientes desventajas:

- Consumen mucho tiempo, por lo tanto, son un cuello de botella y son costosas.
- No son confiables ya que las personas no son tan buenas en tareas repetitivas.
- Los desarrolladores tienen que esperar mucho tiempo para obtener feedback sobre los cambios.
- Si los desarrolladores no son responsables de probar su propio código, no sabrán cómo escribir un código que funcione correctamente.

Para mejorar la calidad del software se deben ejecutar de manera continua tanto pruebas automatizadas como pruebas manuales para validar la funcionalidad y la arquitectura del sistema en desarrollo. El informe concluye que los equipos serán más eficientes cuando los testers trabajen junto a los desarrolladores a lo largo del proceso de desarrollo y entrega del software. También tendrán mejor desempeño cuando se realicen pruebas manuales y pruebas de aceptación a lo largo del proceso de entrega.

El conjunto de pruebas automatizadas debe incluir pruebas unitarias que prueben un solo método, clase o función de forma aislada y pruebas de aceptación que prueban una aplicación o servicio en ejecución.

4.4.1.6 GESTIÓN DE CAMBIOS EN LA BASE DE DATOS

Para implementar una gestión de cambios efectiva hay que tener en cuenta tanto el aspecto técnico como cultural. El informe muestra que los equipos tienen un mejor desempeño cuando se discuten los cambios con los responsables de gestionar la base de datos.

Otro aspecto crucial es la visibilidad sobre el proceso de cambios, para facilitar esto es bueno utilizar una herramienta que registre qué cambios se han ejecutado en qué entornos y cuáles fueron los resultados. También es bueno mantener todos los cambios en el esquema de la base de datos en control de versiones.

Para una buena implementación de la gestión de cambios en la base de datos hay que tener en cuenta estos aspectos:

- **Comunicación efectiva de los cambios:** Discutir los cambios con los administradores de bases de datos en producción es importante ya que estos expertos pueden asesorar sobre la mejor manera de lograr resultados e identificar

problemas. Es crucial también que todos los equipos implicados tengan información sobre el progreso de los cambios.

- **Tratar todos los cambios en el esquema de la base de datos como migraciones:** Un patrón ampliamente utilizado para versionar los cambios en la base de datos es recoger cada cambio como un script de migración que se guarda en control de versiones. Cada script de migración deberá tener un número de secuencia único, para que se sepa en qué orden aplicar las migraciones
- **Cambios en la base de datos sin tiempo de inactividad:** Muchas organizaciones programan tiempos de inactividad o mantenimiento para sus servicios al realizar cambios en el esquema de la base de datos debido a la necesidad de coordinarlos con implementaciones de la aplicación o debido al bloqueo de tablas de la base de datos durante la ejecución de dichos cambios. La entrega continua tiene como objetivo eliminar el tiempo de inactividad para las implementaciones.

4.4.1.7 AUTOMATIZACIÓN DE LA IMPLEMENTACIÓN

La automatización de la implementación permite implementar software en entornos de prueba y producción pulsando un solo botón. Tiene las siguientes entradas:

- Paquetes creados por el proceso de integración continua (CI).
- Scripts para configurar el entorno, implementar paquetes y realizar una prueba de implementación.
- Información de configuración específica del entorno.

Es importante que los scripts y la información de configuración estén almacenados en el control de versiones. Estos scripts deberán ser capaces de preparar el entorno de destino, implementar los paquetes necesarios, realizar tareas relacionadas con la implementación del software, realizar todas las configuraciones necesarias y realizar pruebas de implementación para asegurar que los servicios externos necesarios sean accesibles.

Algunos de los fallos comunes son la **complejidad** ya que automatizar un proceso manual es complejo y frágil. El segundo obstáculo que nos podemos encontrar son las **dependencias** puesto que muchos procesos de implementación requieren un orden particular a la hora de implementar sus componentes. El tercer obstáculo al que hay que enfrentarse son aquellos componentes que **no están diseñados para la automatización** y requieren inicios de sesión en consola o interacciones manuales. Por último, la **mala colaboración entre equipos** puede provocar configuraciones diferentes de los servicios provocando que estos dejen de ser compatibles.

El nivel de automatización se puede medir contando el número de pasos manuales necesarios para la implementación o el porcentaje de automatización a la hora de implementar el servicio.

4.4.1.8 POTENCIAR A LOS EQUIPOS PARA ELEGIR HERRAMIENTAS

Para obtener un buen rendimiento de entrega de software se debe empoderar a los equipos para tomar decisiones informadas sobre las herramientas y tecnologías que utilizan para desarrollar su trabajo.

Esto se ha observado en empresas como Google y Netflix, donde se respalda una herramienta por defecto. Pero si un equipo cree firmemente que una herramienta o tecnología diferente es la mejor para su caso, tienen la libertad de elegirla. Los equipos comprenden que su elección conlleva la responsabilidad de realizar el trabajo de respaldar esta nueva herramienta.

Para llevar a cabo esto es crucial crear una línea base entre los equipos, para ello representantes de los equipos y de otras áreas multifuncionales deberán establecer una línea base de herramientas aprobadas que pueda cubrir la mayoría de las necesidades de la organización. También será necesario una revisión periódica de estas herramientas que brindará la oportunidad para discutir sobre las herramientas actuales y poder añadir nuevas.

Por último, también será importante definir un proceso para excepciones, cuando se utilice una nueva herramienta que no esté en la línea base creada anteriormente se deberá documentar qué herramienta es y por qué se está utilizando ya que estos datos serán vitales para la resolución de problemas y mantenimiento del proyecto.

Arquitectura de acoplamiento flexible

4.4.1.9 ARQUITECTURA DE ACOPLAMIENTO FLEXIBLE

El informe muestra que la arquitectura es vital para lograr la entrega continua ya que el uso de Kubernetes o mainframes independientemente permite a los equipos adoptar prácticas que fomenten un mejor rendimiento en la entrega de software.

Cuando la arquitectura del sistema está diseñada para permitir que los equipos prueben, implementen y cambien sistemas sin depender de otros equipos, los equipos requieren poca comunicación para llevar a cabo su trabajo. En otras palabras, tanto la arquitectura como los equipos están débilmente acoplados. En cambio, con una arquitectura fuertemente acoplada, cambios pequeños pueden resultar en fallos en cascada a gran escala.

Las arquitecturas de microservicios están destinadas a permitir estos resultados, al igual que cualquier arquitectura verdaderamente orientada a servicios. En la práctica, muchas arquitecturas denominadas como orientadas a servicios no permiten probar e implementar servicios de manera independiente entre sí, por lo que los equipos no pueden lograr un mayor rendimiento en la entrega de software.

Moverse hacia microservicios o una arquitectura orientada a servicios también cambia muchas cosas en toda la organización. En su discurso sobre plataformas, Steve Yegge presenta varias lecciones críticas aprendidas al moverse hacia una SOA:

- Las **métricas y el monitoreo** se vuelven más importantes y escalarlos se vuelven más difíciles porque un problema que surge en un servicio podría venir de un servicio que está a muchas llamadas de servicio de distancia.
- Los servicios internos pueden generar problemas tipo **Denegación de Servicio** (DOS), por lo que las cuotas y la limitación de mensajes son importantes en cada servicio.

- La **calidad** y el **monitoreo** comienzan a mezclarse, porque el monitoreo debe ser integral y debe ejercer la lógica empresarial y los datos del servicio.
- Cuando hay **muchos servicios**, tener un mecanismo de descubrimiento de servicios se vuelve importante para el funcionamiento eficiente del sistema.

4.4.1.10 MONITOREO Y OBSERVABILIDAD

Según el informe el monitoreo y la observabilidad contribuyen positivamente a la entrega continua. Estos términos son definidos de la siguiente manera:

- **Monitoreo:** Herramienta o solución técnica que permite a los equipos observar y comprender el estado de sus sistemas. Se basa en recopilación de conjuntos predefinidos de métricas o registros.
- **Observabilidad:** Herramienta o solución técnica que permite a los equipos depurar activamente su sistema. Se basa en explorar propiedades y patrones no definidos anteriormente.

Hay algunas claves para la implementación efectiva del monitoreo y la observabilidad. En primer lugar, su monitoreo debe decir qué está roto y ayudar a comprender por qué, antes de que se cause demasiado daño.

Existen dos formas de alto nivel de ver un sistema: el **monitoreo de caja negra**, donde no se conocen el estado interno y los mecanismos del sistema, y el **monitoreo de caja blanca**, donde se conocen.

También hay 4 factores vitales para un buen sistema de monitoreo como son la **instrumentación**, esto significa que se puede agregar código a un sistema para exponer su estado interno. Otro sería la **correlación**, las métricas se deben poder recopilar desde la aplicación, así como sus sistemas subyacentes. También estaría el **cómputo**, después de recopilar datos de diversas fuentes, el sistema generará estadísticas y datos agregados en varios ámbitos. Por último, es importante el **aprendizaje** y la **mejora** ya que parte de operar un sistema es aprender de interrupciones y errores.

4.4.1.11 ADELANTAR TRATAMIENTO DE LA SEGURIDAD

La investigación del Informe DORA 2016 muestra que los equipos de alto rendimiento dedican un 50 por ciento menos de tiempo a remediar problemas de seguridad que los equipos de bajo rendimiento. Al integrar de manera más efectiva los objetivos de seguridad de la información en el trabajo diario, los equipos pueden lograr niveles más altos de rendimiento en la entrega de software y construir sistemas más seguros. Integrar los objetivos de seguridad en el trabajo diario mejora el rendimiento en la entrega de software y crea sistemas más seguros. Conocido como "cambiar hacia la izquierda", esto implica abordar las preocupaciones de seguridad desde el inicio del desarrollo.

Para implementar esto hay ciertos cambios necesarios:

- Involucrar al equipo de Seguridad Informática en la fase de diseños de todos los proyectos, una revisión de seguridad se puede añadir como factor de bloqueo para terminar una etapa de desarrollo.

- Desarrollar herramientas de seguridad previamente aprobadas. Un equipo de Seguridad informática puede ayudar a estandarizar el código de los desarrolladores ya que el uso de un código estándar facilita pruebas automatizadas que verifiquen que los desarrolladores estén usando librerías preaprobadas.
- Desarrollar pruebas automatizadas. Esto significará que el código puede ser probado continuamente sin requerir una revisión manual. Estas pruebas podrán identificar vulnerabilidades de seguridad.

4.4.1.12 GESTIÓN DE DATOS DE PRUEBA

La capacidad de ejecutar un conjunto completo de pruebas de unidad, integración y sistema es esencial para verificar que su aplicación o servicio se comporte según lo esperado y se pueda implementar de manera segura en producción. Para asegurarse de que sus pruebas están validando escenarios realistas, es fundamental suministrarles datos realistas.

Las siguientes prácticas ayudan a utilizar datos de prueba de manera eficiente:

1. Favorecer las pruebas de unidad: Deben ser independientes entre sí y de cualquier otra parte del sistema
2. Minimizar la dependencia de datos de prueba
3. Aislarnos de los datos de prueba.
4. Minimizar la dependencia de datos de prueba almacenados en bases de datos
5. Hacer que los datos de prueba estén fácilmente disponibles

4.4.1.13 DESARROLLO TRUNK-BASED

El desarrollo Trunk-based es una práctica requerida para la integración continua (CI). La integración continua es la combinación de la práctica del desarrollo basado en tronco y el mantenimiento de una suite de pruebas automatizadas rápidas que se ejecutan después de cada confirmación en el tronco para asegurarse de que el sistema siempre esté funcionando.

Para este tipo de desarrollo, es vital que los desarrolladores comprendan cómo dividir su trabajo en pequeños lotes. Según el informe los equipos logran niveles más altos de entrega software siguiendo estas 3 prácticas:

- Tener tres o menos ramas activas en el repositorio de código de la aplicación.
- Fusionar ramas en el tronco al menos una vez al día.
- No tener bloqueos de código ni fases de integración.

También es recomendable implementar pruebas automatizadas completas, tener una rápida compilación y realizar revisiones de código síncronas para garantizar que los cambios no tengan que esperar horas para sincronizarse.

4.4.1.14 CONTROL DE VERSIONES

Los sistemas de control de versiones como Git, Subversion y Mercurial proporcionan un medio lógico para organizar archivos y coordinar su creación, acceso controlado, actualización, etc.

El informe muestra que el uso integral del control de versiones, entre otras capacidades, predice la entrega continua. En particular, el control de versiones te ayuda a cumplir con estos requisitos críticos:

- **Reproducibilidad.** Los equipos deben ser capaces de provisionar cualquier entorno de manera completamente automatizada y saber que cualquier nuevo entorno reproducido a partir de la misma configuración es idéntico.
- **Trazabilidad.** Los equipos deben poder seleccionar cualquier entorno y determinar de manera rápida y precisa las versiones de cada dependencia utilizada para crear ese entorno.

Para mejorar el control de versiones es recomendable las siguientes medidas:

- Asegurar que cada confirmación en el control de versiones active la **creación automatizada de paquetes** que se pueden implementar en cualquier entorno utilizando sólo información en el control de versiones.
- Facilitar la **creación de entornos** de prueba similares a producción bajo demanda utilizando solo scripts e información de configuración del control de versiones y crea paquetes utilizando el proceso automatizado descrito en el punto anterior
- **Crear infraestructura de prueba** y producción mediante scripts de manera que los equipos puedan agregar capacidad o recuperarse de desastres de manera completamente automatizada.

4.4.2 TPA DE DEVOPS

Con el objetivo de crear un acuerdo de equipo para desarrolladores en el ámbito educativo surge este TPA. Estas métricas están diseñadas para mejorar la calidad del código, la eficiencia del equipo de desarrollo y mantener un flujo de trabajo organizado y ágil.

El objetivo de este conjunto de buenas prácticas es poder llevarlo a la práctica para medir el cumplimiento de estas prácticas en el ámbito educativo.

CTP1: Evitar pull requests de tamaño excesivo

- Descripción: Esta métrica busca garantizar que las pull requests (PR) sean lo suficientemente pequeñas como para ser revisadas fácilmente. Las PR grandes pueden ser difíciles de entender y revisar, lo que aumenta el riesgo de introducir errores.
- Parámetro: Se mide el tamaño de cada PR en términos de líneas de código modificadas (agregadas, eliminadas o cambiadas).
- Umbral: El umbral máximo permitido es de 200 líneas de código.
- Objetivo: Al menos el 90% de las PR deben cumplir con este umbral cada semana.

ITP2: Evitar pull requests de tamaño excesivo (por individuo)

- Descripción: Similar a CTP1, esta métrica se centra en el tamaño de las PR realizadas por cada miembro del equipo, buscando evitar que un solo desarrollador genere PR demasiado grandes.
- Parámetro: Se mide el tamaño de las PR de cada individuo en términos de líneas de código.
- Umbral: El umbral máximo permitido es de 200 líneas de código.
- Objetivo: Al menos el 90% de las PR individuales deben cumplir con este umbral cada semana.

ITP3: Evitar ramas activas sin Pull Request

- Descripción: Esta métrica se enfoca en asegurarse de que las ramas activas no queden sin pull requests, promoviendo un ciclo de desarrollo continuo y evitando que el trabajo en progreso se estanque.
- Parámetro: Número de ramas activas que tienen una PR asociada (abierta o cerrada).
- Periodo: Un sprint (un ciclo de desarrollo definido, generalmente de 1-2 semanas).
- Objetivo: Todas las ramas activas deben tener al menos una PR en el periodo establecido.

ITP4: Evitar complejidad alta de la Pull Request (IC)

- Descripción: Esta métrica mide la complejidad de las PR utilizando el índice de complejidad ciclomática (CCM), que evalúa la cantidad de caminos lógicos en el código. Un CCM alto puede indicar código difícil de mantener y propenso a errores.

- Parámetro: CCM de cada PR.
- Umbral: El valor máximo permitido de CCM es de 6.
- Objetivo: Al menos el 90% de las PR deben tener un CCM de 6 o menos semanalmente.

CTP5: Evitar un número alto de ramas activas en el repositorio

- Descripción: Mantener el número de ramas activas bajo control ayuda a evitar la fragmentación del código y facilita la gestión del repositorio.
- Parámetro: Número de ramas activas en el repositorio.
- Cálculo: El número de ramas activas debe ser menor o igual al número de miembros del equipo más uno.
- Frecuencia: Semanal.

CTP6: Evitar revisiones de PR asíncronas

- Descripción: Esta métrica garantiza que las PR reciban atención rápida, mejorando la colaboración y acelerando el ciclo de desarrollo.
- Parámetro: Tiempo desde que se abre una PR hasta que recibe el primer comentario.
- Umbral: Un tiempo límite establecido (TLTP6).
- Objetivo: Al menos el 90% de las PR deben recibir un comentario en menos del tiempo límite.

CTP7: Reducir el porcentaje de PR fusionadas sin push

- Descripción: Asegura que las PR contengan el trabajo más reciente y no se fusionen sin actualizaciones recientes.
- Parámetro: Proporción de PR fusionadas sin push recientes.
- Objetivo: Minimizar esta proporción para asegurar la integración continua del trabajo más reciente.

CTP8: Evitar confirmaciones de código sin ejecutar pruebas automáticas antes

- Descripción: Asegura que todas las confirmaciones de código (commits) pasen por pruebas automáticas antes de ser aceptadas, garantizando la calidad del código.
- Parámetro: Proporción de commits que pasan por pruebas automáticas antes de ser aceptados.
- Objetivo: Garantizar que todas las confirmaciones sigan este proceso.

CTP9: Evitar un número alto de PR rechazadas

- Descripción: Reducir el número de PR rechazadas mejora la calidad del código y la eficiencia del equipo.
- Parámetro: Proporción de PR rechazadas semanalmente.
- Objetivo: Minimizar esta proporción para asegurar que el trabajo presentado cumple con los estándares esperados.

CTP10: Evitar que las PR estén sin revisor durante mucho tiempo

- Descripción: Las PR deben ser asignadas a un revisor rápidamente para asegurar un ciclo de revisión ágil.
- Parámetro: Tiempo que una PR permanece sin asignar a un revisor.
- Objetivo: Minimizar este tiempo para acelerar el proceso de revisión.

CTP11: Evitar periodos largos de revisión de código

- Descripción: Asegura que las PR no queden estancadas en el proceso de revisión.
- Parámetro: Tiempo total que tarda una PR en ser revisada y aprobada.
- Objetivo: Reducir este tiempo para mantener un flujo de trabajo continuo.

CTP12: Controlar el tiempo de ejecución de pruebas

- Descripción: Garantizar que las pruebas se ejecuten rápidamente para no retrasar el flujo de trabajo.
- Parámetro: Tiempo de ejecución de las pruebas.
- Umbral: Un límite máximo aceptable.
- Objetivo: Mantener el tiempo de ejecución de las pruebas dentro de este límite.

CTP13: Fusionar con la rama principal a la frecuencia establecida

- Descripción: Mantener la integración continua mediante fusiones regulares con la rama principal.
- Parámetro: Frecuencia de fusiones con la rama principal.
- Objetivo: Asegurar que estas fusiones ocurran con la frecuencia establecida.

CTP14: Desencadenar flujos de trabajo

- Descripción: Asegura que cada PR desencadene un flujo de trabajo automatizado, que incluya pruebas y otras verificaciones.
- Parámetro: Proporción de PR que desencadenan correctamente estos flujos.
- Objetivo: Garantizar que todos los PR activen los flujos de trabajo esperados.

CTP15: Lanzar pruebas automatizadas tras cada pull request

- Descripción: Asegura que cada PR desencadene un conjunto de pruebas automatizadas para verificar la calidad del código.
- Parámetro: Proporción de PR que lanzan pruebas automatizadas.
- Objetivo: Garantizar que todas las PR pasen por pruebas automatizadas.

ITP16: Lanzar pruebas automatizadas tras cada push

- Descripción: Similar a CTP15, pero aplicado a cada push a una rama.
- Parámetro: Proporción de pushes que lanzan pruebas automatizadas.
- Objetivo: Asegurar que todos los pushes sean verificados automáticamente.

CTP17: Comprobar automáticamente cada Pull Request

- Descripción: Cada PR debe ser verificada automáticamente mediante herramientas que aseguren la calidad del código.
- Parámetro: Proporción de PR verificadas automáticamente.
- Objetivo: Asegurar que todas las PR pasen por este proceso de verificación.

ITP18: Comprobar automáticamente cada push a la rama

- Descripción: Similar a CTP17, pero aplicado a cada push individual a las ramas del repositorio.
- Parámetro: Proporción de pushes verificados automáticamente.
- Objetivo: Asegurar que todos los pushes pasen por esta verificación.

CTP19: Evitar subidas con errores

- Descripción: Garantizar que los flujos de trabajo automatizados se completen sin errores.
- Parámetro: Proporción de flujos exitosos.
- Umbral: Un máximo tolerable de fallos.
- Objetivo: Minimizar la proporción de errores en los flujos de trabajo.

CTP20: Agilizar la resolución de flujos de trabajo fallidos

- Descripción: Cuando un flujo de trabajo falla, debe ser resuelto rápidamente.
- Parámetro: Tiempo que toma resolver los fallos en los flujos de trabajo.
- Objetivo: Reducir este tiempo para asegurar la continuidad del desarrollo.

CTP21: Verificar la ausencia de Bug Issues sin Pruebas Asociadas

- Descripción: Cada bug identificado debe tener pruebas asociadas para garantizar su correcta resolución.
- Parámetro: Número de bugs sin pruebas correspondientes.
- Objetivo: Asegurar que todos los bugs tengan pruebas asociadas.

CTP22: Reducir tiempos entre una pull request y la ejecución del workflow

- Descripción: Esta métrica mide el tiempo que transcurre entre la creación de una PR y la ejecución de los flujos de trabajo asociados, estableciendo un límite máximo aceptable.
- Parámetro: Tiempo entre la creación de la PR y la ejecución del workflow.
- Objetivo: Minimizar este tiempo para asegurar una rápida verificación del código.

CTP23: Evitar releases con bugs

- Descripción: Se debe minimizar el número de errores en las versiones liberadas.
- Parámetro: Proporción de versiones con bugs.

- Objetivo: Reducir esta proporción al mínimo, asegurando la calidad de las releases.

CTP24: Reducir tiempo de vida de bugs

- Descripción: Los bugs deben ser resueltos rápidamente.
- Parámetro: Tiempo que un bug permanece abierto.
- Objetivo: Reducir este tiempo para asegurar la pronta resolución de los problemas.

Estas métricas y objetivos están diseñados para mejorar la eficiencia del equipo de desarrollo, asegurar la calidad del código y mantener un flujo de trabajo ágil y organizado. Cada práctica está respaldada por métricas específicas y parámetros definidos que permiten evaluar el cumplimiento de los objetivos establecidos.

CAPITULO 5: ANÁLISIS

5.1 OBJETIVO DEL SISTEMA

El objetivo principal de este proyecto es desarrollar una interfaz intuitiva para los usuarios, que les permita añadir Team Practices Agreements (TPA) de una forma sencilla, probada, eficiente y visual.

Esta interfaz no solo facilitará la prueba de acuerdos descritos en un lenguaje formal, sino que también permitirá la extracción de estas funcionalidades para su implementación en otros proyectos independientes. Para esto deberá permitir extraer métricas de aquellos Team Practices Agreements que ya estén dentro de Bluejay y permitirte probar esas mismas métricas para otros acuerdos de equipo.

El primer paso será la creación de una herramienta que permita probar métricas de forma individual. Esto es crucial porque cada métrica dentro de un acuerdo de prácticas del equipo (TPA, Team Practice Agreement) debe ser evaluada y validada de forma aislada antes de integrarse en el sistema global. La herramienta debe proporcionar una manera eficiente y clara de verificar la funcionalidad y precisión de cada métrica.

Esta herramienta, deberá permitir el siguiente ciclo de vida para la creación de una métrica. En primer lugar, habrá que crear cada métrica individualmente. Cuando la tengamos, debemos de poder probar que devuelve los datos correctos al realizar las acciones que mida la métrica. Una vez comprobado que la métrica funcione correctamente, se deberá poder añadir a un acuerdo de equipo.



Para probar las métricas que creamos en un entorno real, debemos tener una herramienta que nos permita hacer pruebas de forma eficiente. Para ello será necesario una interfaz que nos permita interaccionar con Github y tras ellos probar si nuestra métrica ha detectado los cambios tal y como lo queríamos.

Para ello vamos a necesitar 2 formas de hacerlo, una más fácil visualmente, y otra menos visual pero más eficiente.

- La forma más visual será más fácil para el usuario ya que se interaccionará de una forma más gráfica y deberá ser una herramienta que permita hacer todas las acciones de Github que después podamos medir una métrica.
- Y la otra forma, menos visual, pero mucho más eficiente y replicable ya que deberá permitirnos ejecutar un caso de prueba que ejecute todas las acciones que pueda medir la métrica que queremos probar.

El objetivo es que ya sea con la forma más visual, o la más eficiente se pueda llegar a hacer las acciones que mide la métrica y comprobar el correcto funcionamiento de esta.

Una vez ya podamos tener métricas individuales probadas, entonces debemos estar en disposición de agruparlas en algo mayor, es decir en un TPA. Para ello se desarrollará una herramienta o página que permita la gestión completa de los TPA. Esto incluye la capacidad de actualizar, eliminar o subir nuevos TPA de forma gráfica e intuitiva a la plataforma Bluejay. Esta funcionalidad es esencial para mantener los acuerdos de prácticas de equipo actualizados y alineados con las necesidades y expectativas cambiantes del equipo.

Una vez desarrollada la herramienta de gestión completa de TPAs, el siguiente objetivo es la creación formal de un Team Practice Agreement (TPA). Esto implica definir claramente las métricas y garantías que serán parte del TPA. Una vez establecido, se utilizará la herramienta TP-Tester para implementar y verificar estas métricas en la práctica, asegurando su correcto funcionamiento.

Además de las herramientas de gestión y prueba, se investigará el uso de visualizaciones gráficas en el dashboard de Bluejay, que actualmente utiliza Grafana. El objetivo es evaluar las visualizaciones existentes y explorar la posibilidad de añadir nuevas visualizaciones personalizadas. Esto no solo mejorará la capacidad de análisis y seguimiento del rendimiento de las métricas, sino que también enriquecerá la experiencia del usuario al proporcionar una representación más comprensible y detallada de los datos.

En resumen, los objetivos técnicos del proyecto se pueden desglosar en:

- Desarrollo de una herramienta de extensión para Bluejay que permita la creación, edición y eliminación de métricas, así como la gestión completa de los TPA, y que incluya funcionalidades para crear scripts de pruebas automáticas.
- Elaboración de un documento formal de acuerdos de equipo que se convertirá en un TPA, con sus métricas y garantías definidas.
- Investigación y mejora de las visualizaciones gráficas en Grafana, añadiendo nuevas visualizaciones personalizadas para mejorar la interpretación de los datos que se han obtenido con la creación de este TPA.

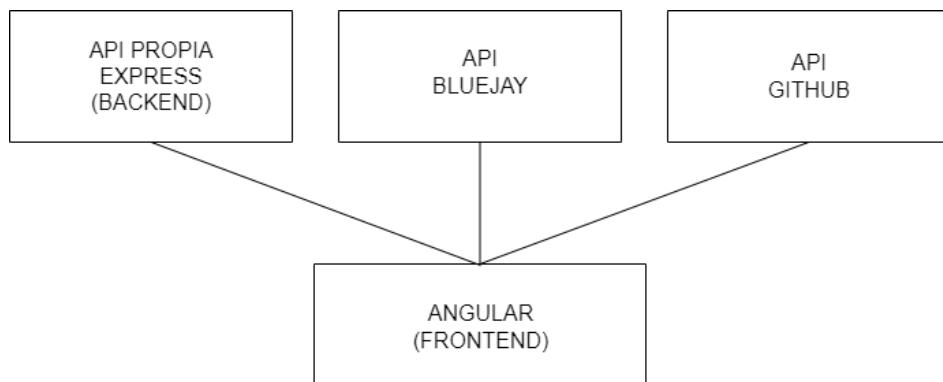
5.2 ARQUITECTURA Y DISEÑO FUNCIONAL

El diseño y la arquitectura de este proyecto se basan en la creación de una aplicación frontend robusta desarrollada en Angular, que interactuará tanto con una API backend propia construida en Express como directamente con otras APIs externas, incluyendo la API de Bluejay y la API de GitHub. Esta arquitectura asegura una comunicación eficiente entre los diferentes componentes del sistema y una gestión centralizada de las funcionalidades críticas.

5.2.1 FRONTEND

La aplicación frontend en Angular se organizará en varias capas para mantener una estructura clara y modular, permitiendo la escalabilidad y el mantenimiento eficiente del código.

Además, el frontend deberá ser capaz de comunicarse con todas aquellas APIs que son necesarias para la subida de las métricas y las pruebas de estas métricas en github. También debe ser capaz de comunicarse con la API propia que será la encargada del guardado local de métricas y scripts de tests.



Los **modelos** serán las clases que definen la estructura y el esquema de los datos que manejará la aplicación. Estos modelos serán claves para tipar las respuestas y peticiones HTTP, asegurando que los datos recibidos y enviados cumplen con los formatos esperados. Por ejemplo, un modelo de métrica podría definir los atributos necesarios como el nombre de la métrica, su valor, y su descripción, proporcionando una estructura clara y consistente para las operaciones de CRUD (crear, leer, actualizar, borrar).

Los **servicios** se encargarán de la comunicación con el backend y otras APIs externas. Los servicios realizarán las llamadas HTTP a la API de Express, a la API de Bluejay y a la API de GitHub. Cada servicio estará especializado en una tarea específica:

- **Servicio de Repositorios:** Gestionará las operaciones relacionadas con los repositorios locales, como el clonado, la creación de archivos, y la ejecución de scripts.
- **Servicio de TPAs:** Permitirá la creación, actualización, y eliminación de Team Practice Agreements (TPAs).

- **Servicio de Bluejay:** Interactuará directamente con la API de Bluejay y sus microservicios, facilitando la integración de las funcionalidades de Bluejay en la aplicación.
- **Servicio de GitHub:** Realizará llamadas a la API de GitHub para obtener issues, crear nuevas issues, y realizar otras operaciones necesarias en los repositorios de GitHub.

Los **componentes** serán utilizados como bloques reutilizables. Cada componente encapsula una parte específica de la UI y puede ser reutilizado en diferentes páginas para evitar la duplicación de código. Los componentes proporcionarán funcionalidades como:

- **Formulario de Gestión de TPAs:** Permite a los usuarios crear, editar y eliminar TPAs.
- **Panel de Métricas:** Muestra las métricas actuales y permite su gestión.
- **Integración con GitHub:** Facilita la visualización y gestión directamente de servicios de Github como pueden ser las issues abiertas o el nombre de las ramas que tiene el repositorio directamente desde el frontend.
- Ejemplos de posibles **casos de usos** en el yaml de pruebas automáticas.
- Botones que abrirán **pop-up** con tutoriales o información extra sobre el funcionamiento de alguna página.
- Componentes que se reutilizan en todas las páginas como por ejemplo el header y el footer de la página.

Las **páginas** representan vistas completas de la aplicación que agrupan varios componentes. Cada página se enfocará en una funcionalidad principal:

- **Página de Gestión de TPAs:** Centraliza todas las operaciones relacionadas con los TPAs.
- **Página de Métricas:** Permite a los usuarios visualizar y gestionar las métricas.
- **Página de Tests con GitHub:** Proporciona una interfaz para interactuar con los repositorios de GitHub, gestionando issues, Pull requests, ramas y otros aspectos relevantes.
- **Página de Test Automatizados:** Proporciona una interfaz para la ejecución de scripts en formato yaml que interactúen tanto con las API de github, como con el backend.

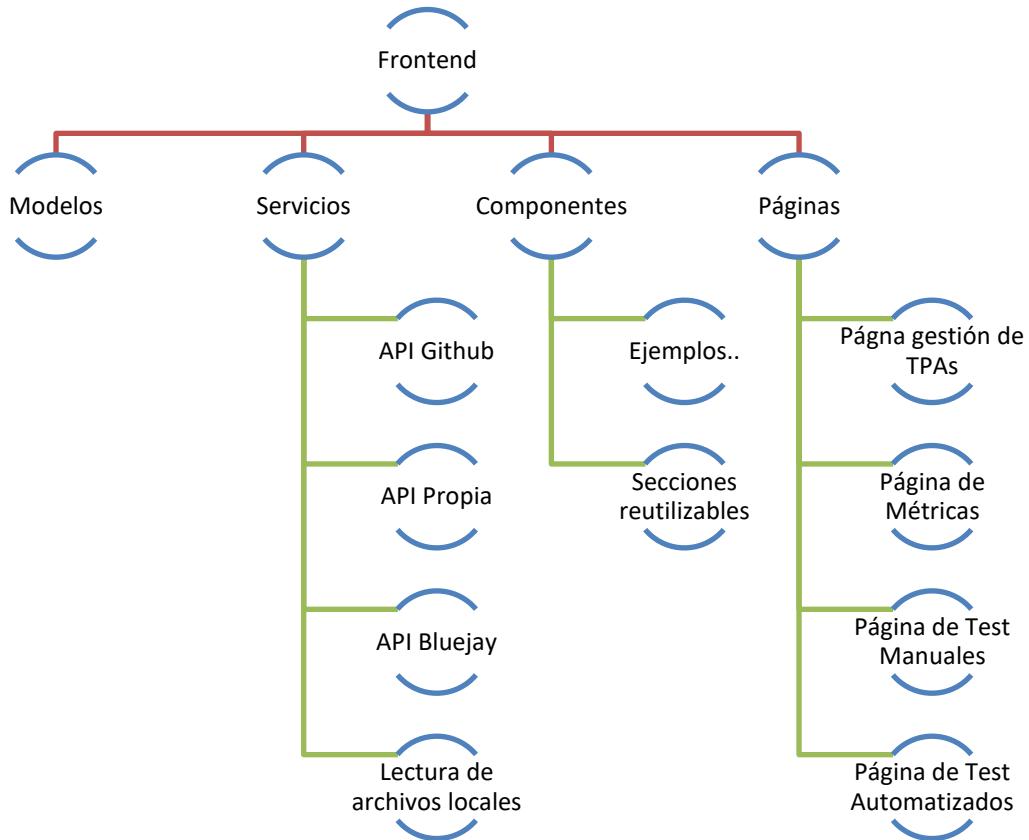


Ilustración 57. Estructura del proyecto Angular

5.2.2 BACKEND

El backend se desarrollará en Express y se organizará en varias capas para manejar las tareas de manera eficiente y facilitar la integración con otros servicios.

El backend gestionará repositorios locales, incluyendo:

- Clonado de Repositorios: Utilizando comandos de Git para clonar repositorios en el servidor local.
- Gestión de Archivos: Creación, modificación y eliminación de archivos en los repositorios.
- Ejecución de Scripts: Permite ejecutar scripts de prueba en los repositorios clonados para verificar la funcionalidad y precisión de las métricas.
- Generación de hashes para aquellos archivos que sea necesario.
- Traducción de archivos yaml, a código más legible por javascript.

El backend será responsable de almacenar y gestionar las métricas, scripts y TPAs. Esto incluirá:

- Operaciones CRUD para Métricas: Crear, leer, actualizar y eliminar métricas.
- Gestión de TPAs: Permitir la creación, actualización y eliminación de TPAs, asegurando que los acuerdos de prácticas del equipo estén siempre actualizados.
- Interacción con APIs Externas.
- Operaciones CRUD para Scripts YAMLS.

Ilustración 58. Vista de las operaciones de la API desde Swagger

Para mejorar la documentación y facilitar el uso de la API, el backend incluirá una página de información utilizando Swagger. Este proporcionará una interfaz gráfica donde se podrán ver todas las llamadas disponibles en la API, junto con su descripción, parámetros requeridos y ejemplos de respuestas. Esta documentación interactiva permitirá a los desarrolladores explorar y probar las diferentes funcionalidades de la API de manera fácil y eficiente. Además de ello, en caso de fallo, permitirá realizar pruebas directamente sobre una interfaz gráfica sin ser necesario el uso de otros programas como postman.

5.2.3 FLUJO DE INFORMACIÓN

El flujo de datos en la aplicación seguirá un camino bien definido, asegurando una comunicación eficiente entre el frontend, el backend y las APIs externas:

- Interacción del Usuario: El usuario interactúa con la interfaz de Angular, generando eventos que requieren comunicación con el backend o con APIs externas.
- Servicios de Angular: Los servicios en Angular envían las peticiones HTTP correspondientes a la API de Express, la API de Bluejay o la API de GitHub.
- Procesamiento en el Backend: La API de Express recibe las peticiones, realiza las operaciones necesarias (como interactuar con repositorios locales, gestionar métricas y TPAs), y devuelve la respuesta adecuada.
- Actualización del Frontend: Los servicios en Angular procesan las respuestas y actualizan los modelos y componentes de la UI, reflejando los cambios realizados.

Esta arquitectura modular y bien definida asegura que la aplicación sea escalable, fácil de mantener y capaz de integrarse eficazmente con múltiples servicios externos, proporcionando una experiencia de usuario coherente y eficiente.

CAPITULO 6: IMPLEMENTACIÓN

6.1. ENTORNO DE DESARROLLO

Para el desarrollo del TP-Tester, se han utilizado diversas herramientas y tecnologías que han permitido un flujo de trabajo eficiente y una gestión integral del proyecto. La selección de cada herramienta ha sido cuidadosamente considerada para maximizar la productividad, garantizar la calidad del código y facilitar la colaboración entre los miembros del equipo. Estas herramientas no solo ofrecen funcionalidades avanzadas para la codificación, depuración y pruebas, sino que también aseguran que cada aspecto del desarrollo esté bien gestionado y optimizado para cumplir con los estándares más altos de la industria.

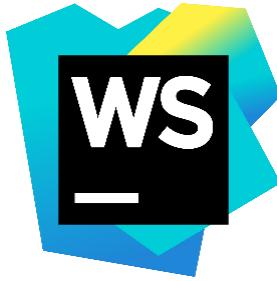
El entorno de desarrollo elegido cubre todos los aspectos del ciclo de vida del software, desde la escritura del código y la gestión de versiones hasta el despliegue y la monitorización en producción. La elección de WebStorm como IDE principal, por ejemplo, se basa en su capacidad para manejar tanto el frontend como el backend en un solo proyecto, ofreciendo características como autocompletado de código, refactorización avanzada y navegación inteligente entre archivos y proyectos. Estas características son fundamentales para mejorar la productividad y la eficiencia del desarrollo.

Además de WebStorm, se han utilizado otras herramientas esenciales como Node.js y npm para la ejecución de JavaScript en el servidor y la gestión de paquetes respectivamente. Estas herramientas permiten desarrollar aplicaciones de manera rápida y eficiente, aprovechando el modelo de I/O asíncrono y no bloqueante de Node.js. Docker también ha sido una elección clave para la contenerización de aplicaciones, asegurando que el software se ejecute de manera consistente en cualquier entorno. Herramientas de prueba como Postman y de control de versiones como Git y GitHub Desktop facilitan la colaboración y aseguran la calidad del código. Finalmente, Toggl Track se ha utilizado para el seguimiento del tiempo, permitiendo una gestión eficiente del tiempo y la optimización de los recursos. A continuación, se describen en detalle estas herramientas y tecnologías, junto con sus beneficios específicos y la razón de su elección.

6.1.1 WEBSTORM

WebStorm es el IDE elegido para este proyecto ya que era el que era el que más se ajustaba para un desarrollo tanto frontend como backend dentro del mismo proyecto [13].

Es un IDE completo; A diferencia de VS Code, que es un editor de texto con extensiones, WebStorm está diseñado como un IDE (Entorno de Desarrollo Integrado) con una amplia gama de características integradas.



Esto incluye autocompletado de código, refactorización avanzada, navegación inteligente entre archivos y proyectos, y una integración profunda con Git y otros sistemas de control de versiones.

Toda esta gama de posibilidades resulta en una **productividad mejorada**, reduciendo así la necesidad de instalar y configurar múltiples extensiones, lo que puede ahorrar tiempo y minimizar los problemas de compatibilidad y configuración.

Soporte Específico para Angular y Node.js:

- Soporte para **Angular**: Ofrece soporte de primer nivel para Angular, con características como el análisis de código Angular, resaltado de sintaxis específico, y la integración de Angular CLI. Esto facilita el desarrollo de aplicaciones Angular con herramientas como la generación de componentes, servicios y módulos con comandos simples.
- Soporte para **Node**: WebStorm proporciona una excelente integración con Node.js, incluyendo depuración avanzada, ejecución de pruebas y un buen rendimiento del conjunto. La configuración de servidores de desarrollo y las opciones de depuración son intuitivas y potentes, lo que acelera el ciclo de desarrollo y la resolución de problemas.

Depuración y Herramientas de Desarrollo:

- **Depuración Avanzada**: La depuración en WebStorm es robusta y fácil de usar, con soporte para depuración de Node.js y Angular, incluyendo puntos de interrupción, inspección de variables, y seguimiento de la ejecución. Esta funcionalidad es más profunda y directa en comparación con las soluciones de depuración disponibles en VS Code.
- **Herramientas de Análisis y Calidad de Código**: Incluye herramientas integradas para análisis de código estático, refactorización avanzada, y control de calidad del código, ayudando a mantener un alto estándar de calidad y consistencia en el desarrollo.

Ilustración 59. Herramienta GIT dentro de Webstorm.

Cómo se puede ver, dentro del propio entorno de desarrollo, tenemos una sección para poder gestionar el control de versiones, haciendo commits, pudiendo vincularlos con las issues ya creadas. Además de esto también incluye otro tipo de funcionalidades como resolución de conflictos.

Características extras:

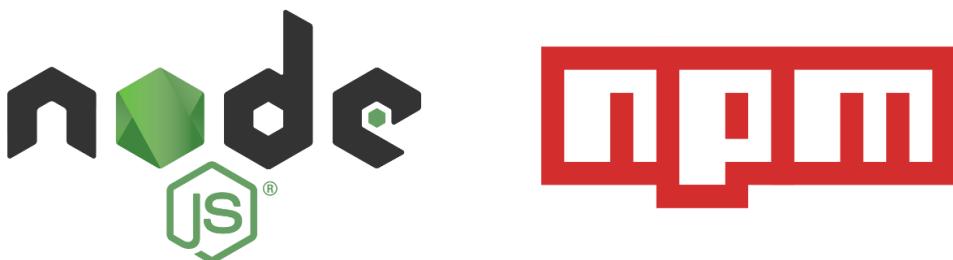
- Interfaz Intuitiva y Configurable:** La interfaz está diseñada para mejorar la experiencia del desarrollador, con una organización clara y opciones de personalización que permiten adaptar el entorno a las preferencias individuales. Esto incluye el manejo eficiente de proyectos grandes y la gestión de múltiples archivos y ventanas.
- Integración con Herramientas Modernas:** WebStorm soporta una amplia gama de herramientas y tecnologías modernas, incluyendo Docker, Kubernetes, y bases de datos SQL/NoSQL, lo que facilita el trabajo con arquitecturas modernas y servicios en la nube.
- Actualizaciones Constantes:** JetBrains, la empresa detrás de WebStorm, proporciona actualizaciones regulares con nuevas características y mejoras, asegurando que el IDE se mantenga al día con las últimas tendencias y tecnologías del desarrollo web.

- **Soporte Técnico de Calidad:** Al tratarse de un editor de pago, (aunque también dispone de licencias gratuitas para estudiantes y comunidad educativa), los usuarios tienen acceso a un soporte técnico de calidad y documentación extensa, lo que es útil para resolver problemas y aprovechar al máximo todas las funcionalidades del IDE.

La elección de este entorno de desarrollo se justifica por su enfoque como un IDE completo con soporte avanzado para Angular y Node.js, lo que incrementa significativamente la productividad y la eficiencia en el desarrollo. Aunque VS Code es altamente personalizable y puede ser una opción válida, WebStorm ofrece una solución más integrada y optimizada para proyectos complejos que requieren un conjunto robusto de herramientas de desarrollo y depuración. Esta elección permitirá una experiencia de desarrollo más fluida y eficaz para tu aplicación Angular y backend en Node.js.

6.1.2 NODEJS y NPM

Node.js y npm son herramientas fundamentales en el desarrollo de aplicaciones modernas. Node.js permite ejecutar JavaScript en el servidor, mientras que npm gestiona los paquetes necesarios para el desarrollo. A continuación, se describen sus características principales y sus beneficios en el desarrollo de este proyecto.



6.1.2.1 NODEJS

Node.js [14] es un entorno de ejecución rápido y eficiente gracias a su uso del motor V8 de Chrome, que compila JavaScript directamente a código máquina, ofreciendo un rendimiento excepcional.

Además, Node.js utiliza un modelo **asíncrono y no bloqueante**, lo que permite manejar múltiples operaciones de entrada y salida sin bloquear la ejecución. Esto es ideal para aplicaciones que requieren alta escalabilidad y eficiencia, como servidores web y APIs, ya que puede gestionar miles de conexiones concurrentes con un bajo consumo de recursos.

Es especialmente adecuado para el desarrollo de **aplicaciones en tiempo real**. Es ideal para aplicaciones de chat y colaboración debido a su capacidad para manejar múltiples conexiones simultáneas y su bajo tiempo de respuesta. La arquitectura basada en eventos permite una comunicación bidireccional eficiente entre el cliente y el servidor. Además, su eficiencia en la gestión de flujos de datos lo hace perfecto para aplicaciones de streaming de medios y otras que requieren procesamiento en tiempo real, transmitiendo grandes cantidades de datos de manera continua y eficiente.

El **ecosistema de Node.js** es otro de sus puntos fuertes. La comunidad ha creado una vasta cantidad de módulos y bibliotecas que extienden su funcionalidad, facilitando el desarrollo de aplicaciones complejas.

Estos módulos, disponibles a través de npm, abarcan desde frameworks web hasta herramientas de prueba y utilidades de desarrollo. Una comunidad activa y en crecimiento asegura soporte continuo, actualizaciones frecuentes y una amplia gama de recursos educativos, lo que facilita el aprendizaje y la resolución de problemas.

6.1.2.2 NPM

npm (Node Package Manager) [15] es el gestor de paquetes predeterminado para Node.js. Se utiliza ampliamente en la comunidad de desarrollo para gestionar y compartir paquetes de código. npm permite a los desarrolladores instalar, compartir y distribuir código, así como gestionar las dependencias de sus proyectos de una manera sencilla y eficiente.

Otro punto de npm es la gran comunidad que apoya el proyecto y colabora con él. Gracias a esto, npm cuenta con más de un millón de paquetes disponibles, ofreciendo así soluciones para casi cualquier necesidad de desarrollo, desde frameworks frontend hasta herramientas de pruebas y utilidades de desarrollo.

Además, permite definir scripts personalizados en el archivo package.json, facilitando la automatización de tareas comunes como pruebas, construcción y despliegue o por ejemplo en caso de uso del proyecto, va a permitir ejecutar mediante un único script tanto el frontend como el backend del proyecto.

Estos scripts simplifican la gestión del ciclo de vida de los proyectos, permitiendo enfocarse en la escritura de código en lugar de en tareas repetitivas.

En relación con el proyecto, otra gran ventaja de usar npm es que es el gestor de paquetes oficial y predeterminado para Node.js, por lo que garantiza una total compatibilidad e integración con el backend a usar.

Uno de los puntos débiles de npm en el pasado era que no era tan eficiente en rendimiento como sus competidores, ya sea pnpm, deno o yarn. Pero en las últimas versiones se han introducido nuevas características como workspaces que permiten gestionar los monorepositorios de una manera mucho mas eficiente reduciendo así los tiempos anteriores.

Comparación con otros gestores:

- **Yarn:** Aunque Yarn fue creado para abordar algunos problemas de rendimiento y seguridad de npm, las versiones recientes de npm **han cerrado la brecha** en términos de velocidad y seguridad. Dado que npm es el gestor de paquetes oficial de Node.js, tiene un nivel de adopción y soporte más amplio.
- **Deno** es una plataforma de ejecución para JavaScript y TypeScript que incluye su propio sistema de gestión de dependencias, pero está diseñado para ser una **alternativa a Node.js**, no solo un gestor de paquetes. Elegir npm mantiene la compatibilidad con el amplio ecosistema de Node.js, mientras que Deno aún está en crecimiento y adopción.
- **PNPM:** Aunque pnpm es compatible con la mayoría de las bibliotecas de npm, algunas herramientas específicas pueden no estar completamente optimizadas para pnpm. Utilizar pnpm **no garantiza** una compatibilidad total con todas las herramientas y bibliotecas del ecosistema de JavaScript.

La elección de npm para la gestión de paquetes en este proyecto se basa en su integración nativa con Node.js, su popularidad y comunidad robusta, su simplicidad y mejoras recientes en velocidad y rendimiento. Aunque Yarn, Deno y pnpm ofrecen características atractivas, npm sigue siendo la opción más equilibrada y confiable para este proyecto, garantizando compatibilidad, eficiencia y acceso a un ecosistema rico en recursos y soporte.

6.1.3 DOCKER

Docker es una plataforma de contenerización que ha revolucionado el desarrollo y despliegue de aplicaciones. Al permitir a los desarrolladores empaquetar aplicaciones y sus dependencias en contenedores ligeros y portátiles, Docker asegura que el software se ejecute de manera consistente en cualquier entorno, desde el desarrollo hasta la producción [16].



Una de las principales características de Docker es su capacidad para crear contenedores ligeros y portátiles. Cada contenedor ejecuta una **instancia aislada de la aplicación**, incluyendo todas sus dependencias, lo que evita conflictos entre aplicaciones y facilita el mantenimiento. Además, los contenedores pueden ejecutarse de manera consistente en cualquier entorno que los soporte, ya sea en un entorno de desarrollo local, un servidor de producción o en la nube, asegurando una portabilidad óptima.

Docker permite la creación de imágenes reutilizables que encapsulan el código de la aplicación y todas sus dependencias, desde bibliotecas hasta configuraciones del sistema. Estas imágenes pueden ser almacenadas y compartidas a través de **Docker Hub** o registros privados, facilitando la distribución y el despliegue de aplicaciones. La gestión de la infraestructura también se simplifica, ya que Docker asegura que las aplicaciones se desplieguen de manera consistente en diferentes entornos, eliminando el clásico problema de "funciona en mi máquina". Los contenedores pueden ser fácilmente escalados hacia arriba o hacia abajo, permitiendo a las aplicaciones manejar variaciones en la carga sin problemas.

En términos de desarrollo ágil y eficiente, Docker permite a los desarrolladores crear entornos de desarrollo que son idénticos a los de producción, asegurando que el código funcione de la misma manera en todos los entornos. Docker se integra perfectamente con sistemas de **integración continua (CI)**, permitiendo la construcción y prueba automatizada de imágenes de contenedores en cada cambio de código. Esto asegura que el desarrollo sea ágil y eficiente, reduciendo errores y mejorando la calidad del software.

El despliegue de aplicaciones se simplifica considerablemente con Docker. Con **Docker Compose**, los desarrolladores pueden definir y ejecutar aplicaciones multi-contenedor de manera declarativa, simplificando la gestión de la infraestructura. Las aplicaciones empaquetadas en contenedores pueden ser desplegadas rápidamente, asegurando consistencia y reduciendo el tiempo de inactividad. Esto es particularmente

beneficioso para la gestión de despliegues en entornos de producción, donde la consistencia y la rapidez son cruciales.

La escalabilidad y la gestión de clústeres también se ven mejoradas con Docker. Herramientas como Docker Swarm y Kubernetes permiten gestionar y orquestar múltiples contenedores, facilitando el despliegue de aplicaciones distribuidas y escalables. La automatización de tareas comunes, como la gestión de recursos y la recuperación ante fallos, permite a las aplicaciones mantener alta disponibilidad y rendimiento, lo que es esencial para aplicaciones críticas y de gran escala.

En cuanto a los casos de uso, Docker es ideal para el desarrollo y despliegue de **microservicios**, donde cada servicio se ejecuta en su propio contenedor, facilitando la independencia y escalabilidad de los servicios. Los microservicios en contenedores permiten una clara separación de preocupaciones, lo que mejora el mantenimiento y la evolución de las aplicaciones.

En conclusión, Docker ha transformado la forma en que se desarrollan, despliegan y gestionan las aplicaciones. Al proporcionar una plataforma de **contenerización** que asegura consistencia, portabilidad y eficiencia, permite a los desarrolladores y equipos de operaciones trabajar juntos de manera más efectiva. Desde el desarrollo ágil hasta el despliegue escalable, Docker simplifica y optimiza el ciclo de vida de las aplicaciones, ofreciendo una solución robusta para los desafíos del desarrollo y despliegue moderno.

6.1.4 POSTMAN

Postman es una plataforma de colaboración y una herramienta potente para desarrollar, probar y documentar APIs (Application Programming Interfaces). Su popularidad ha crecido exponencialmente entre desarrolladores y equipos de QA debido a sus características integradas y su facilidad de uso. A continuación, se detallan las principales ventajas y aplicaciones de Postman [17].



Una de las ventajas más destacadas de Postman es su interfaz intuitiva y fácil de usar. Ofrece una interfaz gráfica de usuario (GUI) que facilita la creación, configuración y prueba de solicitudes HTTP. Además, permite definir diferentes entornos (desarrollo, pruebas, producción) y gestionar variables de entorno, simplificando la configuración y ejecución de pruebas en múltiples escenarios.

En cuanto al **desarrollo y pruebas de APIs**, Postman permite a los desarrolladores crear solicitudes HTTP (GET, POST, PUT, DELETE, etc.) con facilidad, añadiendo encabezados, parámetros, cuerpos de solicitud y autenticación de manera sencilla. Incluye una funcionalidad potente para escribir scripts de prueba utilizando JavaScript, lo que permite automatizar las pruebas de API, verificando respuestas, tiempos de respuesta y validando datos, asegurando que las APIs funcionen según lo esperado.

También facilita la colaboración y documentación. Permitiendo a los equipos colaborar en tiempo real, compartiendo **colecciones de solicitudes y entornos**. Las colecciones pueden versionarse, facilitando el seguimiento de cambios y la colaboración entre desarrolladores y testers. Además, Postman puede generar documentación detallada de las APIs directamente desde las colecciones de solicitudes, mejorando la comunicación.

A screenshot of the Postman application interface. The left sidebar shows 'My Workspace' with several collections listed: 'POST Collector Copy 3', 'POST Collector Copy 4', 'GET New Request Copy', 'POST Collector', 'POST 223', and a expanded section for 'MongoDB Data API' containing 'POST Insert Document', 'POST Find Document', 'POST Update Document', 'POST Delete Document', 'POST Insert Multiple Documents', 'POST Find Multiple Documents', 'POST Update Multiple Documents', 'POST Delete Many Documents', and 'POST Run Aggregation Pipeline'. The main panel displays the 'Overview' tab for the 'MongoDB Data API' collection. It includes a rich text editor with a preview of the collection's purpose: 'This collection is an introduction to the MongoDB Data API. The Data API provides you with a REST-like access to your data in MongoDB Atlas, the database-as-a-service offering by MongoDB.' Below this is a 'Getting Started' section and a link to 'View complete documentation →'. On the right side, there are details about the collection: 'Created by You' and 'Created on 29 Mar 2023, 12:53 PM'. The top navigation bar includes 'Home', 'Workspaces', 'API Network', 'Upgrade', and various tool icons.

Ilustración 60. Ejemplo de colecciones dentro de Postman

Las integraciones y el ecosistema de Postman son otras de sus fortalezas. Se integra con herramientas de integración continua y entrega continua (CI/CD) como Jenkins, Travis CI y GitHub Actions, **permitiendo automatizar las pruebas de API** como parte del proceso de despliegue, asegurando la calidad y fiabilidad de las aplicaciones. Además, ofrece una red de APIs pública donde los desarrolladores pueden explorar y probar APIs de terceros, y sus monitores permiten ejecutar pruebas de API periódicamente, verificando su disponibilidad y rendimiento.

En términos de depuración y análisis, Postman proporciona herramientas que permiten a los desarrolladores inspeccionar las solicitudes y respuestas, ver los encabezados y cuerpos de las respuestas, y rastrear problemas de manera eficiente. Genera informes detallados sobre las **pruebas ejecutadas**, proporcionando información valiosa sobre el rendimiento y el comportamiento de las APIs, los cuales pueden exportarse y compartirse para su análisis y discusión.

En conclusión, Postman es una herramienta integral que mejora significativamente el proceso de desarrollo y pruebas de APIs. Su interfaz intuitiva, capacidades de automatización, funciones de colaboración y amplias integraciones lo convierten en una elección indispensable para desarrolladores y equipos de QA. Utilizar Postman no solo acelera el desarrollo de APIs, sino que también **asegura la calidad** y la eficiencia del software, facilitando la creación de aplicaciones robustas y fiables.

6.1.5 GIT Y GITHUB DESKTOP

Git es un sistema de control de versiones distribuido que permite a los desarrolladores rastrear los cambios en el código fuente a lo largo del tiempo. GitHub Desktop es una aplicación que facilita la interacción con Git y GitHub, proporcionando una interfaz gráfica de usuario (GUI) amigable para gestionar repositorios. A continuación, se detallan las principales ventajas y aplicaciones de Git y GitHub Desktop:



6.1.5.1 GIT: CONTROL DE VERSIONES DISTRIBUIDOS

Rastreo de Cambios:

- **Historial Completo:** Git permite mantener un historial completo de todos los cambios realizados en el código, facilitando la revisión de modificaciones y la recuperación de versiones anteriores en caso de errores.
- **Comentarios y Mensajes:** Cada cambio (commit) se acompaña de un mensaje descriptivo, lo que ayuda a documentar el propósito y contexto de cada modificación, mejorando la comprensión y colaboración dentro del equipo.

Ramas y Fusión:

- **Desarrollo Paralelo:** Git permite crear ramas (branches) para desarrollar nuevas funcionalidades o corregir errores de manera aislada. Esto facilita el desarrollo paralelo y reduce el riesgo de conflictos entre desarrolladores.
- **Fusión Eficiente:** Git ofrece herramientas avanzadas para fusionar ramas, resolviendo automáticamente muchos conflictos y permitiendo a los desarrolladores integrar cambios de manera segura y controlada.

Distribución y Copias de Seguridad:

- **Sistema Distribuido:** Cada copia del repositorio es completa y contiene todo el historial del proyecto, lo que mejora la resiliencia y permite trabajar sin conexión. Esto también facilita la colaboración distribuida y la recuperación ante fallos.
- **Clonación y Forking:** Los repositorios pueden ser clonados y bifurcados (forked) para crear copias independientes, permitiendo a los desarrolladores experimentar y contribuir sin afectar el repositorio original.

6.1.5.2 GITHUB DESKTOP: INTERFAZ GRÁFICA PARA GIT Y GITHUB

Facilidad de Uso:

Interfaz Intuitiva: GitHub Desktop ofrece una GUI amigable que facilita la gestión de repositorios, commits, ramas y fusiones sin necesidad de utilizar la línea de comandos [18]. Esto es especialmente útil para desarrolladores que prefieren una interfaz visual.

Integración con Github:

Integración con GitHub: GitHub Desktop se integra perfectamente con GitHub, facilitando la clonación de repositorios, la sincronización de cambios y la creación de pull requests directamente desde la aplicación.

Un ejemplo de esto, es el tratamiento de issues. Como dicen las buenas prácticas de git, se debería crear una nueva rama por cada nueva funcionalidad, bug, o desarrollo.

Para ello, desde la propia web de Github te permite la creación de una nueva rama desde la propia página de la issue.

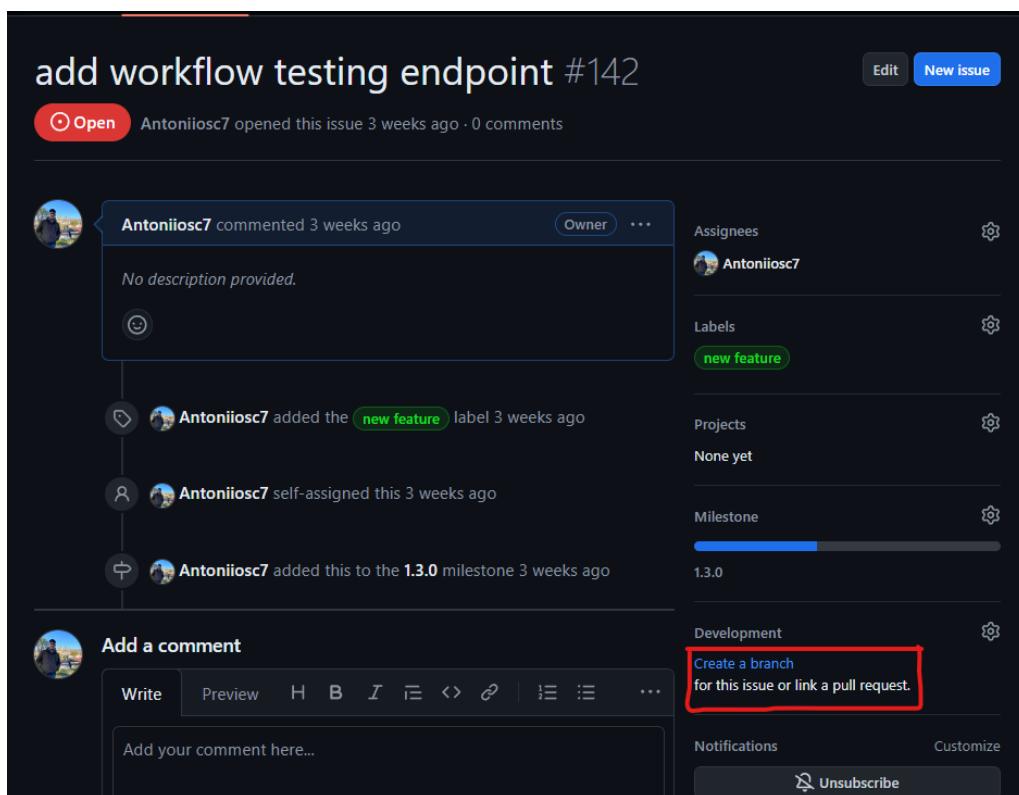


Ilustración 61. Creación de ramas desde página de Github.

Cuando creamos la nueva rama, podemos ver perfectamente un ejemplo de la integración entre Github y Github desktop ya que nos permite posicionarnos automáticamente en esta nueva rama con Github desktop.

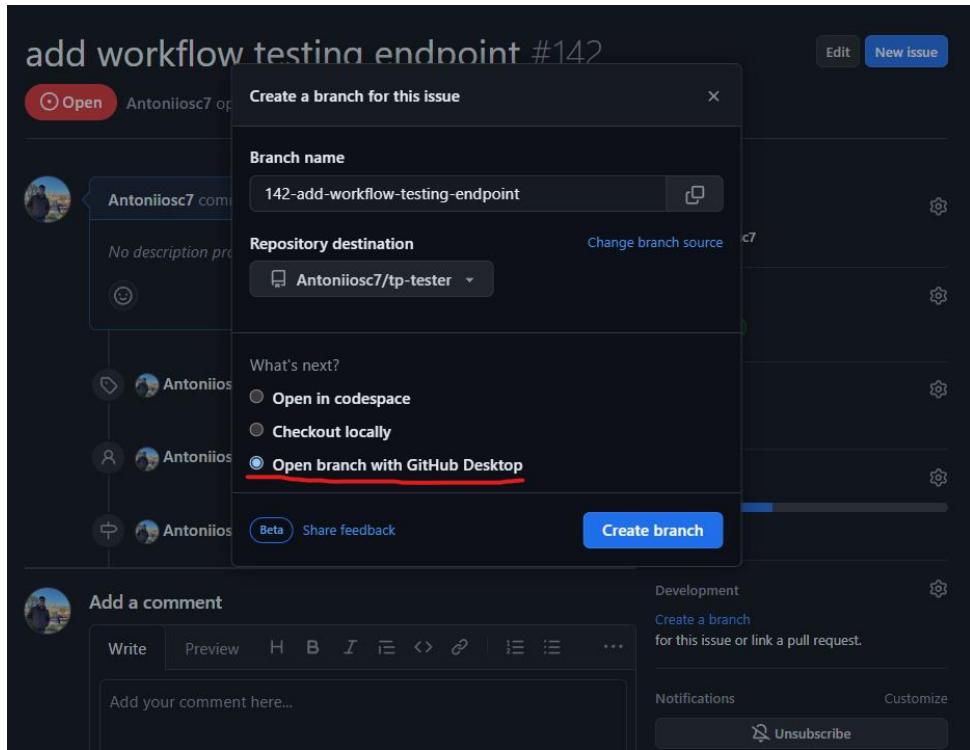


Ilustración 62. Ejemplo de abrir la nueva rama en github desktop.

Y si tenemos instalado Github Desktop en nuestro equipo, automáticamente nos saldrá un aviso para que si queremos llevarnos los cambios que tengamos sin subir a la nueva rama o dejarlos en la rama que estemos actualmente.

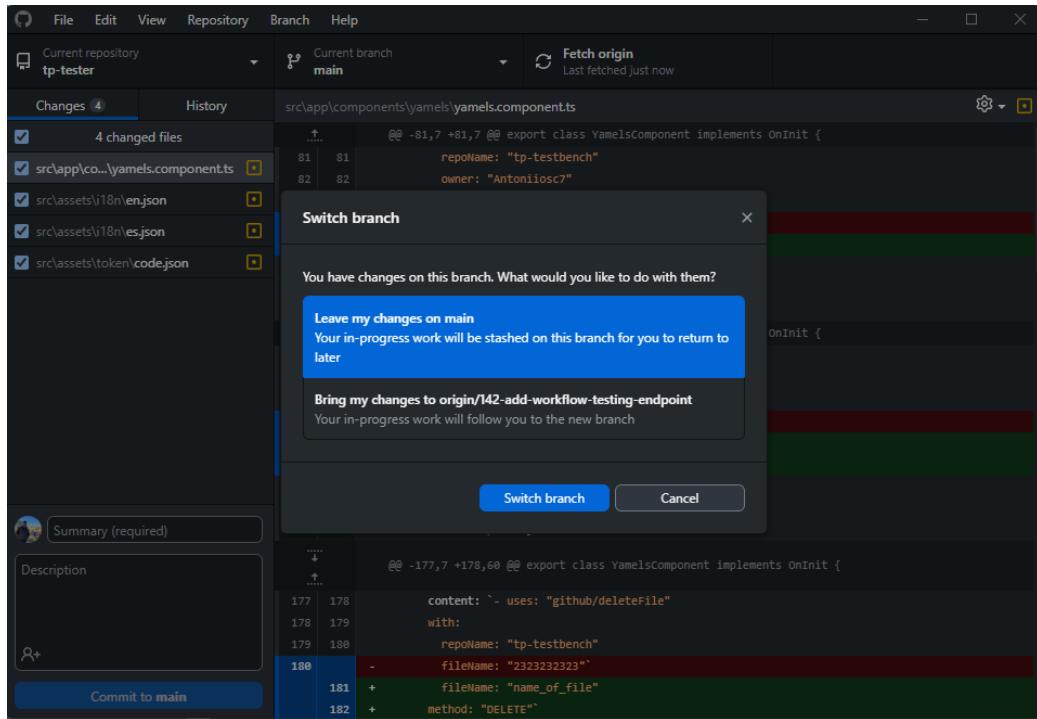


Ilustración 63. Cambio de rama en local con Github Desktop

Esta funcionalidad permite, además, que cuando cerramos esta nueva rama que hemos creado mediante una pull request a nuestra rama principal, automáticamente se cerrará la issue desde la que se creó la rama.

Gestión de Ramas y Fusiones:

- **Ramas Visuales:** GitHub Desktop proporciona una vista gráfica de las ramas, lo que facilita la creación, cambio y fusión de ramas. Los desarrolladores pueden visualizar el estado del proyecto y gestionar ramas de manera intuitiva.
- **Resolución de Conflictos:** La aplicación ofrece herramientas para ayudar a resolver conflictos de fusión, guiando a los usuarios a través del proceso de integración de cambios de manera segura.

Conclusión

Git y GitHub Desktop son herramientas fundamentales para cualquier desarrollador que trabaje en proyectos de software. Git ofrece un control de versiones robusto y flexible, permitiendo un desarrollo paralelo eficiente y un historial completo de cambios. GitHub Desktop, por su parte, proporciona una interfaz gráfica fácil de usar que simplifica la interacción con Git y GitHub, facilitando la colaboración y mejorando la productividad del equipo. Utilizar estas herramientas no solo mejora la gestión del código fuente, sino que también asegura un desarrollo ágil y colaborativo, esencial para el éxito de cualquier proyecto de software.

6.1.6 TOGGL TRACK

Toggl Track es una **herramienta de seguimiento del tiempo** que ayuda a individuos y equipos a gestionar su tiempo de manera efectiva. Esta aplicación permite rastrear cuánto tiempo se dedica a diferentes tareas y proyectos, proporcionando información valiosa para mejorar la productividad y eficiencia [19].

Una de las principales ventajas de Toggl Track es su capacidad de **seguimiento preciso del tiempo**. Permite registrar el tiempo exacto dedicado a cada tarea con solo un clic, lo que facilita el monitoreo de actividades y asegura un control preciso del tiempo invertido en cada proyecto. Además, cuenta con una función de temporizador automático que comienza a contar el tiempo automáticamente, evitando olvidos y garantizando un registro preciso incluso en días ocupados.



Otra ventaja significativa de es su capacidad para generar informes y análisis detallados. La herramienta produce informes personalizados que muestran cómo se **distribuye el tiempo en diferentes tareas** y proyectos, lo cual es útil para identificar áreas donde se puede mejorar la eficiencia. Además, ofrece gráficos y visualizaciones que facilitan la comprensión de los datos, ayudando a los usuarios a identificar patrones y tomar decisiones informadas sobre la gestión del tiempo.

También destaca por sus **integraciones** y automatización. Se integra con una amplia variedad de herramientas de gestión de proyectos, comunicación y productividad, como Asana, Trello, Slack y más, lo que permite un flujo de trabajo más fluido y una mejor sincronización de.

En cuanto a la mejora de la productividad personal, Toggl Track ayuda a identificar distracciones y áreas donde el tiempo no se está utilizando de manera efectiva, permitiendo implementar mejoras en la rutina diaria. También ayuda a los usuarios a establecer metas de tiempo para cada tarea, promoviendo una mejor gestión del tiempo y el cumplimiento de plazos.

En conclusión, Toggl Track es una herramienta poderosa para cualquier persona o equipo que busque mejorar su gestión del tiempo y aumentar la productividad. Con su capacidad para rastrear el tiempo de manera precisa, generar informes detallados y automatizar tareas rutinarias, facilita la optimización del tiempo y la eficiencia operativa. Utilizar Toggl Track no solo ayuda a gestionar mejor los proyectos y la facturación, sino que también proporciona **información valiosa para la mejora continua**, esencial en un entorno de trabajo dinámico y competitivo.

6.2 DISEÑO DE LA APLICACIÓN

El diseño de la aplicación sigue una arquitectura de microservicios, separando claramente las responsabilidades del backend y el frontend. Pero gracias a la biblioteca “concurrently” que veremos posteriormente en 6.4.6 BIBLIOTECAS UTILIZADAS, que permite la ejecución mediante scripts de varios a la aplicaciones a la vez, para el usuario funcionará como una única aplicación.

6.2.1 BACKEND

El backend está implementado usando Node.js y Express. Este servidor proporciona la API RESTful que maneja las solicitudes del frontend, gestiona la lógica de negocio y accede a la base de datos. Las características clave del backend incluyen:

- **API RESTful:** Exposición de varios puntos finales para interactuar con las métricas de TP, archivos guardados (ya sean métricas, scripts o archivos) y los TPAs (Acuerdos de Prácticas de Equipo).
- Documentación y pruebas con **Swagger:** Integración de Swagger para documentar y probar los endpoints de manera interactiva, facilitando el desarrollo y la integración.
- Gestión de **repositorios locales:** Clonado, creación, modificación y eliminación de archivos en los repositorios.
- Ejecución de **scripts:** Permite ejecutar scripts de prueba en los repositorios clonados para verificar la funcionalidad y precisión de las métricas.
- **Nedb:** Es una base de datos embebida para Node.js que se asemeja a MongoDB en términos de API, pero está diseñada para ser utilizada en aplicaciones pequeñas y medianas que no requieren la complejidad de un sistema de base de datos completo. Gracias a ella se podrá guardar los resultados de los tests ejecutados y comprobar los resultados posteriormente.

6.2.2 FRONTEND

El frontend está desarrollado con Angular, un framework robusto para construir aplicaciones web con gran porcentaje de código reutilizable gracias a su gran modularización. Las características clave del frontend incluyen:

- **Arquitectura de componentes:** Angular facilita la creación de componentes reutilizables y modulares, lo que mejora la mantenibilidad del código.
- **Comunicación con el backend:** Utilización de servicios Angular para realizar llamadas HTTP a la API de Express y a otras APIs externas.
- **Interfaz de usuario dinámica** y responsive: Proporciona una experiencia de usuario moderna y eficiente, permitiendo la gestión de TPAs, visualización de métricas y otras tareas relacionadas de una forma interactiva y actualizada.

6.3 STACK TECNOLÓGICO

6.3.1 ANGULAR

Angular es el framework de frontend utilizado para construir la interfaz de usuario. Proporciona un conjunto completo de herramientas para desarrollar aplicaciones web complejas, incluyendo gestión de rutas, servicios para llamadas HTTP, y componentes reutilizables. Angular facilita el desarrollo de una interfaz moderna y eficiente que se comunica con el backend de manera asíncrona.

El proyecto utiliza la versión 13.3.11. Me decanté por usar Angular y en concreto por la versión 13 ya que esta versión Angular incluye características como el cambio de detección, la detección de zona y la renderización del lado del servidor que ayudan a mejorar el rendimiento de las aplicaciones, y es muy cómoda la interacción con distintas APIs como pueden ser la API propia o la API de github. El proyecto se encuentra dividido en 3 grupos, componentes, páginas y servicios como detallaré en la sección de detalles de la implementación.

6.3.2 NODE

Node.js es la base del servidor backend, elegido por su capacidad para manejar aplicaciones escalables y de alto rendimiento. Su modelo de E/S no bloqueante y basado en eventos es ideal para aplicaciones que requieren manejar muchas conexiones simultáneas, como una API RESTful.

6.3.3. SISTEMA DE DESPLIEGUE (DOCKER)

Docker se utiliza para encapsular tanto el frontend como el backend en contenedores, asegurando que la aplicación se ejecute de manera consistente en cualquier entorno. El uso de Docker simplifica la implementación y la administración de las dependencias, eliminando problemas de configuración del entorno.

6.4 DETALLES DE LA IMPLEMENTACIÓN

6.4.1 TRADUCCIONES

En la implementación del sistema de traducciones para la aplicación, se ha utilizado la biblioteca **ngx-translate/core**, en combinación con **ngx-translate/http-loader**. Este sistema permite la gestión eficiente de múltiples idiomas en una aplicación Angular, facilitando así la internacionalización y localización del contenido. A continuación, se detalla cómo se ha configurado.

Configuración app.module

En el archivo app.module.ts, se configura el módulo de traducción. Esto incluye la configuración del cargador de traducciones, que se encargará de cargar los archivos JSON que contienen las cadenas traducidas:

```
import { HttpClientModule, HttpClient } from '@angular/common/http';
import { TranslateModule, TranslateLoader } from '@ngx-translate/core';
import { TranslateHttpLoader } from '@ngx-translate/http-loader';
import { AppComponent } from './app.component';

export function HttpLoaderFactory(http: HttpClient) {
  return new TranslateHttpLoader(http, './assets/i18n/', '.json');
}

@NgModule({
  declarations: [
    AppComponent,
    ...
  ],
  imports: [
    ...
    BrowserModule,
    HttpClientModule,
    TranslateModule.forRoot({
      defaultLanguage: 'en',
      loader: {
        deps: [HttpClient],
        provide: TranslateLoader,
        useFactory: HttpLoaderFactory
      }
    })
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Código 12. Configuración de las traducciones en el app.module

Estructura de archivos html

Para que las traducciones funcionen correctamente, ahora no se introducirá ningún texto directamente en el html, sino que se hará referencia a la traducción que se deberá usar en ese sitio.

Por ejemplo, el archivo html de la página que carga las acciones de github es así:

```
<div class="container">
<div class="row">
<div class="col-12 mt-3">
  <div class="alert alert-warning" role="alert">
    {{ 'GH_SIMULATOR.ADVER_INFO_1' | translate }}<br>
    <a href="https://docs.github.com/en/enterprise-server@3.9/authentication/keeping-your-account-and-data-secure/managing-your-personal-access-tokens" class="alert-link" target="_blank">Github token documentation</a>
    {{ 'GH_SIMULATOR.ADVER_INFO_2' | translate }}<br>
  </div>
</div>
</div>
<div class="row">
  <div class="col-12 d-flex justify-content-center mt-3">
    <div class="card mx-2 card-full-height" style="width: 18rem;">
      <div class="card-body">
        <h5 class="card-title">{{ 'GH_SIMULATOR.NEW_REPO' | translate }}</h5>
        <p class="card-text">{{ 'GH_SIMULATOR.NEW_REPO_INFO' | translate }}<br>
          <button class="btn btn-primary btn-edit" (click)="editAll()>{{ 'GH_SIMULATOR.NEW_REPO' | translate }}</button>
        </div>
      </div>
    <div class="card mx-2 card-full-height" style="width: 18rem;">
      <div class="card-body">
        <h5 class="card-title">{{ 'GH_SIMULATOR.CLONED_REPO' | translate }}<br>
        <p class="card-text">{{ 'GH_SIMULATOR.CLONED_REPO_INFO' | translate }}<br>
          <button class="btn btn-secondary btn-edit" (click)="editSections()>{{ 'GH_SIMULATOR.CLONED_REPO' | translate }}</button>
        </div>
      </div>
    </div>
  </div>
</div>
```

Código 13. Uso de las traducciones en archivos html

Como se puede observar en el texto resaltado en amarillo, simplemente se referencia en la traducción.

Estructura de Archivos de Traducción

Los archivos de traducción están ubicados en el directorio assets/i18n/x.json. Cada archivo JSON contiene las cadenas traducidas para un idioma específico. Por ejemplo, un archivo en.json para inglés y un es.json para español:

assets/i18n/en.json	assets/i18n/es.json
<pre>"GH_SIMULATOR": { "ADVER_INFO_1": "To clone a new repository you will need a valid token with access to that repository. For more information, visit:", "ADVER_INFO_2": "The token must be introduced in the clone new repository page.", "NEW_REPO": "Clone new repository", "NEW_REPO_INFO": "Here you can see all the repositories in which you have access and clone them to test metrics.", "CLONED_REPO": "Cloned repositories", "CLONED_REPO_INFO": "All the repositories that you have previously cloned.", }</pre>	<pre>"GH_SIMULATOR": { "ADVER_INFO_1": "Para clonar un nuevo repositorio necesitarás un token válido con acceso a ese repositorio. Para más información, visita:", "ADVER_INFO_2": "El token debe ser introducido en la página de clonar nuevo repositorio.", "NEW_REPO": "Clonar nuevo repositorio", "NEW_REPO_INFO": "Aquí puedes ver todos los repositorios a los que tienes acceso y clonarlos para probar métricas.", "CLONED_REPO": "Repositorios clonados", "CLONED_REPO_INFO": "Todos los repositorios que has clonado previamente.", }</pre>

Actualmente, el proyecto solo cuenta con los archivos en inglés y en español, pero para añadir nuevos idiomas en TP-Tester bastaría con la creación de un nuevo json en ese directorio con las mismas llaves y con otros valores.

Selección de idioma

Para seleccionar el idioma, por defecto, se seleccionará el idioma que usa el navegador por defecto, aunque esto también se podrá cambiar dentro de la página, en la parte superior derecha seleccionando la bandera del idioma que queramos usar.



Ilustración 64. Selección de idiomaç

6.4.2 GESTIÓN DE ARCHIVOS LOCALES

Para el funcionamiento de TP-Tester, es necesario guardar muchos archivos en el directorio local, ya que tenemos funcionalidades que permiten guardar las métricas, TPAs, variables de entornos, claves de Github y scripts de pruebas del proyecto.

Como el proyecto está pensando para el uso local y que no sea necesario tener levantado todo el ecosistema de bluejay, no se ha optado por el uso de una base de datos para guardar estos archivos, por lo tanto, este guardado se realiza en el directorio local del proyecto.

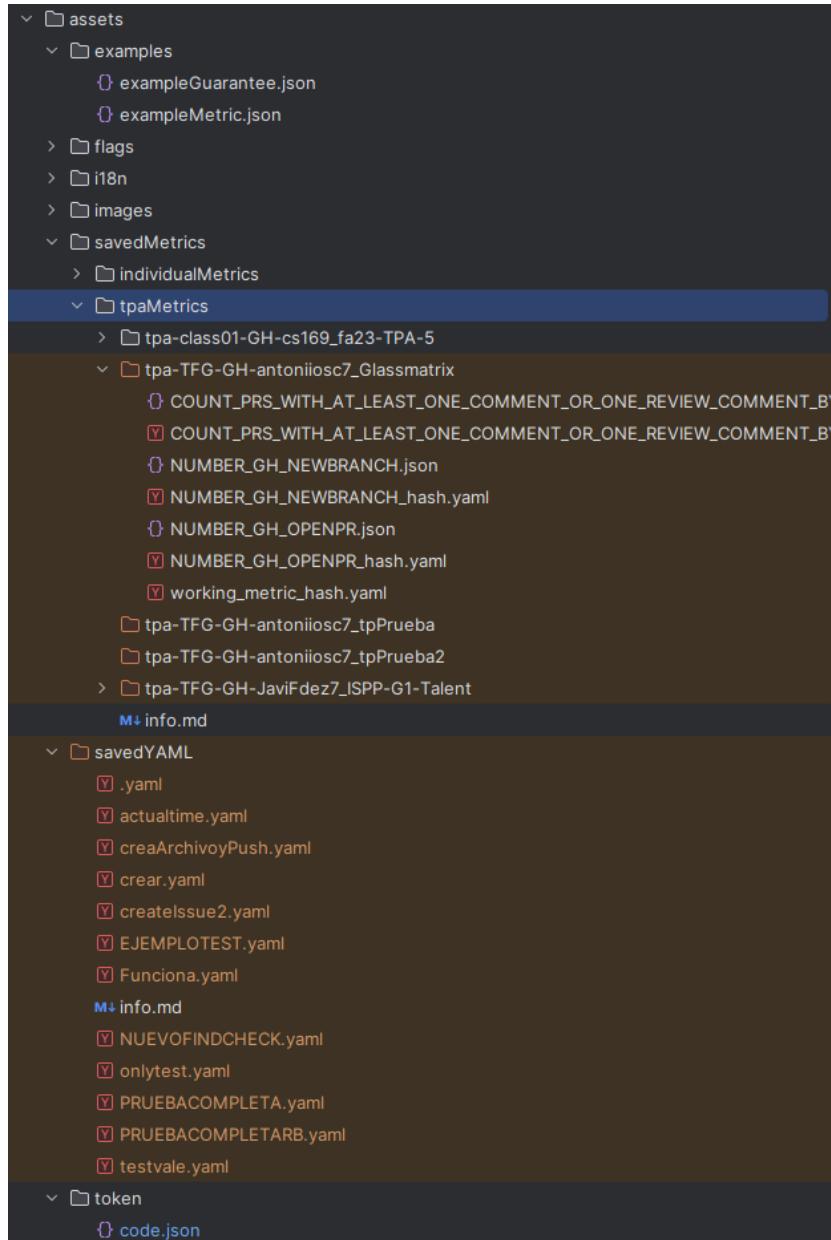


Ilustración 65. Estructura de carpetas de archivos guardados

Sabiendo esto, el guardado de los archivos se realizará en la carpeta assets del proyecto Angular. Dentro de el tendremos las siguientes carpetas:

- **savedMetrics**: Es la carpeta principal usada para el guardado de las métricas. Dentro de ella tendremos dos nuevos directorios, uno para el guardado de métricas individuales y otro para el guardado de métricas que provienen de un TPA.
 - **individualMetrics**: En esta carpeta se guardarán las métricas individuales, es decir, aquellas que no provienen de un TPA y que han sido guardadas directamente desde la carpeta de métricas. Estas métricas se guardan en formato .json.
 - **tpaMetrics**: Esto contendrá a su vez, una subcarpeta con el nombre de cada TPA del que hayamos guardado una métrica de forma individual. Y cada subcarpeta contendrá dos archivos por cada métrica guardada, un archivo .json que contendrá el contenido de la métrica y a su vez otro con el formato _hash.yaml, que guardará el hash original de la métrica cuando se guarda por primera vez.
- **savedYAML**: En esta carpeta se guardan todos los archivos de prueba de ejecución. Estos son unos scripts que irán ejecutando una serie de pasos para probar que la métrica funcione correctamente y están en formato .yaml.
- **token**: La carpeta token es la encargada de guardar el token de github que se haya guardado.
- **examples**: Aquí se encuentran ejemplos tanto de métricas como garantías que se cargan cuando desde el front se pulsa el botón de cargar un ejemplo.

Para este guardado, la aplicación frontend, Angular, mandará el contenido de estos archivos y el nombre que debe tener al backend, que será el encargado de crear los nuevos archivos o actualizar los ya existentes.

POST	/glassmatrix/api/v1/bluejay/findCheck	▼
POST	/glassmatrix/api/v1/tpa/save	▼
POST	/glassmatrix/api/v1/tpa/saveTPAMetric	▼
POST	/glassmatrix/api/v1/tpa/update	▼
POST	/glassmatrix/api/v1/tpa/updateTPAMetric	▼
GET	/glassmatrix/api/v1/tpa/files	▼
GET	/glassmatrix/api/v1/tpa/loadFolders	▼
GET	/glassmatrix/api/v1/tpa/loadFolders/{subdirectory}	▼
GET	/glassmatrix/api/v1/tpa/loadFolders/{subdirectory}/{file}	▼
GET	/glassmatrix/api/v1/tpa/files/{fileName}	▼
DELETE	/glassmatrix/api/v1/tpa/files/{fileName}	▼
DELETE	/glassmatrix/api/v1/tpa/files/tpaFile/{subdirectory}/{fileName}	▼

Ilustración 66. Endpoints usados para el manejo de ficheros

En la Ilustración 66 se pueda observar, los endpoints que se pueden consumir para manejar estos ficheros. No han sido necesarios métodos puts, ya que, al actualizar los ficheros, se realiza un post para volver a crear aquellos archivos que también crean un hash asociado.

La creación de archivos con hashes es necesario sobre todo para aquellos archivos que se crean a partir de un TPA. Al crearse, generan un hash y gracias a este hash podemos indicar en el frontend si es el archivo original o si ha sido modificado.

This metric is **NUMBER_GH_MERGEDPR.json** from **tpa-class01-GH-cs169_fa23-TPA-5**

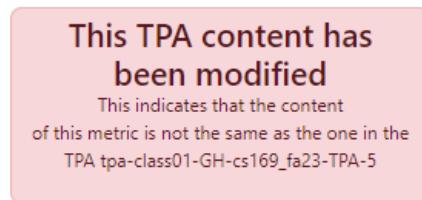


Ilustración 67. Advertencia de archivo modificado.

Para estos mensajes, cada archivo se guarda junto a un hash, y después existe el siguiente endpoint:

```
app.post(apiName + '/calculateSHA', (req, res) => {
  const data = req.body;
  const dataString = JSON.stringify(data);
  const hash = crypto.createHash('sha256');
  hash.update(dataString);
  const hashedContent = hash.digest('hex');
  res.json({ sha256: hashedContent });
});
```

Código 14. Endpoint para el calculo de hashes.

Este endpoint devuelve el hash actual del archivo que se cargue, y si es el mismo se mostrará un mensaje indicando que el archivo no ha sido modificado, y en caso contrario el mensaje que hemos visto anteriormente.

6.4.3 GESTIÓN DE VARIABLES DE ENTORNO

Para que facilitar el despliegue de la aplicación tanto en entornos locales, ya sea de desarrollo o de pruebas o en entornos de producción, hay creado en el proyecto dos formas de configurar estas URL de configuración del entorno. Se puede hacer desde la propia aplicación frontend o directamente desde código.

Desde cualquier editor de código, en la carpeta raíz del proyecto, se encuentra el archivo config.js que contiene lo siguiente:

```
1. const BASE_URL = 'http://localhost';
2. const DEFAULT_COLLECTOR = 'EVENTS';
3. const COLLECTOR_EVENTS_URL = 'http://localhost:5500/api/v2/computations';
4. const AGREEMENTS_URL = 'http://localhost:5400/api/v6/agreements';
5. const SCOPES_URL = 'http://host.docker.internal:5700/api/v1/scopes/development';
6.
7. module.exports = {
8.   BASE_URL,
9.   DEFAULT_COLLECTOR,
10.  COLLECTOR_EVENTS_URL,
11.  AGREEMENTS_URL,
12.  SCOPES_URL
13. };
14.
```

Código 15. Contenido de archivo de variables de entorno (config.js)

Estas variables serán utilizadas tanto desde el back como del front cuando se levante el proyecto. Para ello, cuando se levanta el backend se ejecutará la siguiente función:

```
1. const configData = fs.readFileSync(path.join(__dirname, 'config.js'), 'utf8');
2. let configTs = configData.replace(/const/g, 'export const');
3. configTs = configTs.replace(/module\.exports = {[^}]*};/g, '');
4. fs.writeFileSync(path.join(__dirname, 'lockedConfig.ts'), configTs);
5. let config = {
6.   BASE_URL,
7.   DEFAULT_COLLECTOR,
8.   COLLECTOR_EVENTS_URL,
9.   AGREEMENTS_URL,
10.  SCOPES_URL
11. };
12.
```

Código 16. Función que exporta las variables de entorno a TypeScript

Este fragmento de código realizará las siguientes operaciones:

1. Lee el contenido del archivo `config.js` en la misma ubicación que el script actual (`__dirname`) y lo almacena en la variable `configData`.
2. Reemplaza todas las apariciones de la palabra `const` con `export const` en `configData` y almacena el resultado en `configTs`. Esto convierte las declaraciones de constantes en JavaScript a declaraciones exportables en TypeScript.
3. Elimina la línea que exporta los módulos en `configTs` (la línea que comienza con `module.exports = {...};`).

4. Escribe el contenido de `configTs` en un nuevo archivo llamado `lockedConfig.ts` en la misma ubicación que el script actual.

5. Crea un objeto `config` con las constantes `BASE_URL`, `DEFAULT_COLLECTOR`, `COLLECTOR_EVENTS_URL`, `AGREEMENTS_URL` y `SCOPES_URL`.

En resumen, este código convierte un archivo de configuración JavaScript a un formato exportable de TypeScript y crea un archivo de configuración llamado `lockedConfig.ts` que guardará las variables para que el frontend también las pueda utilizar o cambiarlas.

Desde el frontend también se podrán editar. Entrando en la página de configuración, como posteriormente veremos en la Ilustración 100, podremos acceder a la sección para editar estas variables.

Constants

You should restart the server to apply the changes
×

Base URL	<input type="text" value="http://localhost"/>
Default Collector	<input type="text" value="EVENTS"/>
Collector Events URL	<input type="text" value="http://localhost:5500/api/v2/computations"/>
Agreements URL	<input type="text" value="http://localhost:5400/api/v6/agreements"/>
Scopes URL	<input type="text" value="http://host.docker.internal:5700/api/v1/scopes/development"/>
<input style="background-color: #0072bc; color: white; border: 1px solid #0072bc; padding: 2px 10px; border-radius: 5px; font-weight: bold; width: fit-content;" type="button" value="Update"/>	

Ilustración 68. Cambio de variables de entorno de manera gráfica.

Al usar el botón update, esto llamará al siguiente endpoint que actualizará el archivo config.js, y si reiniciamos TP-Tester, ya estarán cargada la nueva configuración.

```

1. app.put(apiName + '/config', (req, res) => {
2.   const keys = Object.keys(req.body);
3.   let configData = fs.readFileSync(path.join(__dirname, 'config.js'), 'utf8');
4.
5.   keys.forEach((key) => {
6.     if (configData.includes(key)) {
7.       const regex = new RegExp(`\${key} = ).*;`);
8.       configData = configData.replace(regex, `\$1'${req.body[key]}'`);
9.     }
10.   });
11.
12.   fs.writeFileSync(path.join(__dirname, 'config.js'), configData);
13.   res.json({ message: 'Config updated successfully' });
14. });
15.

```

Código 17. Endpoint de actualización de variables de entorno.

6.4.4 EJECUCIÓN DE CASOS DE PRUEBA

La página de testeo de las métricas es sin lugar a duda la más compleja del proyecto. Ya que esta página es la encargada de ejecutar los casos de prueba que pueden ser totalmente configurables por el usuario en función de la métrica que se quiera usar.

Execution zone

Nombre del archivo	Save	Set format

Execute tests

El comienzo de la funcionalidad principal de la página comenzará cuando se pulse el botón de “execute tests”. Esto hará que se ejecute la función executeYaml que se ejecuta a continuación en Código 18-

```

executeYaml(): void { Show usages ± Antonio Saborido *
  this.isLoading = true;
  this.http.post<YamlData>({ url: `${BASE_URL}:6012/api/convertYaml` , body: { yaml: this.yamlContent } }).subscribe( next: data : YamlData =>
    this.response = '';
    data.steps.reduce((prevPromise : Promise<void>, step: Step) => {
      return prevPromise.then(() => {
        return new Promise( executor: resolve => setTimeout(resolve, timeout: 3000))
          .then(() : Promise<...> | Promise<...> => {
            // @ts-ignore
            const handler = this.stepHandlers[step.method][step.usos];
            if (handler) {
              return handler(step).then((response: Response) : void => {
                if (response !== undefined) {
                  const responseString : string = step.usos + ' ' + JSON.stringify(response, replacer: null, space: 2) + '\n\n';
                  this.response += responseString;
                } else {
                  this.response += step.usos + '\n\n';
                }
              }).catch((error: any) : void => {
                console.error(`Error in step ${step.method} ${step.usos}:`, error);
              });
            } else {
              console.error(`No handler found for method ${step.method} and uses ${step.usos}`);
              return Promise.reject( reason: `No handler found for method ${step.method} and uses ${step.usos}` );
            }
          });
        });
      }, Promise.resolve()).then(
        () : void => {
          this.isLoading = false;
        }
      ).catch(error => {
        console.error(error);
        this.errorMessage = `Se produjo un error durante la ejecución: ' + error.message;
        this.isLoading = false;
      });
    });
}

```

Código 18. Función executeYaml

La función executeYaml() realiza las siguientes operaciones:

1. Realiza una solicitud HTTP POST a url/api/convertYaml con el contenido del text area como cuerpo de la solicitud. Este endpoint devolverá el contenido del yaml, pero en formato javascript para poder seguir procesándolo en el frontend.
2. Cuando la respuesta de la solicitud HTTP llega, se procesa como sigue:
 - Se recorren los pasos en la respuesta (asumiendo que la respuesta es un objeto con una propiedad steps que es un array).
 - Para cada paso, se busca un manejador stepHandlers que corresponda al método y uso del paso (se puede ver la estructura de este manejador en el anexo 9.2.7).
 - Si se encuentra un manejador, se llama a este manejador con el paso como argumento. El manejador se supone que devuelve una promesa que se resuelve con una respuesta.

- Si la respuesta no es undefined, se añade a mensaje, como una cadena JSON. Esto será lo que se muestre posteriormente en el textarea de los resultados de la ejecución
3. Si ocurre un error durante el procesamiento de los pasos o durante la solicitud se registra el error en la consola, se establece el mensaje de error, esto mostrará un mensaje de error al usuario y finalizará el spinner de cargado.

Por tanto, **executeYaml** toma un archivo YAML, lo envía a un servidor para que lo convierta en un objeto JavaScript, luego procesa los pasos en el objeto resultante utilizando manejadores específicos para cada combinación de método y uso.

Si está ejecución utiliza alguno de los pasos que tengan method: TEST, esto guardará los resultados de la ejecución una base de datos “**nedb**”. Esto facilitará que posteriormente se pueda comprobar fácilmente si se han encontrado el campo deseado en los resultados anteriores mediante uno de los dos siguientes endpoint:

```

1. app.get(apiName+ '/getData/:field', (req, res) => {
2.   const field = req.params.field;
3.
4.   db.find({}, function (err, docs) {
5.     if (err) {
6.       res.status(500).send(err);
7.     } else {
8.       let fieldDocs = [];
9.
10.      docs.forEach(doc => {
11.        if (doc.computations && Array.isArray(doc.computations)) {
12.          doc.computations.forEach(computation => {
13.            if (computation.evidences && Array.isArray(computation.evidences)) {
14.              computation.evidences.forEach(evidence => {
15.                if (evidence[field] !== undefined) {
16.                  fieldDocs.push({
17.                    value: computation.value,
18.                    [field]: evidence[field],
19.                    createdAt: evidence.createdAt,
20.                    authorLogin: evidence.author.login
21.                  });
22.                }
23.              });
24.            }
25.          });
26.        }
27.      });
28.
29.      if (fieldDocs.length > 0) {
30.        res.status(200).send(fieldDocs);
31.      } else {
32.        let response = {};
33.        response[field] = "not found";
34.        res.status(200).send([response]);
35.      }
36.    }
37.  });
38. });
39.

```

La función **getData**, realizará los siguientes pasos:

1. Extrae el parámetro **field** de la ruta de la solicitud HTTP. Este parámetro se espera que sea el nombre de un campo en los documentos de la base de datos.

2. Realiza una búsqueda en la base de datos utilizando **db.find()**. La búsqueda no tiene ningún filtro, por lo que devuelve todos los registros en la base de datos.
3. Si la búsqueda es exitosa, recorre cada documento devuelto. Para cada documento, recorre cada objeto en el array computations y luego cada objeto en el array evidences dentro de cada objeto computation.
4. Si el objeto **evidence** tiene una propiedad con el nombre especificado por **field**, crea un nuevo objeto con las propiedades value, createdAt, authorLogin y field y lo agrega a un array fieldDocs.
 - a. Si el array fieldDocs tiene al menos un elemento, envía el array como respuesta en formato JSON.
 - b. Si el array fieldDocs está vacío, crea un objeto con una propiedad con el nombre especificado por field y un valor de "not found", y envía este objeto como respuesta en formato JSON.

En resumen, esta función busca en la base de datos documentos que tienen una propiedad con un nombre específico en el array evidences dentro del array computations y devuelve estos documentos.

```

1. app.post(apiName + '/bluejay/findCheck', (req, res) => {
2.   const { values } = req.body;
3.
4.   // Miro las evidencias que tengo que buscar
5.   const evidences = values.flatMap(value => value.evidences);
6.
7.   // Construyo una query para llamar al db.find
8.   const query = {
9.     'computations.value': { $in: values.map(value => value.value) },
10.    $or: evidences.map(evidence => {
11.      const evidenceQuery = {};
12.      for (const key in evidence) {
13.        if (key === 'login') {
14.          evidenceQuery[`computations.evidences.author.${key}`] = evidence[key];
15.        } else if (key === 'bodyText') {
16.          evidenceQuery[`computations.evidences.comments.nodes.${key}`] =
evidence[key];
17.        } else {
18.          evidenceQuery[`computations.evidences.${key}`] = evidence[key];
19.        }
20.      }
21.      return evidenceQuery;
22.    })
23.  };
24.
25.  db.find(query, (err, docs) => {
26.    if (err) {
27.      console.error(err);
28.      res.status(500).json({ message: 'An error occurred while querying the database.' });
29.    } else {
30.      res.json(docs);
31.    }
32.  });
33. });
34.

```

Findcheck extraerá el campo values del cuerpo de la solicitud. Se espera que values sea un array de objetos, donde cada objeto tiene una propiedad evidences. Por ello, se creará otro array evidences que es una combinación de todos los arrays evidences en los objetos en values.

Y cuando devuelvan estos datos al front, el manejador de las opciones (stepHandlers) será el encargado de pintar en un cuadro de texto el resultado de estas búsquedas, siendo exitosa en caso de que findCheck o getData devuelvan los datos esperados o errónea en caso contrario.

6.4.5 EJECUCIÓN EN MODO APLICACIÓN

Para permitir que la aplicación se ejecute en modo ventana, se ha utilizado Electron. Es una biblioteca que combina Chromium y Node.js, lo que permite desarrollar aplicaciones de escritorio con tecnologías web como JavaScript, HTML y CSS. Esta combinación permite que las aplicaciones web se comporten como aplicaciones nativas de escritorio, brindando una experiencia de usuario enriquecida y cohesiva.

Al combinar el motor de renderizado de Chromium (el mismo que usa Google Chrome) con la potencia de Node.js, Electron permite ejecutar código JavaScript tanto en el lado del cliente como en el lado del servidor en una sola aplicación.

- **Chromium:** Es un proyecto de navegador web de código abierto que forma la base de varios navegadores web, incluyendo Google Chrome. En el contexto de Electron, Chromium se utiliza para renderizar el contenido de la aplicación, permitiendo el uso de HTML, CSS y JavaScript.
- **Node.js:** Es un entorno de ejecución de JavaScript del lado del servidor que permite utilizar JavaScript para escribir aplicaciones del lado del servidor con acceso a funcionalidades del sistema operativo, como el sistema de archivos.

Por tanto, Electron encaja perfectamente con el proyecto permitiendo levantar con node el backend necesario, y gracias al navegador Chromium, permite que el frontend Angular funcione correctamente.

Ventajas de utilizar Electron

- **Consistencia en la experiencia de usuario:** Las aplicaciones Electron se ejecutan en su propio contenedor, lo que garantiza que la aplicación se vea y funcione de la misma manera en todos los sistemas operativos soportados (Windows, macOS, Linux). No hay dependencia de las versiones de los navegadores instalados en el sistema del usuario.
- **Acceso a funcionalidades del sistema:** A través de Node.js, las aplicaciones Electron pueden acceder a funcionalidades del sistema operativo que no están disponibles en el navegador. Esto incluye acceso al sistema de archivos, notificaciones del sistema, manejo de ventanas, y más.
- **Desarrollo con tecnologías web:** Los desarrolladores pueden utilizar tecnologías web familiares para construir aplicaciones de escritorio. Esto reduce la necesidad de aprender nuevos lenguajes o frameworks para crear aplicaciones de escritorio, permitiendo así el uso de frameworks como Angular, que es el caso del proyecto.
- **No se necesita un navegador externo:** no dependen de un navegador web externo para ejecutarse. Todo lo necesario para ejecutar la aplicación está empaquetado junto con la aplicación, proporcionando una experiencia autónoma y sin necesidad de configuraciones adicionales por parte del usuario final.

Implementación de Electron en el proyecto

Para la integración de electron en el proyecto en primer lugar, ha sido necesaria la instalación de la biblioteca mediante npm (con el comando: `npm install electron --save-dev`). Una vez que tenemos la biblioteca correctamente instalada, es necesario la creación de un archivo llamado “`main.js`” en la carpeta raíz del proyecto.

Este archivo, tendrá que contener la función `createMainWindow()`, la cual indicará las opciones para la creación de la aplicación. En este caso la mía es:

```
function createMainWindow() {  
  
    mainWindow = new BrowserWindow({  
        width: 1280,  
        height: 720,  
        title: 'Glassmatrix',  
        resizable: true,  
        webPreferences: {  
            contextIsolation: false,  
            nodeIntegration: true,  
        },  
        icon: path.join(__dirname, 'src/favicon.ico'),  
    });  
  
    mainWindow.loadURL(`file://${__dirname}/dist/index.html`);  
  
    // Consola abierta  
    //mainWindow.webContents.openDevTools();  
  
    mainWindow.on('closed', () => {  
        mainWindow = null;  
    });  
}
```

Código 19. Contenido de `main.js` de electron

En este caso, estoy indicando que las dimensiones por defecto de la aplicación serán de 1280 por 720 pixeles, pero al establecer “`resizable`” como verdadero, estoy indicando que el usuario podrá modificar estas dimensiones a su gusto. También tengo comentado que una función que abre la aplicación Electron con la consola de desarrollo, por si estamos utilizando Electron para desarrollar la aplicación directamente.

Además de esto, en el archivo `package.json`, también he añadido el propio script para poder ejecutar la aplicación en modo Electron en desarrollo de una forma sencilla.

```
"scripts": {  
  ...  
  "electron": "concurrently \"ng build --base-href ./ && electron .\" \"node server.js\"",  
},
```

Código 20. Script de ejecución electron

Este script levanta de forma conjunta el front y el back dentro de Electron para que no sea necesario ningún otro comando.

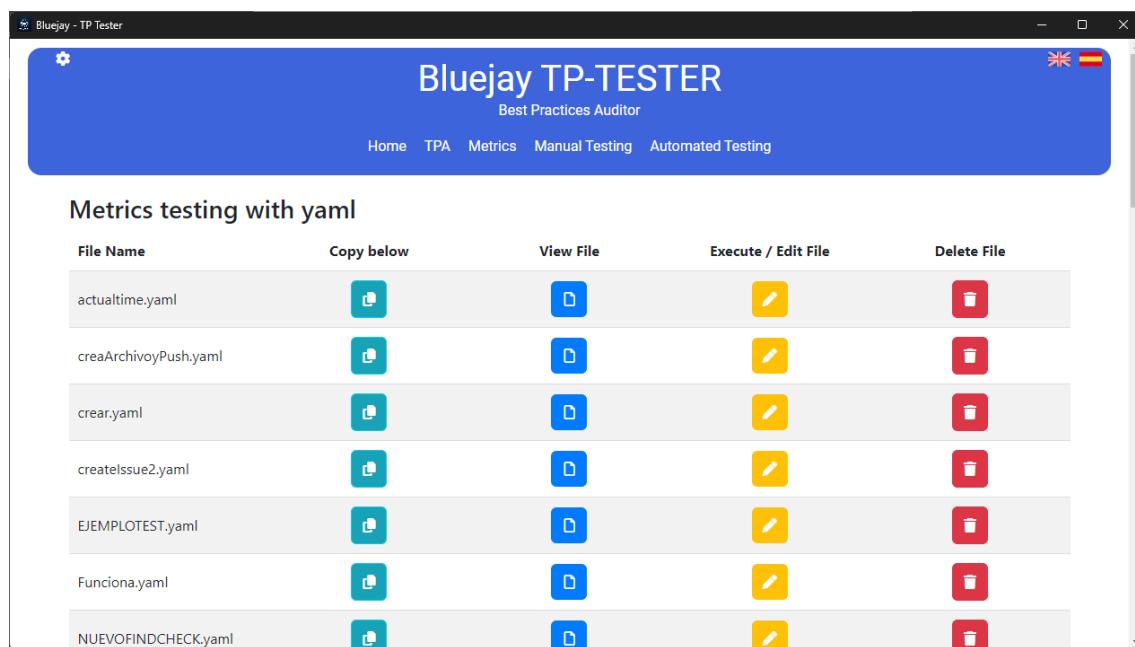


Ilustración 69. Aplicación ejecutada con electron

Como se puede observar en la Ilustración 69, se ejecuta un chromium, con el icono que se seleccione y dentro de ella se muestra la página angular con el backend node funcionando correctamente.

Aunque no está implementado en el proyecto, Electron también permite el empaquetado de la aplicación, haciendo que con el uso de un instalador que creará un ejecutable de arranque no sea necesario tener nada más instalado en nuestro equipo.

Esta implementación asegura que la aplicación pueda ser distribuida como una aplicación de escritorio independiente, proporcionando una experiencia de usuario consistente y aprovechando las capacidades avanzadas del sistema operativo.

6.4.6 BIBLIOTECAS UTILIZADAS

Backend

- axios (1.6.8): Utilizado para hacer solicitudes HTTP desde el backend.
- body-parser (1.20.2): Middleware para analizar cuerpos de solicitudes entrantes en JSON y otros formatos.
- cors (2.8.5): Middleware para habilitar el intercambio de recursos entre diferentes dominios en la API.
- express (4.18.3): Framework para construir aplicaciones web y APIs en Node.js.
- swagger-jsdoc (6.2.8), swagger-ui-express (5.0.0): Utilizados para documentar y visualizar la API con Swagger.
- dockerode (4.0.2): Cliente para interactuar con el demonio Docker, utilizado para la gestión de contenedores.
- js-yaml (4.1.0), json-to-pretty-yaml (1.2.2): Para trabajar con archivos YAML en el backend.
- nedb (1.8.0): Base de datos embebida para aplicaciones Node.js de pequeña escala.
- simple-git (3.23.0): Cliente para interactuar con Git de forma programática.

Frontend

- angular/animations (~13.3.0): Para agregar animaciones y efectos visuales a la interfaz de usuario.
- angular/cdk (^13.3.9): Proporciona componentes de diseño para crear interfaces interactivas y accesibles.
- angular/common (~13.3.0), angular/core (~13.3.0), angular/forms (~13.3.0), angular/platform-browser (~13.3.0), angular/router (~13.3.0): Librerías esenciales para el desarrollo de aplicaciones Angular.
- angular/material (^13.3.9): Implementa Material Design en Angular, ofreciendo una amplia gama de componentes y estilos predefinidos.
- fortawesome/fontawesome-free (^6.5.1): Proporciona una colección de iconos vectoriales para mejorar la experiencia del usuario.
- bootstrap (^5.3.3): Framework de diseño frontend que facilita la creación de interfaces de usuario responsivas y atractivas.
- zone.js (~0.11.4): Biblioteca que implementa la zona de cambio Angular, interceptando y rastreando las operaciones asíncronas en JavaScript.
- ngx-translate/core (^14.0.0), ngx-translate/http-loader (^7.0.0): Utilizadas para manejar las traducciones en Angular y cargar archivos de traducción desde una ubicación externa. [20]

CAPITULO 7: MANUALES

7.1 MANUAL DE DESPLIEGUE

Los requisitos para poder levantar este proyecto localmente son:

- Node.js v16.10.0
- Angular CLI v13.3.11
- Docker (opcional)

Los scripts disponibles para la puesta en marcha son los siguientes:

```
"scripts": {  
  "ng": "ng",  
  "start": "concurrently \"ng serve\" \"node server.js\"",  
  "build": "ng build",  
  "watch": "ng build --watch --configuration development",  
  "test": "ng test",  
  "docker": "docker-compose up --build",  
  "server": "concurrently \"ng serve --host 0.0.0.0 --disable-host-check\" \"node server.js\"",  
  "electron": "concurrently \"ng build --base-href ./ && electron .\" \"node server.js\""  
},
```

Código 21. Scripts disponibles en la aplicación

Se podrá usar cualquiera de los scripts simplemente con “npm run nombre_script.”

Aunque el proyecto TP Tester se puede ejecutar con cualquiera de las opciones que se detallan a continuación, es importante mencionar que, al ser una extensión de Bluejay, para el correcto funcionamiento de todas las herramientas será necesario tener levantadas el resto de microservicios propios de Bluejay.

Despliegue de Bluejay

Para desplegar Bluejay, se puede hacer siguiendo la documentación de Bluejay (<https://docs.bluejay.governify.io/>) o siguiendo los pasos que se indican a continuación. En primer lugar, hay que clonar el repositorio de infraestructura, que contiene los archivos de Docker y configuración necesarios:

```
git clone https://github.com/governify/bluejay-infrastructure.git
```

Dentro de la carpeta bluejay-infrastructure, crea un archivo .env con el siguiente contenido:

```
GOV_INFRASTRUCTURE=http://host.docker.internal:5200/api/v1/public/in  
frastructure-local.yaml  
NODE_ENV=development  
  
# Repository branch that will be cloned into assetsmanager
```

```

ASSETS_REPOSITORY_BRANCH=develop

# Influx database url
INFLUX_URL=http://host.docker.internal:5002

# EVENT COLLECTOR
KEY_GITHUB=<token>

# FRONTENDS ACCESS ACCOUNT
USER_RENDER=bluejay
PASS_RENDER=bluejay
USER_ASSETS=bluejay
PASS_ASSETS=bluejay

# ASSETS MANAGER
KEY_ASSETS_MANAGER_PRIVATE=bluejay-assets-private-key

# SCOPE MANAGER
KEY_SCOPE_MANAGER=bluejay-scopes-private-key

# COMPOSE CONFIG
COMPOSE_HTTP_TIMEOUT=200

```

Código 22. Contenido de archivo .env para levantar microservicios de bluejay.

Una vez clonado el repositorio y creado el archivo .env, podremos levantar la infraestructura de bluejay mediante el siguiente comando:

```
docker-compose -f docker-bluejay/docker-compose-local.yaml -env-file
.env up -d
```

Una vez tenemos listo la infraestructura bluejay levantada, podremos levantar correctamente TP-Tester.

7.1.1 MODO DESARROLLO

Para levantar Bluejay-TP Tester en modo de desarrollo, sigue estos pasos:

1. Clona el repositorio de Bluejay-TP Tester.

```
git clone https://github.com/Antoniosc7/tp-tester
```

2. Navega al directorio del proyecto:

```
cd tp-tester
```

3. Instala las dependencias con npm install.

```
npm install
```

4. Levanta el proyecto con angular y express.

```
npm start
```

Esto sería suficiente ya que el proyecto usa concurrently y levanta tanto el servidor express como angular al mismo tiempo. El servidor Express (API de GlassMatrix) se levanta en el puerto 6012 y la aplicación Angular en el puerto 4200.

7.1.2 MODO PRODUCCIÓN (DOCKER)

Para levantar Bluejay-TP Tester con docker, sigue estos pasos:

1. Clona el repositorio de Bluejay-TP Tester.

```
git clone https://github.com/Antoniosc7/tp-tester
```

2. Navega al directorio del proyecto:

```
cd tp-tester
```

3. Instala las dependencias con npm install.

```
npm install
```

4. Levanta el proyecto docker.

```
npm run docker
```

Con esto, tendríamos el proyecto levantado en el puerto 6011 la web angular, y el servidor express en el puerto 6012.

7.1.3 MODO APLICACIÓN (ELECTRON)

Electron es una librería que está basada en Chromium y Node.js y esto permite el desarrollo de aplicaciones de escritorio [21]. Por tanto, para levantarla como una aplicación de escritorio será necesario únicamente seguir los siguientes pasos:

1. Clona el repositorio de Bluejay-TP Tester.

```
git clone https://github.com/Antoniosc7/tp-tester
```

2. Navega al directorio del proyecto:

```
cd tp-tester
```

3. Instala las dependencias con npm install.

```
npm install
```

4. Levanta el proyecto con electron.

```
npm run electron
```

7.2 MANUAL DE USO

7.2.1 PÁGINA DE INICIO

Cuando se inicie TP-Tester, la página inicial será “Home”. En ella encontraremos de forma organizada por secciones el contenido equivalente al Readme del proyecto, indicando el funcionamiento de cada página.

The screenshot shows the Bluejay TP-TESTER homepage. The top navigation bar includes links for Home, TPA, Metrics, Manual Testing, and Automated Testing, along with language selection buttons for English and Spanish. The main content area is titled "Metrics" and contains the following sections:

- The Metrics Loader page**: A brief description stating it is designed to manage and test metrics, providing a user-friendly interface for viewing, creating, and managing metrics.
- Existing metrics**: A table listing saved metrics, categorized into two large tables: one from the TPA section and another for individual metrics not related to any specific TPA.
- Create new metric**: A section for inputting new metric content via a text area, with buttons for "Post" (to API) and "Get Computation" (to API).
- A code editor window displays JSON configuration code:

```
{  
  "config": {  
    "scopeManager": "http://host.docker.internal:5700/api/v1/scopes/development"  
  },  
  "metric": {  
    "computing": "actual",  
    "element": "number",  
    "event": {  
      "githubGQL": {  
        "custom": {  
          "type": "graphQL",  
          "title": "Get pull requests with at least one comment by member",  
          "steps": {  
            "0": {  
              "label": "Analyze pull requests"  
            }  
          }  
        }  
      }  
    }  
  }  
}
```

Ilustración 70. Página incial TP-Tester

En la parte izquierda, se podrá navegar a todas las secciones disponibles de la documentación. Y en la parte superior derecha, podremos seleccionar el idioma en el que se quiere visualizar la página haciendo “click” en la bandera del país. Actualmente solo está disponible la española o la inglesa.

7.2.2 PÁGINA TESTEAR MÉTRICAS

La página de Carga de Métricas está diseñada para gestionar y probar las métricas. Proporciona una interfaz de usuario amigable para visualizar, crear y gestionar métricas.

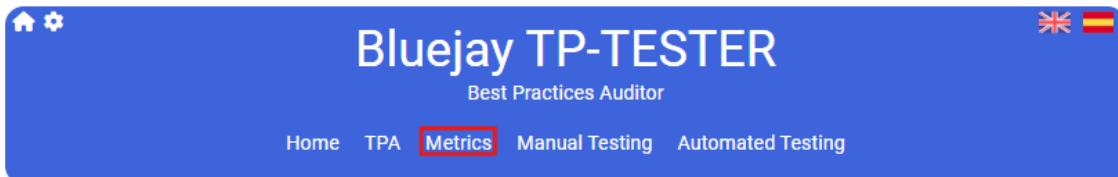


Ilustración 71. Navegación a la pantalla de métricas

El acceso a esta página será posible desde cualquier página de la web, haciendo click en “Metrics”.

7.2.2.1 ESTRUCTURA DE LA PÁGINA

Aquí se pueden ver todas las métricas guardadas, clasificadas en dos grandes tablas. Una tabla consta de aquellas guardadas desde la sección de TPA, que estarán relacionadas con el TPA al que pertenecían. La otra tabla contiene métricas individuales guardadas, que no están relacionadas con ningún TPA específico.

Saved metrics			
TPA Metrics			
File Name	View File	Execute / Edit File	Delete File
tpa-class01-GH-cs169_fa23-TPA-5 NUMBER_GH_MERGEDPR.json			
Individual Metrics			
File Name	View File	Execute / Edit File	Delete File
additions_metric.json			
METRICA_ADDITIONS.json			
METRICA_ADDITIONS3.json			
NPRAC.json			
tp10.json			
tp11.json			

Ilustración 72. Visualización de métricas guardadas

Cada fila de la tabla representa una única métrica, mostrando su nombre y proporcionando botones de acción:

- El botón Ver navega a una vista detallada de la métrica.
- El botón Ejecutar/Editar navega a una página donde puedes ejecutar o editar la métrica.
- El botón Eliminar elimina la métrica.

Crear nueva métrica

Si navegamos al inferior de la página, tras ver todas las métricas que se encuentran guardadas podemos encontrar una nueva sección que permite crear una nueva métrica. Para ello proporciona un área de texto donde puedes introducir el contenido de la nueva métrica.

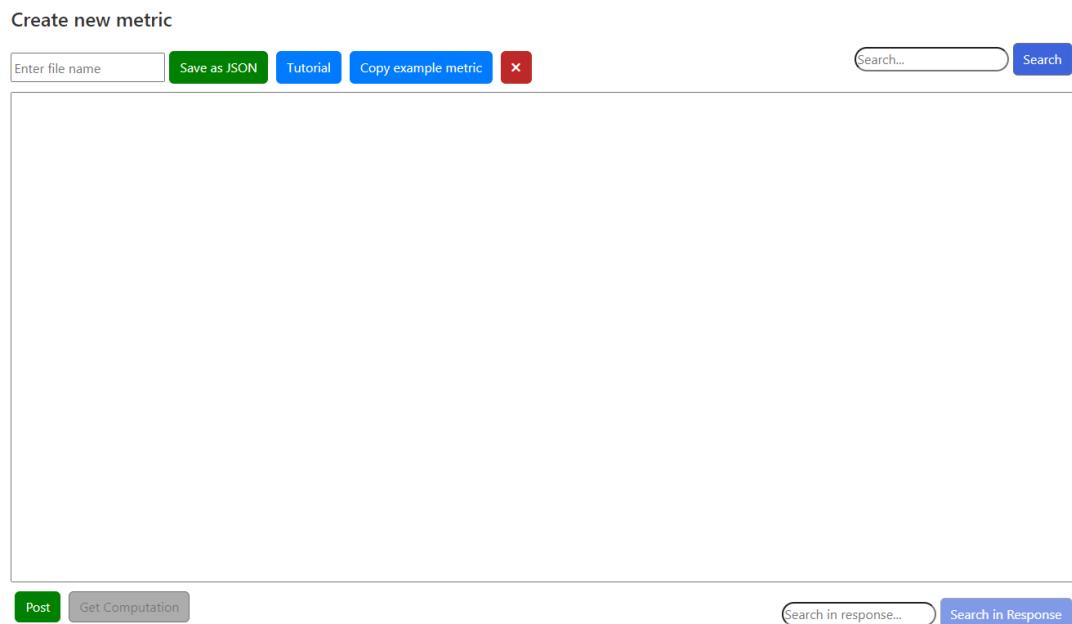


Ilustración 73. Sección para crear una nueva métrica

En la parte superior de la zona de texto podemos encontrar varios botones:

- “Save as JSON” : Este botón permitirá guardar el contenido actual que haya en el textarea en un json con el nombre que indiquemos. Una vez guardada esta métrica se podrá ver en la sección anterior.
- Tutorial: Ese botón abrirá un pop-up con un tutorial y varios ejemplos de cómo tiene que ser el formato de la métrica que se quiere probar.

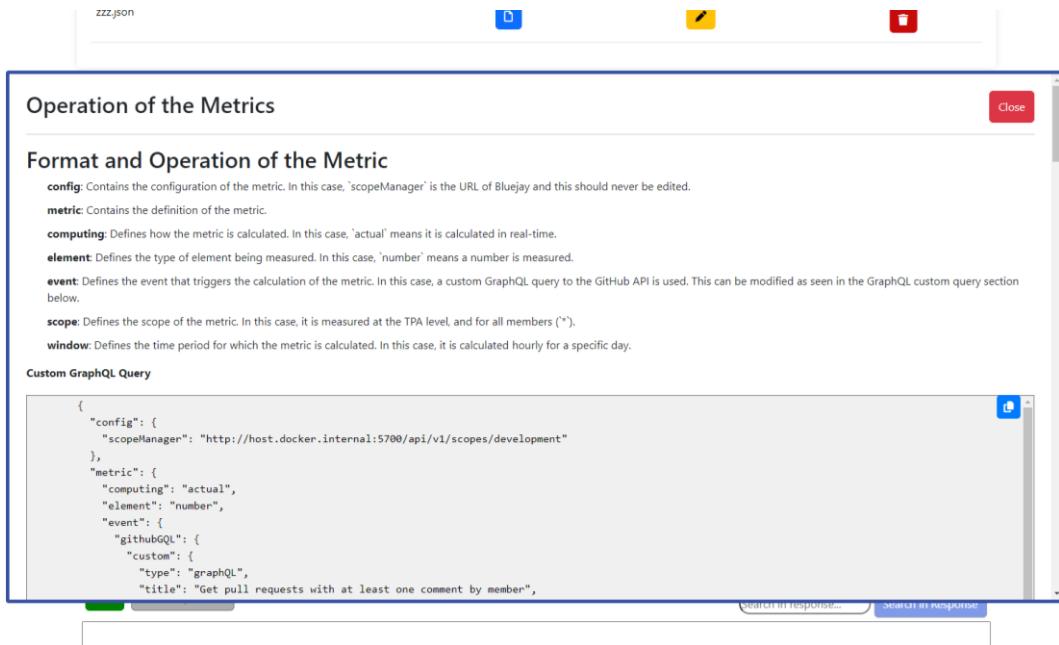


Ilustración 74. Pop Up tutorial de cómo crear una métrica

- “Copy example metric”: Ese botón como dice su traducción, lo que hará será copiar en el cuadro de texto una métrica de ejemplo para que se pueda comprobar cómo funciona.

Depués podemos encontrar otros dos botones en la parte inferior del textarea:

- El botón “post” publica el contenido que has introducido en el área de texto en la API de Bluejay para su cálculo.
- El botón “Get computation” recupera los resultados del cálculo de la API de Bluejay. Para ejecutar este botón deberemos haber pulsado antes el botón “Post” ya que hasta este momento el botón se encontrará bloqueado.

Create new metric

Save as JSON Tutorial Copy example metric X
Search... Search

```

MERGED: && Math.max(new Date(variables.from), new Date(pullRequest.createdAt)) <= Math.min(new Date(variables.to), new Date(pullRequest.mergedAt)) \n\n
for (let comment of pullRequest.comments.nodes) \n\n
    if (comment.createdAt > new Date(variables.from) && new Date(comment.createdAt) < new Date(variables.to)) \n\n
        pullRequestsMergedAndOpenWithAtLeastOneComment.push(pullRequest) \n\n
    if (isPullRequestAlreadyAdded == true)\n\n
        break\n\n
    } \n\n
} \n\n
if (isPullRequestAlreadyAdded) continue\n\n
for (let review of pullRequest.reviews.nodes) \n\n
    if (review.bodyText.length > 0 && review.author.login == `%MEMBER.github.username%` && new Date(review.createdAt) > new Date(variables.from) && new Date(review.createdAt) < new Date(variables.to)) \n\n
        pullRequestsMergedAndOpenWithAtLeastOneComment.push(pullRequest)\n\n
        break\n\n
    } \n\n
} \n\n
return pullRequestsMergedAndOpenWithAtLeastOneComment\n\n
}
}
}
}

"scope": [
    "project": "TFG-GH-antoniosc7_Glassmatrix",
    "class": "TFG",
    "member": "**"
],
"window": {
    "type": "static",
    "period": "hourly",
    "initial": "2024-06-04T23:00:00.000Z",
    "from": "2024-06-04T23:00:00.000Z",
    "end": "2024-06-04T23:59:00.000Z",
    "timeZone": "America/Los_Angeles"
}
}
}

```

Post Get Computation
Search in response... Search in Response

```

{
  "code": 200,
  "message": "OK",
  "computations": [
    {
      "scope": [
        "project": "TFG-GH-antoniosc7_Glassmatrix",
        "class": "TFG",
        "member": "Antonio"
      ],
      "period": [
        "from": "2024-06-04T23:00:00.000Z",
        "to": "2024-06-04T23:59:00.000Z"
      ],
      "evidences": [],
      "value": 0
    }
  ]
}

```

Ilustración 75. Prueba de métrica y su resultado

En resumen, en el cuadro de texto superior se encontrará la métrica que se quiere probar, por lo que ese cuadro de texto será editable y en el inferior, que no será editable, se mostrarán los resultados de la computación.

7.2.2.2 Visualización de métricas guardadas

A la página de visualización de métricas se accede mediante el botón de “View file” y esta página está compuesta un cuadro de texto no editable cuyo contenido es una métrica que ha sido previamente guardada.

File Name	View File	Execute / Edit File	Delete File
additions_metric.json	D	E	X

Ilustración 76. Acceso a página de visualización de métricas

7.2.2.3 Ejecutor de Métricas

La pantalla será algo diferente si accedemos a la página de edición o ejecución de una métrica individual o una métrica que pertenece a un TPA.

Métricas individuales

La página del Ejecutor de Métricas te permite ejecutar una métrica específica. Proporciona una interfaz de usuario amigable para ver y modificar los detalles de la métrica, y para ejecutar la métrica. Para acceder a esta página, será desde la misma tabla

en la que se accede a la visualización de métricas guardadas, pero en este caso será usando el botón de “Execute / Edit File”. Una vez entremos encontraremos una página así:

```
{
  "config": {
    "scopeManager": "http://host.docker.internal:5700/api/v1/scopes/development"
  },
  "metric": {
    "computing": "actual",
    "element": "number",
    "event": {
      "githubGQL": {
        "custom": {
          ...
        }
      }
    }
  }
}
```

Ilustración 77. Página de ejecución o de edición de métricas.

En la parte superior podremos encontrar el nombre del archivo guardado que estamos editando, y tras ello tenemos dos grandes secciones. La sección de contexto de la métrica y la sección de ejecución.

Sección de contexto.

Esta sección muestra los detalles de la métrica. Proporciona varios campos de entrada donde puedes modificar los detalles de la métrica:

- El campo **Proyecto** te permite especificar el proyecto asociado a la métrica.
- El campo **Clase** te permite especificar la clase asociada a la métrica.
- El campo **Miembro** te permite especificar un miembro específico para esta métrica o todos los miembros.

Búsqueda en la computación.

Esta sección te permite especificar los parámetros para el cálculo de la métrica. Proporciona varios campos de entrada:

- El campo **Tipo** te permite especificar el tipo de cálculo.

- El campo **Periodo** te permite especificar el periodo para el cálculo.
- El campo **Inicial** te permite especificar la fecha inicial para el cálculo.
- El campo **Desde** te permite especificar la fecha de inicio para el cálculo.
- El campo **Hasta** te permite especificar la fecha de finalización para el cálculo.
- El campo **Zona horaria** te permite especificar la zona horaria para el cálculo.

También hay varios botones:

- El botón Establecer a la hora actual establece los parámetros de cálculo a la hora actual.
- El botón Guardar como JSON guarda la métrica y sus detalles como un archivo JSON, en este caso no hay que especificar el nombre del archivo ya que se guardará en el mismo que se está editando.
- El botón Tutorial abre un pop up igual que el que podíamos encontrar en la página principal de las métricas en el que te explica cómo funcionan las métricas y su formato.

Sección de Ejecución

Esta sección te permite ejecutar la métrica. Proporciona un área de texto donde puedes ver la representación JSON de la métrica. El funcionamiento de esta sección es el mismo que el de la página principal de las métricas.

También hay varios botones:

- El botón “Post” envía la métrica a la API de Bluejay para su cálculo.
- El botón “Get Computation” recupera los resultados del cálculo de la API de Bluejay.

Métricas de TPA

La página de edición y ejecución en las métricas de TPA tiene únicamente una diferencia.

[Back](#)

This metric is **NUMBER_GH_MERGEDPR.json** from **tpa-class01-GH-cs169_fa23-TPA-5**

This TPA content has been modified

This indicates that the content of this metric is not the same as the one in the TPA tpa-class01-GH-cs169_fa23-TPA-5

Context Section

Scope Information

Project class01-GH-cs169_fa23-TPA-5	Class class01	Member *
--	------------------	-------------

Computation Search

Type static	Period hourly	Initial 2024-04-01T00:00:00.000Z
From 2024-04-01T00:00:00.000Z	End 2024-04-01T23:59:59.999Z	Timezone America/Los_Angeles

[Actual hour computation](#) [Save as json](#) [Tutorial](#)

Execution Section

[Search](#)

Ilustración 78. Diferencia página de métrica de TPA con métricas individuales.

La diferencia es que si la métrica no es igual a la que está cargada originalmente en el TPA al que pertenece, habrá un mensaje indicándolo, aparte que junto al nombre de la métrica también indica al TPA que pertenece.

This metric is **NUMBER_GH_OPENPR.json** from **tpa-class01-GH-cs169_fa23-TPA-5**

Same content as TPA

This indicates that the content of this metric is the same as the one in the TPA tpa-class01-GH-cs169_fa23-TPA-5

Ilustración 79. Mensaje de métrica no editada

En caso de que la métrica no haya sido editada, también habrá un mensaje indicándolo.

7.2.3 TESTEO MANUAL

La página de Pruebas Manuales proporciona una interfaz amigable para probar la funcionalidad de la aplicación interactuando con los repositorios a los que tenga acceso el token que se introduzca en la página de la configuración.

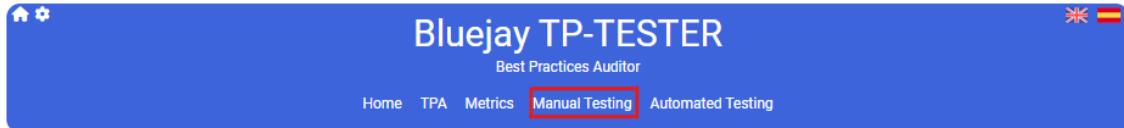


Ilustración 80. Acceso a página de testeo manual

Para acceder a esta página, se hará desde el menú principal, haciendo click en “Manual Testing”.

Al entrar en esta página, tendremos dos opciones principales: clonar un nuevo repositorio o trabajar con un repositorio ya clonado previamente.

A screenshot of the "Manual Testing" page. At the top, there is a note: "To clone a new repository you will need a valid token with access to that repository. For more information, visit: [Github token documentation](#). The token must be introduced in the clone new repository page." Below this, there are two main sections: "Clone new repository" (described as a place to see all repositories with access and clone them to test metrics) and "Cloned repositories" (described as all previously cloned repositories). Each section has a blue button labeled "Clone new repository" and "Cloned repositories" respectively. The footer of the page includes the text "© Antonio Saborido - 2024" and a small circular logo.

Ilustración 81. Pantalla de inicio al entrar en el testing manual

7.2.3.1 CLONAR NUEVOS REPOSITORIOS

En esta página se podrán ver todos los repositorios a los que tenga acceso el token que se haya introducido en la pantalla de configuración. Si aún no se ha introducido ningún token, en esta misma página se podrá introducir un token o actualizar el que esté introducido.

The screenshot shows a user interface for managing GitHub tokens. At the top right is a "Back" button. Below it is a message "Token found" with the token value "Token: ghp_SJtsVlf1yieqvCsEw". A blue "Edit Token" button is present. The main area displays a table of repositories:

Repository Name	Number of Branches	Last Update	View	Edit
Antoniiosc7.github.io	1	2024-01-15T23:03:47Z		
SnakeGamePD	1	2023-09-18T11:35:27Z		
JerseyDetection	2	2024-01-10T23:53:57Z		
TFG-Grafana	1	2023-09-30T18:51:24Z		
Antoniiosc7	1	2024-01-10T23:20:42Z		
JumpMarius	5	2024-01-15T18:20:03Z		
TFG-Angular	1	2023-08-29T02:14:12Z		
pintagraficas	1	2024-06-05T21:52:52Z		
ISSIProject	1	2022-11-25T12:57:33Z		
tp-testbench	30	2024-06-04T23:14:49Z		
TwitchClips	3	2022-05-24T13:41:50Z		
tp-tester	4	2024-06-02T16:26:59Z		

Ilustración 82. Visualización de repositorios disponibles con el token

Cada repositorio tendrá dos botones: el botón "View", que te redirigirá a la página de GitHub del repositorio, y el botón "Edit", que te llevará a la página de clonación.

La página de Clonación te permite clonar un repositorio específico. Muestra los detalles del repositorio, incluyendo su nombre, descripción, propietario, tiempo de creación, tiempo de actualización, lenguaje, visibilidad, y una lista de todas las ramas. También proporciona un botón para clonar el repositorio.

The screenshot shows a repository cloning page for the user "Antoniiosc7". The repository is described as "My Personal Repository". Key details include:

- Owner: Antoniiosc7
- Created at: 2024-01-10T23:20:42Z
- Updated at: 2024-01-10T23:20:42Z
- Language:
- Visibility: public

The "Branches" section lists "main". A prominent blue "Clone Repository" button is at the bottom.

Ilustración 83. Pantalla de descarga de repositorio

El clonado consistirá en descargar el repositorio para poder realizar acciones sobre él en el Testing de la pantalla de repositorios clonados.

7.2.3.2 REPOSITORIOS CLONADOS

Esta sección te permite trabajar con algunos de los repositorios clonados previamente. Proporciona una breve descripción del proceso y un botón para comenzar a trabajar con el repositorio clonado.

Repository name	Branches	Pull Requests	Actions	Delete
JerseyDetection				
TFG-Angular				
TFG-Grafana				
TFG-React				
TFG-Server				
tp-testbench				

Ilustración 84. Página de acciones sobre repositorios clonados

Página de Ramas

A esta pantalla se accederá haciendo click en el botón de “Branches” en el listado de los repositorios clonados. La página te permite gestionar las ramas de un repositorio. Ofrece opciones para ver las ramas, crear una nueva rama, eliminar una rama, y cambiar la rama actual.

TFG-Angular

[View in GitHub](#)

Branches Repository

main
* testbranch

New branch name

Create Branch

Delete Branch

Delete Branch

Change Branch

Change Branch

Ilustración 85. Página de ramas en el testing manual.

Página de Pull Requests

A esta pantalla se accederá haciendo click en el botón de “Branches” en el listado de los repositorios clonados. La página de Pull Requests proporciona una interfaz amigable para gestionar las pull requests de un repositorio. Ofrece opciones para crear una nueva pull request, ver las pull requests abiertas, y fusionar una pull request.

The screenshot shows the TFG-Server interface with three main sections:

- Create Pull Request**: A form with fields for Title, Head, Base, and Body, each with a text input area. A green "Create Pull Request" button is at the bottom.
- Open Pull Requests**: A section stating "There is not any pull request open."
- Merge Pull Request**: A form with fields for Number and Commit Message, each with a text input area. A green "Merge Pull Request" button is at the bottom.

Ilustración 86. Acciones en la página de pull requests.

Página de Acciones

A esta pantalla se accederá haciendo click en el botón de “Actions” en el listado de los repositorios clonados. La página de Acciones es una interfaz para gestionar un repositorio. Ofrece opciones para ver las ramas disponibles, cambiar la rama actual, crear un nuevo archivo, crear un nuevo commit, empujar cambios, ver archivos en el repositorio, y gestionar issues.

TFG-Angular | Actions

[View in GitHub](#)

Available Branches

main
* testbranch

Current Branch:

main [Change Branch](#)

Create New File

New file name

New file content

[Create File](#)

Create New Commit

Commit message

[Create Commit](#)

Push

[Push](#)

The screenshot shows a GitHub repository interface for a project named 'TFG-Angular'. At the top right is a 'Back' button. Below it is a GitHub icon and a link to 'View in GitHub'. The main area is titled 'Actions' and contains several sections: 'Available Branches' (listing 'main' and '* testbranch'), 'Current Branch:' (set to 'main' with a 'Change Branch' button), 'Create New File' (with fields for 'New file name' and 'New file content', and a 'Create File' button), 'Create New Commit' (with a 'Commit message' field and a 'Create Commit' button), and a 'Push' section with a 'Push' button.

Ilustración 87. Acciones generales en el repositorio

7.2.3. PÁGINA DE TESTING AUTOMÁTICO

La página de Testing Automático permite crear y probar scripts de pruebas mediante un archivo de formato yaml, que puedan ser reutilizados y que prueben el correcto funcionamiento de una métrica creando en los repositorios de Github las acciones que mida la métrica.



Ilustración 88. Acceso a página de "Automated Testing"

Para acceder a esta página, se hará desde el menú principal, haciendo click en “Automated Testing”.

Estructura de la página

Esta página es una interfaz de usuario para ejecutar y probar scripts. Estos script tendrán formato “.yaml”, e interactuarán con la API propia y los repositorios de Github para probar métricas ya creadas. La página se divide en varias secciones:

Metrics testing with yaml

File Name	Copy below	View File	Execute / Edit File	Delete File
actualtime.yaml				
creaArchivoYPush.yaml				
crear.yaml				
createIssue2.yaml				
EJEMPLOTEST.yaml				
Funciona.yaml				
NUEVOFINDCHECK.yaml				
onlytest.yaml				
PRUEBACOMPLETA.yaml				
PRUEBACOMPLETARB.yaml				
testvale.yaml				

Ilustración 89. Scripts YAMLs guardados

En la parte superior de la página, hay una tabla que muestra todos los archivos YAML guardados. Los usuarios pueden copiar el contenido del archivo en el cuadro de texto inferior, ver el archivo, editar el archivo o eliminar el archivo.

Zona de ejecución

En la sección de ejecución, los usuarios tienen un cuadro de texto en el que se introduce el archivo yaml que se quiera probar. Además se podrá ingresar el nombre del

archivo YAML para poder guardarlo y que se muestre con el resto de archivos yaml que se muestren en la zona de guardado.

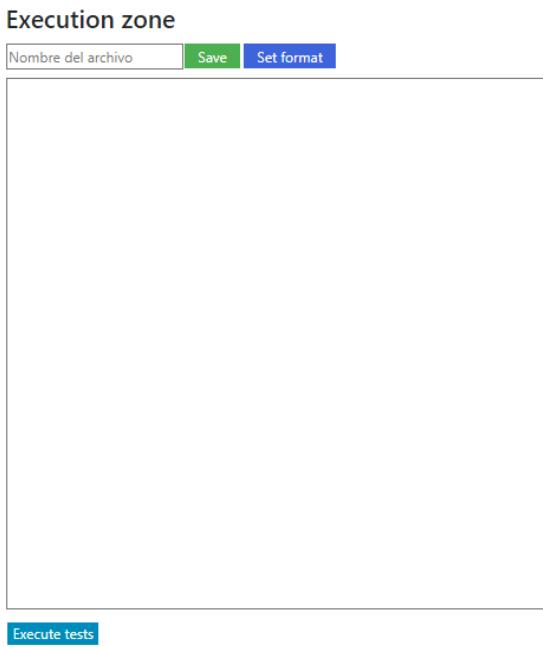


Ilustración 90. Zona de ejecucion en el test yaml

En este cuadro de texto los usuarios podrán seguir una serie de “steps”. Los “steps” son acciones que se pueden realizar en el sistema. Estos steps están predefinidos y realizan métodos (como ‘GET’, ‘POST’, ‘PUT’, ‘DELETE’, ‘TEST’) con sus acciones correspondientes.

En la zona derecha al textarea, se encuentran también ejemplos de los “steps” que se pueden ejecutar en el textarea y botones de copiar, para que solo haya que pegarlos en la zona del cuadro de texto que deseemos.

Examples

Search metric... **Search** **X**

Repository branches

```
- uses: "github/getBranches"
with:
repoName: "tp-testbench"
method: "GET"
```

Create branch

```
- uses: "github/createBranch"
with:
repoName: "tp-testbench"
branchName: "branchNameHere"
method: "POST"
```

Delete branch

```
- uses: "github/deleteBranch"
with:
repoName: "tp-testbench"
branchName: "branchNameHere"
method: "DELETE"
```

View issues

```
- uses: "github/getIssue"
with:
repoName: "tp-testbench"
owner: "Antoniosc7"
method: "GET"
```

Create issues

```
- uses: "github/createIssue"
with:
repoName: "tp-testbench"
owner: "Antoniosc7"
title: "titleText"
```

<< Previous **1** 2 3 4 Next >>

Ilustración 91. "Steps" disponible en la página de testing automático

Visualización de la ejecución

Después de ejecutar el bloque de tests, los resultados se muestran en un área de texto de solo lectura. Si el script realizó una computación, los resultados de esa computación también se muestran en un área de texto de solo lectura.

```

github/getBranches {
  "branches": [
    "32144211234",
    "3231",
    "hole",
    "jsdjadsojk",
    "main",
    "nombreDeLaRam123212a",
    "nombreDeLaRam32123a",
    "nombreDeLaRama2",
    "nombreDeLaRama333213",
    "ramaHoy",
    "ramaHoy2",
    "ramaHoy34",
    "ramaHoy433",
    "ramates",
    "ramates2",
    "ramates3",
    "ramates4",
    "ramates44",
    "ramates445",
    "ramates4455",
    "ramates44555",
    "ramates44552",
    "ramatestToday"
  ]
}

{
  "code": 200,
  "message": "OK",
  "computation": "/api/v2/computations/30643ccfd0fea903"
}
{
  "code": 200,
  "message": "OK",
  "computations": [
    {
      "scope": {
        "project": "TFG-GH-antoniosc7_Glassmatrix",
        "class": "TFG",
        "member": "Antonio"
      }
    },
    "period": {
      "from": "2024-06-09T02:00:00.000Z",
      "to": "2024-06-09T02:59:59.000Z"
    },
    "evidences": []
  ]
}

```

Ilustración 92. Visualización de la ejecución

Resultados de las pruebas

En la columna derecha de la página, debajo de los ejemplos de “steps” disponibles, los usuarios pueden ver los resultados de las pruebas que se han ejecutado. Cada resultado de la prueba se muestra en su propia tarjeta, y los usuarios pueden eliminar resultados de las pruebas haciendo click en la “x”.

Test Results

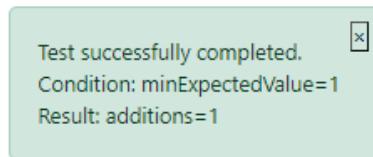


Ilustración 93. Resultados de las pruebas

Pantalla de edición de script YAML

La página de edición o ejecución de scripts YAMLs, funcionará exactamente igual que la página principal, con la diferencia que aquí solo podrás guardar los cambios de ese mismo script YAML o ejecutarlo para comprobar su funcionamiento.

You are viewing the file: PRUEBACOMPLETA.yaml

Execution zone

Set format

Update file

```
steps:  
  - uses: "github/getBranches"  
    with:  
      repoName: "tp-testbench"  
      method: "GET"  
  - uses: "github/pullCurrentBranch"  
    with:  
      repoName: "tp-testbench"  
      method: "GET"  
  - uses: "github/createBranch"  
    with:  
      repoName: "tp-testbench"  
      branchName: "ramatestToday223"  
      method: "POST"  
  - uses: "github/getBranches"  
    with:  
      repoName: "tp-testbench"  
      method: "GET"  
  - uses: "github/createFile"  
    with:
```

Execute tests

Examples

Search metric...

Search



Repository branches

```
- uses: "github/getBranches"  
  with:  
    repoName: "tp-testbench"  
    method: "GET"
```



Create branch

```
- uses:  
  "github/createBranch"  
  with:  
    repoName: "tp-  
testbench"
```



Delete branch

```
- uses:  
  "github/deleteBranch"  
  with:  
    repoName: "tp-  
testbench"
```



View issues

```
- uses: "github/getIssue"  
  with:  
    repoName: "tp-  
testbench"  
    owner: "Antoniiosc7"
```



Ilustración 94. Página de edición/ejecución de scripts YAMLs.

7.2.4 PÁGINA TESTEO DE TPAs

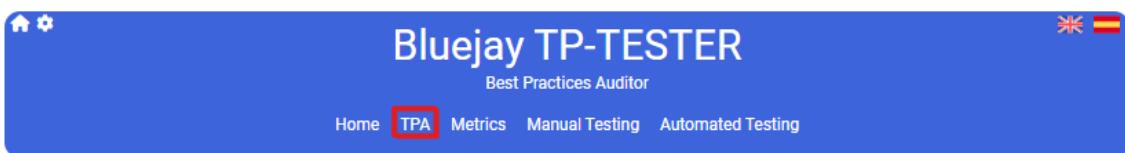


Ilustración 95. Acceso pantalla de TPA

El acceso a esta página será posible desde cualquier página de la web, haciendo click en “TPA”.

7.2.4.1 ESTRUCTURA DE LA PÁGINA

En primer lugar, al entrar en la página de gestión de TPA, encontraremos una tabla con todos los TPAs ya cargados dentro de Bluejay, esta tabla te permite visualizar e interactuar con todos estos TPAs.

Existings TPAs

ID	Project	Class	Options
tpa-TFG-GH-antoniosc7_tpPrueba2	TFG-GH-antoniosc7_tpPrueba2	TFG	
tpa-TFG-GH-antoniosc7_tpPrueba	TFG-GH-antoniosc7_tpPrueba	TFG	
tpa-TFG-GH-antoniosc7-tpPrueba	TFG-GH-antoniosc7-tpPrueba	TFG	
tpa-class01-GH-cs169_fa23-TPA-5	class01-GH-cs169_fa23-TPA-5	class01	
tpa-TFG-GH-JaviFdez7_ISPP-G1-Talent	TFG-GH-JaviFdez7_ISPP-G1-Talent	TFG	
tpa-TFG-GH-antoniosc7_Glassmatrix	TFG-GH-antoniosc7_Glassmatrix	TFG	

Ilustración 96. Tabla con todos los TPAs ya cargados

Esta sección muestra una tabla de todos los TPAs existentes. Cada fila de la tabla representa un solo TPA, mostrando su ID, proyecto y clase.

También hay varios botones de acción para cada TPA:

- El botón **Copiar** permite copiar el contenido de un TPA en el área de texto inferior.
- El botón **Ver** navega a una vista detallada del TPA.
- El botón **Editar** navega a una página donde puedes editar el TPA.
- El botón **Eliminar** elimina el TPA de la base de datos de Bluejay.

Si seguimos navegando por esta página encontraremos la sección de crear un nuevo TPA. Para ello proporciona un área de texto donde puedes introducir el contenido del nuevo TPA.

Create TPA

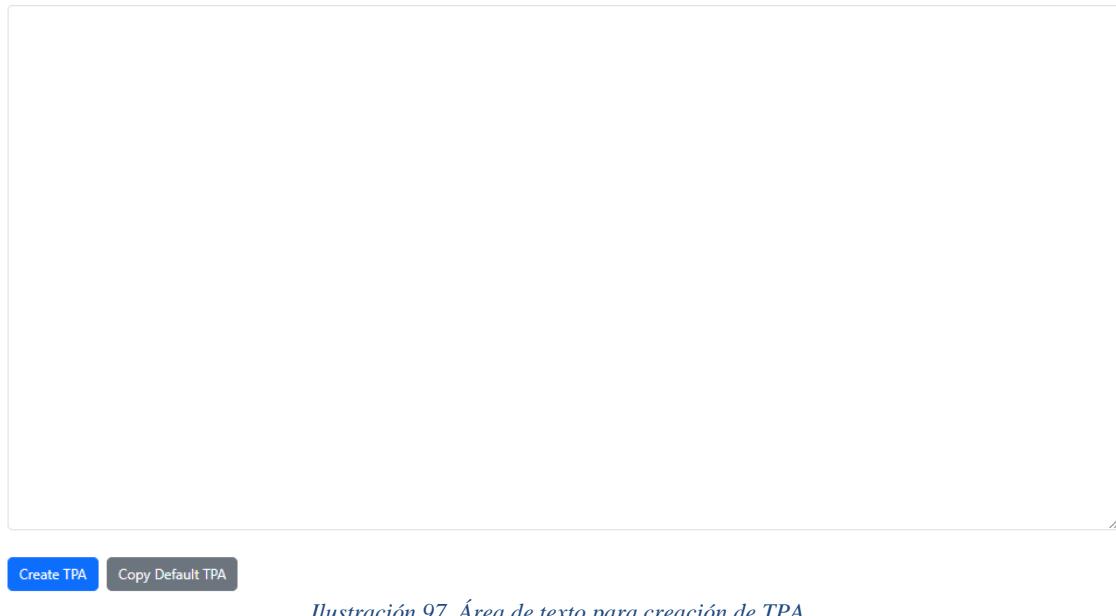


Ilustración 97. Área de texto para creación de TPA

Hay dos botones:

- El botón Crear TPA crea un nuevo TPA con el contenido que has introducido en el área de texto.
- El botón Copiar TPA por defecto llena el área de texto con el contenido de un TPA por defecto, que luego puedes modificar y crear como un nuevo TPA.

7.2.4.2 EDICIÓN DE TPAs

Cuando haces clic en el botón de edición de un TPA, entrarás en una subpágina que te dará acceso a otras dos subpáginas.

You are about to enter the TPA editing screen. Once edited, you cannot go back. For more information, visit:
<https://docs.bluejay.governify.io/customization/agreement-modeling/team-practices-agreements>



Ilustración 98. Selección de modo de edición de TPAs

- Una donde puedes editar todo el TPA.
- Y otra donde puedes editar el TPA por secciones.

EDITAR TPA COMPLETO

Aquí podrás editar el contenido del TPA en un área de texto con el contenido actual del TPA y un botón de guardar que actualizará ese contenido

TPA Update

TPA JSON:

```
{  
  "_id": "660aec99030c09d63737594d",  
  "id": "tpa-TFG-GH-antoniosc7_Glassmatrix",  
  "version": "1.0.0",  
  "type": "agreement",  
  "context": {  
    "validity": {  
      "initial": "2019-01-01",  
      "timeZone": "America/Los_Angeles"  
    },  
    "definitions": {  
      "schemas": {},  
      "scopes": {  
        "development": {  
          "project": {  
            "name": "Project",  
            "description": "Project",  
            "type": "string",  
            "default": "TFG-GH-antoniosc7_Glassmatrix"  
          },  
          "class": {  
            "name": "Class",  
            "description": "Group some Projects",  
            "type": "string",  
            "default": "TFG"  
          }  
        }  
      }  
    }  
  }  
}
```

Ilustración 99. Visualización de pantalla de edición de TPA Completo.

Al usar el botón de “TPA Update”, se actualizará también el TPA en todos los microservicios de Bluejay.

EDITAR TPA POR SECCIONES

En esta página puedes editar el TPA por secciones, editando o eliminando cada métrica y garantía del TPA. Y también puedes añadir nuevas métricas y garantías. Si seleccionas el botón “más”, generará una métrica individual (que se añadirá a la página de metrics) para que pueda ser revisada y probada, y te redirigirá a la nueva página con esa métrica creada.

Create metric or guarantee

Context Section

In this section, you can find the most important parts of the TPA, the metrics and guarantees.

METRICS

COUNT_DONEISSUES_MERGEDPR	+	🔗	trash	▼
COUNT_INPROGRESSISSUES_NEWBRANCH	+	🔗	trash	▼
COUNT_INREVIEWISSUES_OPENPR	+	🔗	trash	▼
COUNT_MERGED_PR_WITH_POSITIVE_REVIEWS_MEMBER	+	🔗	trash	▼
COUNT_PRS_WITH_AT_LEAST_ONE_COMMENT_OR_ONE_REVIEW_COMMENT_BY_MEMBER	+	🔗	trash	▼
NUMBER_GH_MERGEDPR	+	🔗	trash	▼
NUMBER_GH_NEWBRANCH	+	🔗	trash	▼
NUMBER_GH_OPENPR	+	🔗	trash	▼

GUARANTEES

CORRELATION_INPROGRESSISSUES_NEWBRANCH	🔗	trash	▼
CORRELATION_INREVIEWISSUES_OPENPR	🔗	trash	▼
CORRELATION_DONEISSUES_MERGEDPR	🔗	trash	▼
NUMBER_APPROVEDMERGEDPULLREQUEST_MEMBER	🔗	trash	▼
NUMBER_COUNTPRSWITHATLEASTONECOMMENTORONEREVIEWCOMMENTBYMEMBER	🔗	trash	▼

7.2.5. PÁGINA DE CONFIGURACIÓN

La página de Configuración ofrece una interfaz amigable para gestionar la configuración de la aplicación. Proporciona opciones para ver los contenedores Docker activos, actualizar la configuración de la aplicación, ver la documentación de Swagger y consultar la documentación de la aplicación.

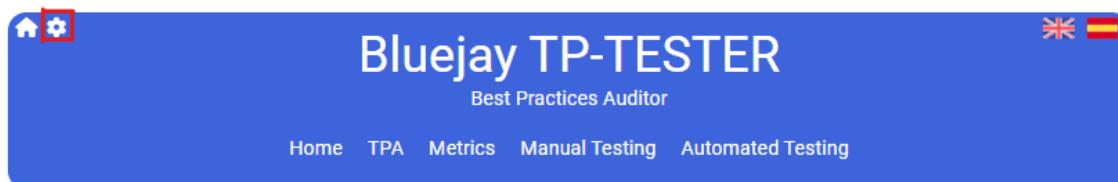


Ilustración 100. Acceso a pantalla de configuración

El acceso a esta página será posible desde cualquier página de la web, en la esquina superior izquierda, el ícono de herramientas o configuración.

La página está dividida en varias secciones:

- La sección **Docker Activos** muestra una tabla de los contenedores Docker activos en ese momento. Cada fila de la tabla representa un contenedor Docker y proporciona información sobre el ID del contenedor, su nombre, URL y puerto.

Active Docker

Container ID	Container Name	URL	Port
2094f5b4e50cd6fc0cfdea4a8dbe741762366ed1d4b3de5c813a40d593ac955	bluejay-dashboard	http://localhost:5600	5600
d8e7e66f587af7c953aa110cb7480ea94985f9dc1239cbcda30325873cd9c	bluejay-scope-manager	http://localhost:5700	5700
82d445b5549134047ee8389811da6723aea43fa58d16d38ac97750d6deeb3dc3	bluejay-reporter	http://localhost:5300	5300
a5b479fb3caaca5f941778b6b60d66093ded147a03eeada35b2b2df132b2c1	bluejay-render	http://localhost:5100	5100
2ac53e1735209c9ddc82bae26e9ea46c3ceeeddb492a8702dee9fd400a15ac07	bluejay-director	http://localhost:5800	5800
bbe73580ce68ed596b4b8b9ae7209e45841d73d9b1eedb3307467d8bb54b82e	bluejay-collector-events	http://localhost:5500	5500
dbaf285ff0b177ea4d7e1af1365dfe0cda894176bca690427230fb2d44b0499	bluejay-registry	http://localhost:5400	5400
9f50f708a60458a3f6fc0372cabda28f63f74af74aad12147ae8b71e401a3e3	bluejay-assets-manager	http://localhost:5200	5200
67d0117faa8baf8bdcfa7a2735a6e8fa480aa7d2b345428c5e98309b7045a55	bluejay-join	http://localhost:6001	6001
e99ef3f1443033095c548a8b6e3edc690b2d9712643a44a34cbe6d0fcba316d73	bluejay-mongo-registry	http://localhost:5001	5001
f667916a1bdce078af07377ce7fbfdce927f15a218bac353a3859bb58260e0	bluejay-influx-reporter	http://localhost:5002	5002
12ab70155123f7b1fe7875b9fd0969c0e02b75a1b1ca1045bc27527fd2899f6f	bluejay-redis-ec	http://localhost:5003	5003
878e93d6b7a12318b67713b33969b261817250c8512f8ad7ee89c186b5bec66	bluejay-tpa-designer	http://localhost:5173	5173

Ilustración 101. Visualización de contenedores dockers activos

Esta sección permitirá de una forma rápida y muy visual si hay algún micro servicio que no esté en ejecución o en que puerto será posible encontrar cada uno de los microservicios.

- La sección **Token de Github** proporciona instrucciones sobre cómo obtener un token de Github. También incluye un botón para abrir un diálogo con ayuda adicional.

Github Token

[How to get a GitHub token](#)
Help

Token found

Token: tokentest1	Edit Token
-------------------	---

<input style="width: 150px; height: 20px; border: 1px solid #ccc; border-radius: 5px; padding: 2px; margin-right: 10px;" type="text"/> Enter new token	Update Token
--	---

Ilustración 102. Sección de actualización de Token Github

Aquí permite cambiar el token de Github que usa la web, ya que esto es completamente necesario para el testeo de las métricas que usen repositorios de Github. Además, el botón help abrirá un pop up con detalles de cómo obtener este token.

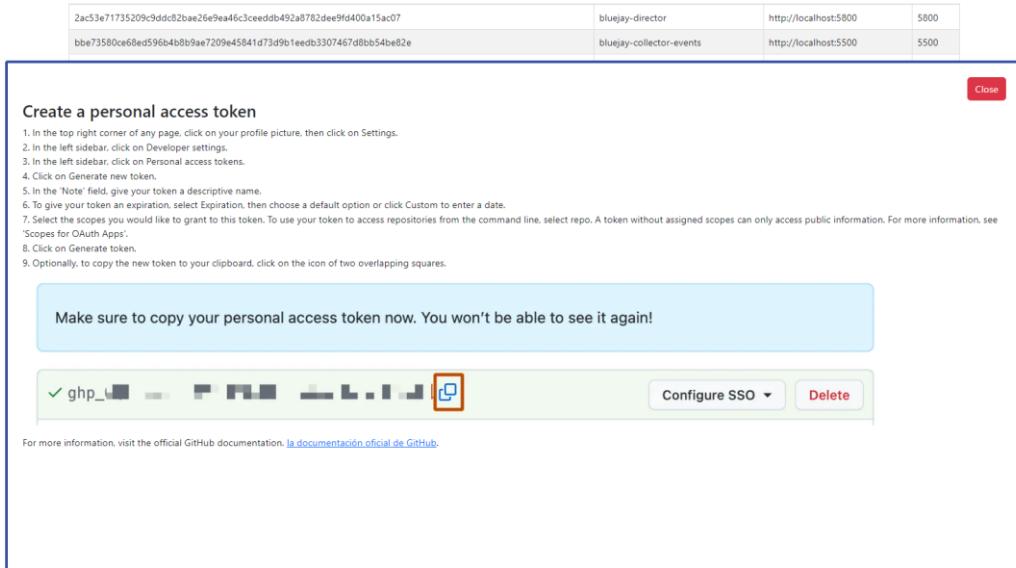


Ilustración 103. Pop Up de ayuda para obtener token de Github

- La sección **Constantes** ofrece un formulario para actualizar la configuración de la aplicación. El formulario incluye campos para la URL base, el colector por defecto, la URL de eventos del colector, la URL de acuerdos y la URL de alcances. Hay un botón para enviar el formulario y actualizar la configuración.

Constants

You should restart the server to apply the changes

Base URL	http://localhost
Default Collector	EVENTS
Collector Events URL	http://localhost:5500/api/v2/computations
Agreements URL	http://localhost:5400/api/v6/agreements
Scopes URL	http://host.docker.internal:5700/api/v1/scopes/development

Update

Ilustración 104. Constantes configurables en la aplicación

Para que los cambios se lleven a cabo será necesario usar el botón update y reiniciar la página para que al levantarse se ejecute con las nuevas URLs.

- La sección Swagger proporciona un enlace a la documentación de Swagger.

OpenAPI Specification

Swagger is a tool used to design, build, document, and use RESTful web services. It allows developers and users to understand and work with a web service API in an easier and more visual way.
From swagger, you can make calls to the API itself and check its correct operation without the need for other applications like Postman.

Open OpenAPI Specification (Swagger)

Ilustración 105. Punto de acceso a la documentación de la API propia

La API que requiere la aplicación implementa Swagger para tener siempre todos los endpoints de la API en un solo lugar junto con una pequeña documentación sobre el funcionamiento de cada Endpoint. Gracias a Swagger, puedes ver todos los puntos finales y probarlos en su interfaz gráfica que está disponible en: <http://url:6012/api-docs/>

7.3 GUIA DE REFERENCIA DE TPA TESTER.YAML

La página de testing, tiene sus propios métodos para interaccionar con los repositorios y realizar las pruebas oportunas para comprobar el correcto funcionamiento de nuestra métrica.

Para probar utilizaremos diferentes pasos o “steps”. Estos steps los dividimos en 5 categorias.

1. Los métodos GETs que son aquellos que obtienen algún tipo de información.
2. Los POSTs son aquellos que mandan una petición (normalmente a github)
3. Métodos PUTs utilizados para actualizar alguna información.
4. Métodos DELETEs para eliminar ya sea una rama, fichero o texto.
5. Y los métodos TESTs que son utilizados para realizar pruebas sobre el contenido que devuelven las computaciones

1. Métodos GETs

- `github/getIssue`: Este step obtiene los issues de un repositorio específico en GitHub.

View issues

```
- uses: "github/getIssue"
  with:
    repoName: "tp-testbench"
    owner: "Antoniiosc7"
    method: "GET"
```

- `github/getOpenPR`: Este step obtiene los pull requests abiertos de un repositorio específico en GitHub.

Open PRs

```
- uses: "github/getOpenPR"
  with:
    repoName: "tp-testbench"
    owner: "Antoniiosc7"
    method: "GET"
```

- `github/pullCurrentBranch`: Este step realiza un pull de la rama actual en un repositorio específico.

Pull the branch

```
- uses: "github/pullCurrentBranch"  
  with:  
    repoName: "tp-testbench"  
    method: "GET"
```

- `github/listRepos`: Este step lista todos los repositorios.

List of repositories

```
- uses: "github/listRepos"  
  method: "GET"
```

- `github/getBranches`: Este step obtiene todas las ramas de un repositorio específico.

Create branch

```
- uses: "github/createBranch"  
  with:  
    repoName: "tp-testbench"  
    branchName: "branchNameHere"  
    method: "POST"
```

- `github/getRepoInfo`: Este step obtiene información sobre un repositorio específico y una rama específica.

Info from a repository

```
- uses: "github/getRepoInfo"  
  with:  
    repoName: "tp-testbench"  
    branchName: "main"  
    method: "GET"
```

2. Métodos POSTs

- `github/mergeLastOpenPR`: Este step fusiona el último pull request abierto en un repositorio específico en GitHub.

Merge last created PR

```
uses: "github/mergeLastOpenPR"
with:
  repoName: "tp-testbench"
  owner: "Antoniiosc7"
  mergeMessage: "mergedPR"
  method: "POST"
```

- `bluejay/compute/tpa`: Este step carga los datos de un archivo y luego realiza una computación en los datos.

Compute TPA Metrics

```
- uses: "bluejay/compute/tpa"
with:
  collector: "EVENTS"
  tpa: "tpa-TFG-GH-antoniosc7_Glassmatrix"
  metric: "working_metric.json"
  method: "POST"
```

- `bluejay/compute/metric`: Este step carga los datos de un archivo y luego realiza una computación en los datos.

Compute Metric

```
- uses: "bluejay/compute/metric"
with:
  actualTime: "true/false"
  collector: "EVENTS"
  metric: "additions_metric.json"
  method: "POST"
```

- `bluejay/checkContain`: Este step (que está marcado como deprecado) comprueba si un valor específico está presente en los datos obtenidos de una API.

Old check in response

```
- uses: "bluejay/checkContain"
with:
  key: "additions"
  minExpectedValue: "5"
  method: "POST"
```

- `github/createIssue`: Este step crea un nuevo issue en un repositorio específico en GitHub.

Create issues

```
- uses: "github/createIssue"
  with:
    repoName: "tp-testbench"
    owner: "Antoniiosc7"
    title: "titleText"
    body: "bodyText"
    method: "POST"
```

- `github/createPR`: Este step crea un nuevo pull request en un repositorio específico en GitHub.

Create PR

```
- uses: "github/createPR"
  with:
    repoName: "tp-testbench"
    owner: "Antoniiosc7"
    title: "title1"
    head: "main"
    base: "branchX"
    body: "bodyText"
    method: "POST"
```

- `github/cloneRepo`: Este step clona un repositorio específico.

Clone repository

```
- uses: "github/cloneRepo"
  with:
    owner: "Antoniiosc7"
    repoName: "tp-testbench"
    method: "POST"
```

- `github/createBranch`: Este step crea una nueva rama en un repositorio específico.

Create new branch

```
uses: "github/createBranch"
  with:
    repoName: "tp-testbench"
    branchName: "branch2"
    method: "POST"
```

- `github/createFile`: Este step crea un nuevo archivo en un repositorio específico.

Create file

```
- uses: "github/createFile"
  with:
    repoName: "tp-testbench"
    fileName: "file.txt"
    fileContent: "content"
    method: "POST"
```

- `github/commitAllChanges`: Este step realiza un commit de todos los cambios en un repositorio específico.

Make commit

```
- uses: "github/commitAllChanges"
  with:
    repoName: "tp-testbench"
    commitMessage: "commit message"
    method: "POST"
```

- `github/pushChanges`: Este step realiza un push de todos los cambios en un repositorio específico.

Push

```
- uses: "github/pushChanges"
  with:
    repoName: "tp-testbench"
    method: "POST"
```

3. Métodos PUTs

- `github/mergePR`: Este step fusiona un pull request específico en un repositorio específico en GitHub.

Merge a PR

```
- uses: "github/mergePR"
  with:
    repoName: "tp-testbench"
    owner: "Antoniosc7"
    prNumber: "1"
    mergeMessage: "mergead"
    method: "PUT"
```

- `github/changeBranch`: Este step cambia a una rama específica en un repositorio específico.

Change branch

```
- uses: "github/changeBranch"  
  with:  
    repoName: "tp-testbench"  
    branchToChangeTo: "branch2"  
    method: "PUT"
```

4. Métodos DELETEs

- `github/deleteRepo`: Este step elimina un repositorio específico.

Delete repository

```
- uses: "github/deleteRepo"  
  with:  
    repoName: "tp-testbench"  
    method: "DELETE"
```

- `github/deleteBranch`: Este step elimina una rama específica de un repositorio.

Delete branch

```
- uses: "github/deleteBranch"  
  with:  
    repoName: "tp-testbench"  
    branchName: "branchNameHere"  
    method: "DELETE"
```

- `github/deleteFile`: Este step borra un archivo de un repositorio

Delete file

```
- uses: "github/deleteFile"  
  with:  
    repoName: "tp-testbench"  
    fileName: "name_of_file"  
    method: "DELETE"
```

5. Métodos TESTs

- **`bluejay/check`**: Este step realiza una serie de comprobaciones en los datos obtenidos de una API. Comprueba si el valor de una clave específica cumple con ciertas condiciones (como un valor mínimo esperado, un valor máximo esperado, o un valor esperado exacto).

Check in response

```
- uses: "bluejay/check"
value: "1" //optional
with:
- key: "additions"
conditions:
minExpectedValue: "5"
maxExpectedValue: "12"
- key: "additions"
conditions:
minExpectedValue: "5"
maxExpectedValue: "12"
- key: "key1"
conditions:
expectedValue: "value1"
method: "TEST"
```

- **`bluejay/findCheck`** dasadssadasddas

Check in response 2

```
- uses: "bluejay/findCheck"
with:
values:
- value: 1
computationCount: 1
evidences:
login: "Antoniiosc7"
bodyText: "wip"
method: "TEST"
```

Cada step se ejecuta en función de los datos proporcionados en el cuadro de texto que se está procesando.

7.3.1 VARIABLES A USAR

actualTime

Para que la métrica se compute con la hora actual, se deberá añadir `actualTime: "true"` al método `"bluejay/compute/metric"`. Si por el contrario se quiere que use el tiempo original de la métrica se puede eliminar `"actualTime"` o establecerlo como `"false"`.

```

steps:
  - uses: "bluejay/compute/metric"
    with:
      collector: "EVENTS"
      metric: "additions_metric.json"
      actualTime: "true"
    method: "POST"
  - uses: "bluejay/check"
    with:
      - key: "additions"
        conditions:
          expectedValue: "49"
    method: "TEST"

```

Código 23. Uso de Actual Time en test.yml

value

Si hay varios resultados, se podrá utilizar la métrica “value” para que únicamente se compruebe aquellas evidencias que tengan el campo “value” en el valor establecido.

```

steps:
  - uses: "bluejay/compute/metric"
    with:
      collector: "EVENTS"
      metric: "additions_metric.json"
    method: "POST"
  - uses: "bluejay/check"
    value: "1"
    with:
      - key: "additions"
        conditions:
          minExpectedValue: "5"

```

Código 24. Especificación de evidencia en los que buscar los resultados

minExpectedValue

El test será exitoso si hay algún campo llamado como el campo key, en este caso “additions” cuyo valor sea numérico y sea mayor que 5.

```

  - uses: "bluejay/check"
    value: "1"
    with:
      - key: "additions"
        conditions:
          minExpectedValue: "5"

```

Código 25. Uso de minExpectedValue en testing

maxExpectedValue

El test será exitoso si hay algún campo llamado como el campo key, en este caso “additions” cuyo valor sea numérico y sea menor que 12.

```
- uses: "bluejay/check"
  value: "1"
  with:
    - key: "additions"
      conditions:
        maxExpectedValue: "12"
```

expectedValue

El test será exitoso si hay algún campo llamado como el campo key, en este caso “additions” cuyo valor sea numérico o no y sea exactamente igual al valor de expected value.

```
- uses: "bluejay/check"
  with:
    - key: "additions"
      conditions:
        expectedValue: "49"
      method: "TEST"
```

7.4 TUTORIAL CICLO DE CREACIÓN DE UN TPA

Para la creación de un TPA completo deberemos ir paso por paso. En primer lugar, deberemos elaborar un documento en el que se definan las métricas y garantías que deban tener los acuerdos de equipo que hayamos elaborado.

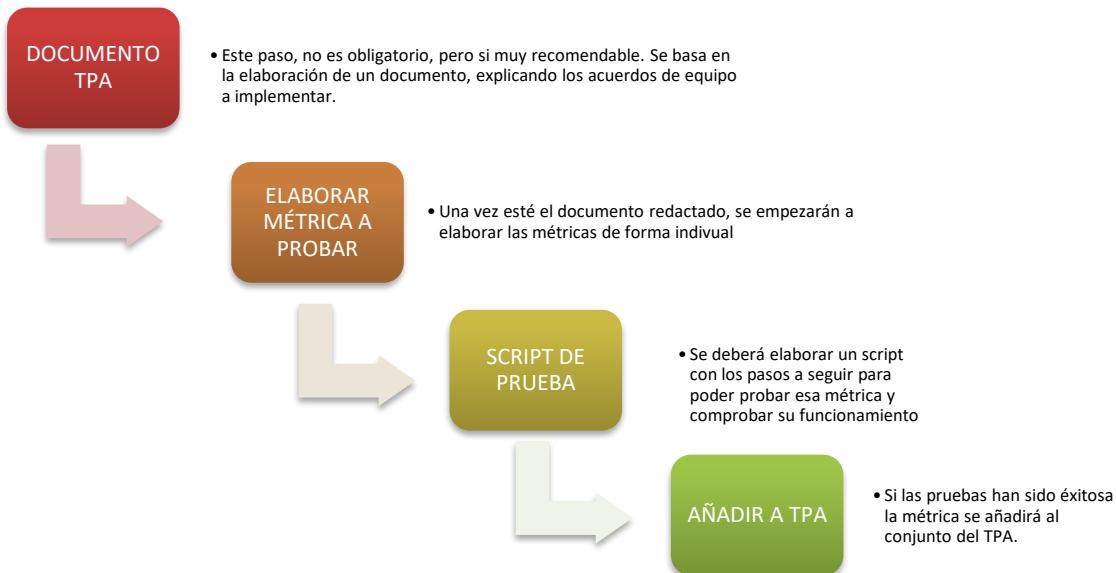


Ilustración 106. Ciclo de vida de creación de un TPA.

A continuación, se detallará un ejemplo de este ciclo de vida, realizado sobre un repositorio de prueba en Github llamado TP-Testbench [22].

DOCUMENTO TPA

Crear un documento con el funcionamiento de la métrica no es obligatorio, pero es muy recomendable para tener claro el objetivo de la métrica y los parámetros que medirá.

Este documento tendrá 5 secciones, una descripción de este, los requisitos que tengan que estar establecidos previamente, las métricas que mide y los parámetros que use la métrica.

Para este ejemplo, vamos a usar la primera métrica elaborada en el apartado 4.4.2 TPA DE DEVOPS. Esta métrica busca evitar pull request de tamaño excesivo, ya que no es recomendable que en una pull request se concentre una gran variedad de cambios y es una mejor práctica la fragmentación de ello en funcionalidades.

Una vez sabiendo tanto, la parte de documentación para esta métrica será:

Descripción:

Evaluar el tamaño de las solicitudes de fusión es esencial para garantizar un proceso de revisión de código eficiente. Esta TP se centra en medir el tamaño de cada solicitud de fusión. Se asume que se debe intentar dividir lo más posible las tareas de forma que se puedan completar lo más rápido posible disponiendo de un tamaño reducido.

Requisitos previos:

- El equipo sigue un proceso formal de revisión de código.
- El equipo debe establecer qué porcentaje máximo de cambio en el código se considerará aceptable en una Pull Request y qué porcentaje de casos podrán sobrepasar esta medida.

Métricas:

- **NPRM_T**: Número de pull request del equipo a la semana que cumplan que el número de líneas de código cambiadas (creadas, modificadas o borradas) sea menor al umbral **PM_{TP1}**.
- **NPR_T**: Número de pull request del equipo a la semana.

Objetivo:

$$\frac{NPRM_T}{NPR_T} \times 100 \geq U_{TP1}$$

Semanalmente por Equipo

Parámetros:

$$\boxed{\begin{array}{c} \mathbf{PM_{TP1}}= 200 \text{ líneas} \\ \mathbf{U_{TP1}}= 90 \end{array}}$$

- **PM_{TP1}** representaría el número de líneas máximas a modificar en una pull request. Esto dependerá de las políticas y preferencias de tu equipo.
- **U_{TP1}** sería el umbral de porcentaje de casos en los que se espera que se cumpla la condición anterior. Esto nuevamente dependerá de las políticas del equipo de trabajo.

ELABORACIÓN DE METRICA

Una vez que ya tenemos definida la métrica que queremos crear, ya podemos tratar de plasmar esto dentro de TP-Tester. Para ello, dentro de TP-Tester iremos a la página de *Metrics*.

Una vez aquí, bajaremos a la sección de crear una nueva métrica ya que aquí podremos ver el tutorial de creación de métricas dentro de la página de TP-Tester o copiar un ejemplo de una métrica que funcione.

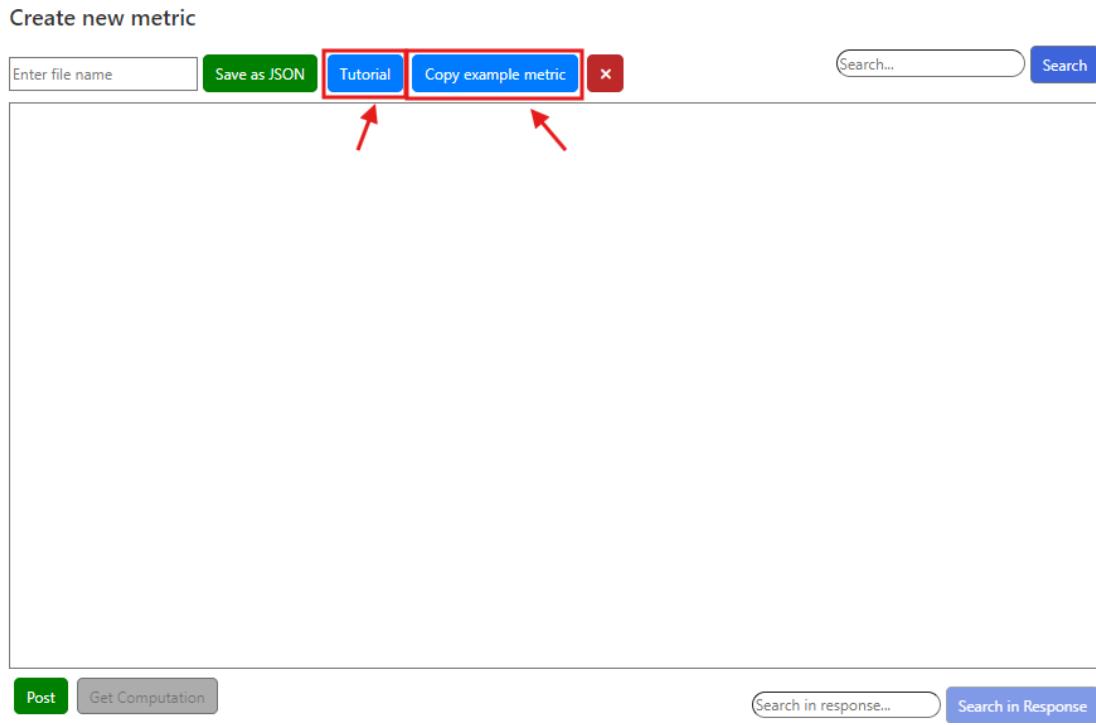


Ilustración 107. Vista de sección para crear una nueva métrica.

Si copiamos la métrica de ejemplo, en el cuadro de texto se nos pondrá la siguiente métrica:

```

1. {
2.   "config": {
3.     "scopeManager": "http://host.docker.internal:5700/api/v1/scopes/development"
4.   },
5.   "metric": {
6.     "computing": "actual",
7.     "element": "number",
8.     "event": {
9.       "githubGraphQL": {
10.         "custom": {
11.           "type": "graphQL",
12.           "title": "Get pull requests with at least one comment by member",
13.           "steps": {
14.             "0": {
15.               "type": "queryGetObject",
16.               "query": "{\r\n      repository(name: \"%PROJECT.github.repository%\",\r\n      owner: \"%PROJECT.github.repoOwner%\") {\r\n        pullRequests(first: 100) {\r\n          pageInfo {\r\n            endCursor\r\n            hasNextPage\r\n          }\r\n          nodes {\r\n            bodyText\r\n            number\r\n            state\r\n            createdAt\r\n            author {\r\n              login\r\n            }\r\n            mergedAt\r\n            mergedBy {\r\n              login\r\n            }\r\n            comments(first: 50) {\r\n              nodes {\r\n                bodyText\r\n                createdAt\r\n                author {\r\n                  login\r\n                }\r\n                reviews(first: 30) {\r\n                  nodes {\r\n                    bodyText\r\n                    author {\r\n                      login\r\n                    }\r\n                    state\r\n                  }\r\n                }\r\n              }\r\n            }\r\n          }\r\n        }\r\n      }\r\n    }\r\n  }
17. },
18. "1": {
19.   "type": "objectGetSubObjects",
20.   "location": "data.repository.pullRequests.nodes"
21. },
22. "2": {
23.   "type": "runScript",

```

```

24.           "variables": {},
25.           "script": "module.exports.generic = function
getPullRequestsWithAtLeastOneCommentByMember(inputData,variables) {\r\n    let
pullRequestsMergedAndOpenWithAtLeastOneComment = []\r\n    for (let pullRequest of
inputData) {\r\n        if (pullRequest.author.login === '%MEMBER.github.username%')
continue;\r\n        let isPullRequestAlreadyAdded = false;\r\n        if
(pullRequest.state === 'OPEN' || pullRequest.state === 'MERGED' && Math.max(new
Date(variables.from), new Date(pullRequest.createdAt)) <= Math.min(new Date(variables.to),
new Date(pullRequest.mergedAt))) {\r\n            for (let comment of
pullRequest.comments.nodes) {\r\n                if (comment.author.login ===
'%MEMBER.github.username%' && new Date(comment.createdAt) > new Date(variables.from) &&
new Date(comment.createdAt) < new Date(variables.to)) {\r\n                    pullRequestsMergedAndOpenWithAtLeastOneComment.push(pullRequest)\r\n                    isPullRequestAlreadyAdded = true;\r\n                    break;\r\n                }\r\n            }\r\n        } else if (isPullRequestAlreadyAdded) continue;\r\n        for (let
review of pullRequest.reviews.nodes) {\r\n            if (review.bodyText.length > 0 &&
review.author.login === '%MEMBER.github.username%' && new Date(review.createdAt) > new
Date(variables.from) && new Date(review.createdAt) < new Date(variables.to)) {\r\n                pullRequestsMergedAndOpenWithAtLeastOneComment.push(pullRequest)\r\n                break;\r\n            }\r\n        }\r\n    }\r\n    return
pullRequestsMergedAndOpenWithAtLeastOneComment;\r\n}\r\n\r\n26.       }
27.     }
28.   }
29. }
30. },
31. "scope": {
32.   "project": "TFG-GH-ANTONIO",
33.   "class": "TFG",
34.   "member": "*"
35. },
36. "window": {
37.   "type": "static",
38.   "period": "hourly",
39.   "initial": "2024-03-14T00:00:00.000Z",
40.   "from": "2024-03-14T00:00:00.000Z",
41.   "end": "2024-03-14T23:59:59.999Z",
42.   "timeZone": "America/Los_Angeles"
43. }
44. }
45. }

```

En ella podemos ver una estructura clara que podremos sustituir para que haga lo que queremos. En primer lugar, tendremos que encontrar una query de Graph QL que obtenga los datos que queremos comparar. Y después también deberemos actualizar el scope del proyecto para que sea el que vayamos a usar.

Para obtener la query, podemos realizar pruebas en Github GQL Explorer [23]. Por ejemplo, en mi caso, la query que devuelve toda la información que devuelve todos los datos que necesito para comprobar que las pull request no sean demasiado grandes será la siguiente:

```

1. {
2.   repository(name: "tp-testbench", owner: "antoniosc7") {
3.     pullRequests(first: 100) {
4.       pageInfo {
5.         endCursor
6.         hasNextPage
7.       }
8.       nodes {
9.         bodyText
10.        number
11.        state
12.        createdAt
13.        author {

```

```

14.      login
15.    }
16.    mergedAt
17.    mergedBy {
18.      login
19.    }
20.    additions
21.    comments(first: 50) {
22.      nodes {
23.        author {
24.          login
25.        }
26.        bodyText
27.        createdAt
28.      }
29.    }
30.    reviews(first: 30) {
31.      nodes {
32.        state
33.        createdAt
34.        bodyText
35.        author {
36.          login
37.        }
38.      }
39.    }
40.  }
41.}
42.}
43.}
44.

```

En el propio Github GQL Explorer [23] se puede ejecutar la query y comprobar que resultados devuelve para ver si nos sirve la query o no. Esto se puede ver en la Ilustración 108.

The screenshot shows the GitHub GraphQL API Explorer interface. At the top, it says "Signed in as Antoniosc7. You're ready to explore! Sign out". Below that, a message reads "Heads up! GitHub's GraphQL Explorer makes use of your real, live, production data." The main area contains a code editor on the left and a results panel on the right. The code editor displays a GraphQL query:

- repository(name: "tp-testbench", owner: "an")
- pullRequests(first: 100) {
- pageInfo {
- endCursor
- hasNextPage
- }
- nodes {
- bodyText
- number
- state
- createdAt
- author {
- login
- }
- mergedAt
- mergedBy {
- login
- }

The results panel shows the JSON response for the first pull request:

- "bodyText": "asd",
- "number": 1,
- "state": "MERGED",
- "createdAt": "2024-03-19T16:21:32Z",
- "author": {
- "login": "Antoniosc7"
- },
- "mergedAt": "2024-03-19T17:06:50Z",
- "mergedBy": {
- "login": "Antoniosc7"
- },
- "additions": 2,
- "comments": {
- "nodes": []
- },
- "reviews": {
- "nodes": []
- },
- "bodyText": "asd".

Ilustración 108. Ejecución de query en Github GQL.

Y como se puede observar, la query devuelve los datos que necesitamos ya que nos devuelve datos como si está mergeada o simplemente creada la pull request, el autor

de esta, si ha recibido revisiones y lo más importante, devuelve “**additions**” que es el número de líneas que se han añadido.

Por tanto, ahora ya si podemos actualizar la métrica de ejemplo con los datos relativos a mi TPA, y con la nueva query. Ahora tendría este formato:

```

1. {
2.   "config": {
3.     "scopeManager": "http://host.docker.internal:5700/api/v1/scopes/development"
4.   },
5.   "metric": {
6.     "computing": "actual",
7.     "element": "number",
8.     "event": {
9.       "githubGraphQL": {
10.         "custom": {
11.           "type": "graphQL",
12.           "title": "Get lines added per closed pull requests",
13.           "steps": {
14.             "0": {
15.               "type": "queryGetObject",
16.               "query": "{\r\n    repository(name: \"%PROJECT.github.repository%\",\r\n    owner: \"%PROJECT.github.repoOwner%\") {\r\n        pullRequests(first: 100) {\r\n            pageInfo {\r\n                endCursor\r\n                hasNextPage\r\n                } \r\n                nodes {\r\n                    bodyText\r\n                    number\r\n                    state\r\n                    createdAt\r\n                    author {\r\n                        login\r\n                        mergedAt\r\n                        mergedBy {\r\n                            login\r\n                            additions\r\n                            comments(first: 50) {\r\n                                nodes {\r\n                                    author {\r\n                                        login\r\n                                        } \r\n                                        bodyText\r\n                                        createdAt\r\n                                        state\r\n                                        reviews(first: 30) {\r\n                                            nodes {\r\n                                                login\r\n                                                bodyText\r\n                                                createdAt\r\n                                                state\r\n                                                reviews(first: 30) {\r\n                                                    nodes {\r\n                                                        login\r\n                                                        bodyText\r\n                                                        reviews(first: 30) {\r\n                                                            nodes {\r\n                                                                login\r\n                                                                bodyText\r\n                                                                reviews(first: 30) {\r\n                                                                    nodes {\r\n                                                                        login\r\n                                                                        bodyText\r\n                                                                        reviews(first: 30) {\r\n                                                                            nodes {\r\n                                                                                login\r\n                                                                                bodyText\r\n                                                                                reviews(first: 30) {\r\n                                                                                    nodes {\r\n                                                                                        login\r\n                                                                                        bodyText\r\n                                                                                        reviews(first: 30) {\r\n                                                                                            nodes {\r\n                                                                                                login\r\n                                                                                                bodyText\r\n                                                                                                reviews(first: 30) {\r\n                                                                                                    nodes {\r\n                                                                                                        login\r\n................................................................
17.             },
18.             "1": {
19.               "type": "objectGetSubObjects",
20.               "location": "data.repository.pullRequests.nodes"
21.             },
22.             "2": {
23.               "type": "runScript",
24.               "variables": {},
25.               "script": "module.exports.generic = function\r\ngetPullRequests(inputData,variables) {\r\n    let pullRequestsMergedAndOpen = []\r\n    for (let pullRequest of inputData) {\r\n        if (pullRequest.author.login ===\r\n            '%MEMBER.github.username%') continue;\r\n        if (pullRequest.state === 'OPEN' ||\r\n            pullRequest.state === 'MERGED' && Math.max(new Date(variables.from), new\r\nDate(pullRequest.createdAt)) <= Math.min(new Date(variables.to), new\r\nDate(pullRequest.mergedAt))) {\r\n            pullRequestsMergedAndOpen.push(pullRequest)\r\n        }\r\n    }\r\n    return pullRequestsMergedAndOpen;\r\n}\r\n"
26.           }
27.         }
28.       }
29.     }
30.   },
31.   "scope": {
32.     "project": "TFG-GH-antoniosc7_Glassmatrix",
33.     "class": "TFG",
34.     "member": "*"
35.   },
36.   "window": {
37.     "type": "static",
38.     "period": "hourly",
39.     "initial": "2024-06-04T23:00:00.000Z",
40.     "from": "2024-06-04T23:00:00.000Z",
41.     "end": "2024-06-04T23:59:00.000Z",
42.     "timeZone": "America/Los_Angeles"
43.   }
}

```

```
44. }
45. }
46.
```

Una vez tenemos el contenido de la métrica ya podemos crearlo en la misma página de creación pulsando el botón “post”. Pero antes de eso, es recomendable crear en github ya sea manualmente, o a través de la página de Testing manual, lo que queremos que mida esta métrica. En este caso, he creado una Pull Request en la que he añadido líneas a un fichero.

Ahora sí, hay datos medibles, al pulsar el botón de “push”, se desbloqueará el botón de “Get Computation”.



Ilustración 109. Paso para obtener la computación de Bluejay

Esto tardará unos ratitos en interactuar con los microservicios de Bluejay, pero cuando acabe, devolverá los datos de esta métrica.

```
"project": "TFG-GH-antoniosc7_Glassmatrix",
"class": "TFG",
"member": "*"
},
>window": {
"type": "static",
"period": "hourly",
"initial": "2024-03-19T16:00:00.000Z",
"from": "2024-03-19T16:00:00.000Z",
"end": "2024-03-19T16:59:00.000Z",
```

The screenshot shows a web-based interface for managing metrics. At the top, there are two buttons: "Post" (green) and "Get Computation" (grey). Below these buttons is a search bar with the placeholder "Search in response...". To the right of the search bar is another button labeled "Search in response". The main area displays a JSON object representing the metric computation results. The JSON includes fields such as "project", "class", "member", "window" (with "type" set to "static" and "period" set to "hourly"), and "evidences" (an array containing detailed information about the metric's state, creation, and authorship). The JSON ends with a closing bracket "].

Ilustración 110. Resultados de la métrica.

Se puede ver, como devuelve que la pull request ha sido mergeada y se han añadido 2 líneas de código. Así que una vez que hemos comprobado que funciona, guardamos la métrica.

Si la métrica se ha guardado correctamente se podrá encontrar en la sección de Métricas Individuales.

File Name	View File	Execute / Edit File	Delete File
additions_metric.json			
METRICA_ADDITIONS.json			
METRICA_ADDITIONS3.json			
NPRAC.json			

Ilustración 111. Métrica guardada en sección de métricas individuales.

SCRIPT DE PRUEBA

Para poder probar bien la métrica, también está la página de *Automated Testing*. En esta página tendremos que crear un script con los pasos que se deberán realizar en Github para poder la métrica. Todos los pasos disponibles se encuentran en la derecha de la página y se pueden copiar dentro del cuadro de texto del script.

The screenshot shows the 'Automated Testing' interface. On the left, there's an 'Execution zone' with a red box around it and a downward arrow pointing to it from the text 'SCRIPT DE EJECUCIÓN'. Below it is a 'Save' button and a 'Set format' button. In the center, there's a large red box labeled 'RESULTADOS DE EJECUCIÓN' with a downward arrow pointing to it from the text 'PASOS DISPONIBLES'. To the right, there's a sidebar titled 'Examples' with several sections: 'Repository branches', 'Create branch', 'Delete branch', 'View issues', 'Create issues', and 'Test Results'. Each section contains examples of GitHub API calls. Red arrows point from the text 'PASOS DISPONIBLES' to the 'Examples' sidebar and from the text 'Resultados de los tests' to the 'Test Results' section. At the bottom of the sidebar, there are navigation buttons for 'Previous', '1', '2', '3', '4', '5', 'Next', and '>'.

Ilustración 112. Distribución de la página de Automated Testing

Por tanto, para probar la métrica que acabamos de crear tendrá que ejecutar los siguientes pasos:

1. Crear una nueva rama
2. Obtener las ramas actuales (paso opcional para comprobar que ha creado bien)
3. Crear un nuevo archivo con el contenido que queramos añadir
4. Hacer un commit
5. Realizar un push a la rama
6. Crear una Pull Request
7. Realizar el merge de esta pull request.
8. Ejecutar la métrica que creamos anteriormente
9. Hacer las pruebas sobre el resultado de esta métrica.

Y una vez se haya subido y probado, deberemos tener otro script (o hacerlo en el mismo) para eliminar estos cambios y devolver al repositorio a su estado original. Así que los pasos serían:

1. Eliminar archivo creado
2. Hacer un commit
3. Realizar un push
4. Crear una Pull Request
5. Cambiar a la rama main (para poder eliminar la rama que hemos creado anteriormente)
6. Mergear la Pull Request
7. Eliminar la rama que habíamos creado

Cuando ya tenemos los pasos decididos, podemos crear los dos scripts.

- Script 1: Creación de archivo

```

1. steps:
2.   - uses: "github/getBranches"
3.     with:
4.       repoName: "tp-testbench"
5.       method: "GET"
6.   - uses: "github/pullCurrentBranch"
7.     with:
8.       repoName: "tp-testbench"
9.       method: "GET"
10.  - uses: "github/createBranch"
11.    with:
12.      repoName: "tp-testbench"
13.      branchName: "ramatestToday223"
14.      method: "POST"
15.  - uses: "github/getBranches"
16.    with:
17.      repoName: "tp-testbench"
18.      method: "GET"
19.  - uses: "github/createFile"
20.    with:
21.      repoName: "tp-testbench"
22.      fileName: "nombre archivo"
23.      fileContent: "contenido fichero"
24.      method: "POST"
25.  - uses: "github/commitAllChanges"
26.    with:
27.      repoName: "tp-testbench"
28.      commitMessage: "cambios"
29.      method: "POST"
30.  - uses: "github/pushChanges"
31.    with:

```

```

32.      repoName: "tp-testbench"
33.      method: "POST"
34.      - uses: "github/createPR"
35.          with:
36.              repoName: "tp-testbench"
37.              owner: "Antoniosc7"
38.              title: "SUBIR CAMBIOS"
39.              head: "ramatestToday223"
40.              base: "main"
41.              body: "cuerpo"
42.              method: "POST"
43.      - uses: "github/mergeLastOpenPR"
44.          with:
45.              repoName: "tp-testbench"
46.              owner: "Antoniosc7"
47.              mergeMessage: "mergeado"
48.              method: "POST"
49.      - uses: "bluejay/compute/metric"
50.          with:
51.              collector: "EVENTS"
52.              actualTime: "true"
53.              metric: "METRICA_ADDITIONS.json"
54.              method: "POST"
55.      - uses: "bluejay/check"
56.          with:
57.              - key: "additions"
58.                  conditions:
59.                      minExpectedValue: "1"
60.              method: "TEST"
61.

```

Código 26. Script 1: Creación de fichero en el repositorio.

- Script 2: Eliminación de cambios

```

1. steps:
2.     - uses: "github/deleteFile"
3.         with:
4.             repoName: "tp-testbench"
5.             fileName: " nombre archivo "
6.             method: "DELETE"
7.     - uses: "github/commitAllChanges"
8.         with:
9.             repoName: "tp-testbench"
10.            commitMessage: "DELETE CHANGES"
11.            method: "POST"
12.     - uses: "github/pushChanges"
13.         with:
14.             repoName: "tp-testbench"
15.             method: "POST"
16.     - uses: "github/createPR"
17.         with:
18.             repoName: "tp-testbench"
19.             owner: "Antoniosc7"
20.             title: "ELIMINAR CAMBIOS"
21.             head: "ramatestToday223"
22.             base: "main"
23.             body: "body"
24.             method: "POST"
25.     - uses: "github/changeBranch"
26.         with:
27.             repoName: "tp-testbench"
28.             branchToChangeTo: "main"
29.             method: "PUT"
30.     - uses: "github/getBranches"
31.         with:
32.             repoName: "tp-testbench"
33.             method: "GET"
34.     - uses: "github/mergeLastOpenPR"

```

```

35.     with:
36.         repoName: "tp-testbench"
37.         owner: "Antoniosc7"
38.         mergeMessage: "mergeado"
39.         method: "POST"
40. - uses: "github/deleteBranch"
41.     with:
42.         repoName: "tp-testbench"
43.         branchName: "ramatestToday223"
44.         method: "DELETE"
45.

```

Código 27. Script 2: Borrado de archivo subido en Script 1

Y ahora sí, se pueden ejecutar los scripts. Al ejecutar el primer script, deberá devolver un resultado de los tests ya que hay puesto el siguiente paso:

```

1. - uses: "bluejay/check"
2.   with:
3.     - key: "additions"
4.       conditions:
5.         minExpectedValue: "1"
6.     method: "TEST"

```

Código 28. Sección de testing dentro del Script 1

El valor mínimo esperado es 1, por lo tanto, si ha funcionado todo bien (con el script creábamos un fichero que añadía 2 líneas), deberías salir que ha funcionado bien la ejecución.

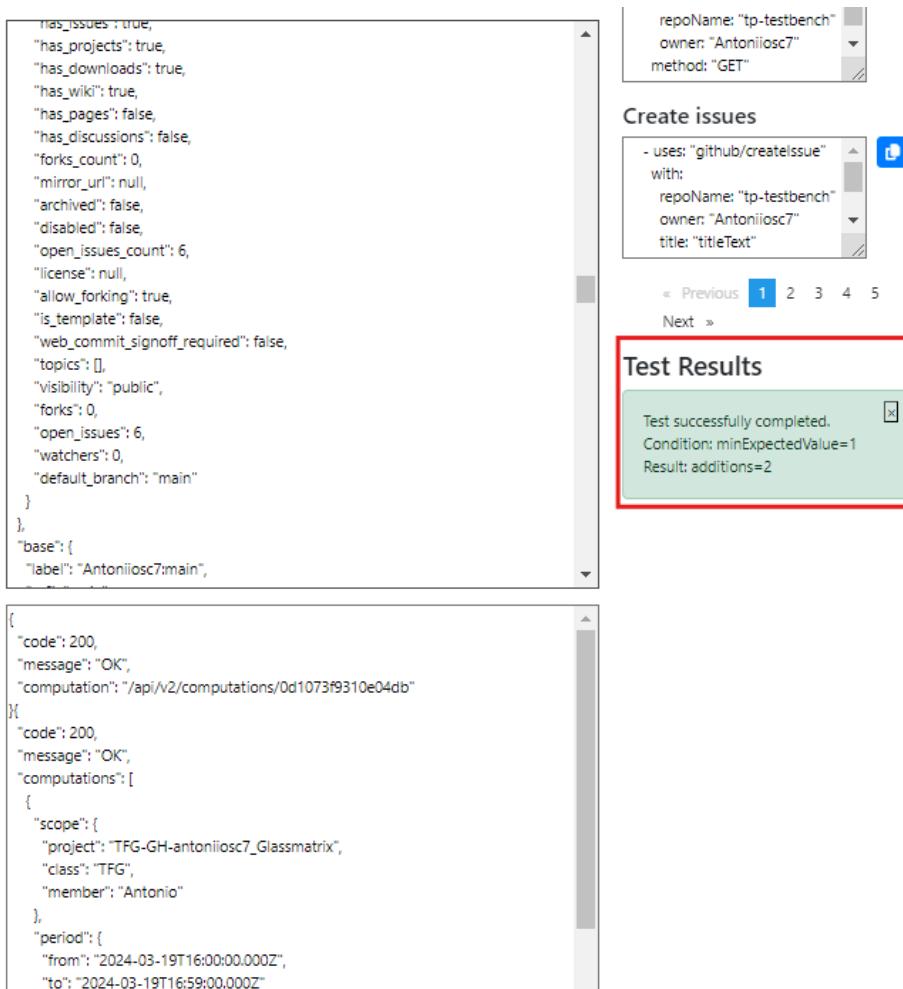


Ilustración 113. Resultados de la ejecución de la métrica.

Por lo tanto, todo ha funcionado como se esperaba y la métrica ya se podría integrar en un TPA. Pero antes de ello tendríamos que ejecutar script 2 para borrar los cambios que no queremos mantener en el repositorio.

INTEGRACIÓN AL TPA

Por último, cuando ya estamos seguro del correcto funcionamiento de la métrica se puede integrar dentro del TPA. Para ello, iremos a la página de TPA. Una vez aquí, seleccionaremos el TPA en el que queremos integrar la métrica recién probada.

Para ello entraremos en la página de edición de ese TPA.

Existing TPAs

ID	Project	Class	Options
tpa-TFG-GH-antoniosc7_tpPrueba2	TFG-GH-antoniosc7_tpPrueba2	TFG	
tpa-TFG-GH-antoniosc7_tpPrueba	TFG-GH-antoniosc7_tpPrueba	TFG	
tpa-TFG-GH-antoniosc7_tpPrueba	TFG-GH-antoniosc7_tpPrueba	TFG	
tpa-class01-GH-cs169_fa23-TPA-5	class01-GH-cs169_fa23-TPA-5	class01	
tpa-TFG-GH-JaviFdez7_ISPP-G1-Talent	TFG-GH-JaviFdez7_ISPP-G1-Talent	TFG	
tpa-TFG-GH-antoniosc7_Glassmatrix	TFG-GH-antoniosc7_Glassmatrix	TFG	

Ilustración 114. Entrada a pantalla de edición de TPA.

Una vez estemos en la página, tenemos un cuadro de texto con el contenido de TPA, en él deberemos añadir la métrica que hemos probado dentro de la sección "metrics". Si quisieramos ver una visualización en Grafana, también tendríamos que añadir la gráfica que queremos ver y su garantía. Y posteriormente actualizaremos el TPA.

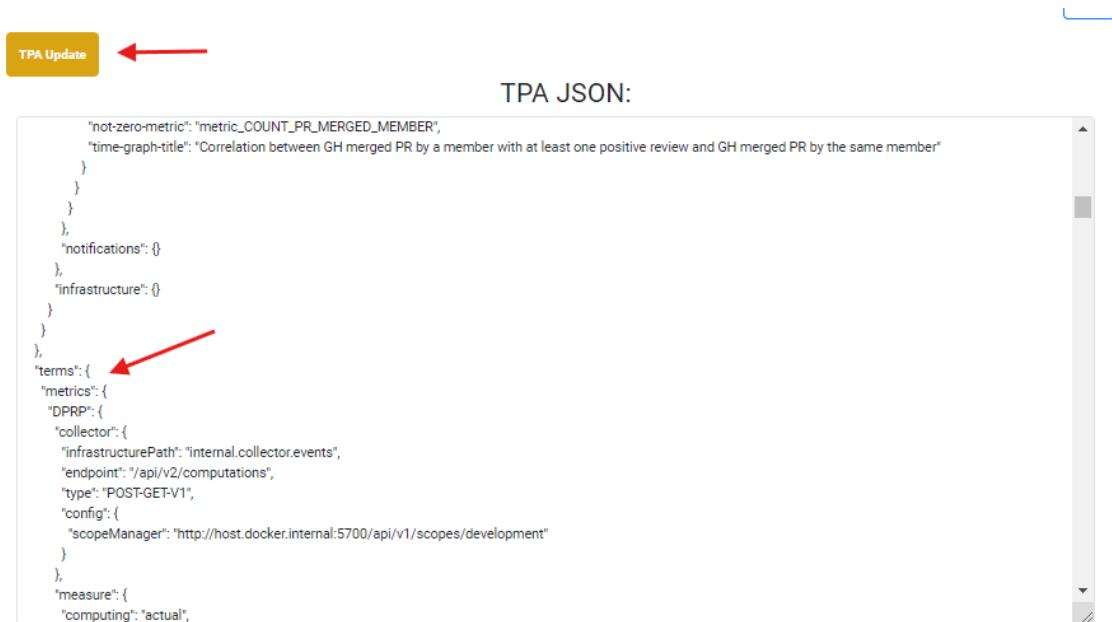


Ilustración 115. Actualización de TPA.

CAPITULO 8: CONCLUSIONES

8.1 ANÁLISIS DE PROBLEMAS

Durante el desarrollo del proyecto, se presentaron varios desafíos significativos. Uno de los problemas principales fue la necesidad de tener los microservicios de Bluejay funcionando localmente en mi equipo. Este requisito complicó el proceso debido a que, al inicio del proyecto, la documentación de Bluejay era insuficiente y poco clara, lo que dificultaba la configuración y el uso de estos servicios.

Por ello, para la puesta en marcha de todo el ecosistema de Bluejay tuve que requerir ayuda de técnicos de Bluejay. Afortunadamente, esta documentación ha mejorado con el tiempo, pero los primeros pasos fueron particularmente complejos debido a esta carencia.

Otro desafío considerable fue diseñar las funcionalidades de la página de Testing Automático de manera suficientemente genérica para que se adaptara a diversas necesidades de los usuarios. La intención era asegurar que la herramienta fuera flexible y útil para diferentes contextos y requerimientos, lo cual requirió un esfuerzo significativo en términos de diseño y desarrollo. Este proceso implicó iteraciones continuas y pruebas exhaustivas para garantizar que cada funcionalidad cumpliera con los estándares esperados.

Además, surgieron problemas relacionados con la integración de diversas tecnologías. Por ejemplo, hubo complicaciones con la integración actual que tiene Bluejay con Grafana. Ya que tras realizar el estudio inicial de las visualizaciones existentes en Grafana o aquellas que se pueden añadir con plugins ya creados se detectó que existían cumplían con las especificaciones necesarias para poder crear todas las gráficas existentes en los estudios previos.

Por tanto, se tuvo que realizar un estudio exhaustivo para desarrollar plugins personalizados que permitieran visualizar los datos de manera efectiva y que fueran compatibles con las demás herramientas utilizadas en el proyecto.

Este desarrollo adicional no solo retrasó el cronograma inicial, sino que también requirió una reestructuración de algunas partes del sistema para asegurar la compatibilidad y la funcionalidad óptima ya que a la hora de elaborar las visualizaciones del TPA conjunto creado en el proyecto nos tendríamos que adaptar a las visualizaciones existentes o crearlas mediante el uso de plugins personalizados.

Además, surgieron problemas relacionados con la integración de diversas tecnologías, la gestión eficiente del tiempo y los recursos, y la adaptación a cambios en los requerimientos del proyecto.

Un ejemplo de ello es que inicialmente, para probar las métricas estaba pensado mediante un Testing más manual, pero en el desarrollo del proyecto, entendimos que era más práctico la creación de scripts de casos de pruebas para lograr así la automatización

de estos tests. Por tanto, a través de un enfoque iterativo y colaborativo, se lograron superar los obstáculos, mejorando la calidad del producto final.

A pesar de estos desafíos, logramos superarlos y completar el proyecto con éxito. La experiencia adquirida y las soluciones implementadas han mejorado significativamente las funcionalidad del sistema, proporcionando una base sólida para futuros desarrollos y mejoras.

8.2 TRABAJO FUTURO

Existen varias áreas en las que el proyecto puede ampliarse y mejorarse en el futuro. Una de las mejoras más evidentes es la optimización de la interfaz de usuario, haciéndola más estilizada y fácil de usar. Se podría trabajar en el diseño para que sea más intuitivo y atractivo, mejorando así la experiencia del usuario.

Otra posible mejora sería la implementación de una funcionalidad que permita crear tests automáticos de manera más visual, por ejemplo, permitiendo a los usuarios arrastrar y soltar los "steps" que debe seguir el script en lugar de escribirlos manualmente. Esto no solo haría el proceso más accesible, sino que también reduciría la posibilidad de errores humanos.

Pero para determinar el trabajo futuro, será clave recibir llevar la herramienta a un entorno de producción para de esta manera poder recibir feedback de usuarios que prueben la herramienta. Ya que esta retroalimentación continua de los usuarios será esencial para guiar estas mejoras y asegurar que el sistema evolucione conforme a sus necesidades reales.

Adicionalmente, se podría explorar la incorporación de nuevas tecnologías que mejoren el rendimiento y la usabilidad del sistema. La automatización de más procesos dentro del flujo de trabajo también sería una dirección valiosa para aumentar la eficiencia.

8.3 CONCLUSIONES PERSONALES

Como resultado del proyecto, se ha creado una nueva herramienta, es decir, un nuevo microservicio que podrá ser integrado dentro del ecosistema de Bluejay, que proporciona una herramienta más para los usuarios con el objetivo de facilitar la creación de métricas.

La realización de este proyecto ha sido una experiencia altamente enriquecedora. Aplicar conocimientos teóricos en un entorno práctico y dinámico ha permitido un aprendizaje profundo y significativo. La colaboración en equipo y la gestión de proyectos han sido aspectos clave para el éxito del desarrollo, destacando la importancia de la comunicación y la coordinación.

Este proyecto también ha resaltado la importancia de la adaptabilidad y la resiliencia frente a los desafíos técnicos y organizativos. La necesidad de enfrentar y superar obstáculos constantes ha fortalecido habilidades críticas como la resolución de problemas y la toma de decisiones bajo presión.

Llevar a cabo este proyecto me ha permitido entender cómo opera un proyecto originado en un grupo de investigación e integrarme a los desarrollos de dicho grupo. Durante este proceso, mi aprendizaje y mis avances han sido aprovechados. Una vez que entendí la herramienta desarrollada, he podido implementar mejoras en el sistema existente, ampliando sus funcionalidades y atendiendo las necesidades que han surgido.

Además, este proyecto me ha brindado la oportunidad de colaborar con otras universidades para poder elaborar así un documento de buenas prácticas de equipo basado en las necesidades reales de un grupo de trabajo, lo cual ha enriquecido aún más mi experiencia y me ha permitido ver diferentes enfoques y metodologías.

En conclusión, este proyecto no solo ha contribuido a mi crecimiento profesional, sino que también ha proporcionado valiosas lecciones personales sobre la importancia de la planificación, la comunicación y la innovación. Las experiencias y conocimientos adquiridos durante este proceso serán fundamentales para mi desarrollo futuro en el campo de la tecnología y la gestión de proyectos.

CAPITULO 9: BIBLIOGRAFÍA Y ANEXOS

9.1 BIBLIOGRAFIA

- [1] "Governify. An agreement-based service governance framework," [Online]. Available: [https://www.softwareimpacts.com/article/S2665-9638\(24\)00017-4/fulltext](https://www.softwareimpacts.com/article/S2665-9638(24)00017-4/fulltext).
- [2] "Documentación Governify," [Online]. Available: <https://docs.governify.io/>.
- [3] "Bluejay: A Cross-Tooling Audit Framework For Agile Software Teams," [Online]. Available: <https://ieeexplore.ieee.org/document/9402165>.
- [4] "Documentación Bluejay," [Online]. Available: <https://docs.bluejay.governify.io/>.
- [5] "Influx DB," [Online]. Available: <https://docs.influxdata.com/influxdb/v2/reference/syntax/influxql/spec/>.
- [6] M. d. T. y. E. Social, "Resolución de 13 de julio de 2023, de la Dirección General de Trabajo, por la que se registra y publica el XVIII Convenio colectivo estatal de empresas de consultoría, tecnologías de la información y estudios de mercado y de la opinión pública.,," [Online]. Available: [https://www.boe.es/eli/es/res/2023/07/13/\(5\)](https://www.boe.es/eli/es/res/2023/07/13/(5)).
- [7] "Estudio de visualizaciones Visualization Universe," [Online]. Available: <https://visualizationuniverse.com/charts/>.
- [8] "Estudio de visualizaciones Data Viz Project," [Online]. Available: <https://datavizproject.com/>.
- [9] P. Vardhan, "Estudio de visualizaciones Gramener," [Online]. Available: <https://gramener.github.io/visual-vocabulary-vega/#/FullList/>.
- [10] "Documentación de Grafana: Creación de plugins," [Online]. Available: <https://grafana.com/developers/plugin-tools/>.
- [11] A. Saborido, "Repositorio Plugin personalizado en Grafana," [Online]. Available: <https://github.com/Antoniiosc7/TFG-Grafana>.
- [12] Google, "Informe DORA," [Online]. Available: https://cloud.google.com/architecture/devops#ways_to_measure_version_control.
- [13] "WebStorm," [Online]. Available: <https://www.jetbrains.com/webstorm/features/>.
- [14] "NodeJS," [Online]. Available: <https://nodejs.org/en/about>.
- [15] "NPM," [Online]. Available: <https://docs.npmjs.com/>.
- [16] "Docker," [Online]. Available: <https://docs.docker.com/>.
- [17] "Postman," [Online]. Available: <https://www.postman.com/product/tools/>.
- [18] "Github Desktop," [Online]. Available: <https://desktop.github.com/>.
- [19] "Toggle Track," [Online]. Available: <https://toggl.com/>.
- [20] "NGX Translate Repository," [Online]. Available: <https://github.com/nginx-translate/core>.
- [21] Electron, «Documentación de electron,» [En línea]. Available: <https://www.electronjs.org/es/docs/latest/>.
- [22] A. Saborido, "Repositorio de pruebas de funcionalidades," [Online]. Available: <https://github.com/Antoniiosc7/tp-testbench>.
- [23] "Github Graph QL Explorrer," [Online]. Available: <https://docs.github.com/en/graphql/overview/explorer>.

- [24] "OAS Tools," [Online]. Available: <https://oas-tools.github.io/docs/cli>.
- [25] A. Saborido, "TP Tester Repositorio," <https://github.com/Antoniiosc7/tp-tester>. [Online]. Available: <https://github.com/Antoniiosc7/tp-tester>.

9.2 ANEXOS

9.2.1 SOPORTE DE GRÁFICAS EN GRAFANA

Soporte de las gráficas de Visual Universe dentro de Grafana.

Visualization Universe	Description	Grafana Support
bar chart	A bar chart or bar graph is a chart or graph that presents grouped data with rectangular bars with lengths proportional to the values that they represent. The bars can be plotted vertically or horizontally.	native
gantt chart	Gantt chart is a chart in which a series of horizontal lines shows the amount of work done or production completed in certain periods of time in relation to the amount planned for those periods.	plugin
histogram	A histogram is a graphical representation of the distribution of numerical data. It is an estimate of the probability distribution of a continuous variable (quantitative variable) and was first introduced by Karl Pearson. It is a kind of bar graph.	native
pie chart	Pie chart is a type of graph in which a circle is divided into sectors that each represent a proportion of the whole.	native
venn diagram	A Venn diagram (also called primary diagram, set diagram or logic diagram) is a diagram representing mathematical or logical sets pictorially as circles or closed curves within an enclosing rectangle (the universal set), common elements of the sets being represented by the areas of overlap among the circles.	need custom plugin
time series	A series of values of a quantity obtained at successive times, often with equal intervals between them.	native
scatter plot	Scatter plot is a graph in which the values of two variables are plotted along two axes, the pattern of the resulting points revealing any correlation present.	plugin
box plot	A box plot is a graphical rendition of statistical data based on the minimum, first quartile, median, third quartile, and maximum. The term "box plot" comes from the fact that the graph looks like a rectangle with lines extending from the top and bottom.	need custom plugin
contour line	A contour line on a map for a function of two variables is a curve connecting points where the function has the same particular value	need custom plugin
heat map	A heat map (or heatmap) is a graphical representation of data where the individual values contained in a matrix are represented as colors.	native
decision tree	A decision tree is a decision support tool that uses a tree-like graph or model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm.	plugin
block diagram	A block diagram is a diagram of a system in which the principal parts or functions are represented by blocks connected by lines that show the relationships of the blocks. They are heavily used in engineering in hardware design, electronic design, software design, and process flow diagrams.	plugin
ishikawa diagram	A fishbone diagram, also called a cause and effect diagram or Ishikawa diagram, is a visualization tool for categorizing the potential causes of a problem in order to identify its root causes.	need custom plugin
candlestick chart	A candlestick chart (also called Japanese candlestick chart) is a style of financial chart used to describe price movements of a security, derivative, or currency. Each "candlestick" typically shows one day; so for example a one-month chart may show the 20 trading days as 20 "candlesticks".	native
word cloud	A tag cloud (word cloud, or weighted list in visual design) is a visual representation of text data, typically used to depict keyword metadata (tags) on websites, or to visualize free form text. Tags are usually single words, and the importance of each tag is shown with font size or color.	plugin
line chart	A line chart or line graph is a type of chart which displays information as a series of data points called 'markers' connected by straight line segments. It is a basic type of chart common in many fields.	native
bubble chart	A bubble chart is a type of chart that displays three dimensions of data. Each entity with its triplet (v1, v2, v3) of associated data is plotted as a disk that expresses two of the vi values through the disk's xy location and the third through its size.	plugin
sankey diagram	Sankey diagrams are a specific type of flow diagram, in which the width of the arrows is shown proportionally to the flow quantity.	plugin
radar chart	A radar chart is a graphical method of displaying multivariate data in the form of a two-dimensional chart of three or more quantitative variables represented on axes starting from the same point. The relative position and angle of the axes is typically uninformative.	plugin
waterfall chart	A waterfall chart is a form of data visualization that helps in understanding the cumulative effect of sequentially introduced positive or negative values. The waterfall chart is also known as a flying bricks chart or Mario chart due to the apparent suspension of columns (bricks) in mid-air.	plugin
tree diagram	Tree diagram is a diagram with a structure of branching connecting lines, representing different processes and relationships.	plugin
tree map	Treemaps display hierarchical (tree-structured) data as a set of nested rectangles. Each branch of the tree is given a rectangle, which is then tiled with smaller rectangles representing sub-branches. A leaf node's rectangle has an area proportional to a specified dimension of the data.	plugin
sparkline	A sparkline is a very small line chart, typically drawn without axes or coordinates. It presents the general shape of the variation (typically over time) in some measurement, such as temperature or stock market price, in a simple and highly condensed way.	native
self-organizing ma	SOMs are useful for visualizing low-dimensional views of high-dimensional data, akin to multidimensional scaling. The artificial neural network introduced by the Finnish professor Teuvo Kohonen in the 1980s is sometimes called a Kohonen map or network. The Kohonen net is a computationally convenient abstraction building on biological models of neural systems from the 1970s and morphogenesis models dating back to Alan Turing in the 1950s.	native
density plot	Also called Curved Bar Chart. In statistics, kernel density estimation (KDE) is a non-parametric way to estimate the probability density function of a random variable. Kernel density estimation is a fundamental data smoothing problem where inferences about the population are made, based on a finite data sample.	native
flow map	Flow maps in cartography are a mix of maps and flow charts, that "show the movement of objects from one location to another, such as the number of people in a migration, the amount of goods being traded, or the number of packets in a network".	need custom plugin
dendrogram	A dendrogram is a tree diagram frequently used to illustrate the arrangement of the clusters produced by hierarchical clustering. Dendograms are often used in computational biology to illustrate the clustering of genes or samples, sometimes on top of heatmaps.	need custom plugin
choropleth map	A choropleth map (from Greek χώρος ("area/region") + πληθωρά ("multitude")) is a thematic map in which areas are shaded or patterned in proportion to the measurement of the statistical variable being displayed on the map, such as population density or per-capita income.	native
cartogram	A cartogram is a map in which some thematic mapping variable – such as travel time, population, or Gross National Product – is substituted for land area or distance. The geometry or space of the map is distorted in order to convey the information of this alternate variable.	native
area chart	An area chart or area graph displays graphically quantitative data. It is based on the line chart. The area between axis and line are commonly emphasized with colors, textures and hatchings. Commonly one compares with an area chart two or more quantities.	need custom plugin
violin plot	A violin plot is a method of plotting numeric data. It is similar to box plot with a rotated kernel density plot on each side. The violin plot is similar to box plots, except that they also show the probability density of the data at different values (in the simplest case this could be a histogram).	need custom plugin
tape diagram	A tape diagram, also known as a bar model, is a pictorial representation of ratios. In mathematics education, it is used to solve word problems.	need custom plugin
doughnut chart	Doughnut chart, just like a pie chart, shows the relationship of parts to a whole, but a doughnut chart can contain more than one data series. Each data series that you plot in a doughnut chart adds a ring to the chart.	native
chord diagram	A chord diagram is a graphical method of displaying the inter-relationships between data in a matrix. The data is arranged radially around a circle with the relationships between the points typically drawn as arcs connecting the data together.	plugin
timeline	A timeline is a way of displaying a list of events in chronological order, sometimes described as a "project artifact". It is typically a graphic design showing a long bar labelled with dates alongside itself and usually events	native
burst chart/diagr	a sunburst diagram or sunburst chart is a multilevel pie chart used to represent the proportion of different values found at each level in a hierarchy.	plugin
stream graph	A streamgraph, or stream graph, is a type of stacked area graph which is displaced around a central axis, resulting in a flowing, organic shape.	need custom plugin
radial tree	A radial tree, or radial map, is a method of displaying a tree structure (e.g., a tree data structure) in a way that expands outwards, radially. It is one of many ways to visually display a tree, with examples extending back to the early 20th century.	need custom plugin
pyramid diagram	A pyramid diagram has the form of a triangle with lines dividing it into sections. A related topic or idea is placed in each section. Because of the triangular shape, each section is a different width from the others; this width indicates a level of hierarchy among the topics.	need custom plugin
proportional symbol n	The proportional symbol technique uses symbols of different sizes to represent data associated with different areas or locations within the map.	native
polar chart	The polar area diagram is similar to a usual pie chart, except sectors are equal angles and differ rather in how far each sector extends from the center of the circle. The polar area diagram is used to plot cyclic phenomena (e.g., count of deaths by month).	need custom plugin
pictorial chart	A record consisting of pictorial symbols, as a prehistoric cave drawing or a graph or chart with symbolic figures representing a certain number of people, cars, factories, etc.	need custom plugin
parallel coordinate	Parallel coordinates. Parallel coordinates are a common way of visualizing high-dimensional geometry and analyzing multivariate data. To show a set of points in an n-dimensional space, a backdrop is drawn consisting of n parallel lines, typically vertical and equally spaced.	need custom plugin

Visualization Universe	Description	Grafana Support
onion diagram	An onion diagram is a kind of chart that shows the dependencies among parts of an organization or process. The chart displays items in concentric circles, where the items in each ring depend on the items in the smaller rings	need custom plugin
marimekko chart	A Mekko chart (sometimes also called marimekko chart) is a two-dimensional stacked chart. In addition to the varying segment heights of a regular stacked chart, a Mekko chart also has varying column widths. Column widths are scaled such that the total width matches the desired chart width.	need custom plugin
kagi chart	The Kagi chart is a chart used for tracking price movements and to make decisions on purchasing stock. It differs from traditional stock charts such as the Candlestick chart by being mostly independent of time.	need custom plugin
hive plot	The hive plot is a perceptually uniform and scalable linear layout visualization for network visual analytics. Nodes are mapped to and positioned on radially distributed linear axes — this mapping is based on network structural properties. Edges are drawn as curved links. Simple and interpretable.	need custom plugin
funnel chart	Funnel charts are a type of chart, often used to represent stages in a sales process and show the amount of potential revenue for each stage. This type of chart can also be useful in identifying potential problem areas in an organization's sales processes. A funnel chart is similar to a stacked percent bar chart.	need custom plugin
dot distribution map	A dot distribution map, or dot density map, is a map type that uses a dot symbol to show the presence of a feature or phenomenon. Dot maps rely on a visual scatter to show spatial pattern.	native
dasymetric map	The dasymetric map is a method of thematic mapping, which uses areal symbols to spatially classify volumetric data.	native
chernoff face	Chernoff faces display multivariate data in the shape of a human face. The individual parts, such as eyes, ears, mouth and nose represent values of the variables by their shape, size, placement and orientation.	need custom plugin
bullet chart	A bullet graph is a variation of a bar graph developed by Stephen Few. Seemingly inspired by the traditional thermometer charts and progress bars found in many dashboards, the bullet graph serves as a replacement for dashboard gauges and meters.	native
arc diagram	An arc diagram is a style of graph drawing, in which the vertices of a graph are placed along a line in the Euclidean plane, with edges being drawn as semicircles in one of the two halfplanes bounded by the line, or as smooth curves formed by sequences of semicircles.	need custom plugin
alluvial diagram	Alluvial diagrams are a type of flow diagram originally developed to represent changes in network structure over time. In allusion to both their visual appearance and their emphasis on flow, alluvial diagrams are named after alluvial fans that are naturally formed by the soil deposited from streaming water.	plugin
stacked bar chart	Stacked Bar Chart is neither Multi-set Bar Chart nor simple Bar Chart. Stacked Bar Chart is multiple datasets on top of each other in order to show how the larger category is divided into the smaller categories and their relations to the total amount.	plugin
dot plot	A dot plot or dot chart is a statistical chart consisting of data points plotted on a fairly simple scale, typically using filled in circles. There are two common versions of the dot chart.	native
slope	The line chart's fraternal twin. Line charts display three or more points in time while slope charts display exactly two points in time.	plugin
network	Network Visualisation (also called Network Graph) is often used to visualise complex relationships between a huge amount of elements.	native
Lollipop	Lollipop charts draw more attention to the data value than standard bar/column and can also show rank effectively	need custom plugin
Bump	Effective for showing changing rankings across multiple dates. For large datasets, consider grouping lines using colour.	plugin

9.2.2 SOPORTE DE GRAFICAS POR VISUAL VOCABULARY

Soporte de graficas en grafana por familias de Visual Vocabulary.

Visualization Universe	Grafana Support	Visual Vocabulary								
		Deviation	Correlation	Ranking	Distribution	Change-over-Time	Magnitude	Part-to-whole	Spatial	Flow
bar chart	native					x				
gantt chart	plugin				x	x				
histogram	native			x						
pie chart	native						x			
venn diagram	need custom plugin							x		
time series	native				x					
scatter plot	plugin	x			x					
box plot	need custom plugin				x					
contour line	need custom plugin	-	-	-	-	-	-	-	-	-
heat map	native		x							
decision tree	plugin	-	-	-	-	-	-	-	-	-
block diagram	plugin	-	-	-	-	-	-	-	-	-
ishikawa diagram	need custom plugin	-	-	-	-	-	-	-	-	-
candlestick chart	native				x					
word cloud	plugin	-	-	-	-	-	-	-	-	-
line chart	native				x					
bubble chart	plugin	x								
sankey diagram	plugin					x			x	
radar chart	plugin					x				
waterfall chart	plugin						x		x	
tree diagram	plugin							x		x
tree map	plugin						x			
sparkline	native				x			x		
elf-organizing ma	native					x			x	
density plot	native			x					x	
flow map	need custom plugin								x	
dendrogram	need custom plugin	-	-	-	-	-	-	-	-	-
choropleth map	native							x		
cartogram	native							x		
area chart	need custom plugin				x					
violin plot	need custom plugin			x						
tape diagram	need custom plugin					x				
doughnut chart	native						x		x	
chord diagram	plugin							x		x
timeline	native				x					
burst chart/diagr	plugin						x		x	
stream graph	need custom plugin				x					
radial tree	need custom plugin	-	-	-	-	-	-	-	-	-
pyramid diagram	need custom plugin			x						
proportional symbol	native					x		x		
polar chart	need custom plugin						x		x	
pictorial chart	need custom plugin					x				
parallel coordinate	need custom plugin					x				
onion diagram	need custom plugin	-	-	-	--	-	-	--	-	-
marimekko chart	need custom plugin							x		
kagi chart	need custom plugin	-	-	-	-	-	-	-	-	-
hive plot	need custom plugin	-	-	-	-	-	-	-	-	-
funnel chart	need custom plugin	-	-	-	-	-	-	-	-	-
ot distribution ma	native								x	
dasymetric map	native							x		
chernoff face	need custom plugin	-	-	-	-	-	--	-	-	-
bullet chart	native					x				
arc diagram	need custom plugin						x			
alluvial diagram	plugin									x
stacked bar char	plugin						x			
dot plot	native		x	x						
slope	plugin		x		x					
network	native									x
Lollipop	need custom plugin		x			x				

9.2.3 SOPORTE DE GRÁFICAS POR FUNCIONES DATAVIZ PROJECT

Soporte de graficas en grafana por funciones de Dataviz Project.

Visualization Universe	Grafana Support	Input Type	Dataviz Project						
			Comparison	Concept-visualization	Correlation	Distribution	Geolocation	Part-to-whole	Trend-over-time
bar chart	native	5	x						x
gantt chart	plugin	9				x			
histogram	native	14				x			
pie chart	native	3	x						
venn diagram	need custom plugin	-			x				
time series	native	11							x
scatter plot	plugin	7			x	x			
box plot	need custom plugin	32	x			x			
contour line	need custom plugin	-	-	-	-	-	-	-	-
heat map	native	34	x		x	x			x
decision tree	plugin	-	-	-	-	-	-	-	-
block diagram	plugin	-	-	-	-	-	-	-	-
ishikawa diagram	need custom plugin	-	-	-	-	-	-	-	-
candlestick chart	native	24	x						x
word cloud	plugin	5	x			x			
line chart	native	14				x			x
bubble chart	plugin	18	x			x			x
sankey diagram	plugin	-	x		x				
radar chart	plugin	31	x			x			
waterfall chart	plugin	20	x			x			
tree diagram	plugin	-	-	-	-	-	-	-	-
tree map	plugin	21	x			x			
sparkline	native	32							x
elf-organizing map	native	-					x		
density plot	native	5	x						
flow map	need custom plugin	-	x		x	x	x		
dendrogram	need custom plugin	-			x				
choropleth map	native	16	x			x	x		
cartogram	native	16	x				x		
area chart	need custom plugin	35				x			x
violin plot	need custom plugin	32	x			x			
tape diagram	need custom plugin	13						x	
doughnut chart	native	3	x					x	
chord diagram	plugin	29	x		x				
timeline	native	15							x
burst chart/diagram	plugin	31	x						
stream graph	need custom plugin	11	x			x			x
radial tree	need custom plugin	-	-	-	-	-	-	-	-
pyramid diagram	need custom plugin	1						x	
proportional symbol map	native	16				x	x		
polar chart	need custom plugin	23	x						x
pictorial chart	need custom plugin	13						x	
parallel coordinate	need custom plugin	19	x						
onion diagram	need custom plugin	-	-	-	-	-	-	-	-
marimekko chart	need custom plugin	10	x					x	
kagi chart	need custom plugin	25	x		x				
hive plot	need custom plugin	-	x		x	x			
funnel chart	need custom plugin	3						x	x
dot distribution map	native	-	-	-	-	-	-	-	-
dasymetric map	native	-	-	-	-	-	-	-	-
chernoff face	need custom plugin	-	-	-	-	-	-	-	-
bullet chart	native	13						x	
arc diagram	need custom plugin	14	x		x				
alluvial diagram	plugin	8	x		x	x			x
stacked bar chart	plugin	19	x					x	
dot plot	native	5	x			x			
slope	plugin	6	x						x

Visualization Universe	Grafana Support	Input Type	Dataviz Project						
			Function						
			Comparison	Concept-visualization	Correlation	Distribution	Geolocation	Part-to-whole	Trend-over-time
network	native	29		X	X				
Lollipop	need custom plugin	5	X						X
Bump	plugin	12	X						X

9.2.4 SOPORTE DE GRÁFICAS POR FAMILIAS DATAVIZ PROJECT

Soporte de graficas en grafana por familias de Dataviz Project.

Visualization Universe	Grafana Support	Input Type	Dataviz Project				
			Family				
			Chart	Diagram	Geospatial	Plot	Table
bar chart	native	5	x				
gantt chart	plugin	9	x				
histogram	native	14	x				
pie chart	native	3	x				
venn diagram	need custom plugin	-		x			
time series	native	11				x	
scatter plot	plugin	7				x	
box plot	need custom plugin	32				x	
contour line	need custom plugin	-	-	-	-	-	-
heat map	native	34	x				x
decision tree	plugin	-	-	-	-	-	-
block diagram	plugin	-	-	-	-	-	-
ishikawa diagram	need custom plugin	-	-	-	-	-	-
candlestick chart	native	24	x				
word cloud	plugin	5	x				
line chart	native	14				x	
bubble chart	plugin	18	x			x	
sankey diagram	plugin	-		x			
radar chart	plugin	31	x				
waterfall chart	plugin	20	x				
tree diagram	plugin	-	-	-	-	-	-
tree map	plugin	21	x				
sparkline	native	32				x	
elf-organizing ma	native	-			x		
density plot	native	5	x				
flow map	need custom plugin	-		x	x		
dendrogram	need custom plugin	-		x			
choropleth map	native	16			x		
cartogram	native	16			x		
area chart	need custom plugin	35				x	
violin plot	need custom plugin	32				x	
tape diagram	need custom plugin	13	x				
doughnut chart	native	3	x				
chord diagram	plugin	29	x	x			
timeline	native	15	x				
burst chart/diagr	plugin	31	x	x			
stream graph	need custom plugin	11	x				
radial tree	need custom plugin	-	-	-	-	-	-
pyramid diagram	need custom plugin	1	x				
proportional symbol	native	16			x		
polar chart	need custom plugin	23	x				
pictorial chart	need custom plugin	13	x				

Visualization Universe	Grafana Support	Dataviz Project					
		Input Type	Family				
			Chart	Diagram	Geospatial	Plot	Table
parallel coordinate	need custom plugin	19	x				
onion diagram	need custom plugin	-	-	-	-	-	-
marimekko chart	need custom plugin	10	x				
kagi chart	need custom plugin	25	x				
hive plot	need custom plugin	-				x	
funnel chart	need custom plugin	3	x				
dot distribution map	native	-	-	-	-	-	-
dasymetric map	native	-	-	-	-	-	-
chernoff face	need custom plugin	-	-	-	-	-	-
bullet chart	native	13	x			x	
arc diagram	need custom plugin	14		x			
alluvial diagram	plugin	8		x			
stacked bar chart	plugin	19	x				
dot plot	native	5	x				
slope	plugin	6	x				
network	native	29		x			
Lollipop	need custom plugin	5	x				
Bump	plugin	12	x			x	

9.2.5 EJEMPLO TPA

```
{  
    "id": "tpa-1010101010",  
    "version": "1.1.0",  
    "type": "agreement",  
    "context": {  
        "validity": {  
            "initial": "2022-01-01",  
            "timeZone": "America/Los_Angeles"  
        },  
        "definitions": {  
            "schemas": {},  
            "scopes": {  
                "development": {  
                    "project": {  
                        "name": "Project",  
                        "description": "Project",  
                        "type": "string",  
                        "default": "1010101010"  
                    },  
                    "class": {  
                        "name": "Class",  
                        "description": "Group some Projects",  
                        "type": "string",  
                        "default": "2020202020"  
                    },  
                    "member": {  
                        "name": "Member",  
                        "description": "Member",  
                        "type": "string",  
                        "default": "*"  
                    }  
                }  
            },  
            "collectors": {  
                "eventcollector": {  
                    "infrastructurePath": "internal.collector.events",  
                    "endpoint": "/api/v2/computations",  
                    "type": "POST-GET-V1",  
                    "config": {  
                        "scopeManager":  
                            "$_[infrastructure.internal.scopes.default]/api/v1/scopes/development"  
                    }  
                }  
            },  
            "dashboards": {  
                "main": {  
                    "base": "$_[infrastructure.internal.assets.default]/api/v1/public/grafana-  
dashboards/tpa/base.json",  
                    "modifier": "$_[infrastructure.internal.assets.default]/api/v1/public/grafana-  
dashboards/tpa/modifier.js",  
                    "overlay": "$_[infrastructure.internal.assets.default]/api/v1/public/grafana-  
dashboards/tpa/overlay.js",  
                    "config": {  
                        "configDashboard": true,  
                        "blocks": {  
                            "0": {  
                                "type": "divider-github-states",  
                                "config": {}  
                            }  
                        }  
                    }  
                }  
            }  
        }  
    }  
}
```

```

        "title": "CS105 2023 Dashboard"
    }
},
"1": {
    "type": "time-graph2-member",
    "guarantee": "NUMBER_INPROGRESSISSUES_HOURLY_UNDER_1",
    "config": {
        "scope-class": "2020202020",
        "time-graph-title": "Number of issues in the 'In Progress' column"
    }
}
},
"notifications": {}
},
"infrastructure": {}
}
},
"terms": {
    "metrics": {
        "COUNT_INPROGRESSISSUES_MEMBER": {
            "collector": {
                "infrastructurePath": "internal.collector.events",
                "endpoint": "/api/v2/computations",
                "type": "POST-GET-V1",
                "config": {
                    "scopeManager": "$_[infrastructure.internal.scopes.default]/api/v1/scopes/development"
                }
            },
            "measure": {
                "computing": "actual",
                "element": "number",
                "event": {
                    "githubGQL": {
                        "custom": {
                            "type": "graphQL",
                            "title": "Get issues in progress",
                            "steps": {
                                "0": {
                                    "type": "queryGetObject",
                                    "query": "{repository(name: \"%PROJECT.github.repository%\", owner: \"%PROJECT.github.repoOwner%\") {\r\n    projectsV2(first: 5) {\r\n        nodes {\r\n            items(first: 100) {\r\n                nodes {\r\n                    content {\r\n                        ... on Issue {\r\n                            bodyText\r\n                            updatedAt\r\n                            number\r\n                            author {\r\n                                login\r\n                            }\r\n                            assignees(first: 5) {\r\n                                nodes {\r\n                                    login\r\n                                }\r\n                            }\r\n                            fieldValues(first: 100) {\r\n                                nodes {\r\n                                    ... on ProjectV2Field {\r\n                                        ProjectV2ItemFieldValue {\r\n                                            field {\r\n                                                ... on ProjectV2Field {\r\n                                                    name\r\n                                                    field {\r\n                                                        ... on ProjectV2Field {\r\n                                                            nameWithOwner\r\n                                                        }\r\n                                                        ... on ProjectV2ItemFieldRepositoryValue {\r\n                                                            field {\r\n                                                                ... on ProjectV2Field {\r\n                                                                    name\r\n                                                                    field {\r\n                                                                        ... on ProjectV2Field {\r\n                                                                            nameWithOwner\r\n                                                                            }\r\n                                                                            ... on ProjectV2ItemFieldTextValue {\r\n                                                                                text\r\n                                                                                field {\r\n                                                                                    ... on ProjectV2Field {\r\n                                                                                        name\r\n                                                                                        field {\r\n                                                                                            ... on ProjectV2Field {\r\n                                                                                                name\r\n                                                                                                field {\r\n                                                                ... on ProjectV2ItemFieldMilestoneValue {\r\n                                                                ... on ProjectV2Field {\r\n                                                                ... on ProjectV2ItemFieldSingleSelectValue {\r\n                                                                ... on ProjectV2SingleSelectField {\r\n................................................................
```
```

```

{\r\n name\r\n } \r\n } \r\n "cache": true
 },
 "1": {
 "type": "objectGetSubObjects",
 "location": "data.repository.projectsV2.nodes.0.items.nodes"
 },
 "2": {
 "type": "objectsFilterObjects",
 "filters": [
 "content.assignees.nodes.*any*.login == '%MEMBER.github.username%'"
]
 },
 "3": {
 "type": "runScript",
 "variables": {},
 "script": "module.exports.generic = function getFieldValues(inputData,\nvariables) {\r\n let result = [];\r\n for (const issue of inputData) {\r\n for (const fieldValue of\nissue.fieldValues.nodes) {\r\n if (fieldValue.name === 'In Progress') {\r\n result.push(issue);\r\n }
 }
 }
},
"scope": {
 "project": {
 "name": "Project",
 "description": "Project",
 "type": "string",
 "default": "1010101010"
 },
 "class": {
 "name": "Class",
 "description": "Group some Projects",
 "type": "string",
 "default": "2020202020"
 }
}
},
"guarantees": [
{
 "id": "NUMBER_INPROGRESSISSUES_HOURLY_UNDER_1",
 "notes": "#### Description\r\n```\r\nTP-1: At most 1 in progress issue every hour.\r\n```\r\n"description": "A developer should only be working on one issue at a time",
 "scope": {
 "project": {
 "name": "Project",
 "description": "Project",
 "type": "string",
 "default": "1010101010"
 },
 "class": {
 "name": "Class",
 "description": "Group some Projects",
 "type": "string",
 "default": "2020202020"
 }
 }
}
]
}

```

```
 },
 "member": "*"
 },
 "of": [
 {
 "scope": {
 "project": "1010101010",
 "member": "*"
 },
 "objective": "COUNT_INPROGRESSISSUES_MEMBER <= 1",
 "with": {
 "COUNT_INPROGRESSISSUES_MEMBER": {}
 },
 "window": {
 "type": "static",
 "period": "hourly",
 "initial": "2022-01-01"
 }
 }
]
}
```

## 9.2.6 EJEMPLO TPA CON FORMATO BLUEJAY

| TP   | Métrica | Métrica Bluejay/ Github                                                                                                                                                                                                                                                  | Garantía                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|------|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CTP1 | NPRM(T) | <pre> "metric": {   "computing": "number",   "element": "number",   "event": {     "pull_request": {       "changes_percentage": {         "operator": "less_than",         "value": 10       }     }   } } </pre>                                                       | <p>"NPRM_OVER_T_NPRT_LESS_THAN_UTP1": {<br/>     "id": "NPRM_OVER_T_NPRT_LESS_THAN_UTP1",<br/>     "notes": "### Descripción\n\nEvitar pull request de tamaño excesivo.\n\nITP-1: Asegurar que el porcentaje de pull request que cumplen con el umbral de cambio sea menor al umbral establecido.",<br/>     "scope": "#context/definitions/scopes/development",<br/>     "Sref": "#context/definitions/scopes/development",<br/>     "of": [{}],<br/>     "scope": {       "project": "1010101010"     },<br/>     "objective": "(NPRT / NPRT) * 100 &gt;= 90",<br/>     "with": {       "NPRT": 0,       "NPRT": 0,       "UTP1": 90     },     "window": {       "type": "static",       "period": "weekly"     }   } }</p>                                                                                                                                                                                    |
|      | NPR(T)  | <pre> "metric": {   "computing": "number",   "element": "number",   "event": {     "pull_request": {}   } } </pre>                                                                                                                                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| ITP2 | NPRM(I) | <pre> "metric": {   "computing": "number",   "element": "number",   "event": {     "pull_request": {       "changes_percentage": {         "operator": "less_than",         "value": 10       }     }   } } </pre>                                                       | <p>"NPRM_OVER_T_NPRI_100_LESS_THAN_UTP2": {<br/>     "id": "NPRM_OVER_T_NPRI_100_LESS_THAN_UTP2",<br/>     "notes": "### Descripción\n\nEvitar pull request de tamaño excesivo.\n\nITP-2: Asegurar que el porcentaje de pull request que cumplen con el umbral de cambio establecido sea mayor o igual al umbral especificado.",<br/>     "description": "Asegurar que el porcentaje de pull request que cumplen con el umbral de cambio establecido sea mayor o igual al umbral especificado.",<br/>     "scope": "#context/definitions/scopes/member",<br/>     "Sref": "#context/definitions/scopes/member",<br/>     "of": [{}],<br/>     "scope": {       "project": "1010101010"     },<br/>     "objective": "(NPRI / NPRI) * 100 &gt;= 90",<br/>     "with": {       "NPRI": 0,       "NPRI": 0,       "UTP2": 90     },     "window": {       "type": "static",       "period": "weekly"     }   } }</p> |
|      | NPRI(I) | <pre> "metric": {   "computing": "number",   "element": "number",   "event": {     "pull_request": {}   } } </pre>                                                                                                                                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| ITP3 | NRA     | <pre> "metric": {   "computing": "actual",   "element": "number",   "event": {     "github": {       "pullRequest": {         "state": ["open", "closed"]       }     }   } } </pre>                                                                                     | <p>"75_PERCENT_NRAI_NRAI_NRAI_BIND": {<br/>     "id": "75_PERCENT_NRAI_NRAI_NRAI_BIND",<br/>     "notes": "### Descripción\n\nEvitar ramas activas sin solicitudes de extracción.\n\nITP-1: Al menos el 75% de las ramas activas deben tener solicitudes de extracción asociadas.",<br/>     "description": "Al menos el 75% de las ramas activas deben tener solicitudes de extracción asociadas.",<br/>     "scope": "#context/definitions/scopes/development",<br/>     "Sref": "#context/definitions/scopes/development",<br/>     "of": [{}],<br/>     "scope": {       "project": "1010101010"     },<br/>     "objective": "NRATI / NRATI &gt;= 0.75",<br/>     "with": {       "NRATI": 0,       "NRATI": 0     },     "window": {       "type": "static",       "period": "sprint",       "initial": "2018-01-01"     }   } }</p>                                                                        |
|      | NRAT    | <pre> "metric": {   "computing": "actual",   "element": "number",   "event": {     "github": {       "activeBranch": true     }   } } </pre>                                                                                                                             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| ITP4 | NPRMI   | <pre> "metric": {   "computing": "actual",   "element": "number",   "event": {     "github": {       "pullRequest": {         "complexity": {           "type": "CCM",           "operator": "&lt;",           "threshold": "%CCMMTP4%"         }       }     } } </pre> | <p>"75_PERCENT_NPRMI_NPRI_NPRMI_BIND": {<br/>     "id": "75_PERCENT_NPRMI_NPRI_NPRMI_BIND",<br/>     "notes": "### Descripción\n\nEvitar high complexity in Pull Requests.\n\nITP-1: At least 75% of Pull Requests must have a complexity index lower than the threshold.",<br/>     "description": "At least 75% of Pull Requests must have a complexity index lower than the threshold.",<br/>     "scope": "#context/definitions/scopes/development",<br/>     "Sref": "#context/definitions/scopes/development",<br/>     "of": [{}],<br/>     "scope": {       "project": "1010101010"     },<br/>     "objective": "NPRI / NPRI &gt;= 0.75",<br/>     "with": {       "NPRI": 0,       "NPRI": 0     },     "window": {       "type": "static",       "period": "weekly",       "initial": "2018-01-01"     }   } }</p>                                                                                     |
|      | NPRI    | <pre> "metric": {   "computing": "actual",   "element": "number",   "event": {     "github": {       "pullRequest": {}     } } </pre>                                                                                                                                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| CTP5 | NM      | <pre> "metric": {   "computing": "number",   "element": "number",   "event": {     "members": {}   } } </pre>                                                                                                                                                            | <p>"NRA_LESS_THAN_OR_EQUAL_TO_NM_PLUS_ONE": {<br/>     "id": "NRA_LESS_THAN_OR_EQUAL_TO_NM_PLUS_ONE",<br/>     "description": "Gestionar eficientemente el número de ramas activas en el repositorio es crucial para mantener un entorno de desarrollo ordenado. Esta ITP se centra en el número de ramas activas y establece un límite basado en el tamaño del equipo.",<br/>     "scope": "#context/definitions/scopes/team",<br/>     "Sref": "#context/definitions/scopes/team",<br/>     "of": [{}],<br/>     "scope": {       "project": "1010101010"     },<br/>     "objective": "NRA &lt;= (NM + 1)",<br/>     "with": {       "NRA": 0,       "NM": 0     },     "window": {       "type": "static",       "period": "weekly"     }   } }</p>                                                                                                                                                           |
|      | NRA     | <pre> "metric": {   "computing": "number",   "element": "number",   "event": {     "active_branches": {}   } } </pre>                                                                                                                                                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |

|       |       |                                                                                                                                |                                                                                                                                                                                                                                                                                                                                                                                                                        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|-------|-------|--------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CTP6  | NPRAC | Número de P/R abiertas que cumplen que han recibido el primer comentario en menos de TLTP6 (contado desde que se abre la P/R). | <pre>"metric": {   "computing": "count",   "event": {     "comments": {       "first_comment": {         "duration": "less_than_2_hours"       }     },     "pull_requests": {       "opened": 0     }   } }  "metric": {   "computing": "count",   "element": "pull_request",   "event": {     "pull_requests": {       "opened": 0     }   } }</pre>                                                                 | <p>"NPRAC_NPRA_LESS_THAN_UTP6": {<br/>hashtag: "Este CTP se centra en medir el tiempo desde que se abre la P/R hasta que se produce un primer comentario derivado de la revisión de la P/R."}<br/>"id": "NPRAC_NPRA_LESS_THAN_UTP6",<br/>"objective": "NPRAC / NPRA * 100 &lt; UTP6",<br/>"scope": {<br/>  "team": {}<br/>},<br/>"window": {<br/>  "period": "weekly"<br/>},<br/>"with": {<br/>  "NPRA": {},<br/>  "NPRAC": {} }</p>                                                                                                                                                                                                                                                                                                                                                                                  |
|       | NPRA  | Número total de P/R abiertas                                                                                                   |                                                                                                                                                                                                                                                                                                                                                                                                                        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| CTP7  | NSF   | Número de solicitudes de fusión que se abren desde una rama que han tenido al menos NPTP7 pushes previos al pull request       | <pre>"metric": {   "computing": "count",   "event": {     "pull_requests": {       "opened": {         "from_branch_with_pushes": {           "at_least": 2         }       }     }   } }  "metric": {   "computing": "count",   "element": "pull_request",   "event": {     "pull_requests": {       "opened": 0     }   } }</pre>                                                                                    | <p>"NSF_NST_LESS_THAN_UTP7": {<br/>descripción: "Esta CTP se centra en medir el porcentaje de solicitudes fusionadas sin recibir ningún push."}<br/>"id": "NSF_NST_LESS_THAN_UTP7",<br/>"objective": "NSF / NST * 100 &lt; UTP7",<br/>"scope": {<br/>  "team": {}<br/>},<br/>"window": {<br/>  "period": "weekly"<br/>},<br/>"with": {<br/>  "NST": {},<br/>  "NSF": {} }</p>                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|       | NST   | Número total de solicitudes de fusión                                                                                          |                                                                                                                                                                                                                                                                                                                                                                                                                        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| CTP8  | NRAP  | Número de ramas "activas" con P/R que han tenido más de NPTP8 Push antes del P/R                                               | <pre>"metric": {   "computing": "count",   "event": {     "pull_requests": {       "opened": {         "from_branch_with_pushes": {           "more_than": 2         }       }     }   } }  "metric": {   "computing": "count",   "element": "pull_request",   "event": {     "pull_requests": {       "opened": 0     }   } }</pre>                                                                                   | <p>"NRAP_NRA_GREATER_THAN_UTP8": {<br/>descripción: "Esta CTP se centra en el número 'pushes' antes de una Pull Request y establece un objetivo para garantizar la ejecución de pruebas."}<br/>"id": "NRAP_NRA_GREATER_THAN_UTP8",<br/>"objective": "NRAP / NRA * 100 &gt; UTP8",<br/>"scope": {<br/>  "team": {}<br/>},<br/>"window": {<br/>  "period": "weekly"<br/>},<br/>"with": {<br/>  "NRA": {},<br/>  "NRAP": {} }</p>                                                                                                                                                                                                                                                                                                                                                                                        |
|       | NRA   | Número de ramas "activas" con P/R.                                                                                             |                                                                                                                                                                                                                                                                                                                                                                                                                        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| CTP9  | NPRR  | Número de P/R rechazadas en la rama principal                                                                                  | <pre>"metric": {   "computing": "actual",   "element": "number",   "event": {     "github": {       "pullRequest": {         "state": "rejected",         "baseBranch": "%GITHUB.REPO_OWNER%:master"       }     }   } }  "metric": {   "computing": "actual",   "element": "number",   "event": {     "github": {       "pullRequest": {         "baseBranch": "%GITHUB.REPO_OWNER%:master"       }     }   } }</pre> | <p>"75_PERCENT_NPRR_NPR_NPR_BIND": {<br/>"id": "75_PERCENT_NPRR_NPR_NPR_BIND",<br/>"notes": "### Description\n\n\"\\n\"\\nCTP9: Avoid a high number of rejected Pull Requests \\n\\nIn\\nTP-1: At least 75% of Pull Requests must be successfully merged into the master branch.",<br/>"description": "At least 75% of Pull Requests must be successfully merged into the master branch.",<br/>"scope": {<br/>  "Smt": "#/context/definitions/scopes/development" },<br/>"of": [<br/>  {     "scope": {<br/>      "project": "1010101010"     },<br/>    "objective": "NPRR / NPR * 100 &lt; 25",<br/>    "with": {<br/>      "NPRR": {},<br/>      "NPR": {}     },<br/>    "window": {<br/>      "type": "static",<br/>      "period": "weekly"     } ] }</p>                                                       |
|       | NPR   | Número total de pull request en la rama principal                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| CTP10 | NPR   | (Number of PR): Número de pull request realizados en la rama principal.                                                        | <pre>"metric": {   "computing": "actual",   "element": "number",   "event": {     "github": {       "pullRequest": {         "baseBranch": "%GITHUB.REPO_OWNER%:master"       }     }   } },</pre>                                                                                                                                                                                                                     | <p>"NPRW_PERCENT_NPR_NPRW_NPRW_BIND": {<br/>"id": "NPRW_PERCENT_NPR_NPRW_NPRW_BIND",<br/>"notes": "### Description\n\n\"\\n\"\\nCTP10: Avoid Pull Requests being without a reviewer for too long.\n\\n\\nIn\\nTP-1: Ensure that the percentage of Pull Requests without a reviewer is within the specified threshold.",<br/>"description": "Ensure that the percentage of Pull Requests without a reviewer is within the specified threshold.",<br/>"scope": {<br/>  "Smt": "#/context/definitions/scopes/development" },<br/>"of": [<br/>  {     "scope": {<br/>      "project": "1010101010"     },<br/>    "objective": "NPRW / NPR * 100 &lt;= 10",<br/>    "with": {<br/>      "NPRW": {},<br/>      "NPR": {}     },<br/>    "window": {<br/>      "type": "static",<br/>      "period": "weekly"     } ] }</p> |
|       | NPRW  | Number of PR without reviewer): Número de pull request que estuvieron sin revisor los primeros                                 | <pre>"metric": {   "computing": "actual",   "element": "number",   "event": {     "github": {       "pullRequest": {         "reviewer": null,         "timeSinceCreation": {           "operator": "&lt;",           "value": "%MTWRTP10%"         }       }     } } },</pre>                                                                                                                                         | <p>"NPRW_PERCENT_NPR_NPRW_NPRW_BIND": {<br/>"id": "NPRW_PERCENT_NPR_NPRW_NPRW_BIND",<br/>"notes": "### Description\n\n\"\\n\"\\nCTP10: Avoid Pull Requests being without a reviewer for too long.\n\\n\\nIn\\nTP-1: Ensure that the percentage of Pull Requests without a reviewer is within the specified threshold.",<br/>"description": "Ensure that the percentage of Pull Requests without a reviewer is within the specified threshold.",<br/>"scope": {<br/>  "Smt": "#/context/definitions/scopes/development" },<br/>"of": [<br/>  {     "scope": {<br/>      "project": "1010101010"     },<br/>    "objective": "NPRW / NPR * 100 &lt;= 10",<br/>    "with": {<br/>      "NPRW": {},<br/>      "NPR": {}     },<br/>    "window": {<br/>      "type": "static",<br/>      "period": "weekly"     } ] }</p> |
| CTP11 | TMR   | Tiempo medio de todas las reviews (desde que se asigna hasta que se aprueba) de las P/R del equipo durante la última semana    | <pre>"metric": {   "computing": "average",   "element": "time",   "event": {     "github": {       "pullRequest": {         "review": {           "status": "approved",           "time": true         }       }     } } },</pre>                                                                                                                                                                                      | <p>"TMR_LESS_THAN_TLTTP11": {<br/>"id": "TMR_LESS_THAN_TLTTP11",<br/>"notes": "### Description\n\n\"\\n\"\\nCTP11: Avoid long code review periods.\n\\n\\nIn\\nTP-1: Ensure that the average review time for Pull Requests is below the specified threshold.",<br/>"description": "Ensure that the average review time for Pull Requests is below the specified threshold.",<br/>"scope": {<br/>  "Smt": "#/context/definitions/scopes/development" },<br/>"of": [<br/>  {     "scope": {<br/>      "project": "1010101010"     },<br/>    "objective": "TMR &lt; TLTTP11",<br/>    "with": {<br/>      "TMR": {},<br/>      "TLTTP11": {}     },<br/>    "window": {<br/>      "type": "static",<br/>      "period": "weekly"     } ] }</p>                                                                          |
|       |       |                                                                                                                                |                                                                                                                                                                                                                                                                                                                                                                                                                        | <p>"TPmax_LESS_THAN_UTP12": {<br/>"id": "TPmax_LESS_THAN_UTP12",<br/>"notes": "### Description\n\n\"\\n\"\\nCTP12: Control the test execution time.\n\\n\\nIn\\nTP-1: Ensure that the maximum test suite execution time is below the specified threshold.",<br/>"description": "Ensure that the maximum test suite execution time is below the specified threshold.",<br/>"scope": {<br/>  "Smt": "#/context/definitions/scopes/development" },<br/>"of": [<br/>  {     "scope": {<br/>      "project": "1010101010"     },<br/>    "objective": "TPmax &lt; UTP12",<br/>    "with": {<br/>      "TPmax": {},<br/>      "UTP12": {}     },<br/>    "window": {<br/>      "type": "static",<br/>      "period": "weekly"     } ] }</p>                                                                                 |

|       |       |       |                                                                                                                              |                                                                                                                                                                                                                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|-------|-------|-------|------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|       | CTP12 | TPmax | El tiempo total que tarda en ejecutarse todo el conjunto de pruebas del proceso.                                             | <pre> "metric": {   "computing": "total",   "element": "time",   "event": {     "continuousIntegration": {       "testSuite": {}     }   } } </pre>                                                                              | <pre> "Sref": "#/context/definitions/scopes/development" "of": [   {     "scope": {       "project": "1010101010"     },     "objective": "TPmax &lt; UTP12",     "with": {       "TPmax": 0,       "UTP12": 0     },     "window": {       "type": "static",       "period": "weekly"     }   } ] </pre>                                                                                                                                                            |
| CTP13 |       | NFR   | Número de fusiones realizadas desde una rama activa a la rama principal.                                                     | <pre> "merge": {   "computing": "actual",   "element": "number",   "event": {     "git": {       "merge": {         "targetBranch": "%GITHUB_REPO_OWNER%:master"       }     }   } } </pre>                                      | <pre> "NFR_DIVIDED_BY_NM_GREATER_THAN_FMTP13": {   "id": "NFR_DIVIDED_BY_NM_GREATER_THAN_FMTP13",   "notes": "### Description\n\nCTP13: Merge to the main branch at the established frequency.\n\nNFR: Ensure that the number of merges to the main branch per team member is at least equal to the specified threshold.\n\nDescription: Ensure that the number of merges to the main branch per team member is at least equal to the specified threshold." } </pre> |
|       |       | NM    | Número de miembros del equipo.                                                                                               | <pre> "metric": {   "computing": "actual",   "element": "number",   "event": {     "team": {}   } } </pre>                                                                                                                       | <pre> "NM": {   "id": "NM",   "notes": "NM: Ensure that the number of team members is above the specified threshold." } </pre>                                                                                                                                                                                                                                                                                                                                       |
| CTP14 |       | NPR   | (Number of PR): Número de pull request realizados en la rama principal.                                                      | <pre> "metric": {   "computing": "actual",   "element": "number",   "event": {     "github": {       "pullRequest": {         "baseBranch": "%GITHUB_REPO_OWNER%:master",         "status": "success"       }     }   } } </pre> | <pre> "NW_PERCENT_NPR_NW_UTP14_BIND": {   "id": "NW_PERCENT_NPR_NW_UTP14_BIND",   "description": "Trigger workflows.",   "notes": "Ensure that the percentage of workflows triggered automatically is above the specified threshold." } </pre>                                                                                                                                                                                                                       |
|       |       | NW    | (Number of Workflow): Número de flujos de trabajo desencadenados automáticamente                                             | <pre> "metric": {   "computing": "actual",   "element": "number",   "event": {     "github": {       "workflow": {         "trigger": "push"       }     }   } } </pre>                                                          | <pre> "NW": {   "id": "NW",   "notes": "NW: Ensure that the percentage of workflows triggered automatically is above the specified threshold." } </pre>                                                                                                                                                                                                                                                                                                              |
| CTP15 |       | NPR   | (Number of PR): El número de pull requests realizados en la rama principal.                                                  | <pre> "metric": {   "computing": "count",   "event": {     "pull_requests": {       "opened": 0     }   } } </pre>                                                                                                               | <pre> "NP_NTT_GREATER_THAN_MBTTTP15": {   "description": "Esta práctica tiene como objetivo proporcionar a los desarrolladores un feedback rápido sobre la calidad de los cambios en su código.",   "id": "NP_NTT_GREATER_THAN_MBTTTP15",   "objective": "NPR &gt; NTT &gt; MBTTTP15",   "scope": {     "team": 0   },   "window": {     "type": "static",     "period": "weekly"   } } </pre>                                                                       |
|       |       | NTT   | (Number of Test Triggered): La cantidad de lanzamientos del conjunto de tests.                                               | <pre> "metric": {   "computing": "count",   "event": {     "tests": {       "triggered": 0     }   } } </pre>                                                                                                                    | <pre> "NP_NTT_GREATER_THAN_MBTTTP15": {   "description": "Esta práctica tiene como objetivo proporcionar a los desarrolladores un feedback rápido sobre la calidad de los cambios en su código.",   "id": "NP_NTT_GREATER_THAN_MBTTTP15",   "objective": "NPR &gt; NTT &gt; MBTTTP15",   "scope": {     "team": 0   },   "window": {     "type": "static",     "period": "weekly"   } } </pre>                                                                       |
| ITP16 |       | NP    | (Number of Push): El número de pushes realizados en la rama principal.                                                       | <pre> "metric": {   "computing": "count",   "event": {     "pushes": {}   } } </pre>                                                                                                                                             | <pre> "NP_NTT_GREATER_THAN_MBTTTP16": {   "description": "Este TP se enfoca en la proporción de cambios en el código que se ejecutan con éxito en relación con el total de cambios ejecutados.",   "id": "NP_NTT_GREATER_THAN_MBTTTP16",   "objective": "NP &gt; NTT &gt; MBTTTP16",   "scope": {     "member": 0   },   "window": {     "type": "static",     "period": "weekly"   } } </pre>                                                                       |
|       |       | NTT   | (Number of Test Triggered): La cantidad de lanzamientos del conjunto de tests.                                               | <pre> "metric": {   "computing": "count",   "event": {     "tests": {       "triggered": 0     }   } } </pre>                                                                                                                    | <pre> "NP_NTT_GREATER_THAN_MBTTTP16": {   "description": "Este TP se enfoca en la proporción de cambios en el código que se ejecutan con éxito en relación con el total de cambios ejecutados.",   "id": "NP_NTT_GREATER_THAN_MBTTTP16",   "objective": "NP &gt; NTT &gt; MBTTTP16",   "scope": {     "member": 0   },   "window": {     "type": "static",     "period": "weekly"   } } </pre>                                                                       |
| CTP17 |       | NTT   | (Number of Test Triggered): Cantidad de tests ejecutados.                                                                    | <pre> "metric": {   "computing": "count",   "event": {     "tests": {       "triggered": 0     }   } } </pre>                                                                                                                    | <pre> "NST_NTT_TIMES_100_GREATER_THAN_UTP17": {   "description": "Este TP se enfoca en la proporción de pruebas automatizadas que se ejecutan con éxito en relación con el total de pruebas ejecutadas.",   "id": "NST_NTT_TIMES_100_GREATER_THAN_UTP17",   "objective": "NST * 100 &gt;= NTT * UTP17",   "scope": {     "team": 0   },   "window": {     "type": "static",     "period": "weekly"   } } </pre>                                                      |
|       |       | NSTT  | (Number of Successful Test Triggered): La cantidad de lanzamientos del conjunto de tests que han tenido un resultado exitoso | <pre> "event": {   "tests": {     "triggered": {       "result": "success"     }   } } </pre>                                                                                                                                    | <pre> "NST_NTT_TIMES_100_GREATER_THAN_UTP17": {   "description": "Este TP se enfoca en la proporción de pruebas automatizadas que se ejecutan con éxito en relación con el total de pruebas ejecutadas.",   "id": "NST_NTT_TIMES_100_GREATER_THAN_UTP17",   "objective": "NST * 100 &gt;= NTT * UTP17",   "scope": {     "team": 0   },   "window": {     "type": "static",     "period": "weekly"   } } </pre>                                                      |
| ITP18 |       | NTT   | (Number of Test Triggered): Cantidad de tests ejecutados.                                                                    | <pre> "metric": {   "computing": "count",   "event": {     "tests": {       "triggered": 0     }   } } </pre>                                                                                                                    | <pre> "NST_NTT_TIMES_100_GREATER_THAN_UTP18": {   "description": "Este TP se enfoca en la proporción de pruebas automatizadas que se ejecutan con éxito en relación con el total de pruebas ejecutadas.",   "id": "NST_NTT_TIMES_100_GREATER_THAN_UTP18",   "objective": "NST * 100 &gt;= NTT * UTP18",   "scope": {     "member": 0   },   "window": {     "type": "static",     "period": "weekly"   } } </pre>                                                    |
|       |       | NSTT  | (Number of Successful Test Triggered): La cantidad de lanzamientos del conjunto de tests que han tenido un resultado exitoso | <pre> "event": {   "tests": {     "triggered": {       "result": "success"     }   } } </pre>                                                                                                                                    | <pre> "NST_NTT_TIMES_100_GREATER_THAN_UTP18": {   "description": "Este TP se enfoca en la proporción de pruebas automatizadas que se ejecutan con éxito en relación con el total de pruebas ejecutadas.",   "id": "NST_NTT_TIMES_100_GREATER_THAN_UTP18",   "objective": "NST * 100 &gt;= NTT * UTP18",   "scope": {     "member": 0   },   "window": {     "type": "static",     "period": "weekly"   } } </pre>                                                    |
|       |       | NW    | (Number of Workflows): El número de flujos de trabajos desencadenados automáticamente.                                       | <pre> "metric": {   "computing": "actual",   "element": "number",   "event": {     "github": {       "workflow": {         "trigger": "push"       }     }   } } </pre>                                                          | <pre> "NSW_LESS_THAN_OR_EQUAL_TO_NW_MINUS_MNWETP19": {   "id": "NSW_LESS_THAN_OR_EQUAL_TO_NW_MINUS_MNWETP19",   "description": "Evitar subidas con errores.",   "notes": "Este TP asegura que el número de flujos de trabajo exitosos sea menor o igual a la diferencia entre el número total de flujos de trabajo activados y el número máximo de flujos de trabajo con errores." } </pre>                                                                          |

|       |       |                                                                                                                                                                                                            |                                                                                                                                                                                                                                                                                                                                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|-------|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CTP19 | NSW   | (Number of Successful Workflows): El número de flujos de trabajo que se ejecutaron con éxito.                                                                                                              | <pre>"metric": {   "computing": "actual",   "element": "number",   "event": {     "github": {       "workflow": {         "status": "completed",         "conclusion": "success",         "trigger": "push"       }     } }</pre>                                                                                                 | <pre>"of": [   {     "scope": {       "project": "1010101010"     },     "with": {       "NSW": 0,       "NW": 0,       "NMWETP19": 3     },     "window": {       "type": "static",       "period": "semanal"     }   } ]</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| CTP20 | RTBW  | (Resolution Time for Broken Workflow): El tiempo que lleva resolver un error en un flujo de trabajo fallido, medido en horas.                                                                              | <pre>"metric": {   "computing": "time",   "element": "timestamp",   "event": {     "github": {       "workflow": {         "status": "failure"       }     } }, "metric": {   "computing": "time",   "element": "timestamp",   "event": {     "github": {       "workflow": {         "status": "success"       }     } }, </pre> | <pre>"RTBW_LESS_THAN_NMHTP20": {   "id": "RTBW_LESS_THAN_NMHTP20",   "notes": "### Description\n\nRTP-1: Ensure that the resolution time for a broken workflow is less than the specified threshold."   "description": "Ensure that the resolution time for a broken workflow is less than the specified threshold.",   "scope": {     "ref": "#/context/definitions/scopes/development"   },   "of": [     {       "scope": {         "project": "1010101010"       },       "objective": "(RTB_failure - RTB_success) &lt; 48",       "with": {         "RTB_failure": 0,         "RTB_success": 0,         "NMHTP20": 48       },       "window": {         "type": "static",         "period": "weekly"       }     }   ] }</pre>                                                                                                                                                                                                                                       |
| CTP21 | NBWT  | (Number of Bug Issues without Test Associated): El número de problemas (bugs) que no tienen pruebas asociadas.                                                                                             | <pre>metric": {   "computing": "count",   "condition": {     "bug_issues": {       "without_test_associated": 0     }   } } </pre>                                                                                                                                                                                                | <pre>"NBWT_LESS_THAN_OR_EQUAL_TO_MNBWTP21": {   "description": "Esta CTP se centra en medir la cantidad de problemas (bugs) sin pruebas asociadas."   "id": "NBWT_LESS_THAN_OR_EQUAL_TO_MNBWTP21",   "objective": "NBWT &lt;= MNBWTP21",   "window": {     "period": "weekly"   },   "with": {     "NBWT": 0   } }</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| CTP22 | DPRP  | (Duration in seconds between push in master and release in production): El tiempo que transcurre entre un push o merge de una P/R en la rama principal y el fin del flujo de trabajo, medido en segundos.  | <pre>"event": {   "github": {     "workflow": {       "trigger": "push",       "status": "success"     }   } }, metric": {   "computing": "time",   "element": "timestamp",   "event": {     "github": {       "workflow": {         "trigger": "pull_request.closed",         "status": "success"       }     } }, </pre>        | <pre>"DPRP_DIFF_LESS_THAN_TFTTP22": {   "id": "DPRP_DIFF_LESS_THAN_TFTTP22",   "notes": "### Description\n\nRTP-1: Reduce times between pull request and workflow execution.\n\nTP-1: Ensure that the difference between the push and pull request closure in the master branch and the end of the workflow is less than the specified threshold."   "description": "Ensure that the difference between the push and pull request closure in the master branch and the end of the workflow is less than the specified threshold.",   "scope": {     "ref": "#/context/definitions/scopes/development"   },   "of": [     {       "scope": {         "project": "1010101010"       },       "objective": "(DPRP_pull_request_close - DPRP_push) &lt;= 3600",       "with": {         "DPRP_push": 0,         "DPRP_pull_request_close": 0,         "TFTTP22": 3600       },       "window": {         "type": "static",         "period": "weekly"       }     }   ] }</pre> |
| CTP23 | NRDUS | (Number of Releases with an Associated Bug Derived from a User Story implemented in that release): El número de releases con un bug derivado de una User Story implementada en esa release.                | <pre>"metric": {   "computing": "count",   "condition": {     "releases": {       "with_bug_derived_from_user_story": 0     }   } } </pre>                                                                                                                                                                                        | <pre>"NRDUS_LESS_THAN_TNR_TIMES_100_LESS_THAN_UTP23": {   "description": "Esta CTP se centra en medir la proporción de cambios que resultan en degradación."   "id": "NRDUS_LESS_THAN_TNR_TIMES_100_LESS_THAN_UTP23",   "objective": "NRDUS / TNR * 100 &lt; UTP23",   "window": {     "period": "monthly"   },   "with": {     "NRDUS": 0,     "TNR": 0   } }</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|       | TNR   | (Total Number of Releases): El número total de releases.                                                                                                                                                   | <pre>"metric": {   "computing": "count",   "condition": {     "releases": 0   } } </pre>                                                                                                                                                                                                                                          | <pre>"NRDUS_LESS_THAN_TNR_TIMES_100_LESS_THAN_UTP23": {   "description": "Esta CTP se centra en medir la proporción de cambios que resultan en degradación."   "id": "NRDUS_LESS_THAN_TNR_TIMES_100_LESS_THAN_UTP23",   "objective": "NRDUS / TNR * 100 &lt; UTP23",   "window": {     "period": "monthly"   },   "with": {     "NRDUS": 0,     "TNR": 0   } }</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| CTP24 | NB    | (Number of bugs): Número de bugs cerrados durante el periodo.                                                                                                                                              | <pre>"event": {   "bug_tracking_system": {     "issue": "bug",     "type": "bug"   } } </pre>                                                                                                                                                                                                                                     | <pre>"BBT_OVER_NB_LESS_THAN_UTP24": {   "id": "BBT_OVER_NB_LESS_THAN_UTP24",   "notes": "### Description\n\nRTP-1: Ensure that the percentage of bugs with a lifetime beyond the threshold is less than or equal to the specified threshold."   "description": "Ensure that the percentage of bugs with a lifetime beyond the threshold is less than or equal to the specified threshold.",   "scope": {     "ref": "#/context/definitions/scopes/development"   },   "of": [     {       "scope": {         "project": "1010101010"       },       "objective": "(BBT / NB) * 100 &lt;= 90",       "with": {         "BBT": 0,         "NB": 0,         "UTP24": 90       },       "window": {         "type": "static",         "period": "weekly"       }     }   ] }</pre>                                                                                                                                                                                              |
|       | BBT   | (Bugs lifetime beyond the threshold): Número de ocasiones en el que el tiempo promedio de vida de los bugs, medido en horas, supera el NMHBTP24 (Número máximo de horas que puede estar sin solucionarse). | <pre>"metric": {   "computing": "time",   "element": "number",   "event": {     "bug_tracking_system": {       "bug": {         "status": "resolved"       }     } }, </pre>                                                                                                                                                      | <pre>"BBT_OVER_NB_LESS_THAN_UTP24": {   "id": "BBT_OVER_NB_LESS_THAN_UTP24",   "notes": "### Description\n\nRTP-1: Ensure that the percentage of bugs with a lifetime beyond the threshold is less than or equal to the specified threshold."   "description": "Ensure that the percentage of bugs with a lifetime beyond the threshold is less than or equal to the specified threshold.",   "scope": {     "ref": "#/context/definitions/scopes/development"   },   "of": [     {       "scope": {         "project": "1010101010"       },       "objective": "(BBT / NB) * 100 &lt;= 90",       "with": {         "BBT": 0,         "NB": 0,         "UTP24": 90       },       "window": {         "type": "static",         "period": "weekly"       }     }   ] }</pre>                                                                                                                                                                                              |

## 9.2.7 FUNCIÓN STEPHANDLERS

Es la función encargada de manejar todas las peticiones que se pueden realizar desde la página de testeo de métricas automatizada.

```
1. private stepHandlers = {
2. 'TEST': {
3. 'bluejay/check': (step: {
4. with: {
5. key: string,
6. conditions: { minExpectedValue?: string, maxExpectedValue?: string,
expectedValue?: string }
7. }[];
8. value?: string,
9. createdAt?: string,
10. authorLogin?: string
11. }) => {
12. return Promise.all(step.with.map(({key, conditions}) => {
13. return new Promise<void>((resolve, reject) => {
14. setTimeout(() => {
15.
this.http.get<any>(`http://localhost:6012/glassmatrix/api/v1/getData/${key}`),
{}).subscribe((data: any) => {
16. if (data)
17. data.forEach((item: any) => {
18. // Si 'value' no está definido en el paso, o si es igual al
'value' en el objeto de datos, entonces procesa el objeto
19. if (item[key] && (step.value === undefined || item['value']
== step.value)) {
20.
21.
22.
23.
24.
25. const value = item[key];
26. // Comprueba si el valor de la clave es "not found"
27. if (value === "not found") {
28. this.testStatuses.push({
29. text: `Test failed. Field '${key}' not found in the
database` ,
30.
31.
32.
33.
34. success: false
35. });
36. resolve();
37. } else {
38. // Comprueba cada condición por separado
39. if (conditions.minExpectedValue !== undefined) {
40. if (value >= Number(conditions.minExpectedValue)) {
41. this.testStatuses.push({
42. text: `Test successfully completed.\nCondition:
minExpectedValue=${conditions.minExpectedValue}\nResult: ${key}=${value}`,
43.
44. success: true
45. });
46.
47. }
48. }
49. if (conditions.maxExpectedValue !== undefined) {
50. if (value <= Number(conditions.maxExpectedValue)) {
51. this.testStatuses.push({
52. text: `Test successfully completed.\nCondition:
maxExpectedValue=${conditions.maxExpectedValue}\nResult: ${key}=${value}`,
53.
54. success: true
55. });
56. }
57. }
58. }
59. }
60. if (value <= Number(conditions.maxExpectedValue)) {
61. this.testStatuses.push({
62. text: `Test failed.\nCondition:
maxExpectedValue=${conditions.maxExpectedValue}\nResult: ${key}=${value}`,
63.
64. success: false
65. });
66. }
67. }
68. }
69. }
70. }
71. }
72. }
73. }
74. }
75. }
76. }
77. }
78. }
79. }
80. }
81. }
82. }
83. }
84. }
85. }
86. }
87. }
88. }
89. }
90. }
91. }
92. }
93. }
94. }
95. }
96. }
97. }
98. }
99. }
100. }
101. }
102. }
103. }
104. }
105. }
106. }
107. }
108. }
109. }
110. }
111. }
112. }
113. }
114. }
115. }
116. }
117. }
118. }
119. }
120. }
121. }
122. }
123. }
124. }
125. }
126. }
127. }
128. }
129. }
130. }
131. }
132. }
133. }
134. }
135. }
136. }
137. }
138. }
139. }
140. }
141. }
142. }
143. }
144. }
145. }
146. }
147. }
148. }
149. }
150. }
151. }
152. }
153. }
154. }
155. }
156. }
157. }
158. }
159. }
160. }
161. }
162. }
163. }
164. }
165. }
166. }
167. }
168. }
169. }
170. }
171. }
172. }
173. }
174. }
175. }
176. }
177. }
178. }
179. }
180. }
181. }
182. }
183. }
184. }
185. }
186. }
187. }
188. }
189. }
190. }
191. }
192. }
193. }
194. }
195. }
196. }
197. }
198. }
199. }
200. }
201. }
202. }
203. }
204. }
205. }
206. }
207. }
208. }
209. }
210. }
211. }
212. }
213. }
214. }
215. }
216. }
217. }
218. }
219. }
220. }
221. }
222. }
223. }
224. }
225. }
226. }
227. }
228. }
229. }
230. }
231. }
232. }
233. }
234. }
235. }
236. }
237. }
238. }
239. }
240. }
241. }
242. }
243. }
244. }
245. }
246. }
247. }
248. }
249. }
250. }
251. }
252. }
253. }
254. }
255. }
256. }
257. }
258. }
259. }
260. }
261. }
262. }
263. }
264. }
265. }
266. }
267. }
268. }
269. }
270. }
271. }
272. }
273. }
274. }
275. }
276. }
277. }
278. }
279. }
280. }
281. }
282. }
283. }
284. }
285. }
286. }
287. }
288. }
289. }
290. }
291. }
292. }
293. }
294. }
295. }
296. }
297. }
298. }
299. }
300. }
301. }
302. }
303. }
304. }
305. }
306. }
307. }
308. }
309. }
310. }
311. }
312. }
313. }
314. }
315. }
316. }
317. }
318. }
319. }
320. }
321. }
322. }
323. }
324. }
325. }
326. }
327. }
328. }
329. }
330. }
331. }
332. }
333. }
334. }
335. }
336. }
337. }
338. }
339. }
340. }
341. }
342. }
343. }
344. }
345. }
346. }
347. }
348. }
349. }
350. }
351. }
352. }
353. }
354. }
355. }
356. }
357. }
358. }
359. }
360. }
361. }
362. }
363. }
364. }
365. }
366. }
367. }
368. }
369. }
370. }
371. }
372. }
373. }
374. }
375. }
376. }
377. }
378. }
379. }
380. }
381. }
382. }
383. }
384. }
385. }
386. }
387. }
388. }
389. }
390. }
391. }
392. }
393. }
394. }
395. }
396. }
397. }
398. }
399. }
400. }
401. }
402. }
403. }
404. }
405. }
406. }
407. }
408. }
409. }
410. }
411. }
412. }
413. }
414. }
415. }
416. }
417. }
418. }
419. }
420. }
421. }
422. }
423. }
424. }
425. }
426. }
427. }
428. }
429. }
430. }
431. }
432. }
433. }
434. }
435. }
436. }
437. }
438. }
439. }
440. }
441. }
442. }
443. }
444. }
445. }
446. }
447. }
448. }
449. }
450. }
451. }
452. }
453. }
454. }
455. }
456. }
457. }
458. }
459. }
460. }
461. }
462. }
463. }
464. }
465. }
466. }
467. }
468. }
469. }
470. }
471. }
472. }
473. }
474. }
475. }
476. }
477. }
478. }
479. }
480. }
481. }
482. }
483. }
484. }
485. }
486. }
487. }
488. }
489. }
490. }
491. }
492. }
493. }
494. }
495. }
496. }
497. }
498. }
499. }
500. }
501. }
502. }
503. }
504. }
505. }
506. }
507. }
508. }
509. }
510. }
511. }
512. }
513. }
514. }
515. }
516. }
517. }
518. }
519. }
520. }
521. }
522. }
523. }
524. }
525. }
526. }
527. }
528. }
529. }
530. }
531. }
532. }
533. }
534. }
535. }
536. }
537. }
538. }
539. }
540. }
541. }
542. }
543. }
544. }
545. }
546. }
547. }
548. }
549. }
550. }
551. }
552. }
553. }
554. }
555. }
556. }
557. }
558. }
559. }
560. }
561. }
562. }
563. }
564. }
565. }
566. }
567. }
568. }
569. }
570. }
571. }
572. }
573. }
574. }
575. }
576. }
577. }
578. }
579. }
580. }
581. }
582. }
583. }
584. }
585. }
586. }
587. }
588. }
589. }
590. }
591. }
592. }
593. }
594. }
595. }
596. }
597. }
598. }
599. }
600. }
601. }
602. }
603. }
604. }
605. }
606. }
607. }
608. }
609. }
610. }
611. }
612. }
613. }
614. }
615. }
616. }
617. }
618. }
619. }
620. }
621. }
622. }
623. }
624. }
625. }
626. }
627. }
628. }
629. }
630. }
631. }
632. }
633. }
634. }
635. }
636. }
637. }
638. }
639. }
640. }
641. }
642. }
643. }
644. }
645. }
646. }
647. }
648. }
649. }
650. }
651. }
652. }
653. }
654. }
655. }
656. }
657. }
658. }
659. }
660. }
661. }
662. }
663. }
664. }
665. }
666. }
667. }
668. }
669. }
670. }
671. }
672. }
673. }
674. }
675. }
676. }
677. }
678. }
679. }
680. }
681. }
682. }
683. }
684. }
685. }
686. }
687. }
688. }
689. }
690. }
691. }
692. }
693. }
694. }
695. }
696. }
697. }
698. }
699. }
700. }
701. }
702. }
703. }
704. }
705. }
706. }
707. }
708. }
709. }
710. }
711. }
712. }
713. }
714. }
715. }
716. }
717. }
718. }
719. }
720. }
721. }
722. }
723. }
724. }
725. }
726. }
727. }
728. }
729. }
730. }
731. }
732. }
733. }
734. }
735. }
736. }
737. }
738. }
739. }
740. }
741. }
742. }
743. }
744. }
745. }
746. }
747. }
748. }
749. }
750. }
751. }
752. }
753. }
754. }
755. }
756. }
757. }
758. }
759. }
760. }
761. }
762. }
763. }
764. }
765. }
766. }
767. }
768. }
769. }
770. }
771. }
772. }
773. }
774. }
775. }
776. }
777. }
778. }
779. }
779. }
```

```

54. }
55. }
56. if (conditions.expectedValue !== undefined) {
57. if (Number(value) === Number(conditions.expectedValue))
58. {
59. this.testStatuses.push({
60. text: `Test successfully completed.\nCondition:
61. expectedValue=${conditions.expectedValue}\nResult: ${key}=${value}`,
62. success: true
63. });
64. } else {
65. this.testStatuses.push({
66. text: `Test failed.\nCondition:
67. expectedValue=${conditions.expectedValue}\nResult: ${key}=${value}`,
68. success: false
69. });
70. }
71. });
72. });
73. } else {
74. // Si no hay datos, empuja un mensaje indicando que el test ha
fallado a this.testStatuses
75. this.testStatuses.push({
76. text: `Test failed. Field '${key}' not found in the
database`,
77. success: false
78. });
79. resolve();
80. }
81. }, reject);
82. }, 1000);
83.);
84.);
85.},
86. 'bluejay/findCheck': (step: { with: { values: any[]; }; }) => {
87. return new Promise(resolve => setTimeout(resolve, 1000))
88. .then(() => {
89. const url = `${BASE_URL}:6012/glassmatrix/api/v1/bluejay/findCheck`;
90. const headers = {'Authorization': `Bearer ${this.token}`};
91.
92.
93. return this.http.post(url, {values: step.with.values},
{headers}).toPromise().then((response: any) => {
94. // Agrega la respuesta a la respuesta existe
95. this.response += 'bluejay/findCheck ' + JSON.stringify(response,
null, 2) + '\n\n';
96.
97.
98. // Comprueba si se encontraron los valores esperados y agrega los
resultados a testStatuses
99. step.with.values.forEach((valueObj: any) => {
100. const foundValue = response.find((res: any) =>
101. res.computations.some((comp: any) =>
102. comp.value === valueObj.value &&
103. comp.evidences.some((evidence: any) => {
104. let allEvidencesFound = true;
105. for (const key in valueObj.evidences) {
106. if (key === 'login') {
107. const foundLogin = evidence.author.login ===
valueObj.evidences.login;
108. allEvidencesFound = allEvidencesFound && foundLogin;
109. if (foundLogin) {
110. this.testStatuses.push({
111. text: `Test successfully completed. login =
"${valueObj.evidences.login}" found.`,
112. success: true
113. });

```

```

114. } else {
115. this.testStatuses.push({
116. text: `Test failed. login =
117. ${valueObj.evidences.login}" not found.`,
118. success: false
119. });
120. } else if (key === 'bodyText') {
121. const foundBodyText =
evidence.comments.nodes.some((comment: any) => comment.bodyText ===
valueObj.evidences.bodyText);
122. allEvidencesFound = allEvidencesFound && foundBodyText;
123. if (foundBodyText) {
124. this.testStatuses.push({
125. text: `Test successfully completed. bodyText =
"${valueObj.evidences.bodyText}" found.`,
126. success: true
127. });
128. } else {
129. this.testStatuses.push({
130. text: `Test failed. bodyText =
"${valueObj.evidences.bodyText}" not found.`,
131. success: false
132. });
133. }
134. } else {
135. const foundEvidence = evidence[key] ===
valueObj.evidences[key];
136. allEvidencesFound = allEvidencesFound && foundEvidence;
137. if (foundEvidence) {
138. this.testStatuses.push({
139. text: `Test successfully completed. ${key} =
"${valueObj.evidences[key]}" found.`,
140. success: true
141. });
142. } else {
143. this.testStatuses.push({
144. text: `Test failed. ${key} =
"${valueObj.evidences[key]}" not found.`,
145. success: false
146. });
147. }
148. }
149. }
150. return allEvidencesFound;
151.)
152.)
153.);
154. if (!foundValue) {
155. this.testStatuses.push({
156. text: `Test failed. Test for value: ${valueObj.value} has
failed.`,
157. success: false
158. });
159. }
160.);
161.);
162.);
163. },
164.],
165. 'GET': [
166. 'github/getIssue': (step: { with: { [x: string]: string; } }) =>
this.githubService.getIssues(this.token, step.with['owner'],
step.with['repoName']).toPromise(),
167. 'github/getOpenPR': (step: { with: { [x: string]: string; } }) =>
this.githubService.getOpenPullRequests(this.token, step.with['owner'],
step.with['repoName']).toPromise(),
168. 'github/pullCurrentBranch': (step: { with: { [x: string]: string; } }) =>
this.glassmatrixService.pullCurrentBranch(step.with['repoName']).toPromise(),
169. 'github/listRepos': () => this.glassmatrixService.listRepos().toPromise(),

```

```

170. 'github/getBranches': (step: { with: { [x: string]: string; } }) =>
171. this.glassmatrixService.getBranches(step.with['repoName']).toPromise(),
172. 'github/getRepoInfo': (step: { with: { [x: string]: string; } }) =>
173. this.githubService.getRepoInfo(step.with['repoName'], step.with['branchName']).toPromise()
174. },
175. 'POST': {
176. 'github/mergeLastOpenPR': (step: { with: { [x: string]: string; } }) => {
177. return this.githubService.mergeLastOpenPullRequest(this.token,
178. step.with['owner'], step.with['repoName'], step.with['mergeMessage']).toPromise();
179. },
180. 'github/undoLastMergedPR': (step: { with: { [x: string]: string; } }) => {
181. return this.githubService.undoLastMergedPullRequest(this.token,
182. step.with['owner'], step.with['repoName']).toPromise();
183. },
184. 'bluejay/compute/tpa': (step: { with: { [x: string]: string; } }) => {
185. const tpa = step.with['tpa'];
186. const metric = step.with['metric'];
187. const time = step.with['actualTime'] === 'true';
188. return this.loadData(tpa, metric, time).toPromise().then((data) => {
189. this.postContent().subscribe(response => {
190. setTimeout(() => {
191. this.getComputation();
192. }, 1000);
193. });
194. });
195. },
196. 'bluejay/compute/metric': (step: { with: { [x: string]: string; } }) => {
197. const metric = step.with['metric'];
198. const time = step.with['actualTime'] === 'true';
199. return new Promise<void>((resolve, reject) => {
200. this.loadIndividualData(metric, time).subscribe(
201. () => {
202. this.postContent().subscribe(response => {
203. setTimeout(() => {
204. this.getComputation();
205. }, 1000);
206. resolve();
207. }, reject);
208. },
209. reject
210.);
211. });
212. },
213. //DEPRECADO
214. 'bluejay/checkContain': (step: { with: { [x: string]: string; } }) => {
215. console.warn("Deprecation Warning: 'bluejay/checkContain' has been
216. deprecated. Please use 'TEST' method with 'bluejay/check' instead.");
217. const key = step.with['key'];
218. const minExpectedValue = Number(step.with['minExpectedValue']);
219. return new Promise<void>((resolve, reject) => {
220. setTimeout(() => {
221.
222. this.http.get<any>(`http://localhost:6012/glassmatrix/api/v1/getData/${key}`,
223. {}).subscribe((data: any) => {
224. this.testStatuses.push({ text: `Deprecation Warning:
225. 'bluejay/checkContain' has been deprecated. Please use 'TEST' method with 'bluejay/check'
226. instead.`, success: false });
227. if (data && data[0] && data[0][key]) {
228. const value = data[0][key];
229. if (value >= minExpectedValue) {
230. this.testStatuses.push({ text: `Test successfully completed.
231. ${key}=${value}`, success: true });
232. resolve();
233. } else {
234. this.testStatuses.push({ text: `Test failed. ${key}=${value}`,
235. success: false });
236. resolve();
237. }
238. } else {
239. this.testStatuses.push({ text: `Test failed. ${key}=${value}`,
240. success: false });
241. resolve();
242. }
243. });
244. });
245. });
246. }
247. }
248.);
249.);
250.);
251.);
252.}

```

```

228. this.testStatuses.push({ text: `No records found for the field
229. `${key}` in the database`, success: false });
230. }
231. },
232. },
233.);
234.},
235. 'github/createIssue': (step: { with: { [x: string]: string; } }) => {
236. const issue = { title: step.with['title'], body: step.with['body'] };
237. return this.githubService.createIssue(this.token, step.with['owner'],
238. step.with['repoName'], issue).toPromise();
239. },
240. 'github/createPR': (step: { with: { [x: string]: string; } }) => {
241. const pr = { title: step.with['title'], head: step.with['head'], base:
242. step.with['base'], body: step.with['body'] };
243. return this.githubService.createPullRequest(this.token,
244. step.with['owner'], step.with['repoName'], pr.title, pr.head, pr.base,
245. pr.body).toPromise();
246. },
247. 'github/cloneRepo': (step: { with: { [x: string]: string; } }) =>
248. this.glassmatrixService.cloneRepo(step.with['owner']).toPromise(),
249. 'github/createBranch': (step: { with: { [x: string]: string; } }) => {
250. const branchFormValue = { branchName: step.with['branchName'] };
251. return this.glassmatrixService.createBranch(step.with['repoName'],
252. branchFormValue).toPromise();
253. },
254. 'github/createFile': (step: { with: { [x: string]: string; } }) =>
255. this.glassmatrixService.createFile(step.with['repoName'], step.with['fileName'],
256. step.with['fileContent']).toPromise(),
257. 'github/createCommit': (step: { with: { [x: string]: string; } }) =>
258. this.glassmatrixService.createCommit(step.with['repoName'], step.with['fileContent'],
259. step.with['commitMessage']).toPromise(),
260. 'github/commitAllChanges': (step: { with: { [x: string]: string; } }) =>
261. this.glassmatrixService.commitAllChanges(step.with['repoName'],
262. step.with['commitMessage']).toPromise(),
263. 'github/pushChanges': (step: { with: { [x: string]: string; } }) =>
264. this.glassmatrixService.pushChanges(step.with['repoName']).toPromise()
265. },
266. 'PUT': {
267. 'github/mergePR': (step: { with: { [x: string]: string; } }) =>
268. this.githubService.mergePullRequest(this.token, step.with['owner'], step.with['repoName'],
269. Number(step.with['prNumber']), step.with['mergeMessage']).toPromise(),
270. 'github/changeBranch': (step: { with: { [x: string]: string; } }) =>
271. this.glassmatrixService.changeBranch(step.with['repoName'],
272. step.with['branchToChangeTo']).toPromise()
273. },
274. 'DELETE': {
275. 'github/deleteRepo': (step: { with: { [x: string]: string; } }) =>
276. this.glassmatrixService.deleteRepo(step.with['repoName']).toPromise(),
277. 'github/deleteBranch': (step: { with: { [x: string]: string; } }) =>
278. this.glassmatrixService.deleteBranch(step.with['repoName'],
279. step.with['branchName']).toPromise(),
280. 'github/deleteFile': (step: { with: { [x: string]: string; } }) =>
281. this.glassmatrixService.deleteGithubFile(step.with['repoName'],
282. step.with['fileName']).toPromise()
283. },
284. }
285. };

```