DESARROLLO DE UN DASHBOARD PARA LA CONSULTA DE INTEGRACIONES Y DESPLIEGUES CONTINUOS EN LA PLATAFORMA DE AZURE DEVOPS



UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

FACULTAD TECNOLÓGICA

TECNOLOGIA EN SISTEMATIZACIÓN DE DATOS

BOGOTÁ D.C

2023

DESARROLLO DE UN DASHBOARD PARA LA CONSULTA DE INTEGRACIONES Y DESPLIEGUES CONTINUOS EN LA PLATAFORMA DE AZURE DEVOPS

Jonatan Stiben Escobar Alzate

Jhon Alexander Garcia Caro

TRABAJO DE GRADO REALIZADO COMO REQUISITO PARA OPTAR AL TÍTULO DE TECNOLOGO EN SISTEMATIZAICÓN DE DATOS

TUTOR

Rocío Rodríguez Guerrero

_____

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

FACULTAD TECNOLÓGICA

TECNOLOGIA EN SISTEMATIZACIÓN DE DATOS

BOGOTÁ D.C

2023

Nota de Aceptación:

_____

_____

_____

_____

_____

Tutor: Rocío Rodríguez Guerrero

_____

Jurado: Gerardo Alberto Castang Montiel

AGRADECIMIENTOS

En primer lugar, extendemos nuestro más sincero agradecimiento a nuestras familias, quienes han brindado su apoyo inquebrantable a lo largo de nuestro viaje de formación. Reconocemos que este camino no ha estado exento de desafíos, pero gracias a los sólidos principios y valores que nos han inculcado, hemos adquirido un profundo sentido de pertenencia y responsabilidad, permitiéndonos alcanzar los logros deseados.

Asimismo, extendemos nuestro agradecimiento a la Universidad Distrital Francisco José de Caldas por habernos brindado la oportunidad de cursar nuestros estudios. Así mismo, queremos reconocer a los docentes que compartieron sus conocimientos, apoyo y tiempo con nosotros, contribuyendo a nuestra formación académica.

Un agradecimiento especial se dirige a nuestra tutora de proyecto de grado, la ingeniera Rocío Rodríguez Guerrero, por haber compartido generosamente su experiencia, conocimientos y apoyo a lo largo del desarrollo de este proyecto. Su orientación fue invaluable para el logro de nuestros objetivos.

Finalmente, queremos expresar nuestra gratitud a nuestros padres, mamá y papá, cuyo incansable esfuerzo, apoyo, motivación y brindarnos todo lo necesario para alcanzar la culminación de nuestros estudios con éxito. Su contribución es inestimable. También a nuestros amigos por todo el apoyo y confianza brindada.

CONTENIDO

AGRADECIMIENTOS i
RESUMEN vii
PALABRAS CLAVE vii
ABSTRACT viii
KEYWORDS viii
INTRODUCCIÓN 1
1. FASE I: Planeación, definición y organización 2
1.1. TÍTULO DEL PROYECTO 2
1.2. TEMA 2
1.3. Planteamiento del problema 2
1.3.1. Descripción del problema 2
1.3.2. Formulación del problema 3
1.4. Objetivos 3
1.4.1. Objetivo general 4
1.4.2. Objetivos específicos 4
1.5. Alcances y Limitaciones 4
1.5.1. Alcances 4
1.5.2. Limitaciones 5
1.6. Justificación 5
1.7. Marco referencial 6
1.7.1. Estado del arte 6
1.7.2. Marco Teórico 10
1.7.3. Marco conceptual 15
1.7.4. Marco Legal 16
1.8. Solución Tecnológica 17
1.8.1. Metodología del proyecto 18
1.9. Factibilidad de desarrollo 21
1.9.1. Factibilidad Técnica 21
1.9.2. Factibilidad económica 23
1.9.2.1. Factibilidad Operativa - Recursos Humanos 23

1.9.2.2. Factibilidad Económica - Recursos Técnicos 24
1.9.2.3. Factibilidad Económica – Costo Total..... 25
1.10. Cronograma de actividades 26
1.10.2. Agenda propuesta 26
1.10.3. Diagrama de Gantt 27
2. FASE II: Análisis del proyecto 28
2.1. Análisis del problema 28
2.2. Identificación de roles 29
2.3 Sprints 30
2.4. Lista de actividades por rol 31
2.4 Historias de usuario.....33
3. Fase III: Diseño del proyecto 35
3.1 Sprint 1 35
3.1.1 Definición de sprint 1 35
3.1.2 Diseño de Sprint 1 36
3.1.3 Prueba del Sprint 1 37
3.2 Sprint 2 37
3.2.1 Definición de sprint 2 38
3.2.2 Diseño de Sprint 2 38
3.2.3 Prueba del Sprint 2 39
3.3 Sprint 3 39
3.3.1 Definición de sprint 3 39
3.3.2 Diseño de Sprint 3 40
3.3.3 Prueba del Sprint 3 40
3.4 Sprint 4 41
3.4.1 Definición de sprint 4 41
3.4.2 Diseño de Sprint 4 42
3.4.3 Prueba del Sprint 4 42
3.5 Sprint 5 43
3.5.1 Definición de sprint 5 43
3.5.2 Diseño de Sprint 5 44
3.5.3 Prueba del Sprint 5 44

3.6 Sprint 6 45
3.6.1 Definición de sprint 6 45
3.6.1 Diseño de Sprint 6 46
3.6.2 Prueba del Sprint 6 50
3.7 Sprint 7 50
3.7.1 Definición de sprint 7 50
3.7.2 Diseño de Sprint 7 52
3.7.3 Prueba del Sprint 7 53
4. Fase IV: Implementación del proyecto 54
4.1 API REST 54
4.2 Dashboard 58
4.3 Monitoreo 60
5. Fase V: Pruebas 62
5.3. Pruebas de calidad 62
5.4 Prueba de visualización de datos 63
5.5 Prueba de rendimiento 64
5.6 Prueba de calidad general 66
6. Fase VI: Resultados 68
7. Fase VII: Trabajos futuros 69
CONCLUSIONES 72
RECOMENDACIONES 73
BIBLIOGRAFÍA 74
ANEXOS 76
Anexo 1: Manual de Usuario 76
Descripción de la aplicación 76
Requerimientos 76
Descargar e instalar backed 77
Descargar e instalar dashboard 81
Anexo 2: Cronograma 1

INDICE DE ILUSTRACIONES

Ilustración 1. Diseño de la metodología Scrum. 21
Ilustración 2. Fases del proyecto 26
Ilustración 3. Actividades del proyecto 27
Ilustración 4 Diagrama de Gantt 27
Ilustración 5. Base de datos creada en Azure. 36
Ilustración 6. lista de Builds y prueba de conexión con las API. 37
Ilustración 7. Get Builds (creación de servicio de chequeo de integraciones). 39
Ilustración 8. Get Deployments (creacion de servicio para obtener la informacion de los despleigues continuos. 40
Ilustración 9. Creación de tablas a manejar en la base de datos (Clasificación Base de datos). 42
Ilustración 10. Conexión a la Base de datos. 42
Ilustración 11. Procedimientos Almacenados creados. 44
Ilustración 12. Plantillas creadas en Grafana Labs para la muestra de información obtenida. 46
Ilustración 13. Dashboard principal informativo (Estado general de compilaciones). 46
Ilustración 14. Tiempos Máximos de Compilación por Pipeline. 47
Ilustración 15. Tiempos Máximos de Compilación por Pipeline 48
Ilustración 16. Estado de las compilaciones. 48
Ilustración 17. Dashboard de builds con tiempos, estados. 49
Ilustración 18.Dashboard número de compilaciones exitosas y no exitosas. 49
Ilustración 19. Integración continua (Log de las integraciones guardadas). 52
Ilustración 20. Despliegue continuo (Log de los despliegues guardados). 53
Ilustración 22. Distribución de microservicios (API Rest) 55
Ilustración 23. Distribución detallada de los folders del proyecto base. 55
Ilustración 24. Distribución detallada de los folders de la capa de Dominio. 56
Ilustración 25. Distribución detallada de los folders de la capa de modelos. 57
Ilustración 26. Distribución detallada de los folders de la capa de repositorio. 57
Ilustración 27. Servicios en Swagger. 58
Ilustración 28. Dashboard principal informativo, con actualizaciones. 59
Ilustración 29. Dashboard de builds. 60
Ilustración 30. Plataforma demo Azure DevOps. 61
Ilustración 31. Pruebas Unitarias. 62
Ilustración 32. Prueba de visualización de Dashboard. 63
Ilustración 33. Visualización Dashboard Grafana. 64
Ilustración 34. Prueba de rendimiento. 66
Ilustración 35. Prueba de calidad general con lighthouse de Chrome DevTools. 68

INDICE DE TABLAS

TABLA 2 ESTADO DEL ARTE 10
TABLA 3. HERRAMIENTAS DE DESARROLLO. 22
TABLA 4. ESPECIFICACIÓN DE RECURSOS FÍSICOS. 22
TABLA 5. ESPECIFICACIÓN DE RECURSOS HUMANOS 23
TABLA 6. COSTO DE RECURSOS HUMANO. 24
TABLA 7. COSTOS DE HERRAMIENTAS DE DESARROLLO. 25
TABLA 8. VALOR TOTAL DE COSTE DEL DESARROLLO. 26
TABLA 9. ROLES. 30
TABLA 10. SPRINTS. 31
TABLA 11. ACTIVIDADES POR ROL 33
TABLA 12.CONSUMO DE API'S. 33
TABLA 13. ALMACENAMIENTO DE INTEGRACIONES Y DESPLIEGUES CONTINUOS EN BD 34
TABLA 14. VISUALIZACIÓN EN GRAFANA LABS. 34
TABLA 15.DEFINICIÓN SPRINT 1 36
TABLA 16. PRUEBAS SPRINT 1. 37
TABLA 17. DEFINICIÓN SPRINT 2 38
TABLA 18. PRUEBAS SPRINT 2. 39
TABLA 19. DEFINICIÓN SPRINT 3. 40
TABLA 20. PRUEBAS SPRINT 3. 41
TABLA 21.DEFINICIÓN SPRINT 4 42
TABLA 22. PRUEBAS SPRINT 4. 43
TABLA 23. DEFINICIÓN SPRINT 5. 44
TABLA 24. PRUEBAS SPRINT 5. 45
TABLA 25. DEFINICIÓN SPRINT 6. 45
TABLA 26. PRUEBAS SPRINT 6. 50
TABLA 27. DEFINICIÓN SPRINT 7. 51
TABLA 28. PRUEBAS SPRINT 7. 53

RESUMEN

En el presente trabajo se podrá encontrar el proceso de diseño y desarrollo de un aplicativo que servirá como base para una integración de diferentes resultados de compilaciones de proyectos empresariales, en el que aplicando metodologías especializadas se propone una solución tecnológica que brinde una mejora al proceso de monitoreo o vigilancia en los diferentes estados de las integraciones y despliegues de Microsoft Azure DevOps como servicio presente en el ciclo de vida del software.

PALABRAS CLAVE

Microsoft Azure, DevOps, Grafana, Monitoreo, Dashboard, Software, arquitectura.

ABSTRACT

In this work you will find the process of design and development of an application that will serve as the basis for an integration of different results of compilations of business projects, in which applying specialized methodologies a technological solution that provides an improvement to the process of monitoring or surveillance in the different states of the integrations and deployments of Microsoft Azure DevOps as a service present in the life cycle of the software is proposed.

KEYWORDS

Microsoft Azure, DevOps, Grafana, Monitoring, Dashboard, Software, Architecture.

INTRODUCCIÓN

Hay muchas empresas que se especializan en brindar servicios de soluciones de software y proveen sus propios servicios, así mismo, en el ciclo de vida del software existen una gran cantidad de herramientas que tratan de solventar la mayor cantidad de necesidades para cada organización, estás necesidades van desde la integración y despliegue continuos del software, durante toda la etapa se requiere un monitoreo para el análisis de cada uno de los procesos de compilación y despliegue del desarrollo pero cuando el número de proyectos crecen dentro de una organización esta tarea se vuelve dificultoso para llevar por una persona y/o equipo.

En la actualidad se trata de utilizar una herramienta que esté presente en el ciclo de vida del software, así mismo se pueda integrar con las herramientas líderes en el mercado, que pueda proporcionar una escalabilidad, usabilidad, monitoreo de forma sencilla, también que logre integrarse de manera rápida con servicios en la nueve, del mismo modo que no genere un alto costo para la organización y es aquí donde las organizaciones puedan tomar decisiones que afecten las fases de planeamiento, desarrollo, entrega y uso del software. Cada fase depende de las demás donde de algún modo todos los roles están involucrados en todas las fases.

Teniendo en cuenta lo anterior, una forma efectiva de hacer uso de una herramienta que este implicada en todo el ciclo de vida del software es Azure DevOps(SaaS), que provee integración con las herramientas líderes del mercado, además que, cuando los activos de una organización crezcan en gran cantidad, se pueda monitorear los procesos de compilación y despliegue de cada uno, para ello Azure DevOps ofrece un conjunto de API’s que se pueden consumir para solucionar el problema de monitorio de las fases anteriormente mencionadas, donde la obtención de datos se puedan mostrar de una manera simple para la toma de decisión en cada fase y que este involucre a los roles de la organización.

1. FASE I: Planeación, definición y organización

i. TÍTULO DEL PROYECTO

DESARROLLO DE UN DASHBOARD PARA LA CONSULTA DE INTEGRACIONES Y DESPLIEGUES CONTINUOS EN LA PLATAFORMA DE AZURE DEVOPS

1.2. TEMA

En este proyecto se muestra la definición y características de una propuesta de diseño de un proyecto de software, aplicada al monitoreo de la integración y despliegue continuo, donde esta información pueda ser analizada de forma rápida y sencilla para la toma de decisiones.

1.3. Planteamiento del problema

1.3.1. DESCRIPCIÓN DEL PROBLEMA

En la actualidad las empresas de software han adquirido la cultura DevOps, esto implica una colaboración más estrecha y una responsabilidad compartida entre el desarrollo y las operaciones por los productos que crean y mantienen. Esto ayuda a las empresas a alinear a su gente, procesos y herramientas hacia un enfoque de cliente más unificado; el beneficio más importante al adoptar una cultura DevOps son los lanzamientos de software optimizados, frecuentes y de alta calidad. Esto no solo aumenta el rendimiento de la empresa, sino también la satisfacción de los empleados.[1]

[1] aws.amazon. ¿Qué es DevOps?. https://aws.amazon.com/es/devops/what-is-devops/. [Consulta: miércoles, 16 de noviembre de 2022]

Debido a la rápida demanda del mercado por la adopción de la cultura DevOps, generalmente requiere que las personas y equipos realicen cambios significativos en la forma que trabajan, durante ese proceso, es imperativo la aceptación de la gerencia y las personas encargadas de la supervisión del proceso de automatización, en este punto es donde los lanzamientos automatizados, integración y despliegue continuo hacen parte, crecen en gran medida dando una entrega de software rápido y efectivo, donde se alcancen altos estándares de calidad, pero cuando al crecer este proceso automático, también se dejan de lado el monitorear resultados, revisar la cantidad de ejecuciones exitosas, se abandona el optimizar los procesos en los pipelines, los cuales pueden aumentar su tiempo de ejecución y al no llevar una trazabilidad crea un cuello de botella en el ciclo de vida del software; según infoQ, las empresas que adoptan el monitoreo necesitan ver sus datos de manera eficiente y sencilla para la toma de decisiones[2].

Con base a lo mencionado anteriormente, se presenta un problema de monitoreo, consolidación de información de las ejecuciones, se presenta un problema de revisión de registros sobre todo cuando las organizaciones crecen y tienen muchas ejecuciones de integración y despliegue continuo.

1.3.2. FORMULACIÓN DEL PROBLEMA

¿El desarrollo de una aplicación de software permitirá llevar el seguimiento y registro de las compilaciones y despliegues continuos para tener un historial que pueda ser de ayuda para el monitoreo, control y toma de decisiones dentro de las organizaciones?

1.4. Objetivos

[2] infoQ. Full-Stack Observability with Grafana and Azure Monitor. https://www.infoq.com/news/2022/05/azure-managed-grafana-preview/. [Consulta: miércoles, 16 de noviembre de 2022]

1.4.1. OBJETIVO GENERAL

Desarrollar una aplicación que permita monitorear el registro de las integraciones y despliegues continuos del desarrollo de software para la toma de decisiones.

1.4.2. OBJETIVOS ESPECÍFICOS

- Analizar los diferentes servicios y API's de la plataforma Azure DevOps que provee información de las integraciones y despliegues continuos cuando se desarrolla software en esta plataforma.
- Diseñar un módulo de consulta de procesos DevOps, que permita visualizar las integraciones y despliegues continuos realizados por una organización en la plataforma Azure DevOps.
- Implementar en Granafa Labs los dashboards con la información procesada resultante del consumo de los API's de los procesos DevOps.
- Creación de un Dashboard en Grafana para la consulta de manera visual de la información que la aplicación monitorea.
- Crear un submódulo en Grafana para el monitoreo de la frecuencia de implementación de las integraciones y despliegues continuos (reporte detallado).
- Crear consultas SQL, para realizar las transacciones en los submódulos orquestados en Grafana.

1.5. Alcances y Limitaciones

1.5.1. ALCANCES

- Se planea un alcance inicial de aplicativos de la plataforma de Microsoft Azure DevOps.

- Entrega de resultados gráficos con ayuda de Grafana.
- Se garantiza la obtención de información de las compilaciones y despliegues de los proyectos alojados en la plataforma de Azure DevOps.

1.5.2. LIMITACIONES

- Debido a que solo se realiza la propuesta para la plataforma Azure DevOps, el desarrollo va enfocado solamente a esta plataforma.
- Los resultados de las consultas de origen de datos están limitados a 80 MB por parte de Grafana.
- Para realizar el monitoreo, todos los proyectos deben tener un pipeline para compilación y/o despliegue.
- El proyecto este enfocado en diseñar y crear un modelo para el uso de las API's de Azure DevOps.
- Debido a que solo se obtendrá información de la plataforma Azure DevOps, el proyecto no modificará particularidades de la plataforma.

1.6. Justificación

En Colombia, con el auge de desarrollo de software y el constante avance en las tecnologías, se ha tomado por las empresas medidas poco automatizadas para poder medir o poder generar reportes de despliegues, compilaciones, módulos.

En las diferentes organizaciones de desarrollo se maneja cada vez más una mayor cantidad de proyectos o mayor número en la cantidad de empleados trabajando sobre el mismo proyecto, se ha visto perdido un poco la trazabilidad de cada actualización realizada o de cada despliegue de algún módulo y el jefe debe es de confiar en resultados generales o asignando cierto apartado únicamente al empleado a evaluar sin poder tener algún sistema automatizado que nos pueda generar una trazabilidad, un reporte de cada despliegue cada cambio con el fin de poder tomar acciones

inmediatamente sin tener que esperar a que finalice una etapa o el proyecto ya en general.

1.7. Marco referencial

1.7.1. ESTADO DEL ARTE

Para llevar a cabo este proyecto es necesario considerar su base de referencia, por lo que se dan a conocer los siguientes proyectos relacionados, sus descripciones y su posible impacto en el desarrollo.

Título	Autor	Enlace	Descripción
Propuesta de diseño de un dashboard para optimizar la planificación de los recursos y perfeccionar el control de los KPI's en la obra Francisco I de la	Gutiérrez Villa, Matías	https://repositorio.unab.cl/xmlui/handle/ria/6676	El proyecto tiene como objetivo optimizar la planificación de los recursos, mejorar el seguimiento y control de los KPI's y perfeccionar la toma de decisiones de la Constructora Beltec en el proyecto "Francisco I" a través del desarrollo de una maqueta funcional de Dashboard. Se diseño de un sistema de control operativo para la empresa, en la base al desarrollo de un bosquejo de Cuadro de Mando Operativo el cual contendrá los principales indicadores de rendimiento de gestión. Luego de analizar si estos

constructora BELTEC			indicadores cumplen con los objetivos estratégicos se creará una maqueta funcional de Dashboard para la Oficina Técnica del proyecto. Se pretende mejorar el seguimiento y control de los KPI's y que se perfeccione la toma de decisiones aportando conocimiento importante y análisis de gráficas
DASHBOARD ASSISTANT: HERRAMIENTA PARA ASISTIR LA CREACIÓN DE TABLEROS DE CONTROL EFECTIVOS	Iván Duarte Bernal	https://repositorio.uniandes.edu.co/bitstream/handle/1992/40324/u808194.pdf	Este proyecto diseña y desarrolla una aplicación web que ayuda a la creación de dashboards aplicando las buenas prácticas de comunicación efectiva con datos. La aplicación busca guiar al usuario, haciendo que este se preocupe por el mensaje a comunicar y no en la selección ni en el diseño de las visualizaciones. Con esta también es posible contar historias a partir de los tableros creados (proceso conocido como storytelling). La aplicación fue desarrollada en React (front-end) y Node.js (back-end) utilizando una gramática de visualización (Vega) para la generación de los gráficos.
EL USO DE DASHBOARD EN LA TOMA DE	Hernán Alberto Báez Moreno	http://fade.espoch.edu.ec/docs/contauditar/Articulo8.pdf	El autor planta como desde un sistema de Dashboard se puede lograr un mejor desempeño y así tener una mejor proyección y

DECISIONES EMPRESARIALES CASO PRÁCTICO EMPRESA REPMAJUSA			mejores resultados, dándonos información al instante de las diferentes variables que pueden hacer crecer la rentabilidad y crecimiento continuo con un buen enfoque en este caso más específicamente refiriéndose a la empresa REPMAJUSA
LA IMPORTANCIA DE UN DASHBOARD INTELIGENTES	EGOSBI	https://www.egosbi.com/importancia-dashboards-inteligentes/	En el artículo se plantea que un dashboard es más efectivo que una gráfica simple porque es personalizable, permite tener toda la información en un solo lugar, puede ser ramificado, permitir profundizar en detalles, mostrar áreas de oportunidad, ser intuitivo y fácil de usar, proporcionar transparencia y accesible a través de dispositivos móviles.
DASHBOARD PARA EL SOPORTE DE DECISIONES EN UNA EMPRESA DEL SECTOR MINERO	Universidad Técnica de Machala	[PDF] Dashboard para el soporte de decisiones en empresas del sector minero (researchgate.net)	El trabajo se enfoca en implementar un sistema de Inteligencia de Negocios (BI) para una empresa minera en la provincia de El Oro. La metodología utilizada combina las mejores prácticas de las metodologías Hefesto y Kimball. El proceso incluye el análisis de los requerimientos, la identificación de los indicadores clave de desempeño (KPI's), el diseño del

			almacén de datos y la implementación de un dashboard. Utilizando herramientas opensource como Pentaho y Postgres, se logró un software de panel de control que consolida la información clave y proporciona la capacidad de análisis multidimensional en línea para apoyar las decisiones de los ejecutivos mineros.
Client Dashboard	Manipal Academy of Higher Education (MAHE)	https://impressions.manipal.edu/mit/67/	Este proyecto tiene como objetivo crear un microservicio que proporcione una API REST sin estado accesible a través de HTTP para ser utilizada en una página web estándar. El enfoque es para hacer que la aplicación sea fácil de construir y mantener, dividiéndola en componentes más pequeños. El marco Spring de JAVA, Microsoft Azure Cosmos DB y Microsoft Azure SQL, Angular 7.0, JWT y Spring MVC son las tecnologías utilizadas en este proyecto. La autenticación JWT se está implementando para asegurar la seguridad de los puntos finales. La finalidad del proyecto es garantizar la utilización y seguridad óptima de los datos almacenados en las fuentes de datos, así como también

			proporcionar una interfaz de usuario para acceder fácilmente a la información. Este proyecto se integrará con un proyecto existente que predice puntajes de anuncios y ayudará a los clientes a verificar puntajes y otra información relacionada a sus anuncios.

Tabla 1 Estado del arte

1.7.2. MARCO TEÓRICO

Dentro de las herramientas tecnológicas y estrategias que se aplicarán en este proyecto, se tienen en cuenta las siguientes:

- **SQL Server 2019**

  SQL Server 2019 abarca todo esto en un solo producto, ofrece un sistema de escalabilidad vertical y horizontal con soporte integrado para Big Data (Apache Spark, Data Lake), estado de la tecnología de virtualización de datos de última generación y con capacidades integradas de aprendizaje automático.

  SQL Server 2019 está diseñado para resolver los desafíos del profesional de datos moderno, incluidos:

  - Almacene datos empresariales en un lago de datos y ofrezca datos generales de capacidad de consulta de SQL y Spark.
  - Reducir la necesidad de aplicaciones de extracción, transformación y carga (ETL) al eliminar el movimiento de datos.
  - Integre y proteja las aplicaciones de aprendizaje automático con un rendimiento escalable.

- Reduzca la necesidad de cambios en la aplicación y la consulta para aumentar el rendimiento.
- Aumente la computación confidencial de datos a través de enclaves de hardware.
- Aumente el tiempo de actividad y la disponibilidad de aplicaciones y bases de datos a través de características como ADR (recuperación avanzada de bases de datos)
- Extienda el poder del lenguaje T-SQL de una manera segura y robusta.
- Ejecute aplicaciones e implemente bases de datos en múltiples sistemas operativos y plataformas con compatibilidad.
- Reduzca el riesgo de actualizaciones mientras usa las nuevas capacidades de SQL cuando esté listo a través de niveles de compatibilidad de base de datos incorporados.

- **NetCore 3.1**

  .Net Core es una nueva versión de .Net framework, desarrollada por Microsoft. Es un marco de código abierto para desarrollar ventanas, aplicaciones web, servicios y aplicaciones móviles y se puede ejecutar en Windows, Mac o Linux. Las aplicaciones .Net Core pueden ejecutarse tanto en .NET CORE como en .NET Framework tradicional (.NET framework 4.x). Se reescribió completamente desde cero y se lanzó inicialmente como .Net 5, pero luego se le cambió el nombre a .NET CORE 1.0.

  .NET Core 3.1 también es conocido por Blazor, la tecnología ASP.NET Core que utiliza WebAssembly como destino de compilación del navegador para lenguajes de programación de orden superior para que puedan usarse en el desarrollo web en lugar de JavaScript. NET Core 3.1 es una versión con soporte a largo plazo (LTS).

Dentro de las principales características se encuentran:

- Multiplataforma: puede ejecutarse en Windows, Linux y Mac.
- Unificado: ofrece un único componente unificado como la biblioteca .NET Standard para todas las plataformas con el mismo código, los mismos lenguajes y las mismas herramientas.
- Mejor rendimiento: .Net core no depende de system.web.dll para la comunicación entre el navegador y el servidor. En .Net core todo son paquetes que necesitamos para nuestra aplicación. Los paquetes reducen la canalización de solicitudes y mejoran el rendimiento de la aplicación.
- Ligero: es un marco liviano. Permite a los desarrolladores implementar bibliotecas y componentes que se necesitan en un conjunto de paquetes.
- Inyección de dependencia: la inyección de dependencia está integrada en .Net Core.
- Hosting: la aplicación .Net Core se puede alojar en varios servidores web, como IIS, Apache, Docker, etc. No depende de IIS.
- Código abierto: es un marco de código completamente abierto. Eso significa que es gratuito y su código fuente está disponible.
- Capacidad de prueba: la prueba unitaria es muy fácil en las aplicaciones principales de .Net.
- Control de versiones de aplicaciones en paralelo: .Net Core se ejecuta en .NET Core, que admite la ejecución simultánea en varias versiones de aplicaciones.[3]

- **Grafana**

[3] c-sharpcorner. Getting Started With .NET Core 3.1 - Part One. https://www.c-sharpcorner.com/article/getting-started-with-net-core-3-1-part-one/. [Consulta: sábado, 12 de noviembre de 2022]

Grafana es una solución de código abierto para ejecutar análisis de datos, extraer métricas que dan sentido a la enorme cantidad de datos y monitorear aplicaciones con la ayuda de paneles personalizables.

Grafana se conecta con todas las fuentes de datos posibles, comúnmente conocidas como bases de datos como Graphite, Prometheus, Influx DB, ElasticSearch, MySQL, PostgreSQL, etc.

Grafana, al ser una solución de código abierto, también permite escribir complementos desde cero para la integración con varias fuentes de datos diferentes.[4]

Dentro de las principales características se encuentran:

- El tablero está bastante equipado y evoluciona continuamente para dar sentido a los datos complejos. Desde mostrar gráficos hasta mapas de calor, histogramas, mapas geográficos. La herramienta tiene una gran cantidad de opciones de visualización para comprender los datos según nuestros requisitos comerciales.
- Grafana tiene soporte nativo para un gran número de base de datos.
- Tiene gran escalabilidad, ya que se puede alojar en cualquier plataforma de la nube.
- Tiene soporte integrado para grafito y expresiones como agregar, filtrar, avg, min, max funciones, etc. para obtener datos personalizados.
- Tiene soporte integrado para Influx DB, Prometheus, ElasticSearch y CloudWatch.

[4] Scaleyourapp. What is Grafana? Why Use It? Everything You Should Know About It. https://scaleyourapp.com/what-is-grafana-why-use-it-everything-you-should-know-about-it/. [Consulta: sábado, 12 de noviembre de 2022]

- **Azure DevOps**

  Azure DevOps es una plataforma de software como servicio (SaaS) de Microsoft que proporciona una cadena de herramientas de DevOps integral para desarrollar e implementar software. También se integra con la mayoría de las herramientas líderes en el mercado y es una excelente opción para orquestar una cadena de herramientas DevOps.

  Azure DevOps admite una cultura colaborativa y un conjunto de procesos que reúnen a desarrolladores, administradores de proyectos y colaboradores para desarrollar software. Permite a las organizaciones crear y mejorar productos a un ritmo más rápido de lo que pueden con los enfoques tradicionales de desarrollo de software.

  Azure DevOps Services admite la integración con repositorios de GitHub.com y GitHub Enterprise Server.

  Azure DevOps Services también proporciona acceso a los servidores de compilación e implementación en la nube, así como a Application Insights.[5]

  Azure DevOps proporciona una colección de API's para diferentes usos y beneficios del usuario.[6]

  Dentro de las principales características se encuentran:

  - Configuración rápida.
  - Operaciones sin mantenimiento.
  - Colaboración sencilla entre dominios.
  - Escalado elástico.
  - Seguridad sólida.

[5] learn.microsoft.com. ¿Qué es Azure DevOps?. https://learn.microsoft.com/es-mx/azure/devops/user-guide/what-is-azure-devops?view=azure-devops. [Consulta: sábado, 12 de noviembre de 2022]
[6] Learn.microsoft.com. Azure DevOps Services REST API Reference. https://learn.microsoft.com/en-us/rest/api/azure/devops/?view=azure-devops-rest-7.1&viewFallbackFrom=azure-devops-rest-5.0. [Consulta: sábado, 02 de enero del 2023]

- **Azure DevOps Services REST API 5.1**

  La API REST de Azure DevOps Services es un conjunto de puntos de conexión de servicio que admiten una serie de operaciones HTTP (métodos), que proporcionan acceso de creación, recuperación, actualización o eliminación del acceso a los recursos del servicio.[7]

  La Azure DevOps Services REST API 5.1 es una interfaz de programación de aplicaciones (API) que permite a los desarrolladores interactuar con Azure DevOps Services para automatizar y personalizar el proceso de construcción y entrega de software.

  Dentro de esta API, existe una función llamada Builds - get, que permite obtener información específica sobre un build (compilación) en particular.

  Para utilizar esta función, el desarrollador debe enviar una solicitud HTTP GET a través de la API, especificando el ID del build que desea obtener. La API devolverá entonces un objeto JSON que contiene información detallada sobre el build, incluyendo su estado actual, los artefactos generados, los errores o problemas encontrados durante la compilación, entre otros detalles.

  La documentación ofrece la siguiente URL de solicitud HTTP para obtener los builds de la plataforma de Azure DevOp; La URL se compone de varios componentes. {organization} y {project} son los nombres de la organización y el proyecto en Azure DevOps, respectivamente. {buildId} es el ID del build específico que se está buscando *"GET https://dev.azure.com/{organization}/{project}/_apis/build/builds/{buildId}?api-version=5.1"*

1.7.3. MARCO CONCEPTUAL

[7] learn.microsoft.com. Referencia de la API rest de Azure DevOps Services. Introducción a las API REST para Azure DevOps Services y Azure DevOps Server - Azure DevOps Services REST API | Microsoft Learn. [Consulta: miércoles, 05 de mayo del 2023]

Inicialmente, debemos entender un término que es muy popular hoy en día, DevOps, que es el nombre que se da a una cultura, que se refiere a un marco de trabajo y una filosofía en constante evolución que promueve un mejor desarrollo de aplicaciones en menos tiempo y la rápida publicación de nuevas o revisadas funciones de software o productos para los clientes.

Bajo un modelo de DevOps, los equipos de desarrollo y operaciones ya no están “aislados”. A veces, los dos equipos se fusionan en uno solo, donde los ingenieros trabajan en todo el ciclo de vida de la aplicación, desde el desarrollo y las pruebas hasta la implementación y las operaciones, y desarrollan una variedad de habilidades no limitadas a una única función.

En algunos modelos de DevOps, los equipos de control de calidad y de seguridad también se integran más con el desarrollo y las operaciones e intervienen durante todo el ciclo de vida de la aplicación. Cuando la seguridad es la prioridad de todos los miembros de DevOps, a veces se conoce como operaciones de seguridad de desarrollo.

Sabiendo que la cultura DevOps está presente en todo el ciclo de vida del software, la integración y despliegue continua de un gran número de aplicaciones por una organización presenta una problemática en la cual no se puede controlar la cantidad, no se pueden llevar métricas; la implementación de un dashboard es una solución para obtener métricas, tomar decisiones y ajustar los procesos de integración y despliegue continuos.

1.7.4. MARCO LEGAL

Ya que en el proyecto se llevar acabo un registro detallado de los procesos de integración y despliegue continuos, se hará uso de .NetCore 3.1. la cual es una plataforma de desarrollo de código abierto, sin costos de licencia, SQL Server 2019 Developer Edition la cual permite a los desarrolladores compilar cualquier tipo de aplicación en SQL Server. Incluye toda la funcionalidad de la edición

Enterprise, pero tiene licencias para usarse como sistema de prueba y desarrollo, no como un servidor de producción, Grafana la cual es una plataforma de desarrollo de código abierto, Azure DevOps, al ser una plataforma SaaS(Software como servicio), tiene costos por servicios usados, para el desarrollo de este proyecto se usara en los usuarios los planes **basic**, que es gratis para los primeros 5 usuarios, para el consumo de información de compilaciones Azure DevOps ofrece 1800 segundos gratuitos, las compilaciones no tienen costos ni servicios de alojamiento de repositorios.

1.8. Solución Tecnológica

La solución tecnológica que se propone a la problemática previamente establecida es el diseño y desarrollo de una aplicación que permita consultar las diferentes integraciones y despliegues continuos realizados en uno o varios proyectos desarrollados en Azure DevOps. Para ello, la aplicación realizará el consumo de las API's "Azure DevOps Services REST API" ofrecidas por Microsoft como PaaS (Platform as a Service); la colección de API's que se utilizan en la implementación del proyecto son de categoría "Builds".

Inicialmente, el proceso para obtener la información de la organización en la plataforma Azure DevOps se realiza mediante el consumo de los procesos de integración y despliegue continuo dentro de la misma, una vez que los datos sean obtenidos se alojan en una base de datos SQL.

La visualización de resultados se realizará en Grafana, donde se creará un Dashboard con una serie de submódulos que permiten visualizar y categorizar la información específica de las compilaciones, número de compilaciones, tasa de éxito de las integraciones y despliegues.

Con la información cargada en los Dashboards, se pretende que la organización plantee estrategias de negocio que apoyen la toma de decisiones en los

proyectos que se requieran donde, en el ciclo de vida DevOps, se puedan evidenciar cuellos de botella en los procesos de integración y despliegue continuo, fallos en las compilaciones y número de compilaciones.

1.8.1. METODOLOGÍA DEL PROYECTO

Scrum es un proceso en el que se utiliza regularmente un conjunto de mejores prácticas para trabajar juntos como grupo a fin de obtener el mejor resultado posible de un plan.

Debido a esto, Scrum es especialmente adecuado para proyectos en entornos complejos, donde se necesitan resultados rápidos, en la que los requisitos cambian o están mal definidos, donde la innovación, la competitividad, la flexibilidad y la productividad son primordiales.

Scrum también se utiliza para resolver situaciones en que no se está entregando al cliente lo que necesita, cuando las entregas se alargan demasiado, los costes se disparan o la calidad no es aceptable, cuando se necesita capacidad de reacción ante la competencia, cuando la moral de los equipos es baja y la rotación alta, cuando es necesario identificar y solucionar ineficiencias sistemáticamente o cuando se quiere trabajar utilizando un proceso especializado en el desarrollo de producto.[8]

Roles Principales Product Owner

El Product Owner representa la voz del cliente. Se asegura de que el equipo Scrum trabaje de forma adecuada desde la perspectiva del negocio. El Product Owner escribe historias de usuario, las prioriza, y las coloca en el Product Backlog.

[8] Proyectosagiles. Qué es SCRUM. https://proyectosagiles.org/que-es-scrum/. Consultado: martes, 25 de octubre de 2020].

ScrumMaster (o Facilitator)

El Scrum es facilitado por un ScrumMaster, cuyo trabajo primario es eliminar los obstáculos que impiden que el equipo alcance el objetivo del sprint. El ScrumMaster no es el líder del equipo (porque ellos se autoorganizan), sino que actúa como una protección entre el equipo y cualquier influencia que le distraiga. El ScrumMaster se asegura de que el proceso Scrum se utiliza como es debido. El ScrumMaster es el que hace que las reglas se cumplan[9].

Equipo Scrum

El equipo tiene la responsabilidad de entregar el producto. Es recomendable un pequeño equipo de 5 a 9 personas con las habilidades transversales necesarias para realizar el trabajo (análisis, diseño, desarrollo, pruebas, documentación, etc.).

Roles Auxiliares

Los roles auxiliares en los "equipos Scrums" son aquellos que no tienen un rol formal y no se involucran frecuentemente en el "proceso Scrum", sin embargo, deben ser tomados en cuenta. Un aspecto importante de una aproximación ágil es la práctica de involucrar en el proceso a los usuarios, expertos del negocio y otros interesados ("stakeholders"). Es importante que esa gente participe y entregue retroalimentación con respecto a la salida del proceso a fin de revisar y planear cada sprint.

Stakeholders (Clientes, Proveedores, Vendedores, etc.)

Son las personas que hacen posible el proyecto y para quienes el proyecto producirá el beneficio acordado que justifica su desarrollo. Únicamente participan directamente durante las revisiones del "sprint".

Administradores (Managers)

[9] Scrum619. SCRUM https://scrum619.wordpress.com/2016/02/26/roles/ [Consulta: martes, 25 de octubre de 2022]

Son los responsables de establecer el entorno para el desarrollo del proyecto.

Proceso

En primer lugar, se define el Product Backlog, lo que nos permitirá realizar nuestros Sprints más adelante.

Product Backlog: Es una “wish list” sobre las funcionalidades del producto. Es elaborado por el Product Owner y las funciones están priorizadas según lo que es más y menos importante para el negocio. El objetivo es que el Product Owner responda la pregunta “¿Qué hay que hacer?”.

Sprint Backlog: Es un subconjunto de ítemes del Product Backlog, que son seleccionados por el equipo para realizar durante el Sprint sobre el que se va a trabajar. El equipo establece la duración de cada Sprint.

Sprint Planning Meeting: Esta reunión se hace al comienzo de cada Sprint y se define cómo se va a enfocar el proyecto que viene del Product Backlog las etapas y los plazos. Cada Sprint está compuesto por diferentes features. Por ejemplo, decidimos que los features del primer Sprint son: diseño del logo, definición, colores y contenido multimedia.

Daily Scrum o Stand-up Meeting: Es una reunión breve que se realiza a diario mientras dura el periodo de Sprint. Se responden individualmente tres preguntas: ¿Qué hice ayer?, ¿Qué voy a hacer hoy?, ¿Qué ayuda necesito? El Scrum Master debe tratar de solucionar los problemas u obstáculos que se presenten.

Sprint Review: Se revisa el sprint terminado, y ya debería haber un avance claro y tangible para presentárselo al cliente.

Sprint Retrospective: El equipo revisa los objetivos cumplidos del Sprint terminado. Se anota lo bueno y lo malo, para no repetir los errores. Esta

etapa sirve para implementar mejoras desde el punto de vista del proceso del desarrollo.[10]

Scrum-Team



Ilustración 1. Diseño de la metodología Scrum.

1.9. Factibilidad de desarrollo

1.9.1. FACTIBILIDAD TÉCNICA

La aplicación se desarrolla como modelo de ejecución en la nube. Para la codificación del aplicativo, se necesitará equipos de cómputo que soporte los programas a utilizar como lo son: Visual Studio, SQL Server 2019, Grafana, navegador web, entre otros.

En la tabla 2. Se describen las herramientas que se tendrán en cuenta para el desarrollo del proyecto.

HERRAMIENTA	DESCRIPCIÓN

[10] Interpolados. METODOLOGÍA SCRUM. https://interpolados.wordpress.com/2016/01/17/metodologia-scrum/. [Consulta: martes, 25 de octubre del 2022].

VISUAL STUDIO CODE	Entorno para el desarrollo de aplicaciones.
SQL SERVER 2019	Entorno visual de diseño de bases de datos.
GRAFANA	Entorno para la visualización de métricas, que nos permitirá ver datos de forma dinámica y se alimentará de la información recolectada de la plataforma Azure DevOps.
AZURE DEVOPS	Plataforma donde se alojará el código fuente, llevando un control de versión, además se consumirá las API's de integración y despliegue, para ver métricas y tomar decisiones en los pipelines.
POSTMAN	Permite el envío de peticiones HTTP REST

Tabla 2. Herramientas de desarrollo.

En la Tabla 3. Se mencionan las especificaciones de los recursos físicos con los que cuentan los equipos para el desarrollo.

RECURSOS UTILIZADOS	CARACTERISTICAS DE LOS RECURSOS	
PC	Procesador: Intel Core I7. Memoria RAM: 16GB DDR3. Disco Duro: 1 Tera Tarjeta gráfica: GeForce 1080 4GB	Procesador: Intel Core I5 8300H. Memoria RAM: 24GB DDR4. Disco Duro: 2 Tera Tarjeta gráfica: NVIDIA GTX 1050 4GB

Tabla 3. Especificación de recursos físicos.

También se muestra en la Tabla 4. Las características de los recursos humanos que se requirieren.

RECURSO HUMANO	CARACTERISTICAS
DESARROLLADORES	Estudiantes de tecnología en sistematización de datos, con conocimiento en lenguajes de programación orientados a

	objetos y con habilidades en plataformas cloud, manejo de servidores y procesos DevOps.
TUTOR	Ingeniero de sistemas con conocimientos en desarrollo de material educativo computarizado para sistemas telemáticos.

Tabla 4. Especificación de recursos humanos.

De esta manera se resumiría que el proyecto es factible a nivel técnico, ya que se cuenta con los recursos necesarios para desarrollarlo.

1.9.2. FACTIBILIDAD ECONÓMICA

En el análisis de la factibilidad económica se determina el presupuesto de precios de los recursos técnicos y humanos necesarios para el desarrollo del proyecto. En las siguientes tablas se describe la factibilidad económica relacionado con los recursos previamente nombrados junto a una breve descripción.

1.9.2.1. FACTIBILIDAD OPERATIVA - RECURSOS HUMANOS

Los recursos humanos constarán de las tutorías por parte del tutor necesarias para la definición y planeación del software, así como el personal de desarrollo de la aplicación.

Tipo	Descripción	Valor-Hora	Cantidad	Total
Tutor	Asesorías para la realización del proyecto, referente a la metodología.	$ 35.000	15	$ 525.000
Desarrollador	Dos estudiantes de Tecnología en	$ 12.000 c/u	320	$ 7.680.000

	sistematización de datos que realicen la planeación e implementación del proyecto.			
Costo Total - Recursos Humanos				$8.205.000

Tabla 5. Costo de recursos humano.

1.9.2.2. FACTIBILIDAD ECONÓMICA - RECURSOS TÉCNICOS

Los recursos técnicos están dados a los equipos de cómputo, suscripciones a los servicios de Microsoft Azure DevOps y materiales necesarios para la implementación del proyecto.

Tipo	Descripción	Valor Unitario	Cantidad	Total
Computadores	Equipos de escritorio para el desarrollo del software.	$2.600.000	2	$5.200.000
Servicio de electricidad	Electricidad necesaria para el uso de los computadores	$18.000	1 pc – 2 meses	$36.000
Servicio de acceso a internet	Servicio de acceso a internet	$100.000	5 meses	$400.000
Servicio de Microsoft Azure DevOps	Servicios DevOps, en el cual estará alojado el repositorio del proyecto, los procesos de	$0	1800 segundos de compilaciones gratuitas con	$0

	integración y despliegue, la herramienta de la cual se consumirán las API's para la obtención de información		los agentes de Microsoft	
Grafana	Plataforma de visualización de métricas en un Dashboard	$0	N/A	$0
SQL Server Developer Edition	Plataforma para compilar aplicaciones SQL Server, licencia de prueba y desarrollo sin costos.	$0	1	$0
Costo Total - Recursos Técnicos				**$ 5.636.000**

Tabla 6. Costos de herramientas de desarrollo.

1.9.2.3. FACTIBILIDAD ECONÓMICA – COSTO TOTAL

El costo total contiene el costo de los recursos humanos, los recursos técnicos, otros costos asociados a la papelería, uso de internet, entre otros, así como el valor de costos imprevistos a lo largo del desarrollo del proyecto.

Recurso	Valor
Costo Total - Recursos Humanos	$ 8.205.000
Costo Total - Recursos Técnicos	$ 5.636.000

Costo Total - Otros recursos	$ 150.000
Costos imprevistos (10%)	$ 210.00
Costo Total	**$ 14.201.000**

Tabla 7. Valor total de coste del desarrollo.

1.10. Cronograma de actividades

1.10.2. AGENDA PROPUESTA

Para el desarrollo del proyecto se propone un cronograma de 82 días los cuales se dividen en 3 faces las cuales se describen en la ilustración 2.

Nombre	Duracion	Inicio	Terminado
⊟Duración estimada del proyecto	82 days?	14/12/22 9:00	7/04/23 9:00
⊞Fase 1: Inicio del proyecto	12 days	14/12/22 9:00	30/12/22 9:00
⊞Fase 2: Desarrollo de proyecto	60 days?	30/12/22 9:00	24/03/23 9:00
⊞Fase 3: Finalización del proyecto	10 days	24/03/23 9:00	7/04/23 9:00

Ilustración 2. Fases del proyecto

Las actividades de las fases anteriores se mencionan en la ilustración 3 y se desarrollaran en el documento.

Nombre	Duracion	Inicio	Terminado
Duración estimada del proyecto	**82 days?**	**14/12/22 9:00**	**7/04/23 9:00**
Fase 1: Inicio del proyecto	**12 days**	**14/12/22 9:00**	**30/12/22 9:00**
Planificación	7 days	14/12/22 9:00	23/12/22 9:00
Diseño y arquitectura	5 days	23/12/22 9:00	30/12/22 9:00
Fase 2: Desarrollo de proyecto	**60 days?**	**30/12/22 9:00**	**24/03/23 9:00**
Análisis de Sprint	15 days	30/12/22 9:00	20/01/23 9:00
Desarrollo de Sprint	15 days?	20/01/23 9:00	10/02/23 9:00
Validaciones de Sprint	15 days?	10/02/23 9:00	3/03/23 9:00
Ajuste del Sprint	15 days?	3/03/23 9:00	24/03/23 9:00
Fase 3: Finalización del proyecto	**10 days**	**24/03/23 9:00**	**7/04/23 9:00**
Encuestas de usuarios	5 days	24/03/23 9:00	31/03/23 9:00
Cierre	5 days	31/03/23 9:00	7/04/23 9:00

Ilustración 3. Actividades del proyecto

1.10.3. DIAGRAMA DE GANTT

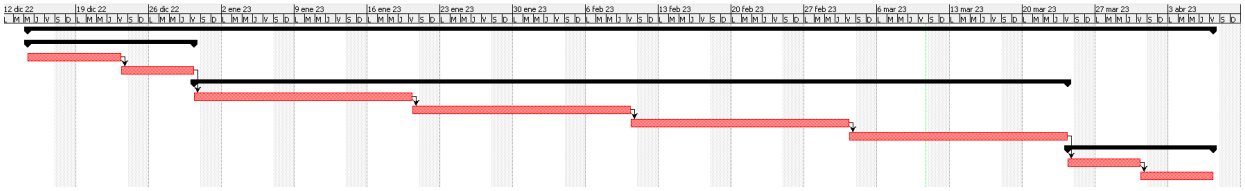


Ilustración 4 Diagrama de Gantt

2. FASE II: Análisis del proyecto

En este capítulo se realiza un breve repaso de la problemática actual y se identifican los roles necesarios para llevar a cabo el proyecto. También se presenta una lista de actividades por rol y se define el Sprint, que es un conjunto de actividades o desarrollos por implementar. Además, se incluye la historia de usuario para determinar qué es lo que busca o espera del aplicativo.

En cuanto a la cultura DevOps, esta implica una colaboración más estrecha y una responsabilidad compartida entre los equipos de desarrollo y operaciones para crear y mantener productos de software. Al adoptar una cultura DevOps, las empresas pueden alinear a su gente, procesos y herramientas hacia un enfoque de cliente más unificado; el beneficio más importante al adoptar una cultura DevOps son los lanzamientos de software optimizados, frecuentes y de alta calidad.

Ya que no existe software o aplicaciones que brinden una solución a la situación planteada anteriormente, necesitaría más información sobre el problema específico que estás tratando de resolver para poder ayudarte mejor. ¿Podrías proporcionar más detalles sobre el problema que estás tratando de resolver?

2.1. Análisis del problema

En la actualidad las empresas dedicadas al desarrollo de software han adoptado la cultura DevOps. Esto implica una colaboración más estrecha y una responsabilidad compartida entre el equipo encargado del desarrollo y el equipo encargado del mantenimiento del producto. Esta práctica ayuda a las empresas a alinear sus procesos y herramientas hacia un enfoque más centrado en el cliente. El beneficio más importante al adoptar esta cultura es que se logran lanzamientos frecuentes y optimizados del software con alta calidad. Esto no solo aumenta el rendimiento empresarial sino también la satisfacción laboral.

Debido a la rápida demanda del mercado por la adopción de esta cultura DevOps, generalmente requiere que las personas y equipos realicen cambios significativos en su forma de trabajo. Durante este proceso es imperativo que tanto la gerencia como las personas encargadas de supervisar el proceso de automatización acepten estos cambios. Según devopstips, "El monitoreo permite identificar problemas y solucionarlos de manera rápida y eficiente, lo que evita interrupciones en el servicio y mantiene la satisfacción del usuario[11]" En este punto es donde los lanzamientos automatizados, integración y despliegue continuo hacen parte del proceso para dar una entrega rápida y efectiva del software con altos estándares de calidad.

Sin embargo, cuando este proceso automático crece también se deja de lado el monitoreo constante para revisar los resultados obtenidos en cada ejecución exitosa o fallida. Además, se abandona la optimización continua del pipeline lo cual puede aumentar su tiempo de ejecución y crear cuellos de botella en el ciclo de vida del software.

según infoQ, "las empresas que adoptan el monitoreo necesitan ver sus datos de manera eficiente y sencilla para la toma de decisiones[12]".

2.2. Identificación de roles

A continuación, se muestra una tabla con los roles correspondientes y la persona responsable de cada uno, junto con una breve descripción de las tareas que deben realizar.

ROL	RESPONSABLE	DESCRIPCIÓN

[11] Devopstips. Devops Monitoring. https://devopstips.com.ar/blog/monitoreo-devops/. [Consulta: lunes, 20 de marzo de 2023]

[12] infoQ. Full-Stack Observability with Grafana and Azure Monitor. https://www.infoq.com/news/2022/05/azure-managed-grafana-preview/. [Consulta: miércoles, 16 de noviembre de 2022]

Scrum Master	Jonatan Stiben Escobar Alzate	La persona encargada de liderar al equipo y guiarlo para que cumpla las reglas y procesos de la metodología es el líder de proyecto.
Product Owner	Jhon Alexander Garcia Caro	El representante de los accionistas y clientes que utilizan el software es el Product Owner. Su enfoque se centra en la parte de negocio y es responsable del ROI del proyecto, es decir, de entregar un valor superior al dinero invertido.
Team	Jonatan Stiben Escobar Alzate Jhon Alexander Garcia Caro	El grupo de profesionales con los conocimientos técnicos necesarios que desarrollan el proyecto de manera conjunta y llevan a cabo las historias comprometidas al inicio de cada sprint son el equipo de desarrollo.

Tabla 8. Roles.

2.3 Sprints

En la siguiente tabla muestra los Sprints con lo que se realiza en el proyecto, con su respectivo nombre y una descripción.

Sprint	Nombre	Descripción
Sprint 1	Configuración de entornos	Configuración de entornos para desarrollo
Sprint 2	Registro de servicios de integraciones continuas (CI)	Consulta de datos de integración continua
Sprint 3	Registro de servicios de despliegues continuos. (CD)	Consulta de datos de despliegues continuas
Sprint 4	Implementación de base de datos	Registro de Base de datos para alojar información
Sprint 5	Creación de procesos almacenados SQL	Creación de SP para las tablas en la base de datos
Sprint 6	Configuración de Grafana para el dashboard.	Configuración de Grafana Labs
Sprint 7	Registro de datos a visualizar en Grafana	Visualización de datos en Grafana Labs

Tabla 9. Sprints.

2.4. Lista de actividades por rol

Se presenta una lista detallada de las tareas asignadas a cada rol, con el objetivo de obtener un orden y una comprensión clara de las responsabilidades de cada miembro del equipo.

ROL	RESPONSABLE	DESCRIPCIÓN
Scrum Master	Jonatan Stiben Escobar Alzate	• Solucionar los conflictos que afecten el avance normal del proyecto.

		• Estimular y motivar al equipo de trabajo. • Promover la autogestión de los colaboradores en el proceso. • Negociar y renegociar las condiciones con el cliente. • Prevenir la intromisión de terceros en las tareas.
Product Owner	Jhon Alexander Garcia Caro	• Comunica las necesidades del negocio al director y su equipo. • Toma decisiones sobre las características funcionales del producto o servicio. • Salvaguarda los intereses del negocio y busca maximizar el valor de la inversión. • Revisa el producto al final de cada iteración. • Propone cambios y adaptaciones al finalizar cada nueva iteración.

Team	Jonatan Stiben Escobar Alzate Jhon Alexander Garcia Caro	• Realizar todas las tareas especificadas en el plan de trabajo. • Utilizar sus habilidades y conocimientos técnicos en beneficio del proyecto.

Tabla 10. Actividades por rol

2.4 Historias de usuario

Para desarrollar las historias de usuario, se preguntó ¿Cuáles son las expectativas para una aplicación que gestione la monitorización de integraciones y despliegues continuos de los proyectos en Azure DevOps, mostrando el estado de estas operaciones y las fechas de uso?; Se proporcionaron sus respuestas, que se registraron en las siguientes tablas.

CONSUMO DE API'S AZURE DEVOPS
Como usuario de la aplicación, quiero poder conectarme a Azure DevOps y obtener las integraciones y despliegues continuos (pipelines) de mis activos, para poder visualizar el estado actual y las fechas de uso.
Criterios de aceptación: • COMO usuario de la aplicación QUIERO poder conectarme a Azure DevOps PARA obtener las integraciones y despliegues continuos (pipelines) de mis activos. • COMO usuario de la aplicación QUIERO visualizar el estado actual y las fechas de uso PARA poder tener una mejor comprensión del estado actual de mis activos.

Tabla 11.Consumo de API's.

ALMACENAMIENTO DE INTEGRACIONES Y DESPLIEGUES CONTINUOS EN UNA BASE DE DATOS SQL
Como usuario de la aplicación, quiero que las integraciones y despliegues continuos se almacenen en una base de datos SQL para poder acceder a ellos de manera más rápida y eficiente.
Criterios de aceptación: • COMO usuario de la aplicación QUIERO que las integraciones y despliegues continuos se almacenen en una base de datos SQL PARA poder acceder a ellos de manera más rápida y eficiente. • COMO usuario de la aplicación QUIERO que los datos almacenados en la base de datos SQL sean precisos y actualizados PARA poder tomar decisiones informadas.

Tabla 12. Almacenamiento de integraciones y despliegues continuos en BD

VISUALIZACIÓN DE RESULTADOS DE PIPELINES EN GRAFANA
Como usuario de la aplicación, quiero poder visualizar los resultados de los pipelines de integración y despliegue continuo en un dashboard en Grafana, para que pueda monitorear su estado de forma eficiente.
• COMO usuario de la aplicación, QUIERO que los resultados de los pipelines de integración y despliegue continuo se muestren en un dashboard en Grafana PARA poder tener una visión rápida y clara del estado de las integraciones y despliegues continuos. • COMO usuario de la aplicación, QUIERO poder filtrar los resultados de los pipelines de integración y despliegue continuo en el dashboard en Grafana POR nombre del pipeline, estado y fecha PARA poder hacer seguimiento a los pipelines específicos que me interesan.

Tabla 13. Visualización en Grafana Labs.

3. FASE III: DISEÑO DEL PROYECTO

En este capítulo se detalla la planificación y diseño de los Sprints establecidos en el capítulo anterior. Se proporciona una serie de tablas para cada Sprint, las cuales contienen la definición, el diseño y las pruebas de validación correspondientes.

3.1 Sprint 1

A continuación, se presenta la reunión de inicio del proyecto, investigación de requisitos, diseño de prototipos.

3.1.1 DEFINICIÓN DE SPRINT 1

<table>
<tr><td>Sprint N.</td><td>1</td><td>Nombre</td><td>Configuración de entornos</td></tr>
<tr><td>Objetivo</td><td colspan="3">Configurar el entorno de desarrollo y crear el proyecto base, así como obtener las credenciales de acceso a las APIs de Azure DevOps.</td></tr>
<tr><td colspan="4">Detalle</td></tr>
<tr><td colspan="4">• Instalación de las herramientas y software necesarios para el proyecto, configuración de las variables de entorno y definición de los roles y permisos del equipo.
• Configuración inicial del repositorio de código, definición de los roles y permisos del equipo y creación de las tareas iniciales.
• Obtención de las credenciales de acceso necesarias para la integración con las APIs de Azure DevOps.</td></tr>
<tr><td colspan="4">Criterios de aceptación</td></tr>
<tr><td colspan="4">• El entorno de desarrollo ha sido configurado correctamente, de acuerdo con las especificaciones técnicas del proyecto.
• Se ha creado el proyecto base en Azure DevOps, incluyendo la definición de los roles y permisos del equipo y las tareas iniciales.</td></tr>
</table>

• Las credenciales de acceso a las APIs de Azure DevOps han sido obtenidas y se han establecido correctamente en el entorno de desarrollo, permitiendo la integración con las APIs para el desarrollo del proyecto.

Tabla 14.Definición Sprint 1.

3.1.2 DISEÑO DE SPRINT 1

Para la configuración de entornos se configuro una instancia en Azure con una base de datos como se muestra en la siguiente ilustración.

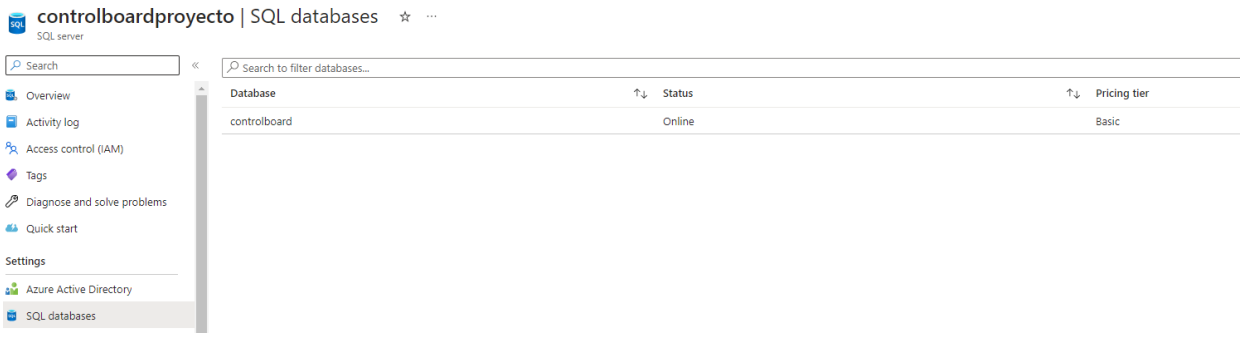


Ilustración 5. Base de datos creada en Azure.

La anterior instancia nos dejara tener una base de datos siempre disponible y actualizada para el consumo de información.

Para el entorno de desarrollo se usó Visual Studio y Visual Studio Code, donde solo se descargó de las páginas oficiales de las herramientas, junto con la versión de .netcore 3.1 https://dotnet.microsoft.com/en-us/download/dotnet/3.1

Se creo una plantilla en Azure DevOps con un correo Microsoft para obtener la plantilla de pruebas, así mismo, se creó el token de acceso y se probó la url de API “”GET https://dev.azure.com/{organization}/{project}/_apis/build/builds/{buildId}?api-version=5.1, como se evidencia en la siguiente ilustración.

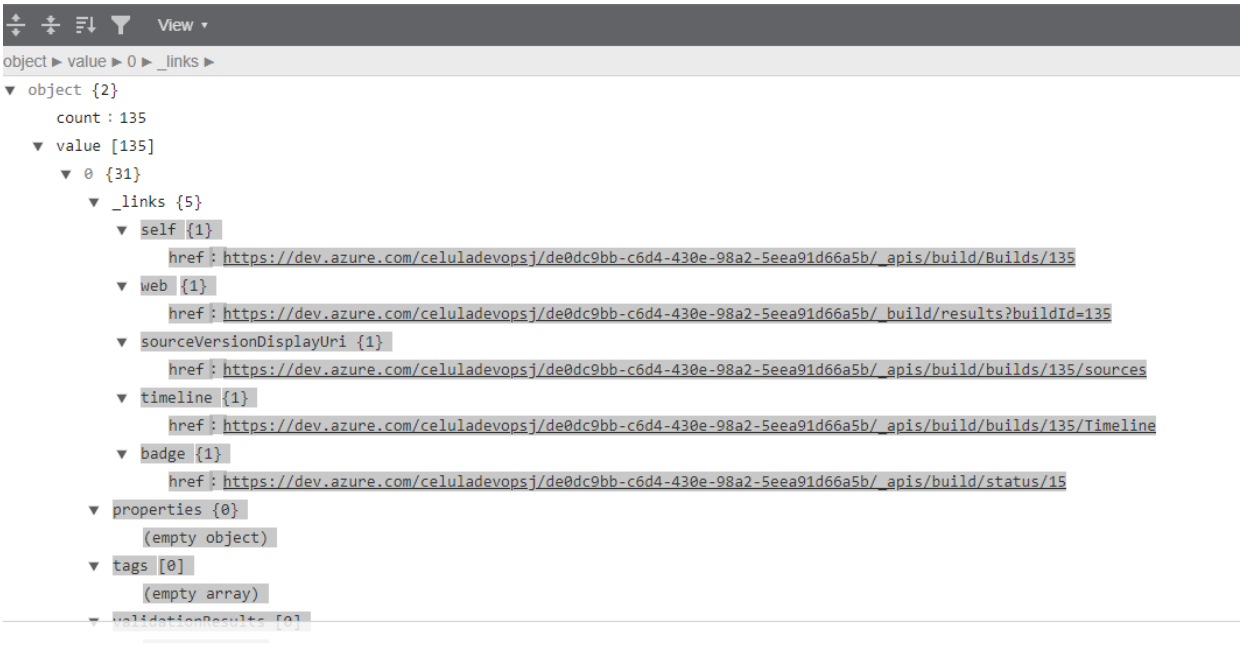


Ilustración 6. lista de Builds y prueba de conexión con las API.

3.1.3 PRUEBA DEL SPRINT 1

En la siguiente tabla se califican los criterios de aceptación del sprint 1 para dar por terminado el sprint una vez se cumplen.

Sprint 1		
Prueba		¿Cumple?
1	El ambiente de base de datos funciona sin errores.	SI
2	El ambiente de desarrollo funciona sin errores.	SI
3	La platilla en Azure DevOps funciona correctamente.	SI
4	El token de Azure DevOps funciona correctamente.	SI

Tabla 15. Pruebas Sprint 1.

3.2 Sprint 2

A continuación, se presenta la reunión de inicio del proyecto, investigación de requisitos, diseño de prototipos.

3.2.1 DEFINICIÓN DE SPRINT 2

Sprint N.	2	**Nombre**	Registro de servicios de integraciones continuas (CI)
Objetivo	Desarrollar la funcionalidad para obtener información de despliegues continuos y registrar los servicios de despliegues continuos en Azure DevOps.		
Detalle			
• Diseñar e implementar la funcionalidad para obtener información sobre los despliegues continuos que se han realizado. • Integrar la funcionalidad desarrollada con las API's de Azure DevOps para obtener información sobre los despliegues continuos. • Documentar y registrar adecuadamente los servicios de despliegues continuos en Azure DevOps.			
Criterios de aceptación			
• La funcionalidad para obtener información de integraciones continuos se ha desarrollado y se integra correctamente con las APIs de Azure DevOps. • El registro de servicios de despliegues continuos está completo y se ha documentado adecuadamente, permitiendo su uso y mantenimiento.			

Tabla 16. Definición Sprint 2

3.2.2 DISEÑO DE SPRINT 2

Se realiza la creación del servicio para obtener la información de las integraciones continuas como se evidencia en la siguiente ilustración.



Ilustración 7. Get Builds (creación de servicio de chequeo de integraciones).

3.2.3 PRUEBA DEL SPRINT 2

En la siguiente tabla se califican los criterios de aceptación del sprint 2 para dar por terminado el sprint una vez se cumplen.

Sprint 2		
Prueba		¿Cumple?
1	La obtención de integraciones continuas cumple los criterios de aceptación	SI
2	Se documenta los servicios correctamente con Swagger	SI
3	Se consume la información de la organización bajo los criterios establecidos	SI

Tabla 17. Pruebas Sprint 2.

3.3 Sprint 3

A continuación, se presenta la reunión de inicio del proyecto, investigación de requisitos, diseño de prototipos.

3.3.1 DEFINICIÓN DE SPRINT 3

Sprint N.	3	**Nombre**	Registro de servicios de despliegues continuos. (CD)
Objetivo	Desarrollar la funcionalidad para obtener información de integraciones continuas y registrar los servicios de integraciones continuas en Azure DevOps.		
Detalle			

• Diseñar e implementar la funcionalidad para obtener información sobre las integraciones continuas que se han realizado. • Integrar la funcionalidad desarrollada con las APIs de Azure DevOps para obtener información sobre las integraciones continuas. • Documentar y registrar adecuadamente los servicios de integraciones continuas en Azure DevOps.
Criterios de aceptación
• La funcionalidad para obtener información de integraciones continuas se ha desarrollado y se integra correctamente con las APIs de Azure DevOps. • El registro de servicios de integraciones continuas está completo y se ha documentado adecuadamente, permitiendo su uso y mantenimiento.

Tabla 18. Definición Sprint 3.

3.3.2 DISEÑO DE SPRINT 3

Se realiza la creación del servicio para obtener la información de los despliegues continuos, documentados con Swagger, como se evidencia en la siguiente ilustración.

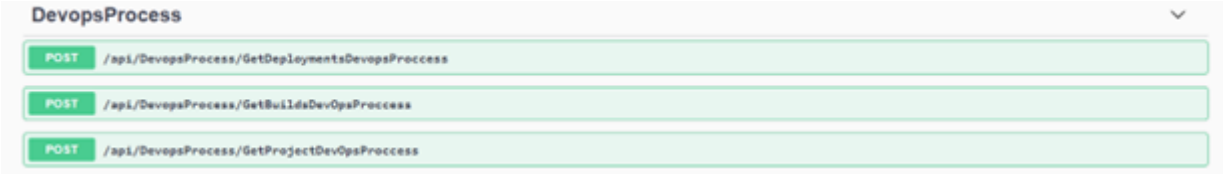


Ilustración 8. Get Deployments (creacion de servicio para obtener la informacion de los despleigues continuos.

3.3.3 PRUEBA DEL SPRINT 3

En la siguiente tabla se califican los criterios de aceptación del sprint 3 para dar por terminado el sprint una vez se cumplen.

Sprint 3		
Prueba		¿Cumple?
1	La obtención de despliegues continuo cumple los criterios de aceptación.	SI

2	Se documenta los servicios correctamente con Swagger.	SI
3	Se consume la información de la organización bajo los criterios establecidos.	SI

Tabla 19. Pruebas Sprint 3.

3.4 Sprint 4

A continuación, se presenta la reunión de inicio del proyecto, investigación de requisitos, diseño de prototipos.

3.4.1 DEFINICIÓN DE SPRINT 4

Sprint N.	4	**Nombre**	Implementación de base de datos
Objetivo	Implementar la base de datos SQL en el proyecto.		
Detalle			
• Diseñar el modelo de base de datos que cumpla con los requisitos de la aplicación. • Implementar el modelo de base de datos en SQL. • Configurar el entorno de base de datos para su uso en el proyecto. • Integrar la base de datos en el proyecto para su uso. • Ejecutar pruebas de integración con la base de datos para garantizar su correcta implementación.			
Criterios de aceptación			
• El modelo de base de datos cumple con los requisitos de la aplicación. • La base de datos se ha implementado correctamente en SQL. • El entorno de base de datos está configurado correctamente. • La integración de la base de datos en el proyecto se ha realizado con éxito.			

• Las pruebas de integración con la base de datos se han ejecutado correctamente, garantizando su correcto funcionamiento en el proyecto.

Tabla 20.Definición Sprint 4.

3.4.2 DISEÑO DE SPRINT 4

Se realiza la creación de las tablas para alojar la información obtenida de las API's en la siguiente ilustración se muestra como están clasificadas.

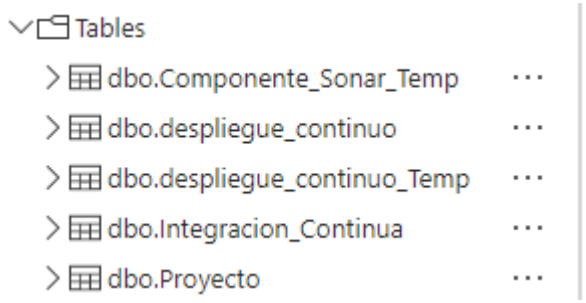


Ilustración 9. Creación de tablas a manejar en la base de datos (Clasificación Base de datos).

Se configura la cadena de conexión en el proyecto para que se comunique con la base de datos y así se pueda alojar información.

```
"ConnectionString": "Server=tcp:controlboardproyecto.database.windows.net,1433;Initial
Catalog=controlboard;Persist Security Info=False;User ID=esco;Password=;MultipleActiveResultSets=False;
Encrypt=True;TrustServerCertificate=False;Connection Timeout=30;",
```

Ilustración 10. Conexión a la Base de datos.

3.4.3 PRUEBA DEL SPRINT 4

En la siguiente tabla se califican los criterios de aceptación del sprint 4 para dar por terminado el sprint una vez se cumplen.

Sprint 4	
Prueba	¿Cumple?

1	La base de datos aloja la información obtenida de los procesos CI/CD.	SI
2	El modelo de Base de datos cumple con los requisitos para la información.	SI
3	La conexión entre los servicios y base de datos funciona correctamente.	SI

Tabla 21. Pruebas Sprint 4.

3.5 Sprint 5

A continuación, se presenta la reunión de inicio del proyecto, investigación de requisitos, diseño de prototipos.

3.5.1 DEFINICIÓN DE SPRINT 5

Sprint N.	5	**Nombre**	Creación de procesos almacenados SQL
Objetivo	Implementar la persistencia de datos en la base de datos SQL mediante la creación de procesos almacenados.		
Detalle			
• Identificar los procesos que se deben almacenar en la base de datos para asegurar la persistencia de los datos. • Crear los procesos almacenados necesarios para almacenar los datos en la base de datos. • Integrar los procesos almacenados en el proyecto para su uso. • Ejecutar pruebas de integración con los procesos almacenados para garantizar su correcta implementación.			
Criterios de aceptación			
• Se han identificado correctamente los procesos que se deben almacenar en la base de datos. • Los procesos almacenados se han creado correctamente y cumplen con los requisitos de la aplicación.			

• La integración de los procesos almacenados en el proyecto se ha realizado con éxito. • Las pruebas de integración con los procesos almacenados se han ejecutado correctamente, garantizando su correcto funcionamiento en el proyecto.

Tabla 22. Definición Sprint 5.

3.5.2 DISEÑO DE SPRINT 5

Se crean los procedimientos almacenados para facilitar el proceso de almacenar datos y que tengan persistencias, en la siguiente ilustración.

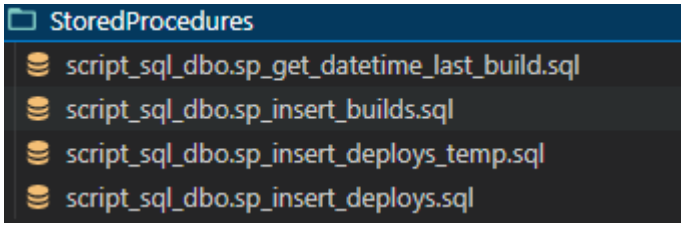


Ilustración 11. Procedimientos Almacenados creados.

3.5.3 PRUEBA DEL SPRINT 5

En la siguiente tabla se califican los criterios de aceptación del sprint 5 para dar por terminado el sprint una vez se cumplen.

Sprint 5		
Prueba		¿Cumple?
1	Los procedimientos se integran con éxito con los criterios establecidos.	SI
2	La ejecución de las API's se integra correctamente con los procedimientos almacenados para alojar la información.	SI

3	La ejecución para obtener la información funciona de manera rápida.	SI

Tabla 23. Pruebas Sprint 5.

3.6 Sprint 6

A continuación, se presenta la reunión de inicio del proyecto, investigación de requisitos, diseño de prototipos.

3.6.1 DEFINICIÓN DE SPRINT 6

Sprint N.	6	**Nombre**	Configuración de Grafana para el dashboard.
Objetivo	Configurar Grafana para el dashboard y diseñar la interfaz gráfica para mostrar los datos de integraciones y despliegues continuos obtenidos a través de la API de Azure DevOps.		
Detalle			
En este sprint se trabajará en la configuración de Grafana para el dashboard, lo que incluirá el diseño de la interfaz gráfica para mostrar los datos de integraciones y despliegues continuos obtenidos a través de la API de Azure DevOps.			
Criterios de aceptación			
• Configuración de Grafana en el entorno de desarrollo. • Diseño de la interfaz gráfica para mostrar los datos obtenidos de la API de Azure DevOps. • Integración de la API de Azure DevOps con Grafana para obtener datos de integraciones y despliegues continuos. • Validación de la correcta visualización de los datos en la interfaz gráfica de Grafana. • Pruebas de integración con el resto de los componentes del proyecto.			

Tabla 24. Definición Sprint 6.

3.6.1 DISEÑO DE SPRINT 6

Se realiza la creación y configuración de una plantilla en Grafana Labs para visualizar el Dashboard, como se muestra en las siguientes ilustraciones.

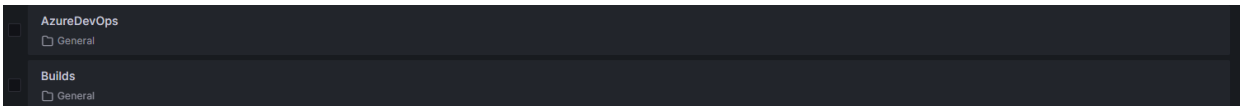


Ilustración 12. Plantillas creadas en Grafana Labs para la muestra de información obtenida.

En la ilustración 12, se presenta el esquema de carpetas, las cuales contienen las plantillas con la información de manera grafica.

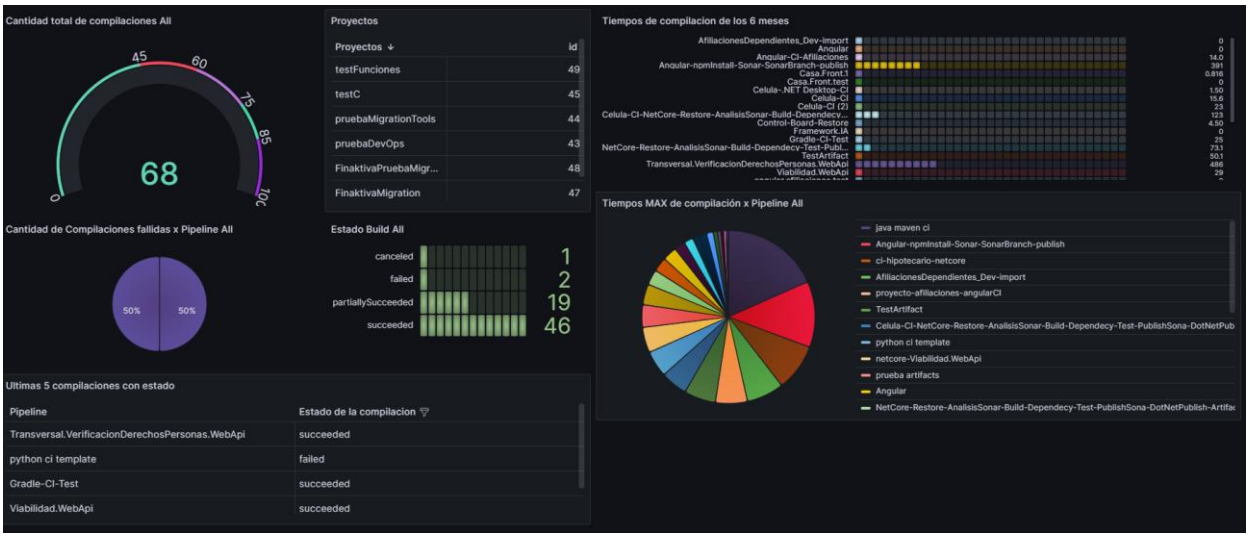


Ilustración 13. Dashboard principal informativo (Estado general de compilaciones).

En la Ilustración 13, se presenta un panel de visualización de datos con diversas gráficas que proporcionan información clave sobre el rendimiento de los pipelines en un entorno DevOps.



Ilustración 14. Tiempos Máximos de Compilación por Pipeline.

La ilustración 14, es un medidor circular que representa un valor único. En este caso, el medidor tiene un rango de valores de 0 a 100, con marcas de graduación en puntos clave, como 0, 45, 60, 75, 85 y 100. El valor actual en el medidor es 68. Esta visualización permite evaluar el rendimiento de un proceso DevOps, y se considera óptimo si supera el 60% en la cantidad de compilaciones semanales.

La utilidad de esta visualización radica en su capacidad para evaluar el rendimiento de un proceso DevOps. Se considera que el rendimiento es óptimo cuando el valor en el medidor supera el umbral del 60%, especialmente en lo que respecta a la cantidad de compilaciones semanales. De esta manera, el medidor proporciona una representación gráfica clara y efectiva de la eficiencia del proceso, permitiendo a los usuarios identificar fácilmente si se están cumpliendo los objetivos de rendimiento deseados.

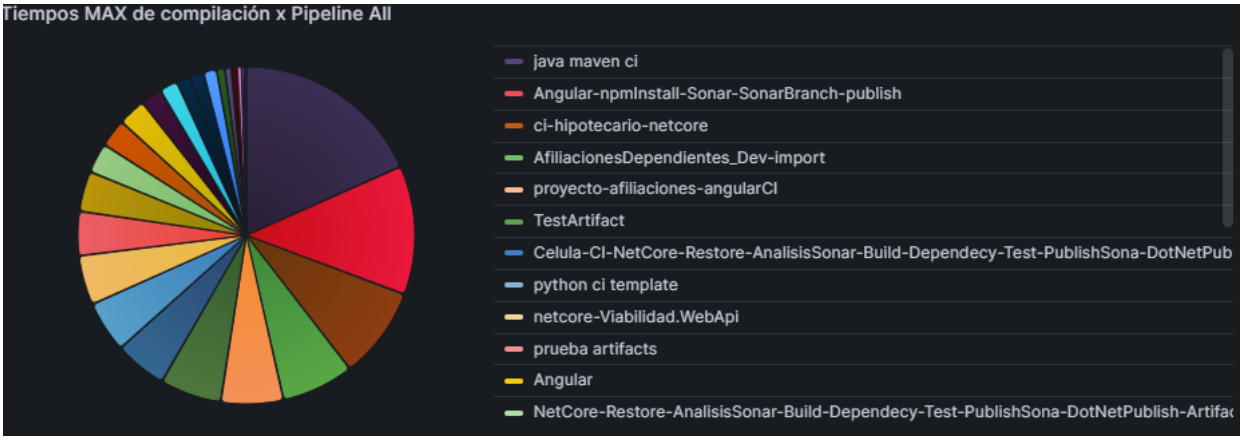


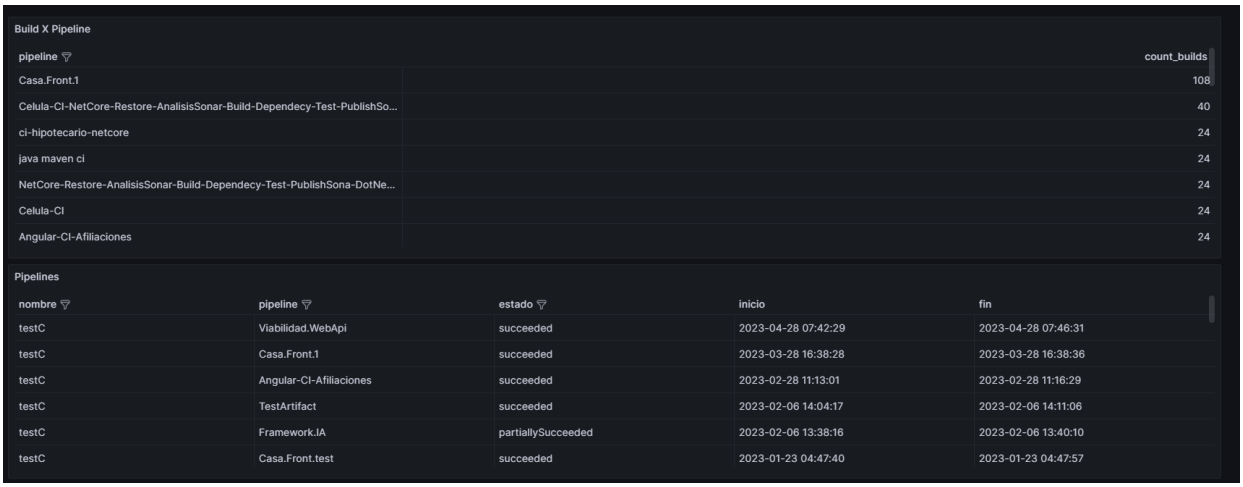
Ilustración 15. Tiempos Máximos de Compilación por Pipeline

En la Ilustración 15, se presenta un gráfico de pastel que proporciona una representación visual de los tiempos máximos de ejecución de varios pipelines en el entorno de DevOps. Cada segmento del gráfico corresponde a un pipeline específico y se identifica claramente mediante su nombre, ejemplificado por etiquetas como "Angular-Restore-AngularSonarBranch-publish" y "NetCore-Restore-AngularSonarBranch-publish-TestPublishSonarDonetPublish-Artifactory". El título del gráfico, "Tiempos Máximos de Compilación por Pipeline", ayuda a contextualizar la información presentada y resalta el objetivo de analizar la duración de ejecución de estos procesos DevOps. En resumen, este gráfico de pastel ofrece una visión panorámica de los tiempos de ejecución de distintos pipelines, permitiendo una comprensión rápida y visual de su rendimiento dentro del proceso de desarrollo y despliegue.



Ilustración 16. Estado de las compilaciones.

La Ilustración 16 presenta un medidor horizontal que permite clasificar los pipelines según su cantidad y estado. Los estados considerados son "cancelado," "fallido," "parcialmente exitoso" y "exitoso." Este medidor posee un rango de valores que va de 0 a 100, lo que facilita una evaluación rápida de cómo se distribuyen los pipelines en los diversos estados. Estos elementos gráficos resultan muy útiles para llevar a cabo un seguimiento efectivo y analizar el rendimiento de los procesos DevOps en una organización.



Build X Pipeline

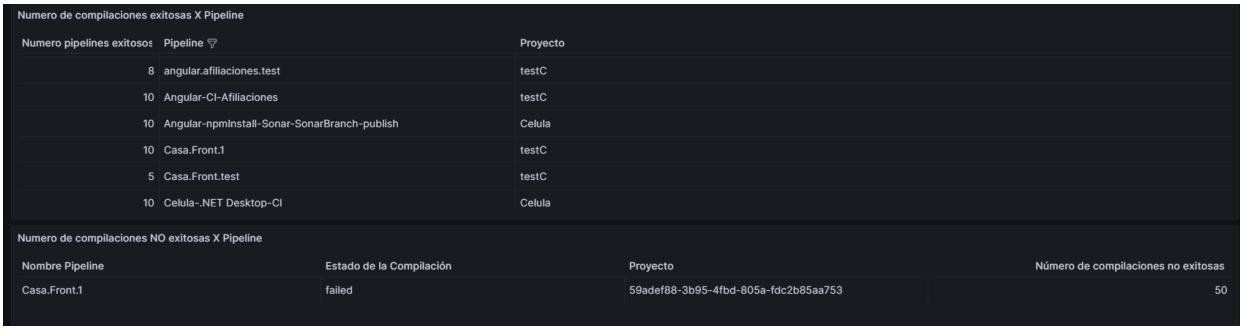
pipeline	count_builds
Casa.Front.1	108
Celula-CI-NetCore-Restore-AnalisisSonar-Build-Dependecy-Test-PublishSo...	40
ci-hipotecario-netcore	24
java maven ci	24
NetCore-Restore-AnalisisSonar-Build-Dependecy-Test-PublishSona-DotNe...	24
Celula-CI	24
Angular-CI-Afiliaciones	24

Pipelines

nombre	pipeline	estado	inicio	fin
testC	Viabilidad.WebApi	succeeded	2023-04-28 07:42:29	2023-04-28 07:46:31
testC	Casa.Front.1	succeeded	2023-03-28 16:38:28	2023-03-28 16:38:36
testC	Angular-CI-Afiliaciones	succeeded	2023-02-28 11:13:01	2023-02-28 11:16:29
testC	TestArtifact	succeeded	2023-02-06 14:04:17	2023-02-06 14:11:06
testC	Framework.IA	partiallySucceeded	2023-02-06 13:38:16	2023-02-06 13:40:10
testC	Casa.Front.test	succeeded	2023-01-23 04:47:40	2023-01-23 04:47:57

Ilustración 17. Dashboard de builds con tiempos, estados.

En la ilustración 17, es una tabla que posee la información más general de los estados de las compilaciones, así mismo, la cantidad de compilaciones que ha realizado cada pipeline.



Numero de compilaciones exitosas X Pipeline

Numero pipelines exitosos	Pipeline	Proyecto
8	angular.afiliaciones.test	testC
10	Angular-CI-Afiliaciones	testC
10	Angular-npmInstall-Sonar-SonarBranch-publish	Celula
10	Casa.Front.1	testC
5	Casa.Front.test	testC
10	Celula-.NET Desktop-CI	Celula

Numero de compilaciones NO exitosas X Pipeline

Nombre Pipeline	Estado de la Compilación	Proyecto	Número de compilaciones no exitosas
Casa.Front.1	failed	59adef88-3b95-4fbd-805a-fdc2b85aa753	50

Ilustración 18.Dashboard número de compilaciones exitosas y no exitosas.

En la ilustración 18, se presenta una tabla que contiene datos relacionados con el número de compilaciones exitosas realizadas por diferentes pipelines, junto con información adicional que incluye el nombre del pipeline y el proyecto al que está asociado. Además, en la tabla se muestra el recuento de compilaciones que no tuvieron éxito. La información en esta tabla proporciona una visión clara de la ejecución exitosa y fallida de las compilaciones, permitiendo una rápida identificación de los pipelines y proyectos que requieren atención o mejoras en el proceso de desarrollo y despliegue.

3.6.2 PRUEBA DEL SPRINT 6

En la siguiente tabla se califican los criterios de aceptación del sprint 6 para dar por terminado el sprint una vez se cumplen.

Sprint 6		
Prueba		¿Cumple?
1	El entorno de Grafana funciona según los criterios.	SI
2	Los dashboards funcionan de manera rápida y sencilla.	SI
3	Las pruebas de información se realizan según lo planteado.	SI

Tabla 25. Pruebas Sprint 6.

3.7 Sprint 7

A continuación, se presenta la reunión de inicio del proyecto, investigación de requisitos, diseño de prototipos.

3.7.1 DEFINICIÓN DE SPRINT 7

Sprint N.	7	**Nombre**	Registro de datos a visualizar en Grafana
Objetivo	El objetivo principal de este sprint es el registro de los datos que se visualizarán en Grafana y la implementación de un dashboard para		

	mostrar la información de integraciones y despliegues continuos. Además, se deberá establecer la conexión entre Grafana y la base de datos SQL para obtener los datos necesarios para el dashboard.
Detalle	
• Identificación de los datos necesarios para la visualización de integraciones y despliegues continuos en el dashboard de Grafana. • Implementación del registro de los datos en la base de datos SQL, considerando las estructuras y relaciones de las tablas. • Configuración y conexión de Grafana con la base de datos SQL para obtener los datos necesarios para el dashboard. • Desarrollo y diseño del dashboard en Grafana, considerando los datos registrados y la información relevante que se desea mostrar. • Validación del correcto registro de los datos en la base de datos SQL y su correcta visualización en el dashboard de Grafana.	
Criterios de aceptación	
• Se han identificado correctamente los datos necesarios para la visualización de integraciones y despliegues continuos en el dashboard de Grafana. • Se han implementado correctamente las estructuras y relaciones de las tablas en la base de datos SQL para el registro de los datos. • La conexión entre Grafana y la base de datos SQL se ha realizado de forma correcta y sin errores. • Se ha desarrollado y diseñado un dashboard funcional y estéticamente agradable en Grafana que muestra los datos de integraciones y despliegues continuos. • Se han realizado pruebas de validación para confirmar que los datos se están registrando correctamente en la base de datos SQL y que se están visualizando correctamente en el dashboard de Grafana.	

Tabla 26. Definición Sprint 7.

3.7.2 DISEÑO DE SPRINT 7

Se realiza las consultas de la base de datos de las tablas de integración y despliegue continuos.

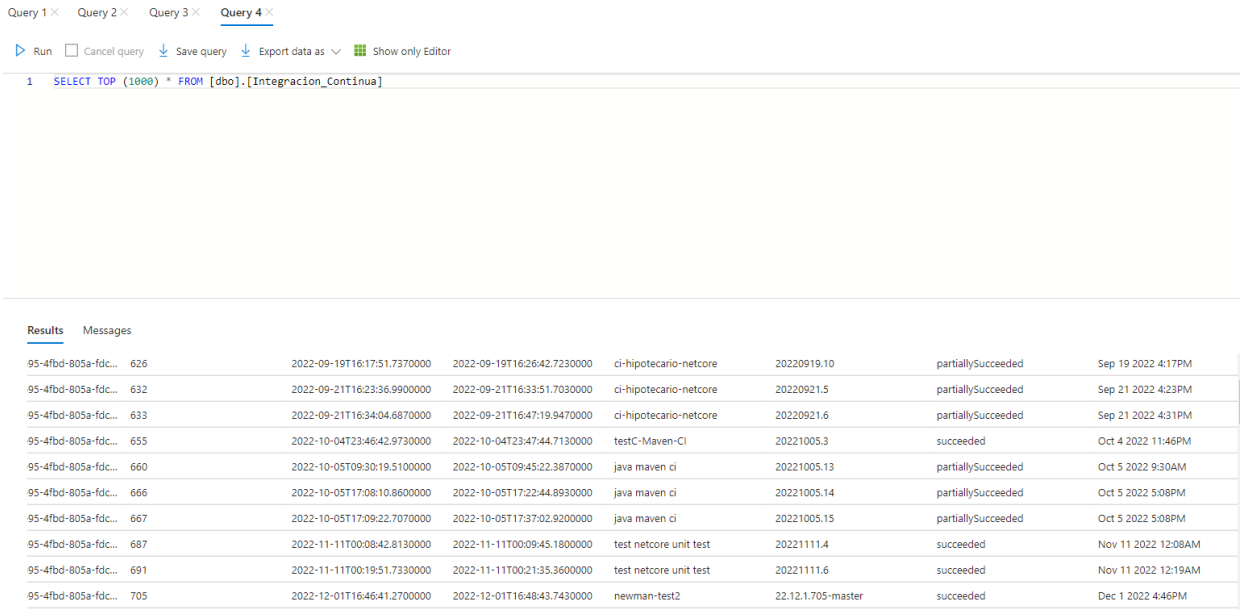


Ilustración 19. Integración continua (Log de las integraciones guardadas).

En la Ilustración 19, se lleva a cabo una operación de consulta de datos en la base de datos, centrándose en la tabla llamada "Integración_Continua." Esta tabla almacena información detallada sobre los pipelines utilizados en las prácticas de integración continua dentro de la plataforma de Azure DevOps. Durante esta consulta, se accede a los registros que contienen datos esenciales relacionados con estos procesos de integración continua, lo que facilita su posterior análisis y gestión.

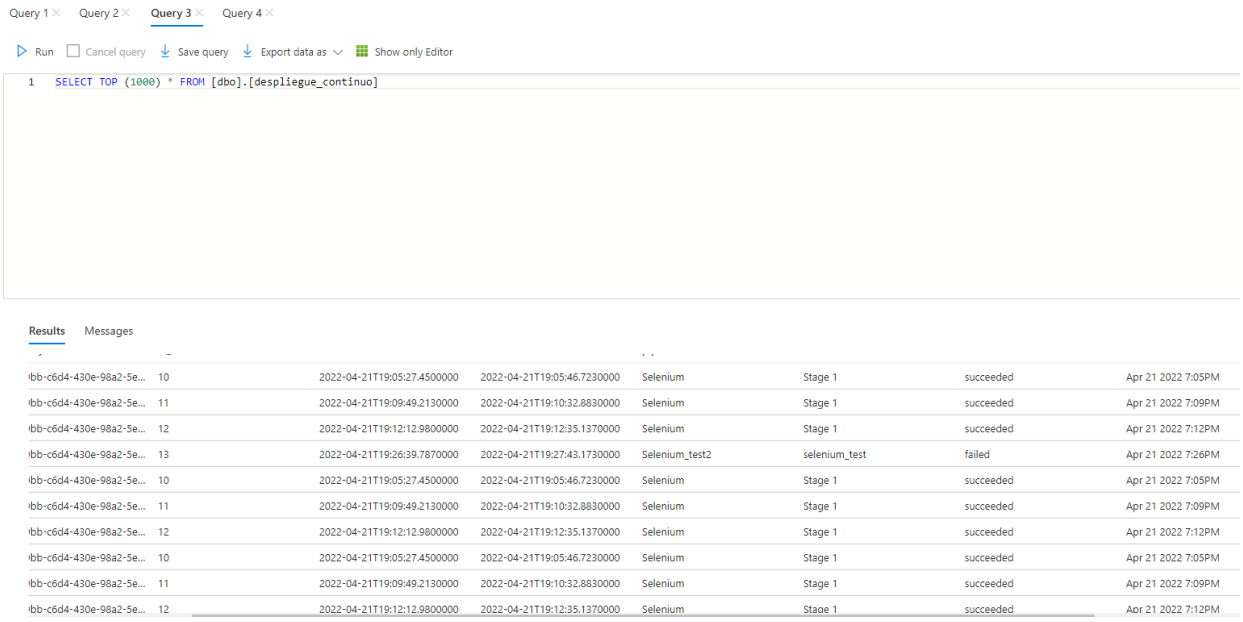


Ilustración 20. Despliegue continuo (Log de los despliegues guardados).

La información se muestra en los dashboard creados en Grafana Labs, como el número total de compilaciones, los proyectos en Azure DevOps, los estados de los pipelines, “Canceled, Failed, PartiallySucceeded, Succeeded”, las ultimas compilaciones, tiempos máximos de compilación.

3.7.3 PRUEBA DEL SPRINT 7

En la siguiente tabla se califican los criterios de aceptación del sprint 6 para dar por terminado el sprint una vez se cumplen.

Sprint 7		
Prueba		¿Cumple?
1	Las consultas en Grafana Labs coinciden con la información de la base de datos y sigue los criterios de seguridad.	SI
2	La información se plasma de manera sencilla y fácil de entender.	SI
3	La estructura de los Dashboard permite la navegación de manera sencilla.	SI

Tabla 27. Pruebas Sprint 7.

4. FASE IV: IMPLEMENTACIÓN DEL PROYECTO

La aplicación utiliza una arquitectura de microservicios para consumir las APIs de Azure DevOps y obtener información sobre las integraciones y despliegues continuos. La información obtenida se almacena en una base de datos SQL para su posterior consulta. Finalmente, la información se muestra en un dashboard en Grafana Labs para que el usuario pueda monitorear el estado de las integraciones y despliegues continuos de manera visual y eficiente. La aplicación es escalable y flexible, lo que permite un fácil manejo y acceso a la información en tiempo real.

4.1 API REST

Este proyecto de microservicios está diseñado para consumir las API REST de Azure DevOps y se basa en una arquitectura n capas, las cuales tienen su funcionamiento en:

- La capa de dominio es responsable de la lógica de negocio y está separada de la capa de infraestructura que maneja la comunicación con la base de datos SQL.
- La capa de modelo se definen los objetos de datos y los modelos de negocio.
- La capa de repositorios, que manejaría la persistencia de datos y la interacción con la base de datos SQL.

El proyecto podrá consumir las API REST de Azure DevOps a través de la integración de los endpoints en la capa de servicios, la cual se encargará de conectar con la API de Azure DevOps y de procesar los datos que se obtienen. Los servicios también pueden proporcionar lógica de negocio adicional que no esté cubierta por la capa de dominio.

Este proyecto ha sido desarrollado en C# 3.1 y ha sido construido siguiendo las mejores de desarrollo. El objetivo principal del proyecto es proporcionar una plataforma escalable y fácil de mantener para el monitoreo de las integraciones y despliegues continuos de los proyectos en Azure DevOps, utilizando una arquitectura modular basada en microservicios y una separación clara de responsabilidades entre las diferentes capas.

La siguiente ilustración representa la distribución de la arquitectura de n capas, con los microservicios construidos para el funcionamiento del proyecto en la obtención de la información de la organización creada en Azure DevOps.

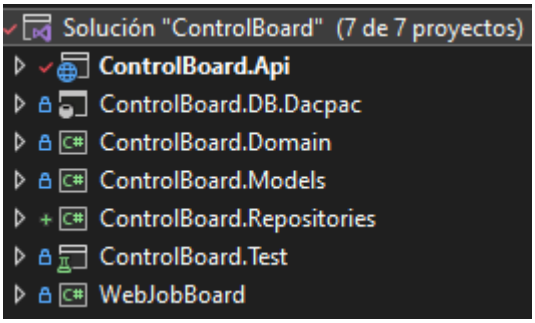


Ilustración 21. Distribución de microservicios (API Rest)

La ilustración del Proyecto Base sirve como cimiento para el desarrollo de la aplicación. Incluye configuraciones, bibliotecas esenciales y estructura inicial del proyecto.

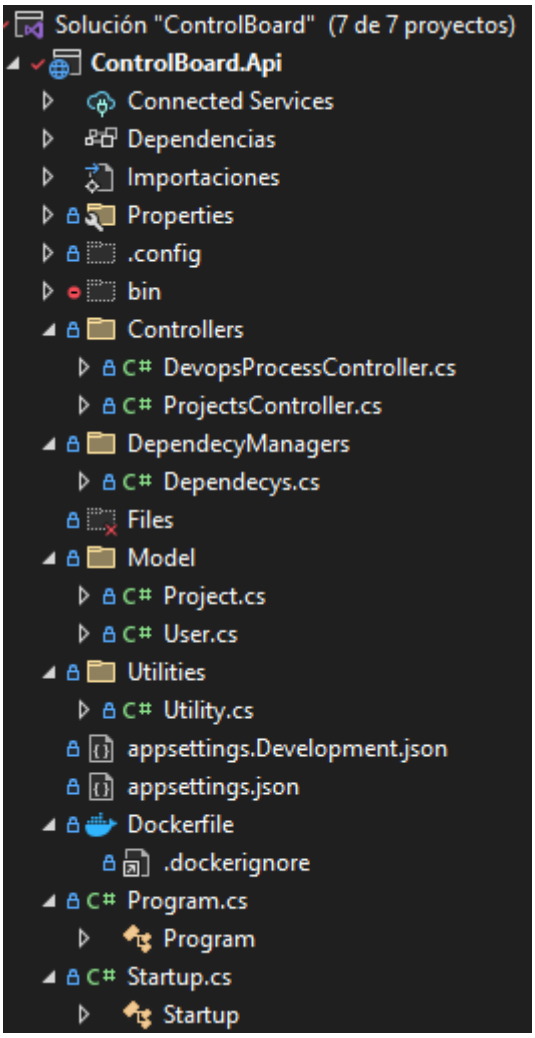


Ilustración 22. Distribución detallada de los folders del proyecto base.

La siguiente ilustración de capa de Dominio asegura una gestión eficiente de la lógica del negocio.

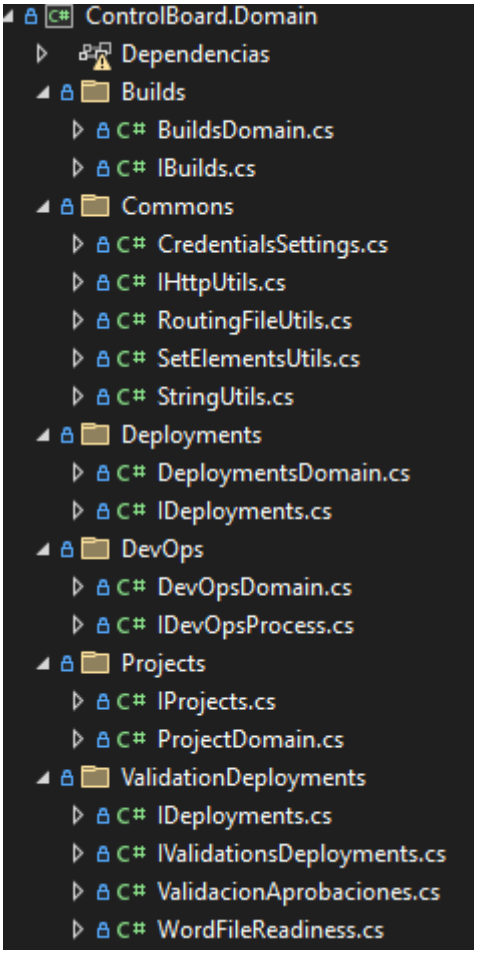


Ilustración 23. Distribución detallada de los folders de la capa de Dominio.

La ilustración muestra la capa de modelos em la que se definen las estructuras de datos y las clases que representan las entidades y objetos en la aplicación, la cual es esencial para la comunicación entre las diferentes capas del sistema.

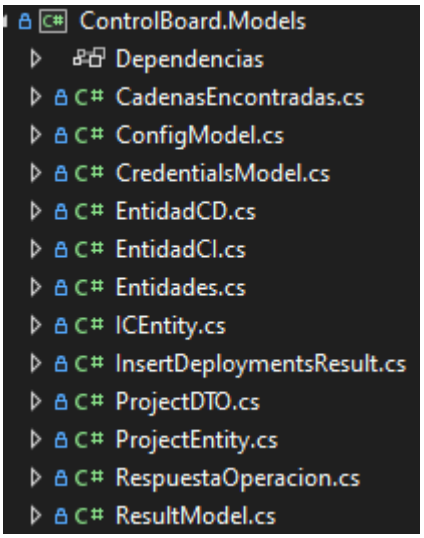


Ilustración 24. Distribución detallada de los folders de la capa de modelos

La ilustración demuestra la capa de repositorio la cual se encarga de la gestión de datos y acceso a la base de datos, esta incluye carpetas.

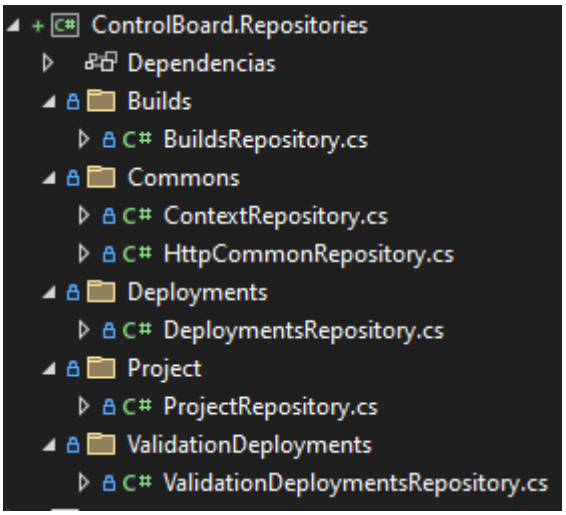


Ilustración 25. Distribución detallada de los folders de la capa de repositorio.

Visualización de los servicios GetDeploymentsDevopsProccess, GetBuildsDevOpsProccess, GetProjectDevOpsProccessen, y SyncProjectsToDataBase en Swagger; esta es una herramienta de documentación y

exposición de APIs. Esta imagen muestra cómo los servicios y endpoints de la aplicación se describen y documentan en formato Swagger, lo que facilita la comprensión y el acceso a la funcionalidad de la aplicación a través de una interfaz de usuario interactiva.

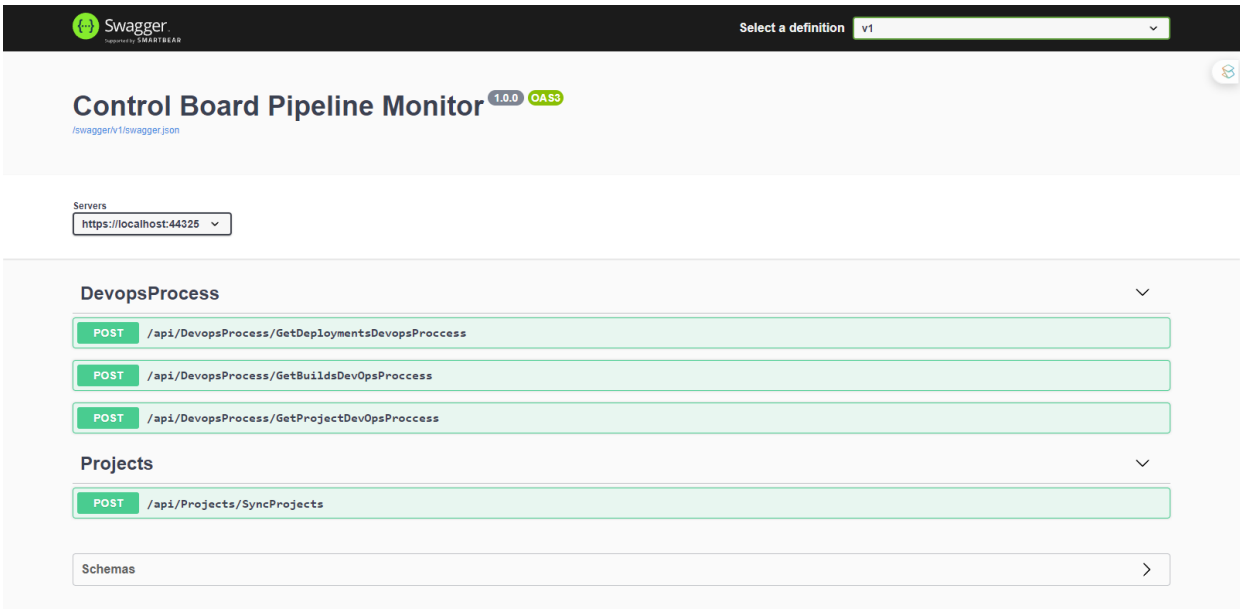


Ilustración 26. Servicios en Swagger.

4.2 Dashboard

La aplicación que utiliza Grafana Labs como plataforma para visualizar información relacionada con los pipelines de integración y despliegue continuo en Azure DevOps. Para obtener los datos necesarios, la aplicación se conecta a una base de datos SQL que almacena información previamente recolectada por un backend que consume las APIs de Azure DevOps.

La interfaz del dashboard proporciona una vista general de la información de los pipelines, permitiendo al usuario obtener una visión rápida del estado actual de los procesos de integración y despliegue continuo. Además, el dashboard permite clasificar la información por otro dashboard específico para los procesos de construcción (builds) de los pipelines, lo que permite al usuario profundizar aún más en la información de los procesos.

Para lograr esto, el proyecto utiliza diversas tecnologías y herramientas, incluyendo lenguajes de programación como C#, la plataforma Grafana Labs y la conexión con bases de datos SQL. El proyecto se ha diseñado con una arquitectura adecuada que permite que las diferentes partes de la aplicación trabajen juntas de manera efectiva, incluyendo capas de modelo, dominio y repositorios.

La siguiente ilustracion proporciona una vista general de información esencial en tiempo real, donde se actualiza el dashboard cada 5 minutos. Este panel presenta datos clave y métricas relevantes para el proyecto, ofreciendo a los usuarios una visión rápida y completa de la situación actual y el rendimiento del sistema en Azure DevOps.

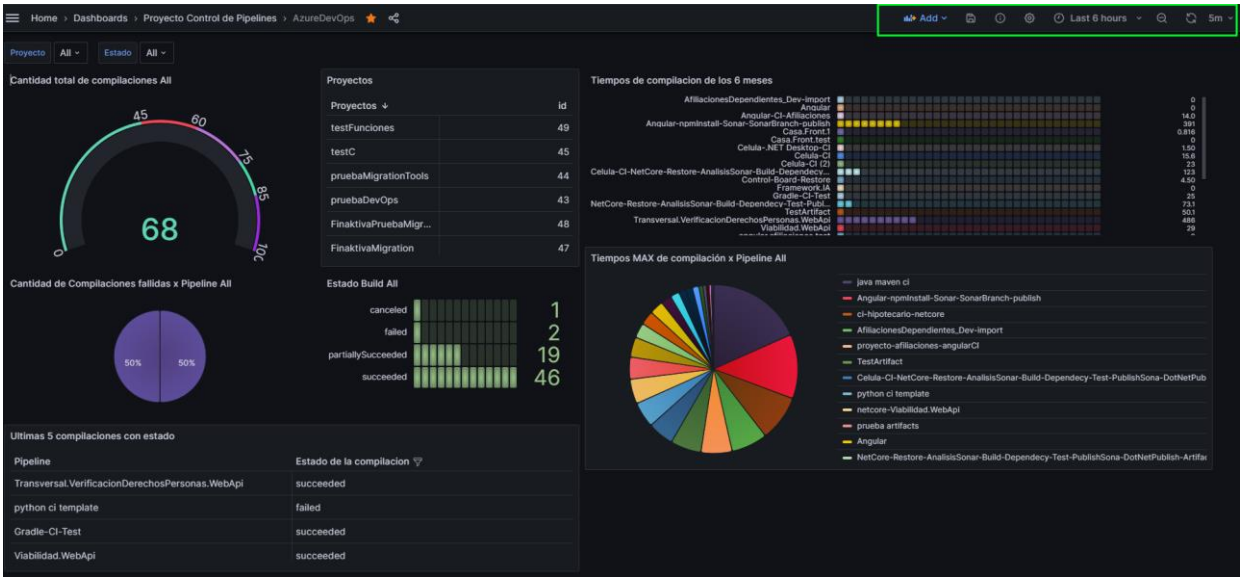


Ilustración 27. Dashboard principal informativo, con actualizaciones.

Esta ilustración muestra el estado y el progreso de las compilaciones y despliegues continuos en Azure DevOps. Este panel ofrece información detallada sobre las compilaciones en curso, histórico de compilaciones exitosas y fallidas, y proporciona

una visión general de la salud y estabilidad del proceso de desarrollo y despliegue en el proyecto.

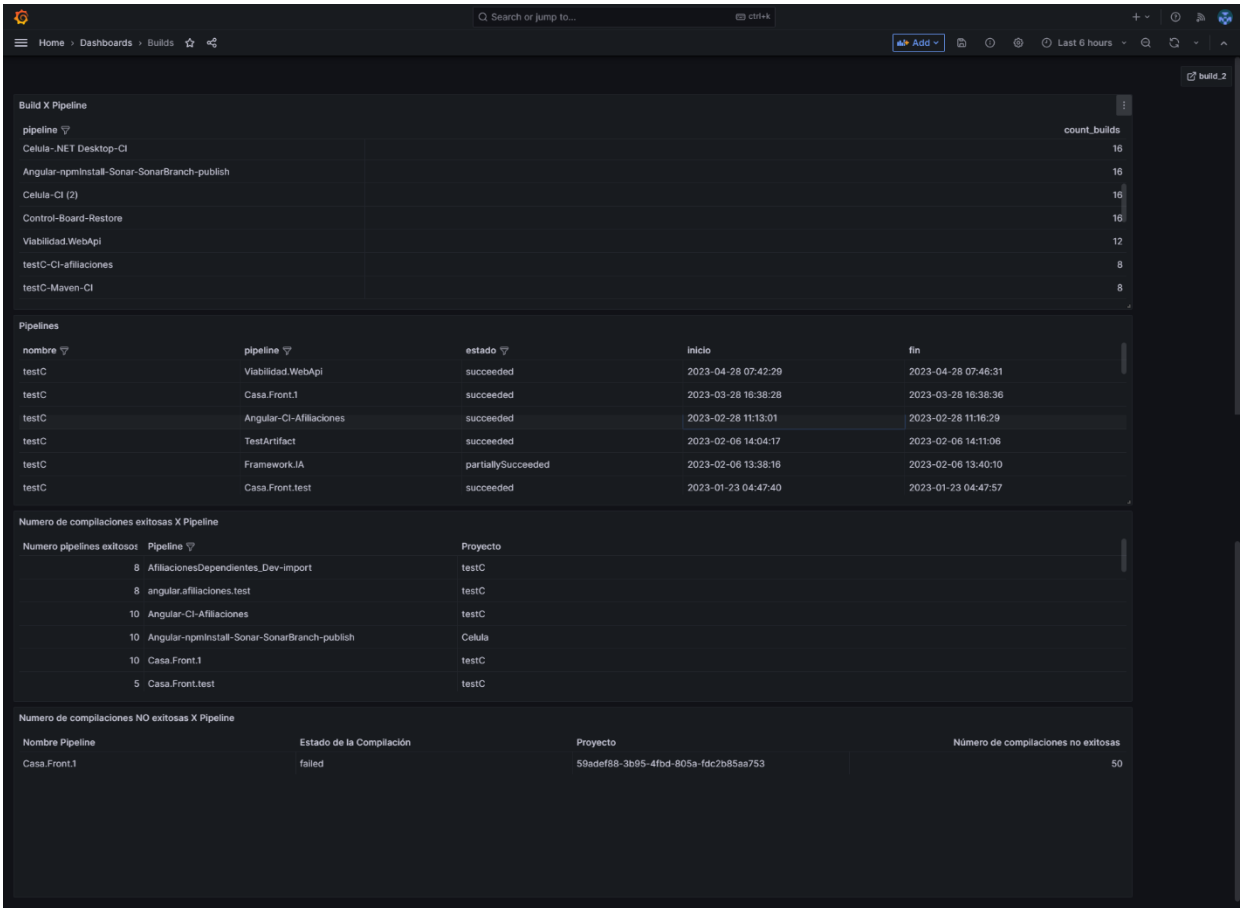


Ilustración 28. Dashboard de builds.

La ilustración 29, ofrece una información general en tablas, donde se explica la cantidad de compilación por pipelines, los pipelines con su estado, organización, fechas de inicio y fin, además de las compilaciones fallidas.

4.3 Monitoreo

Para realizar el monitoreo y la información utilizada en la demostración, creamos una organización en la plataforma Azure DevOps: https://dev.azure.com/celuladevopsj. En esta organización hemos agregado repositorios de varios proyectos desarrollados en diferentes tecnologías. Cada uno de estos repositorios tiene un proceso, que es un conjunto de procesos y herramientas diseñados para recopilar datos sin procesar de

diversas fuentes, analizarlos y presentar los resultados en un formato comprensible. El objetivo final es determinar si el proceso de compilación, ejecución de prueba e implementación es exitoso, parcialmente exitoso o fallido.

Los pipelines que creamos para cargar el panel en Grafana Labs están asociadas con un repositorio específico. Cada uno de estos repositorios representa un proyecto sobre una tecnología específica y la canalización garantiza que el proyecto pueda ejecutarse en un entorno de producción real. Esto significa que, si el proyecto se compila con éxito, pasa las pruebas y se implementa con éxito en el entorno, se considera un proceso exitoso.

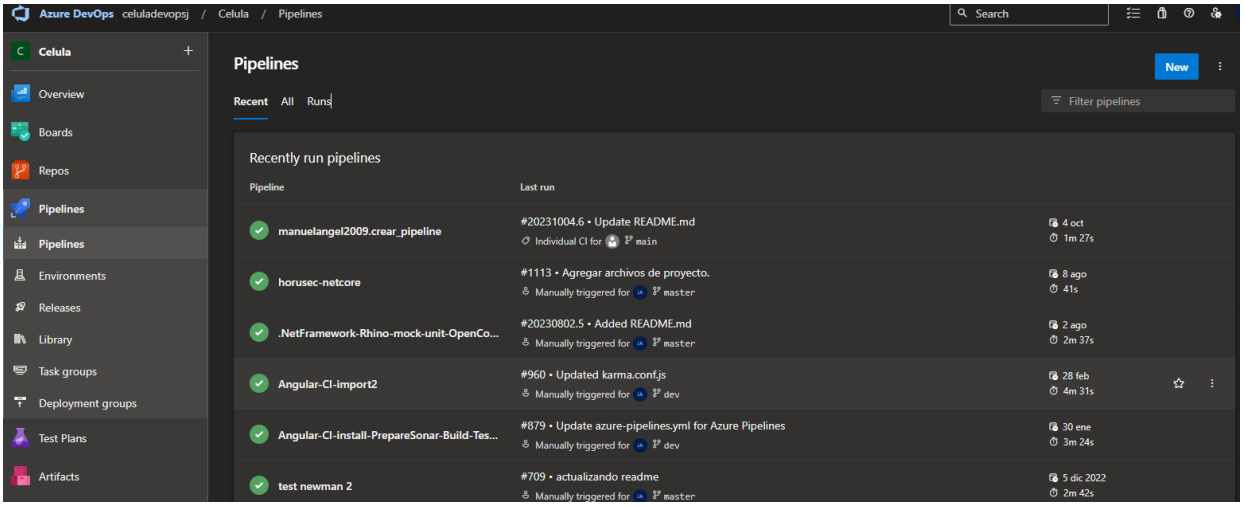


Ilustración 29. Plataforma demo Azure DevOps.

La ilustración 30, proporciona la información alojada en Azure DevOps.

Los resultados derivados de la ejecución de los pipelines son adquiridos a través de una API diseñada para esta finalidad. Esta API recopila los datos resultantes y los almacena en una base de datos en la nube de Azure. Posteriormente, estos datos se utilizan como fuente de información para los dashboards alojados en la plataforma Grafana Labs.

Actualmente, se lleva a cabo el monitoreo de aproximadamente 68 proyectos que cuentan con pipelines en funcionamiento. Es importante destacar que esta cantidad de proyectos podría aumentar, ya que esta configuración se adapta fácilmente para la creación de nuevos pipelines, lo que permite su escalabilidad.

5. FASE V: PRUEBAS

5.3. Pruebas de calidad

En la Ilustración 28 se presentan los resultados obtenidos al llevar a cabo pruebas unitarias en el código, las cuales tienen como objetivo verificar y asegurar el adecuado funcionamiento de la lógica de la aplicación, contribuyendo así a la calidad del proyecto backend.

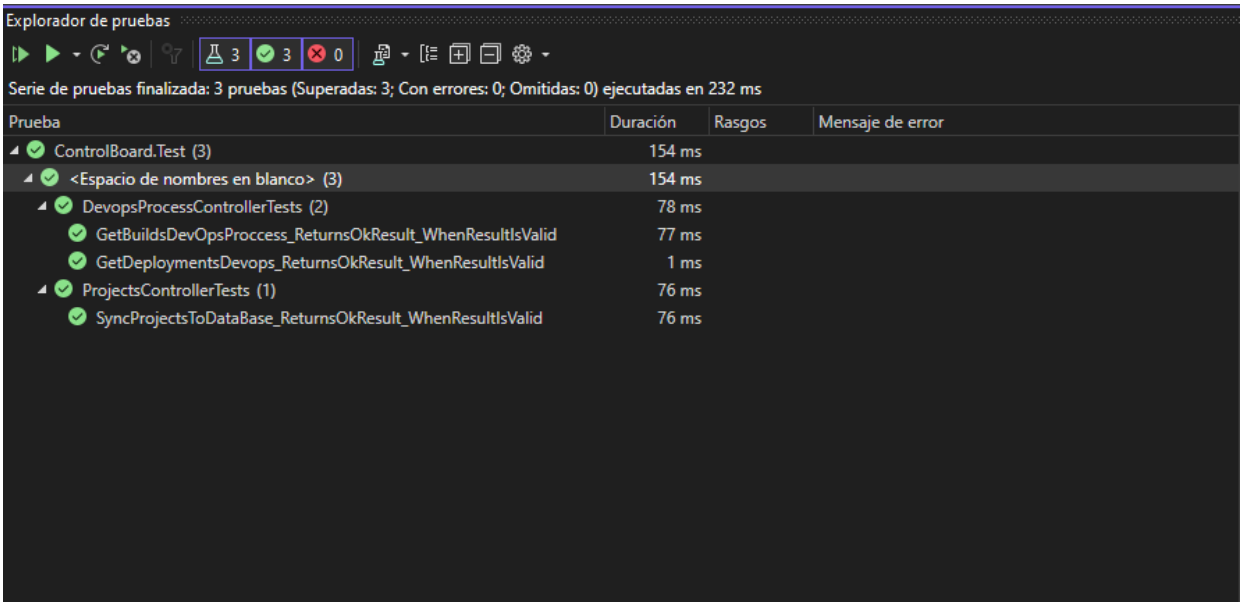


Ilustración 30. Pruebas Unitarias.

La ilustración 31, nos muestra las pruebas unitarias que se realizaron al método GetDeploymentsDevops_ReturnsOkResult_WhenResultIsValid, la cual la primera prueba unitaria, se verifica si la función produce un resultado exitoso, concretamente un resultado exitoso. En particular, se verificó que al ejecutar el programa encargado de obtener información sobre implementaciones en un entorno DevOps, el resultado es positivo, como lo demuestra el estado “OK”.

La primera prueba unitaria se realiza al método GetBuildsDevOpsProccess_ReturnsOkResult_WhenResultIsValid, la cual la primera prueba unitaria, se verifica si la función produce un resultado exitoso, concretamente un resultado exitoso. En particular, se verificó que al ejecutar el programa encargado de

obtener información sobre implementaciones en un entorno DevOps, el resultado es positivo, como lo demuestra el estado “OK”.

La tercera prueba unitaria se realiza al método SyncProjectsToDataBase_ReturnsOkResult_WhenResultIsValid. En la fase de configuración (Arrange), se estableció las condiciones y expectativas utilizando objetos modelo. En concreto, la clase BuildsDomain y se configuro para que cuando se llame al método GetProjectDevOpsProccess con el nombre de proyecto "SampleProject", devuelva la cadena "Project Details".

5.4 Prueba de visualización de datos

La Ilustración 29 muestra una prueba clave en el panel de Grafana para comprobar la precisión de la visualización de datos. Los elementos visuales como gráficos, diagramas, tablas y paneles se prueban para garantizar que reflejen con precisión los datos de la base de datos subyacente. Al mismo tiempo, se eliminan los datos de prueba para simular la recopilación de información actualizada.

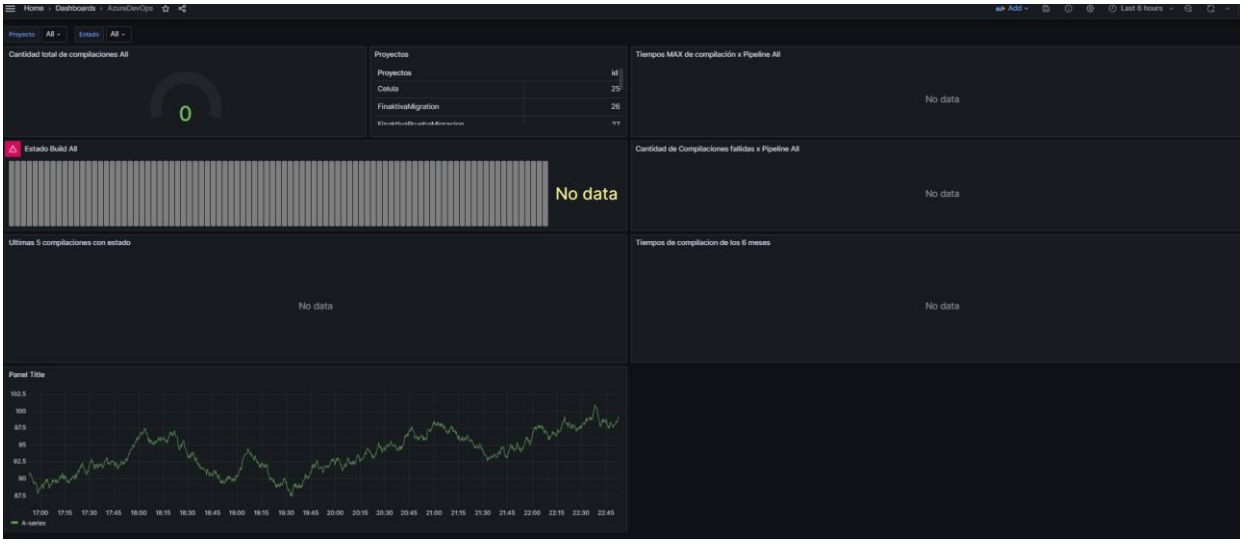


Ilustración 31. Prueba de visualización de Dashboard.

El objetivo principal de la ilustración 32, es brindar un apoyo visual de estas pruebas y garantizar que el cuadro de mando proporcione a los usuarios la información precisa y actualizada necesaria para tomar decisiones con conocimiento de causa. También se

realiza prueba de interactividad para asegurar de que las interacciones funcionen según lo esperado. Esto incluye la capacidad de filtrar datos, hacer zoom, desplazarse en el tiempo y otras acciones interactivas que los elementos visuales del cuadro de mandos cumplen las normas de visualización de datos de la organización.

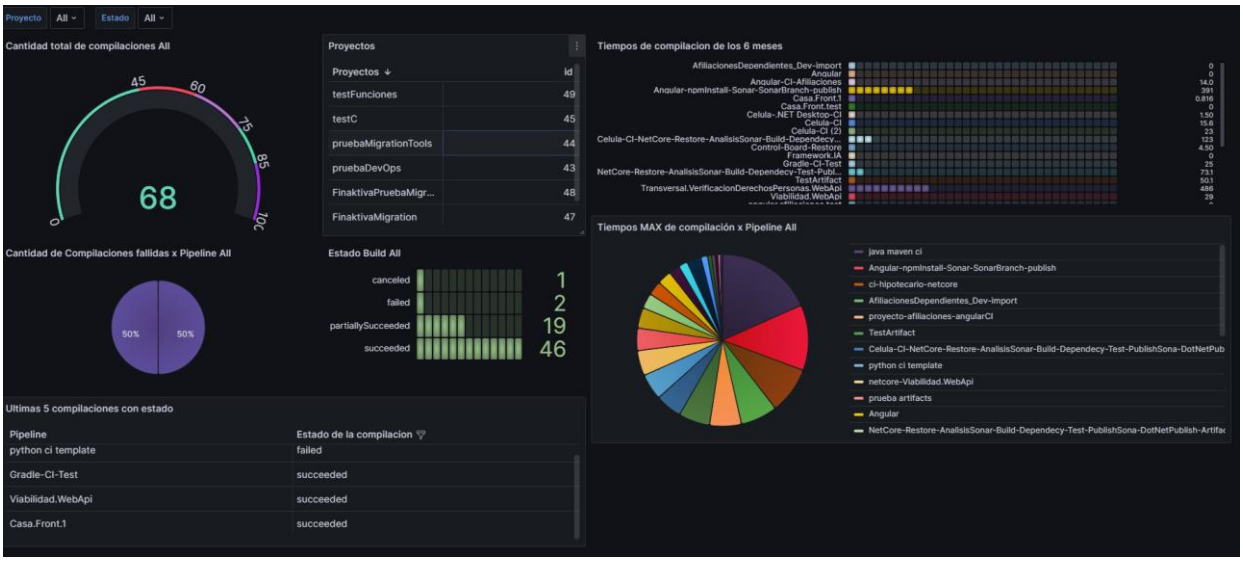


Ilustración 32. Visualización Dashboard Grafana.

La ilustración 33, contine la información de mando general, con los gráficos, los cuales son un apoyo a la toma de decisiones.

5.5 Prueba de rendimiento

Las pruebas de rendimiento se realizaron mediante la herramienta web https://gtmetrix.com/. [13]Se analizó el rendimiento del panel alojado en Grafana Labs, cuya URL es https://escobarjsa1.grafana.net/d/aesppbonz/azuredevops?orgId=1.

En la ilustración 34, se muestran los resultados de la prueba mostraron una calificación de "D" en los detalles de evaluación y puntuación. Recibió una puntuación del 63 % en rendimiento y una puntuación del 79 % en estructura del sitio. Estos valores reflejan el rendimiento y la eficacia del tablero.

[13] GTmetrix. Performance Report. https://gtmetrix.com/reports/escobarjsa1.grafana.net/vtW76YmR/. [Consulta: viernes, 18 de agosto de 2023]

En cuanto a métricas web, se midieron tres aspectos clave:

- Pintura de mayor contenido (LCP) con un tiempo de carga de 3,3 segundos. LCP se refiere al momento en que el contenido principal de una página se vuelve visible para el usuario.
- El tiempo total de bloqueo (TBT) es 0 ms, lo que significa que no hay un bloqueo significativo mientras se carga la página.
- El cambio de diseño total (CLS) es 0, lo que indica que no hay cambios inesperados en el diseño de los elementos de la página que puedan causar molestias al usuario.

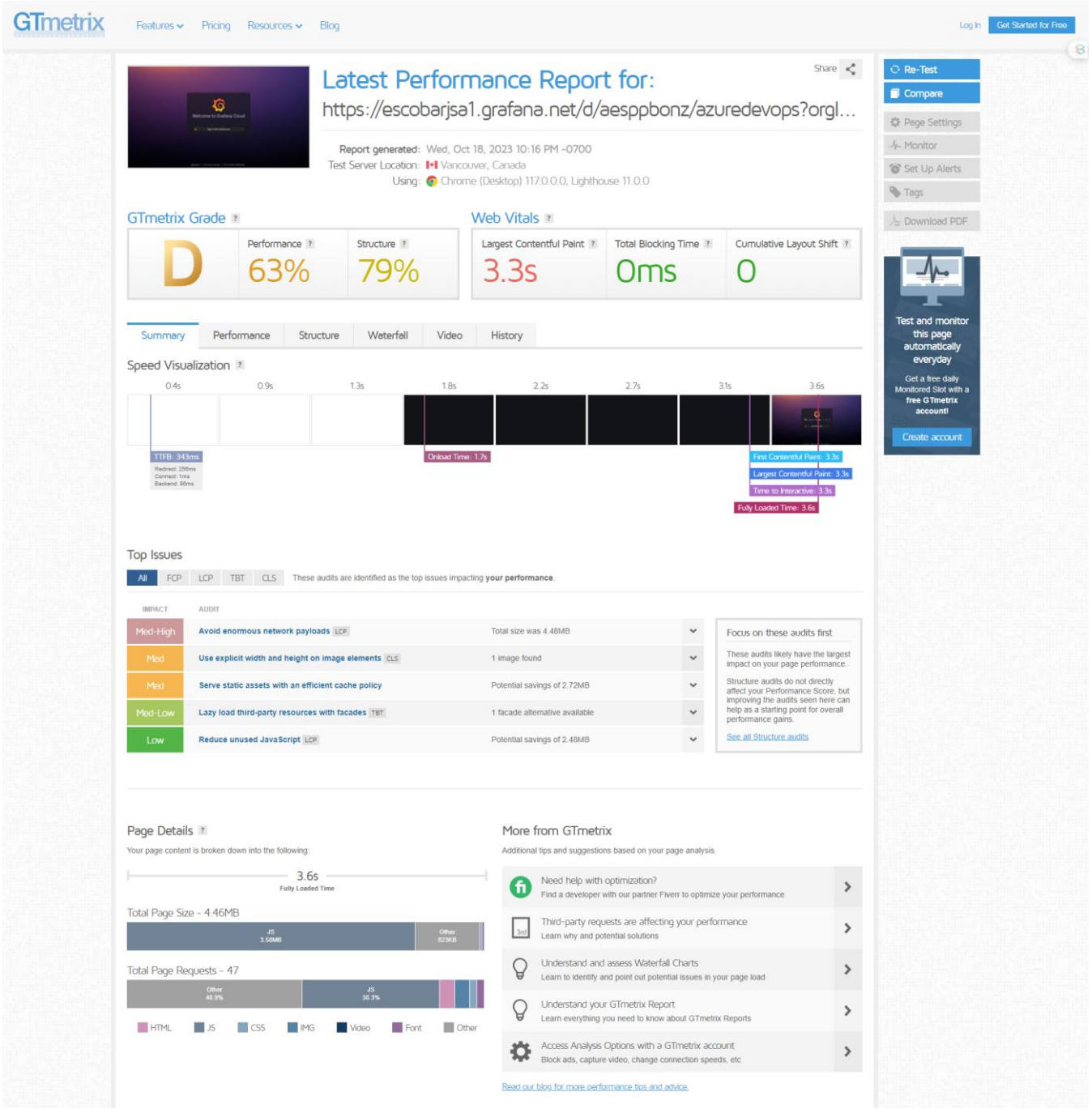


Ilustración 33. Prueba de rendimiento.

5.6 Prueba de calidad general

Para llevar a cabo las pruebas de calidad general, se utilizó la herramienta Lighthouse de Google dentro de las Chrome DevTools. Se evaluó un dashboard alojado en Grafana

Labs, accesible a través de la URL https://escobarjsa1.grafana.net/d/aesppbonz/azuredevops?orgId=1. Estas pruebas proporcionaron una evaluación detallada del rendimiento y la calidad del sitio.

Los resultados de las pruebas se reflejaron en puntajes en diferentes áreas. En cuanto al rendimiento, se obtuvo un puntaje de 40, lo que sugiere que hay margen de mejora. La accesibilidad recibió un puntaje del 88, lo que indica un buen nivel de accesibilidad. Las mejores prácticas se calificaron con un 64, y en SEO se obtuvo un puntaje de 82, lo que sugiere una optimización razonable para los motores de búsqueda.

Además, se evaluaron cinco métricas de velocidad importantes:

- Primera Pintura con Contenido (FCP): 3.5 segundos, que es el tiempo en que el primer contenido visible se carga.
- Pintura con Contenido Más Grande (LCP): 8.0 segundos, que mide el tiempo para cargar el elemento más grande de la página.
- Tiempo Total de Bloqueo (TBT): 330 milisegundos, que cuantifica el tiempo en que la página no responde a la interacción del usuario.
- Cambio de Diseño Acumulativo (CLS): 0.088, que mide la estabilidad de la página.
- Índice de Velocidad (SI): 5.1 segundos, que refleja la velocidad general de carga de la página.

Es importante destacar que estas pruebas se realizaron en un servicio en la nube (Grafana Labs) que aloja el dashboard en una página estática. Los resultados de estas pruebas indican áreas de mejora potencial para optimizar la velocidad y la calidad general del dashboard a nivel de Grafana, pero al ser un servicio Cloud el acceso a estos cambios son limitados.

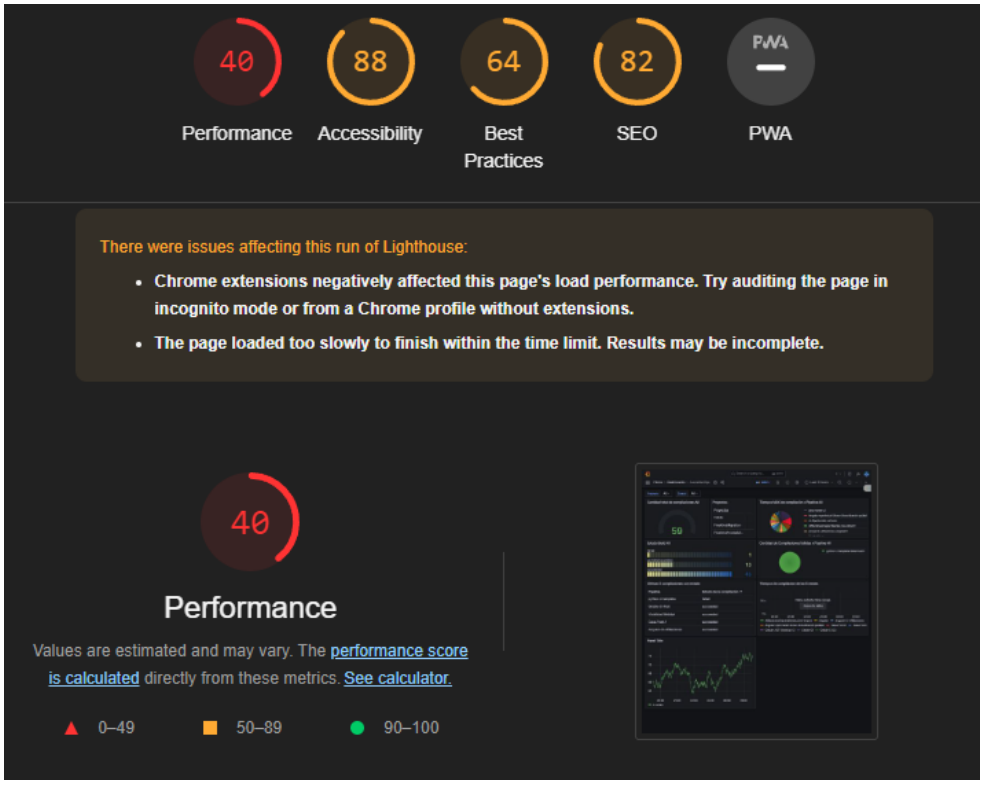


Ilustración 34. Prueba de calidad general con lighthouse de Chrome DevTools.

6. FASE VI: RESULTADOS

Resultado	Indicador
Ejecuto la aplicación y envío una solicitud a los servicios internos de la misma con el propósito de obtener información específica relacionada con el número de proyectos, el estado de los pipelines y el recuento de despliegues continuos alojados en la plataforma Azure DevOps.	Correcto funcionamiento de los servicios que interactúan con las API de creación (build) de Azure DevOps es un elemento esencial para obtener la información de la plataforma.
Ejecuto la base de datos, reviso que las tablas estén pobladas con los datos clasificados	La inicialización de la base de datos se realizó con éxito, y se completó la carga de datos clasificados en las tablas sin ningún problema, contribuyendo así a un

	flujo de trabajo eficiente y sin interrupciones.
Reviso el dashboard en Grafana Labs, verifico que los datos puedan ser visualizados	El dashboard en Grafana Labs se encuentra disponible de manera consistente y es fácilmente accesible. Los datos se presentan de forma efectiva y se muestran de manera organizada para su visualización y monitoreo.
Reviso los gráficos con la información obtenida de Azure DevOps, que pueda servirme para monitorear los proyectos y sus estados.	Los datos se cargan adecuadamente en el dashboard, utilizando un rango de tiempo de los últimos 5 minutos, lo que nos proporciona la información más actualizada para su monitoreo.

7. FASE VII: TRABAJOS FUTUROS

Se ha previsto la integración de futuros proyectos, ya sean extensiones del proyecto actual o iniciativas derivadas, siempre que cumplan con los principios de obtener información del ciclo de vida del software y estén alineados con las prácticas de DevOps, que combinan el desarrollo de software y las operaciones de TI. Además, se considera la posibilidad de consumir API's de servicios externos para ampliar las capacidades y la funcionalidad del proyecto en el futuro, como lo son:

- **Travis CI**: que es un sistema de integración continúa distribuido y gratuito para proyectos de código abierto, y de pago para proyectos privados[14]. Travis CI permite conectar el repositorio de Github y probar después de cada push que se realice, regenerando el proyecto. Soporta múltiples lenguajes como Clojure, Erlang, Node.js, PHP y Ruby. Travis CI, ofrece una colección de API’s en las

[14] genbeta. Travis CI, sistema distribuido de integración continua libre integrado con Github. https://www.genbeta.com/desarrollo/travis-ci-sistema-distribuido-de-integracion-continua-libre-integrado-con-github. [Consulta: viernes, 18 de agosto de 2023].

cuales se pueden consultar en función de lo que se requiera obtener del proceso DevOps.

- **CircleCI**: es una plataforma de integración continua (CI) y despliegue continuo (CD) fundada en el año 2011. Esta plataforma se caracteriza por su velocidad líder en la industria, la base para la entrega a escala, la confianza en cada compromiso y el poder de crear sin límites[15]. CircleCI se integra con varios servicios como son AWS, Slack y Atlassian. CircleCI proporciona una serie de API's para interactuar con los procesos de CI/CD y obtener información relevante.
- **Jenkins**: es una herramienta de integración continua que permite automatizar el proceso de construcción, prueba y despliegue de software. Jenkins proporciona una API REST que permite interactuar con el servidor de Jenkins y realizar tareas como crear, actualizar o eliminar trabajos, obtener información sobre los trabajos en ejecución, etc.
- **GitHub Actions**: es una plataforma de integración y despliegue continuo que permite la automatización de procesos de compilación, implementación y prueba de software. Esta herramienta se integra sin problemas con GitHub y permite crear flujos de trabajo personalizados para los proyectos. Se pueden crear flujos de trabajo para compilar, probar y desplegar código, así como para realizar otras tareas automatizadas[16]. GitHub ofrece una API REST para interactuar con estos flujos de trabajo y obtener información sobre la ejecución de tareas automatizadas.

Además, no solo se considera la integración con herramientas DevOps, como las mencionadas anteriormente, sino también la posibilidad de incluir el consumo de API’s de monitoreo de herramientas como parte integral de la gestión y monitoreo del ciclo de vida del software. Estas consideraciones aseguran una base sólida para la toma de decisiones informadas y la mejora continua en proyectos, se pueden monitorear herramientas como:

[15] KeepCoding, R.Qué es CircleCI. https://keepcoding.io/blog/que-es-circleci/. [Consulta: viernes, 18 de agosto de 2023].
[16] GitHub. Entender las GitHub Actions, Aprende lo básico de las GitHub Actions, incluyendo los conceptos nucleares y la terminología esencial. https://docs.github.com/es/actions/learn-github-actions/understanding-github-actions . [Consulta: viernes, 18 de agosto de 2023].

- **SonarQube**: es una plataforma de código abierto para la inspección continua de la calidad del código a través de diferentes herramientas de análisis estático de código fuente. Proporciona métricas que ayudan a mejorar la calidad del código de un programa permitiendo a los equipos de desarrollo hacer seguimiento y detectar errores y vulnerabilidades en el código. SonarQube proporciona una API REST que permite interactuar con el servidor de SonarQube y realizar tareas como crear, actualizar o eliminar proyectos, obtener información sobre los proyectos en ejecución, etc[17].

[17] SonarQube. Web API. https://docs.sonarsource.com/sonarqube/latest/extension-guide/web-api/. [Consulta: viernes, 18 de agosto de 2023].

CONCLUSIONES

Al analizar los diferentes servicios y API’s de la plataforma Azure DevOps, se puede obtener información valiosa sobre las integraciones y despliegues continuos realizados en esta plataforma.

Como resultado del diseño de un módulo de consulta de procesos DevOps, se logra visualizar de manera clara y organizada los procesos de integración y despliegue continúo llevados a cabo por una organización en Azure DevOps.

La implementación de los dashboards en Grafana, con la información procesada resultante del consumo de los API’s de los procesos DevOps, permite una visualización más intuitiva y amigable para el usuario.

La creación de un Dashboard en Grafana para la consulta visual de la información, se convierte en una herramienta útil para la toma de decisiones, ya que permite ver de manera gráfica el estado de los procesos DevOps.

La creación de un submódulo en Grafana para el monitoreo de la frecuencia de implementación de las integraciones y despliegues continuos, proporciona un reporte detallado sobre el rendimiento de estos procesos, lo que facilita la identificación de áreas de mejora.

Las consultas SQL realizadas en los submódulos orquestados en Grafana permiten una gestión más eficiente de la información, ya que se pueden realizar búsquedas y filtrados específicos de acuerdo con las necesidades de la organización.

Al utilizar la plataforma Azure DevOps en conjunto con Grafana, se logra una integración perfecta que permite una gestión más completa y eficiente de los procesos DevOps.

La implementación de un módulo de consulta en Grafana para procesos DevOps, facilita la supervisión y el seguimiento de estos procesos, lo que reduce los tiempos de respuesta y mejora la eficiencia de la organización.

RECOMENDACIONES

A continuación, se presentan una serie de recomendaciones cruciales para garantizar el funcionamiento óptimo, junto con sugerencias para posibles mejoras en el futuro:

- Se requiere un nivel mínimo de comprensión de la cultura DevOps y la aplicación de buenas prácticas en el ciclo de vida del software para aprovechar al máximo el prototipo.
- Para utilizar Grafana en la visualización de dashboards, se debe tener una cuenta en Grafana Labs. Alternativamente, es posible implementar un entorno de Grafana con Docker o instalar Grafana directamente en un sistema Linux o Windows y luego importar los dashboards necesarios.
- Se recomienda mantener actualizada la versión de .NET a ediciones más recientes para evitar posibles conflictos con paquetes obsoletos o en desuso.
- En futuras versiones, es aconsejable considerar la integración de otras herramientas que abarquen aspectos adicionales del ciclo de vida del software. Esto permitirá una mayor compatibilidad y una evaluación más exhaustiva de los procesos.

BIBLIOGRAFÍA

AMAZON WEB SERVICES. ¿Qué es DevOps? [En línea] https://aws.amazon.com/es/devops/what-is-devops/.

c-sharpcorner. Getting Started With .NET Core 3.1 - Part One. [En línea] https://www.c-sharpcorner.com/article/getting-started-with-net-core-3-1-part-one/.

genbeta. 2012. Travis CI, sistema distribuido de integración continua libre integrado con Github. [En línea] 2012. https://www.genbeta.com/desarrollo/travis-ci-sistema-distribuido-de-integracion-continua-libre-integrado-con-github.

github. 2023. Entender las GitHub Actions, Aprende lo básico de las GitHub Actions, incluyendo los conceptos nucleares y la terminología esencial. [En línea] 5 de Septiembre de 2023. https://docs.github.com/es/actions/learn-github-actions/understanding-github-actions.

InfoQ. Full-Stack Observability with Grafana and Azure Monitor. [En línea] https://www.infoq.com/news/2022/05/azure-managed-grafana-preview/.

interpolados. METODOLOGÍA SCRUM. [En línea] https://interpolados.wordpress.com/2016/01/17/metodologia-scrum/.

KeepCoding. 2023. Qué es CircleCI. [En línea] 5 de octubre de 2023. https://keepcoding.io/blog/que-es-circleci/.

learn.microsoft.com. Referencia de la API rest de Azure DevOps Services. [En línea] https://learn.microsoft.com/es-es/rest/api/azure/devops/?view=azure-devops-rest-7.1&viewFallbackFrom=azure-devops-rest-5.1.

Microsoft. ¿Qué es Azure DevOps? [En línea] https://learn.microsoft.com/es-mx/azure/devops/user-guide/what-is-azure-devops?view=azure-devops.

—. Azure DevOps Services REST API Reference. [En línea] https://learn.microsoft.com/en-us/rest/api/azure/devops/?view=azure-devops-rest-7.1&viewFallbackFrom=azure-devops-rest-5.1.

proyectosagiles. Qué es SCRUM. [En línea] https://proyectosagiles.org/que-es-scrum/.

Report, GTmetrix Performance. 2023. https://gtmetrix.com/reports/escobarjsa1.grafana.net/vtW76YmR/. [En línea] 1 de 09 de 2023.

scaleyourapp. What Is Grafana? Why Use It? Everything You Should Know About It - Scaleyourapp. [En línea] https://scaleyourapp.com/what-is-grafana-why-use-it-everything-you-should-know-about-it/.

scrum619. SCRUM. [En línea] https://scrum619.wordpress.com/2016/02/26/roles/.

SonarQube. 18. Web API. [En línea] 2023 de agosto de 18. https://docs.sonarsource.com/sonarqube/latest/extension-guide/web-api/.

ANEXOS

Anexo 1: Manual de Usuario

DESCRIPCIÓN DE LA APLICACIÓN

La aplicación es un demo que muestra cómo se puede integrar una herramienta de monitoreo de las integraciones y despliegues continuos en Azure DevOps. El demo muestra en un dashboard con la herramienta Grafana el monitoreo, los estados de los pipelines y proporciona información de manera sencilla para la gerencia y toma de decisiones.

La herramienta de monitoreo permite a los usuarios ver el estado de las integraciones y despliegues continuos en Azure DevOps. El dashboard proporciona una vista general de los pipelines y su estado actual. La herramienta Grafana se utiliza para visualizar los datos recopilados por la herramienta de monitoreo.

El demo es útil para aquellos que quieren aprender cómo integrar una herramienta de monitoreo en Azure DevOps. La aplicación proporciona una manera fácil de ver el estado actual de los pipelines y tomar decisiones informadas sobre cómo mejorar el proceso de integración y despliegue continuo.

La aplicación es especialmente útil para la gerencia, ya que proporciona información clara y concisa sobre el estado actual del proceso de integración y despliegue continuo. Esto les permite tomar decisiones informadas sobre cómo mejorar el proceso y reducir el tiempo necesario para implementar nuevas características o correcciones.

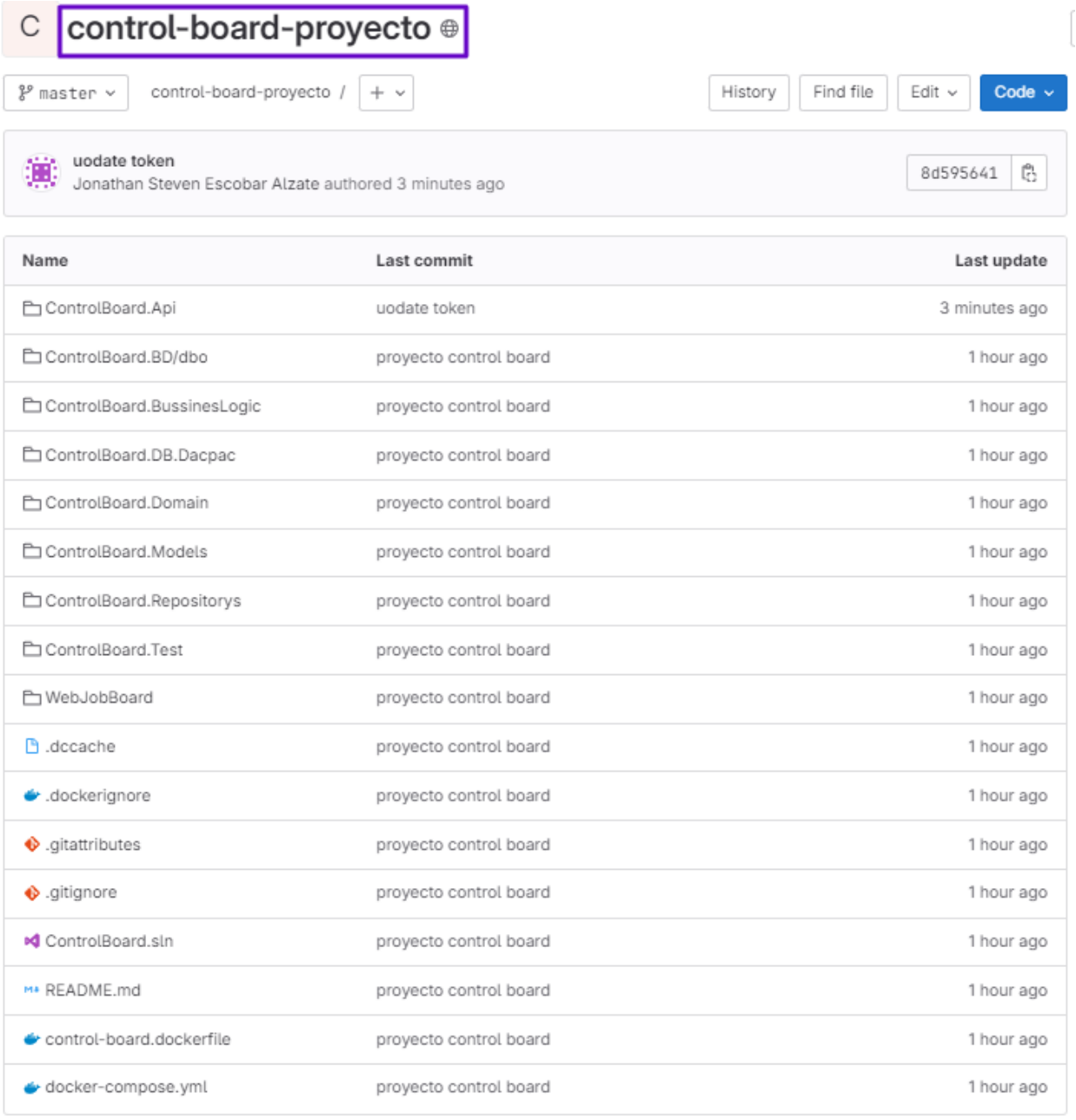
REQUERIMIENTOS

- Cuenta de Azure DevOps, la creación es gratuita.
- Base de datos SQL Server 2019 o posteriores versiones.
- Visual Studio o Visual Studio Code para la ejecución del demo.
- Cuenta de Grafana para importar el dashboard y revisar el monitoreo.

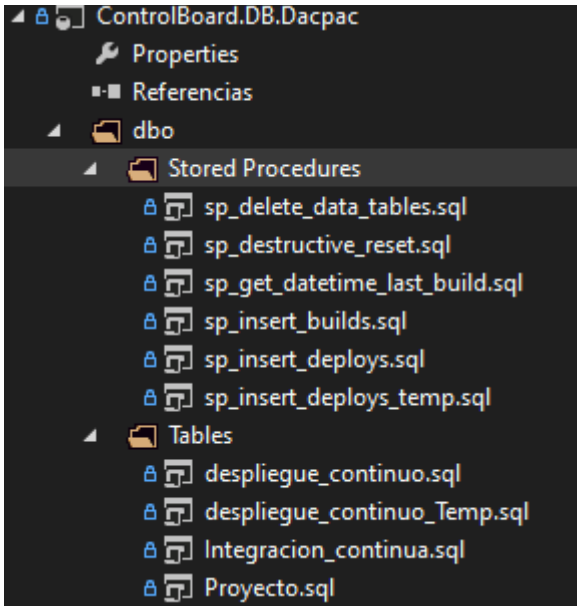
- Conexión a internet.

DESCARGAR E INSTALAR BACKED

1. Ingresar al siguiente link de gitlab para obtener el demo de la aplicación: https://gitlab.com/escobarjsa/control-board-proyecto clonar el repositorio “control-board-proyecto”.



2. Para ejecutar el demo dentro de la carpeta Control-Board está el programa desarrollado en C#, junto con un DacPac con la base de datos lista a ejecutarla a SQL SERVER 2019, Azure SQL o posteriores.



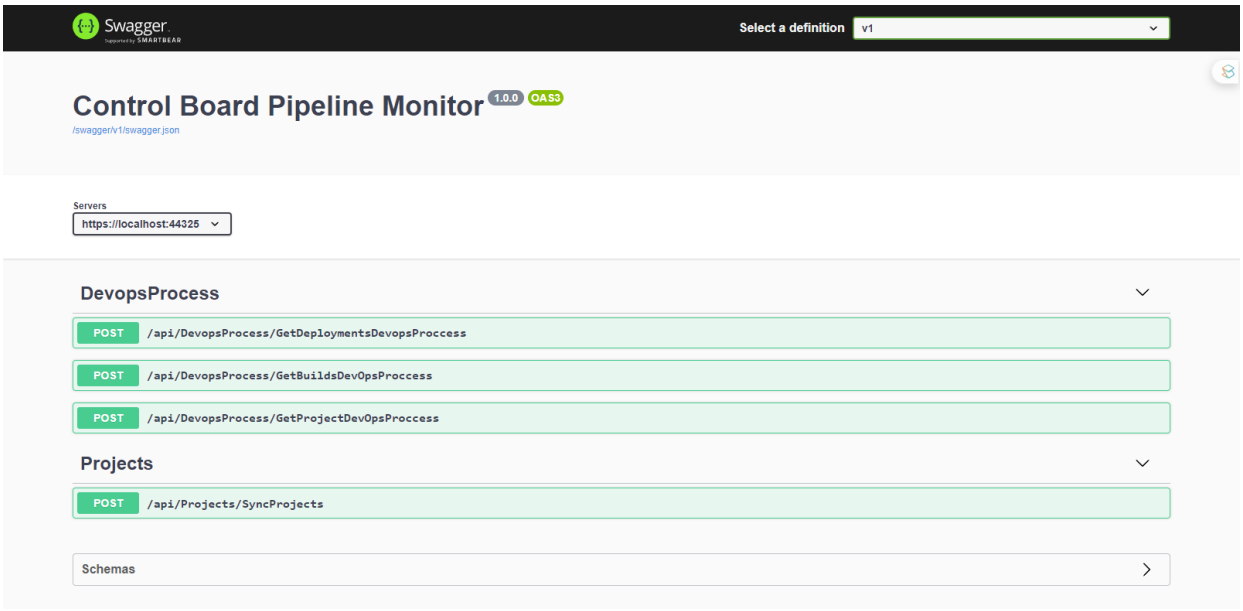
3. Para ejecutar la demo se debe realizar la conexión con la base de datos, para ello en el archivo appsettings.json el parámetro ConnectionString debe tener la conexión con la base de datos que se usara.

```
"ConnectionString": "Server=",
```

4. Realizar la conexión con Azure DevOps en el archivo appsettings.json es necesario generar un Access personal token de Azure DevOps, agregar el usuario asignado al token y la organización.

```
"UsuarioVSTS": "USUARIO_AZURE_DEVOPS",
"PasswordVSTS": "TOKEN_AZURE_DEVOPS",
"CollectionVSTS": "NOMBRE_ORGANIZACION_AZURE_DEVOPS",
"Devops_uri_api": "https://dev.azure.com/",
"Devops_uri_api_release": "https://vsrm.dev.azure.com/",
"Devops_project_url": "https://dev.azure.com/{collection}/_apis/projects",
"Devops_project_url_Projects": "https://dev.azure.com/celuladevopsj/_apis/projects?$top=500&api-version=5.0",
"Devops_Definitions_url": "https://dev.azure.com/{colection}/{project name}/_apis/build/definitions?api-version=5.0",
```

5. Una vez configurado lo anteriormente mencionado ejecutamos el proyecto como cada microservicio viene con una configuración con el complemento Swagger pata tenerlo documentado de manera sencilla y rápida para los servicios Rest expuestos y como consumir los endpoints, se debe usar la url de localhost:<puerto>/ swagger



6. Para ejecutar de manera sencilla los microservicios con Swagger se desplegar y se ejecutan con el botón Try it out para que nos responda a la base de datos con la información obtenida.

Control Board Pipeline Monitor 1.0.0 OAS3

/swagger/v1/swagger.json

Servers
https://localhost:44325

DevopsProcess

POST `/api/DevopsProcess/GetDeploymentsDevopsProccess`

Parameters | Try it out

No parameters

Responses

Code	Description	Links
200	Media type: application/json (Controls Accept header.) Example Value | Schema: `string`	*No links*

POST `/api/DevopsProcess/GetBuildsDevOpsProccess`

Parameters | Try it out

No parameters

Responses

Code	Description	Links
200	Media type: application/json (Controls Accept header.) Example Value | Schema: `string`	*No links*

POST `/api/DevopsProcess/GetProjectDevOpsProccess`

Parameters | Try it out

No parameters

Request body required — application/json

Example Value | Schema

```
{
  "projectName": "string"
}
```

Responses

Code	Description	Links
200	Media type: application/json (Controls Accept header.) Example Value | Schema: `string`	*No links*

Projects

POST `/api/Projects/SyncProjects`

Parameters | Try it out

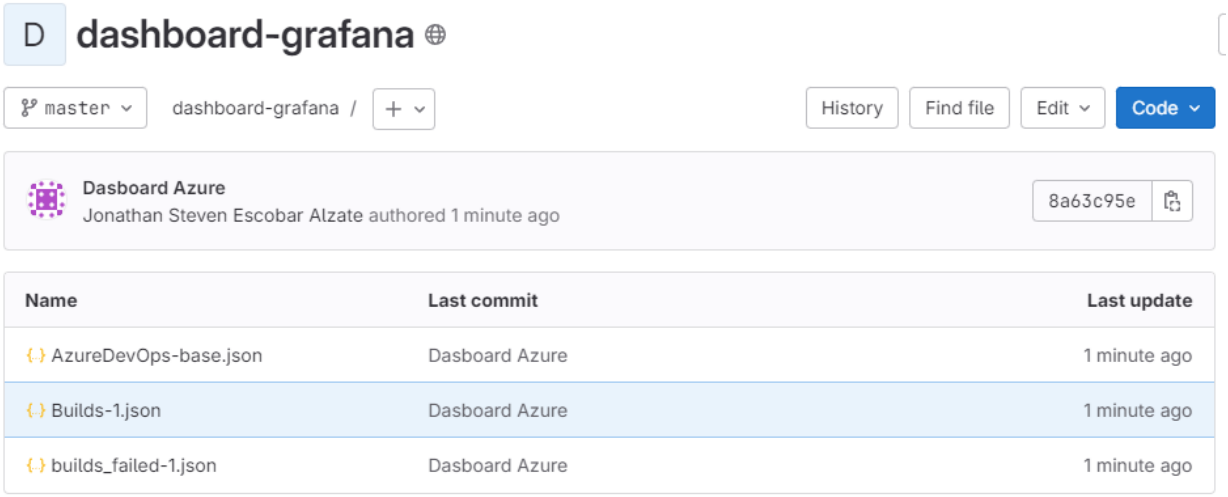
No parameters

Responses

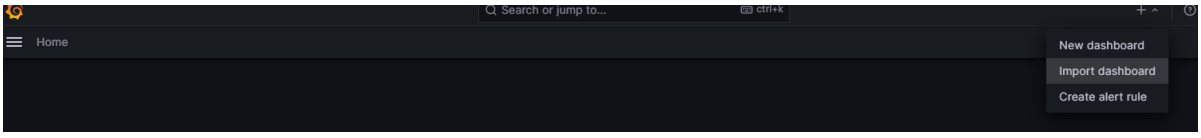
Code	Description	Links
200	Media type: application/json (Controls Accept header.) Example Value | Schema: `string`	*No links*

DESCARGAR E INSTALAR DASHBOARD

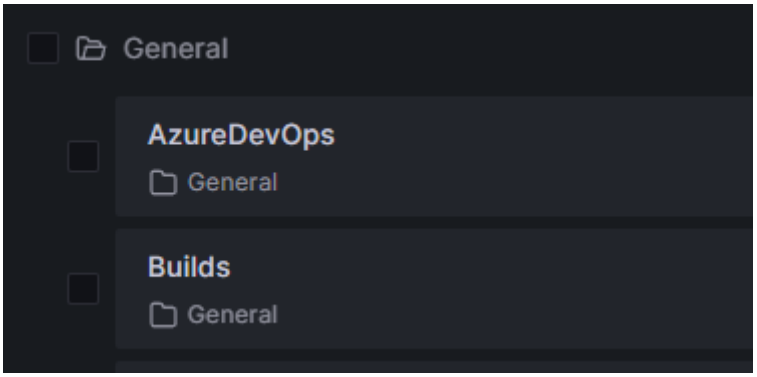
1. Ingresar al siguiente link de gitlab para obtener el demo dashboard en Grafana labs: https://gitlab.com/escobarjsa/dashboard-grafana.git
2. Clonar el repositorio “dashboard-grafana”.



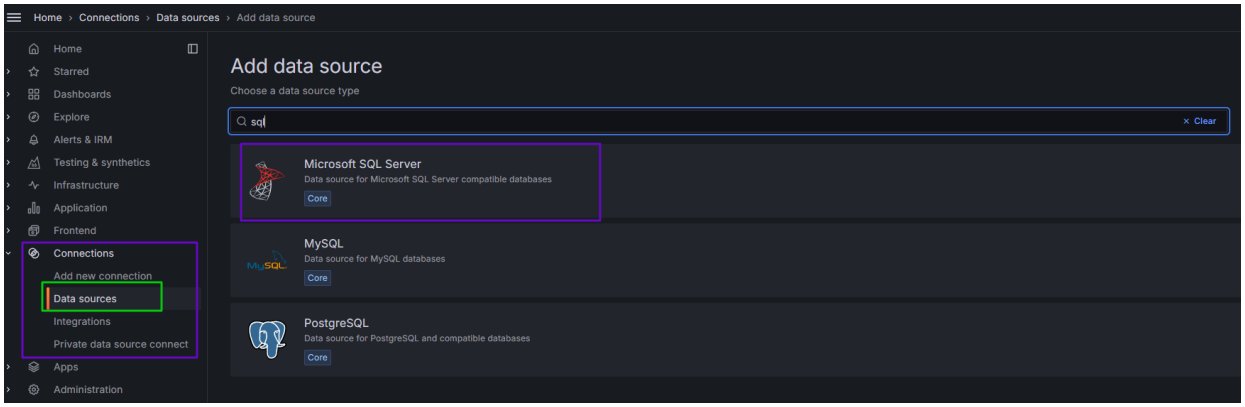
3. Ingresar a https://grafana.com/ iniciar sesión e importar los dashboard que se encuentran en la carpeta.



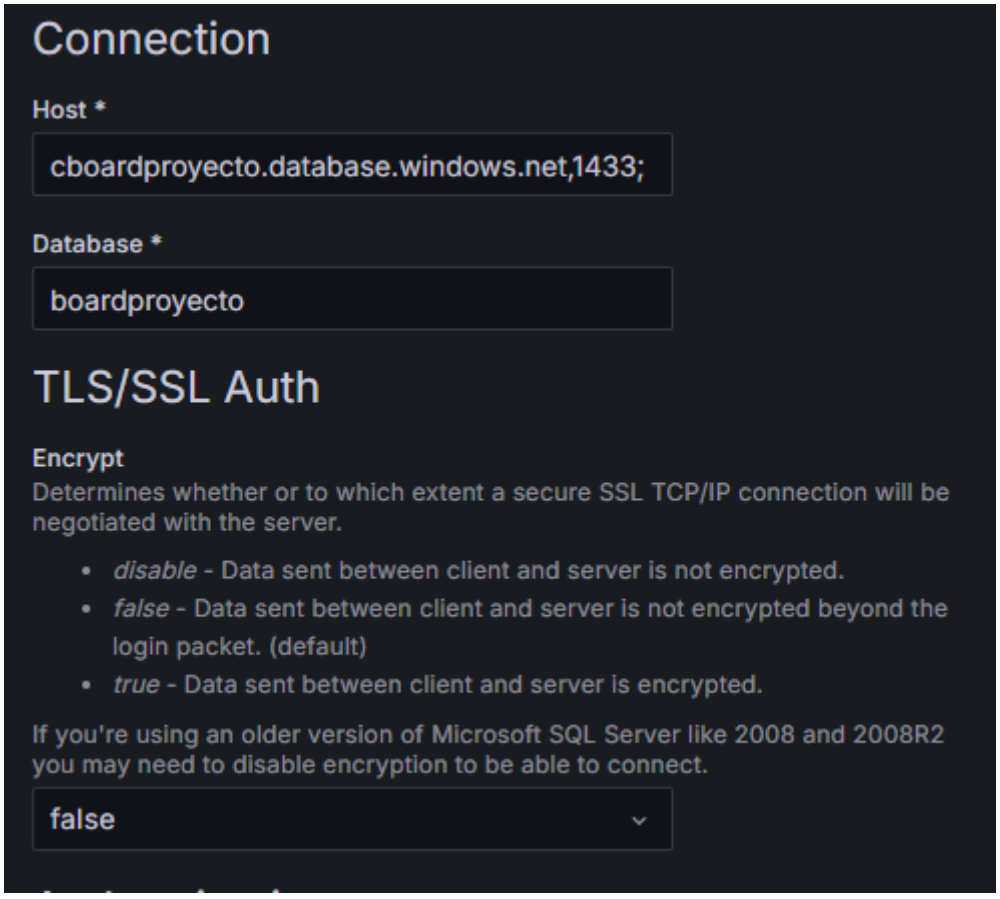
4. Una vez importados los dashboards los encontraremos en la carpeta general.



5. Crear una conexión con la base de datos que se creó previamente con Grafana, en el apartado de Connections, Data sources, Microsoft SQL Server.



6. En el siguiente meno se agrega el host que es la conexión con la base de datos y el puerto, así mismo, el nombre de la base de datos, username y password.


Connection
Host *
cboardproyecto.database.windows.net,1433;
Database *
boardproyecto
TLS/SSL Auth
Encrypt
Determines whether or to which extent a secure SSL TCP/IP connection will be negotiated with the server.
disable - Data sent between client and server is not encrypted.
false - Data sent between client and server is not encrypted beyond the login packet. (default)
true - Data sent between client and server is encrypted.
If you're using an older version of Microsoft SQL Server like 2008 and 2008R2 you may need to disable encryption to be able to connect.
false

Anexo 2: Cronograma

