

UNIVERSIDAD TÉCNICA DEL NORTE



Facultad de Ingeniería en Ciencias Aplicadas

Carrera de Software

**ELABORACIÓN DE UNA GUÍA METODOLÓGICA PARA LA GESTIÓN DE PROYECTOS
DE SOFTWARE USANDO MICROSOFT AZURE DEVOPS Y EL MARCO DE TRABAJO
SCRUM.**

Trabajo de grado previo a la obtención del título de Ingeniero de Software

Autor:

Ervin Patricio Cabascango Santacruz

Director:

MSc. José Antonio Quiña Mera

Ibarra – Ecuador 2022



UNIVERSIDAD TÉCNICA DEL NORTE

BIBLIOTECA UNIVERSITARIA

AUTORIZACIÓN DE USO Y POBLACIÓN

A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

1. IDENTIFICACIÓN DE LA OBRA

En cumplimiento del Art. 144 de la Ley de Educación Superior, hago la entrega del presente trabajo a la Universidad Técnica del Norte para que sea publicado en el Repositorio Digital Institucional, para lo cual pongo a disposición la siguiente información:

DATOS DE CONTACTO			
CÉDULA DE IDENTIDAD:	1004071732		
APELLIDOS Y NOMBRES:	CABASCANGO SANTACRUZ ERVIN PATRICIO		
DIRECCIÓN:	JUAN MONTALVO 10-406		
EMAIL:	epcabascangos@utn.edu.ec		
TELÉFONO FIJO:	062 611545	TELÉFONO MÓVIL:	0995215959

DATOS DE LA OBRA	
TÍTULO:	“ELABORACIÓN DE UNA GUÍA METODOLÓGICA PARA LA GESTIÓN DE PROYECTOS DE SOFTWARE USANDO MICROSOFT AZURE DEVOPS Y EL MARCO DE TRABAJO SCRUM”
AUTOR:	CABASCANGO SANTACRUZ ERVIN PATRICIO
FECHA:	06/10/2022
PROGRAMA:	<input checked="" type="checkbox"/> PREGRADO <input type="checkbox"/> POSGRADO
TÍTULO POR EL QUE OPTA:	INGENIERÍA DE SOFTWARE
ASESOR/DIRECTOR:	MSC. ANTONIO QUIÑA

2. CONSTANCIAS

El autor manifiesta que la obra objeto de la presente autorización es original y se la desarrolló, sin violar derechos de autor de terceros, por lo tanto, la obra es original y que es el titular de los derechos patrimoniales, por lo que asume la responsabilidad sobre el contenido de la misma y saldrá en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, a los 6 días del mes de octubre de 2022.

EL AUTOR



Nombre: Ervin Patricio Cabascango Santacruz

Cédula: 1004071732



CERTIFICADO DEL DIRECTOR DE TRABAJO DE GRADO
UNIVERSIDAD TÉCNICA DEL NORTE
FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CERTIFICACIÓN DEL ASESOR

Certifico que la Tesis previa a la obtención del título de Ingeniera en Software con el tema:

“ELABORACIÓN DE UNA GUÍA METODOLÓGICA PARA LA GESTIÓN DE PROYECTOS DE SOFTWARE USANDO MICROSOFT AZURE DEVOPS Y EL MARCO DE TRABAJO SCRUM” ha sido desarrollada y terminada en su totalidad por el Sr. CABASCANGO SANTACRUZ ERVIN PATRICIO, con cédula de identidad Nro. 1004071732-2 bajo mi supervisión para lo cual firmo en constancia.

MsC. Antonio Quiña Mera

DIRECTOR DE TESIS

DEDICATORIA

El presente trabajo de grado se lo dedico a mi madre Patricia Santacruz por todo el trabajo, esfuerzo, sacrificio amor y apoyo incondicional que me ha brindado, a mi padre Patricio Cabascango, a mis hermanos Odalis Cabascango y Steven Cabascango espero ser un ejemplo para alentarles a cumplir todas sus metas.

A Karol, que con su carisma e inteligencia me ha inspirado a luchar y a perseguir mis sueños, además de hacerme entender de que estoy hecho para grandes cosas.

A mis familiares, amigos que confiaron y me brindaron su apoyo a lo largo de mi carrera universitaria

AGRADECIMIENTO

Quiero agradecer a mis padres por su apoyo incondicional, en especial a mi madre Patricia que me ha inspirado y me ha enseñado a salir de toda adversidad.

Agradezco a mis amigos quienes han hecho de este largo camino universitario una aventura inolvidable.

Agradezco los excelentes docentes que tuve en el desarrollo de mi carrera universitaria, quienes con su conocimiento y profesionalismo me inspiraron a desarrollarme como profesional.

Agradezco a mi tutor MSc. Antonio Quiña por su apoyo y seguimiento necesarios durante todo este proceso de titulación.

Agradezco a mis opositores MSc. Cathy Guevara y MSc. Mauricio Rea por todo su tiempo y apoyo valiosos para la realización de este trabajo de titulación.

TABLA DE CONTENIDO

CERTIFICADO DEL DIRECTOR DE TRABAJO DE GRADO.....	IV
CERTIFICACIÓN DEL ASESOR.....	IV
DEDICATORIA	V
AGRADECIMIENTO	VI
TABLA DE CONTENIDO.....	VII
RESUMEN	XV
ABSTRACT	XVI
Introducción.....	1
Antecedentes:	1
Situación actual.....	1
Planteamiento del Problema	1
Objetivos	2
Objetivo General	2
Objetivos Específicos.....	2
Alcance y Metodología	2
Alcance.....	2
Metodología.....	4
Justificación y Riesgos	4
Justificación.....	4
Riesgos.....	5
CAPÍTULO I	7
1 Marco Teórico.....	7
1.1 Gestión de proyectos de software.	7
1.2 Conceptos generales.	8
1.1.1 Proceso de software.	8
1.2.1 Metodología de desarrollo	9
1.2.2 DevOps	10

1.2	Ciclo de Vida del Software	11
1.2.1	Etapas del ciclo de vida	11
1.2.2	Modelos del ciclo de vida.....	12
1.3	Herramientas para gestión de proyectos de software.....	14
1.4	Marco de trabajo SCRUM.....	17
1.5	Microsoft Azure DevOps	22
1.5.1	Azure DevOps Boards	22
1.5.2	Azure DevOps Repos	23
1.5.3	Azure DevOps Pipelines.....	23
1.5.4	Azure DevOps Test Plans	23
1.5.5	Azure DevOps Artifacts	23
CAPÍTULO II		24
2	Metodología y Desarrollo	24
2.1	Revisión de la Literatura	24
2.1.1	Fases para el desarrollo de una guía metodológica.	25
2.1.2	Objetivos, Alcance y Audiencia	26
2.1.3	Recopilación de Información	26
2.2	Elaboración de la guía	27
2.2.1	Aprobación.....	28
2.2.2	Edición y Diseño	28
2.2.3	Difusión	29
2.3	Metodología para el desarrollo de la guía.	29
2.3.1	Estructura de la guía.....	29
2.3.2	SCRUM en el desarrollo.....	30
2.3.3	Azure DevOps y SCRUM.	30
2.3.4	Prueba de concepto.....	31
2.3.5	Difusión y Medición.....	31
2.4	Diseño de la guía metodológica.	31

2.4.1	Pre-juego.....	33
2.4.2	Juego.....	34
2.4.3	Post-juego.....	35
CAPÍTULO III.....		38
3	DESARROLLO DE LA GUÍA METODOLÓGICA.....	38
3.1	Introducción.....	38
3.2	Desarrollo de la Guía	39
3.2.1	Configuración del proyecto	39
3.2.2	Pre-juego.....	41
3.2.3	Juego.....	52
3.2.4	Post-juego.....	59
3.3	Conclusiones de la elaboración de la guía.....	65
CAPÍTULO IV.....		48
4	VALIDACIÓN DE RESULTADOS	48
4.1	Desarrollo de prueba de concepto usando la guía metodológica.....	48
4.1.1	Configuración Proyecto	48
4.1.2	Fase 1. Pre-Juego.	49
4.1.3	Fase 2. Juego.	53
4.1.4	Fase 3. Post-Juego.....	58
4.2	Resultados de la prueba de concepto.	58
4.2.1	Análisis de resultados.....	58
Conclusiones.....		61
Recomendaciones		63
Referencias.....		64

ÍNDICE DE FIGURAS

Fig. 1. Árbol de problemas.....	2
Fig. 2. Azure DevOps Services	3
Fig. 3. Ciclo de vida de un proyecto	4
Fig. 4. Metodología	4
Fig. 5. Nivel de riesgo y su impacto	6
Fig. 6. Estructura Capítulo I.	7
Fig. 7. Procesos de ingeniería de software.....	8
Fig. 8. DevOps.....	10
Fig. 9. Ciclo de vida de software fuera del ámbito productivo.	12
Fig. 10. Eventos SCRUM.....	20
Fig. 11. Ciclo de vida SCRUM.....	21
Fig. 12. Servicios de Azure DevOps.	22
Fig. 13. Estructura Capítulo II.	24
Fig. 14. Fases para el desarrollo de una guía.....	25
Fig. 15. Metodología para el desarrollo de la guía metodológica.	29
Fig. 16. Diseño de la guía metodológica.....	32
Fig. 17. Planeación en Azure DevOps.	33
Fig. 18. Backlogs en Azure DevOps.	34
Fig. 19. Sprints en Azure DevOps.....	35
Fig. 20. Azure DevOps Repos.....	35
Fig. 21. Azure DevOps Test Plans.	36
Fig. 22. Azure DevOps Pipelines.....	37
Fig. 23. Estructura Guía Azure DevOps y SCRUM.	38
Fig. 24. Página principal Azure DevOps.	40
Fig. 25. Crear proyecto en Azure DevOps.....	40
Fig. 26. Vista principal de un proyecto en Azure DevOps.....	41
Fig. 27. Suite de herramientas de Azure DevOps.	41

Fig. 28. Invitar miembros de equipo.	42
Fig. 29. Agregar miembro al proyecto.	43
Fig. 30. Crear equipo Azure DevOps.	43
Fig. 31. Usuarios Equipo Azure DevOps.	44
Fig. 32. Permisos Azure DevOps.	44
Fig. 33. Clasificación de Permisos Azure DevOps.	45
Fig. 34. Permisos Equipo Azure DevOps.	45
Fig. 35. Opciones de permiso de equipo.	46
Fig. 36. Tableros de Backlog Azure DevOps.	47
Fig. 37. Tableros de Backlog Azure DevOps.	47
Fig. 38. Backlog Azure DevOps.	47
Fig. 39. Tableros con ítems de trabajo Azure DevOps.	48
Fig. 40. Editar ítem de trabajo Tableros.	48
Fig. 41. Sprints del proyecto Azure DevOps.	49
Fig. 42. Agregar iteración.	49
Fig. 43. Distribución Sprint Azure DevOps.	50
Fig. 44. Pila Backlog Boards.	50
Fig. 45. Asignar Backlog a un Sprint.	51
Fig. 46. Seleccionar Sprint.	51
Fig. 47. Agregar ítem Product Backlog.	51
Fig. 48. Sprints Azure DevOps.	52
Fig. 49. Sprint por equipo.	53
Fig. 50. Nuevo ítem en Sprint.	53
Fig. 51. Ítem de trabajo Sprint.	53
Fig. 52. Repos Azure DevOps.	54
Fig. 53. Azure Repos Files.	55
Fig. 54. Crear Rama Azure DevOps.	55
Fig. 55. Ramas en Azure Repos.	56

Fig. 56. Commits Azure Repos.....	56
Fig. 57. Commits por rama Azure Repos.....	56
Fig. 58. Pushes Azure DevOps.....	57
Fig. 59. Detalles de Push Azure DevOps.....	57
Fig. 60. Crear un Tag Azure Repos.....	58
Fig. 61. Tags Azure Repos.....	58
Fig. 62. Retrospectiva Sprint Azure Boards.....	59
Fig. 63. Fusionar ramas Azure Repos.....	60
Fig. 64. Agregar Prueba Azure Boards.....	60
Fig. 65. Configurar una prueba.....	61
Fig. 66. Pruebas desde ítem de trabajo.....	61
Fig. 67. Caso de Prueba.....	62
Fig. 68. Tests Plans Azure DevOps.....	62
Fig. 69. Test Runs Azure DevOps.....	62
Fig. 70. Reporte de progreso.....	63
Fig. 71. Conexión de un Pipeline.....	63
Fig. 72. Selección repositorio en Pipeline.....	64
Fig. 73. Configuración Pipeline.....	64
Fig. 74. Revisión de Pipeline.....	65
Fig. 75. Estructura Capítulo IV.....	48
Fig. 76. Creación de proyecto Prueba de concepto.....	48
Fig. 77. Integrantes de Equipo.....	49
Fig. 78. Sprint de Prueba de concepto.....	50
Fig. 79. Product Backlog Prueba de Concepto.....	50
Fig. 80. Tablero Product Backlog prueba de concepto.....	51
Fig. 81. Sprint 1 Base de datos.....	51
Fig. 82. Sprint 2 BackEnd.....	52
Fig. 83. Sprint 3 FrontEnd.....	53

Fig. 84. Sprint 4 Entrega.....	53
Fig. 85. Sprint 1 Prueba de Concepto.....	54
Fig. 86. Sprint 2 Prueba de Concepto.....	55
Fig. 87. Sprint 3 Prueba de Concepto.....	56
Fig. 88. Repositorio Prueba de Concepto.....	56
Fig. 89. Commits Portafolio web.....	57
Fig. 90. Sprint 4 Prueba de Concepto.....	57
Fig. 91. Integración de actividades Prueba de Concepto.....	58
Fig. 92. Escala de evaluación.....	61

ÍNDICE DE CUADROS

TABLA 1. Modelos del ciclo de vida.....	13
TABLA 2. Herramientas para la gestión.....	15
TABLA 3. Documentos totales en base a parámetros establecidos.	25
TABLA 4. SCRUM y las herramientas de Azure DevOps.	32
TABLA 5. Roles Del Proyecto	49
TABLA 6. Prototipo Sprint 1	54
TABLA 7. Prototipo Sprint 2	54
TABLA 8. Prototipo Sprint 3	55
TABLA 9. Prototipo Sprint 4	57
TABLA 10. Medida de Completitud Funcional	58
TABLA 11. Evaluación Guía de Desarrollo	59

RESUMEN

En la actualidad, la gestión de proyectos de software utilizando herramientas basadas en la nube se han vuelto muy populares. No obstante, el uso de metodologías o marcos de trabajo para el desarrollo de software forman parte de la gestión del software. La herramienta basada en la nube como Azure DevOps nos ayuda a llevar una correcta gestión del software a través de sus herramientas integradas, sin embargo, el desconocimiento y la falta de información que nos brinda una guía metodológica con Azure DevOps que describa un proceso ya probado para seguir, hace que muchos profesionales y estudiantes TI recaigan en el empirismo para la gestión de su proyecto, de esta manera los proyectos de software que se realizan son productos de una causalidad.

El presente trabajo se enfoca en la elaboración de una guía metodológica para la gestión de proyectos de software usando Azure DevOps y el marco de trabajo SCRUM. Dicha guía metodológica está basada en las fases de SCRUM (pre-juego, juego, post-juego) y su correlación con las herramientas de Azure DevOps (Boards, Repos, Pipelines, Test Plans).

Por otro lado, la guía metodológica se validó mediante una prueba de concepto que consistió en el desarrollo de una aplicación web gestionada con las herramientas de Azure DevOps.

Finalmente, la guía fue evaluada en base a una medida de completitud funcional y una escala de evaluación sustentada en base a la norma ISO/IEC 25040.

Palabras clave: Azure DevOps, SCRUM, guía metodológica, gestión software, herramientas.

ABSTRACT

Nowadays, software project management using cloud-based tools have become very popular. However, the use of methodologies or frameworks for software development are part of software management. The cloud-based tool like Azure DevOps helps us to carry out a correct management of the software through its integrated tools, however, the lack of knowledge and the lack of information provided by a methodological guide with Azure DevOps that describes an already proven process To continue, it makes many IT professionals and students fall back on empiricism for project management, in this way the software projects that are carried out are products of causality.

The present work focuses on the elaboration of a methodological guide for the management of software projects using Azure DevOps and the SCRUM framework. This methodological guide is based on the SCRUM phases (pre-game, game, post-game) and its correlation with Azure DevOps tools (Boards, Repos, Pipelines, Test Plans).

On the other hand, the methodological guide was validated through a proof of concept that consisted in the development of a web application managed with Azure DevOps tools.

Finally, the guide was evaluated based on a measure of functional completeness and an evaluation scale based on the ISO/IEC 25040 standard.

Keywords: Azure DevOps, SCRUM, methodological guide, software management, tools.

Introducción

Antecedentes

El desarrollo de software actualmente es un componente de suma importancia para el desarrollo de la sociedad, la demanda del software cada día va en aumento a tal grado de que hoy en día podemos encontrar marcos de trabajo que nos ayudan a agilizar el proceso de desarrollo de software.

Muchas veces el desarrollo de software es manejado con poca experiencia en cuanto a gestión y administración, según el Standish Group los proyectos de software tienen una tasa de más del 70% de fracasos (Techbizdesign, 2018), muchos son los factores que influyen para que un proyecto de software fracase, desde la elección de un marco de trabajo que no cumple con las necesidades del proyecto, hasta el uso de herramientas para la gestión de proyectos de software como el set de Microsoft Azure DevOps Services que ayudan a llevar el proyecto por un buen camino. Sin embargo, el uso empírico que se le da por parte de profesionales y estudiantes TI de ponen en riesgo la sostenibilidad del proyecto (Ocaña, 2019).

En consecuencia, esto implica una relación de tiempo y costo que resultan contraproducentes. Ante estos factores es normal pensar que los proyectos de software están condenados desde el principio.

Situación actual

Actualmente la gestión de proyectos de software que se realizan con el set de Microsoft Azure DevOps Services no hacen uso de un marco de trabajo para llevar a cabo una gestión formal de un proyecto de software, debido a esto muchos profesionales y estudiantes TI recaen en el empirismo para la gestión de su proyecto, de esta manera los proyectos de software que se realizan son producto de una causalidad (Nayebi et al., 2016).

Planteamiento del Problema

Existe un alto índice de fracaso en la gestión de proyectos de software por parte de los estudiantes y docentes de las carreras de Ingeniería en Sistemas (CISIC) e Ingeniería de Software (CSOFT), los motivos devienen desde una pobre gestión en los proyectos, el uso inadecuado de herramientas de gestión o un bajo grado de aplicabilidad de un marco de trabajo que se ajuste al objetivo del proyecto (Zavala, 2004).

La principal razón de este problema es el desconocimiento, la falta de información que nos brinde una guía metodológica la cual es un documento técnico que describe un proceso ya probado para seguir, con la finalidad de llevar a cabo una gestión formal del proyecto (Robles,

2017). A continuación, identificamos los problemas, causa y efectos que conllevan al fracaso de los proyectos de software:



Fig. 1. Árbol de problemas
Fuente propia

Objetivos

Objetivo General

Elaboración de una guía metodológica para la gestión de proyectos de software usando Microsoft Azure DevOps y el marco de trabajo SCRUM.

Objetivos Específicos

- Elaborar un marco teórico y tecnológico para el desarrollo de proyectos de software.
- Elaborar una guía metodológica para la gestión de proyectos de software con Microsoft Azure DevOps basado en el marco de trabajo SCRUM.
- Validar la guía metodológica mediante el desarrollo de una prueba de concepto.

Alcance y Metodología

Alcance

El presente tema se centra en el desarrollo de una guía metodológica la cual facilitará el desarrollo del proyecto de software de una forma más sencilla permitiendo así que estudiantes y docentes de las carreras de Ingeniería en Sistemas (CISIC) e Ingeniería de Software (CSOFT) de la Universidad Técnica del Norte los cuales hacen uso de las tecnologías de la información, puedan llevar a cabo la gestión de proyectos y desarrollo de software basándonos en el marco de trabajo SCRUM y Microsoft Azure DevOps Services como herramienta para la gestión.

Además, validar esta guía metodológica mediante pruebas de concepto que irán de la mano con el desarrollo de un aplicativo en la cual aplicaremos dicho marco de trabajo y herramientas para la gestión de proyectos de software.

Se validará la guía mediante una prueba de concepto de un aplicativo web “portafolio personal” que contará con las siguientes tecnologías:

- Base de datos: PostgreSQL
- IDE de Desarrollo: Visual Studio Code.
- BackEnd: Node JS.
- FrontEnd: Angular con API Rest.
- Servidor de aplicaciones: Express.

La gestión del proyecto se llevará a cabo por medio del set de herramientas que ofrece Microsoft Azure DevOps.

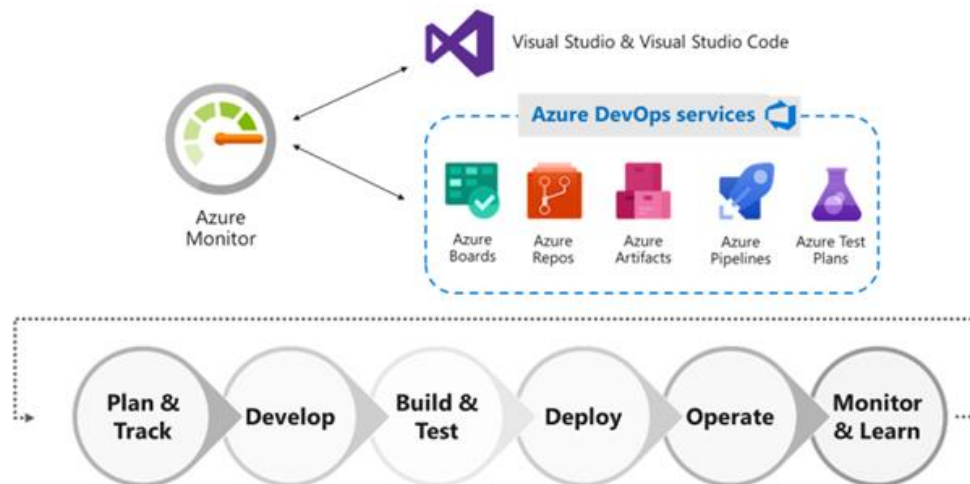


Fig. 2. Azure DevOps Services
Fuente: Conde, A. (2019).

La prueba de concepto utilizara a SCRUM como marco de trabajo para la ejecución de actividades y tiempos para el desarrollo del aplicativo.

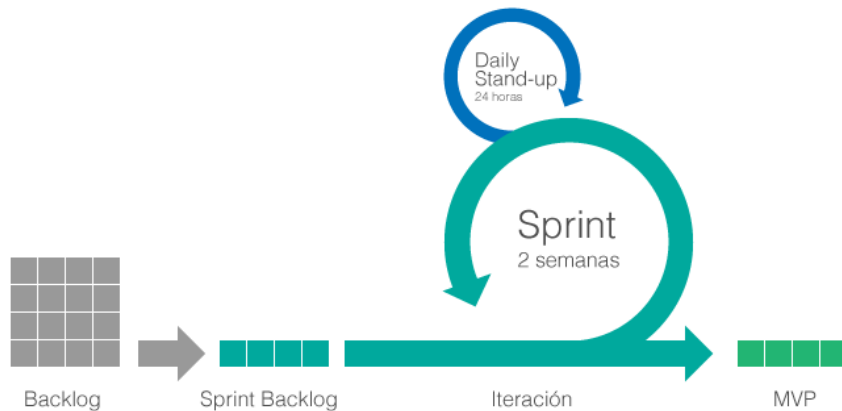


Fig. 3. Ciclo de vida de un proyecto
Fuente: Calle, M. (2020).

Metodología

En el presente proyecto se realizará una guía metodológica para la gestión de proyectos de software, por lo cual se realizará una prueba de concepto que consistirá en el desarrollo de un aplicativo web que nos permitirá validar dicha guía. Se procederá con una revisión de la literatura, definir un marco teórico y tecnológico, diseño de la guía, elaboración de la guía metodológica basada en SCRUM, validación de la guía mediante una prueba de concepto.

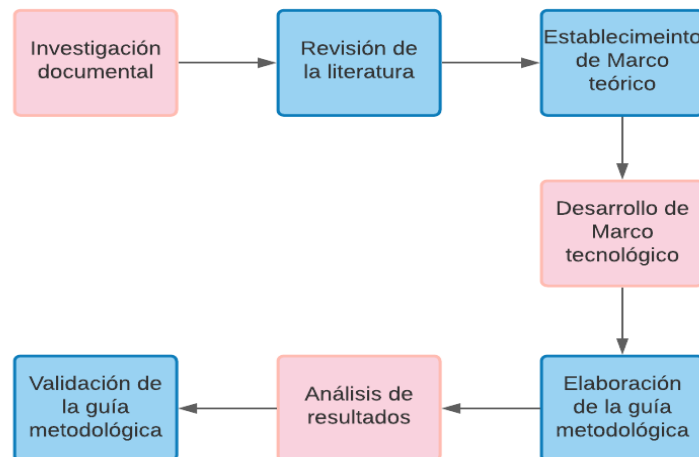


Fig. 4. Metodología
Fuente: Propia

Justificación y Riesgos

Justificación

El presente proyecto está enfocado en dos objetivos del desarrollo sostenible (ODS), a continuación, se describe dichos objetivos:

El objetivo 8: Trabajo decente y crecimiento económico, se enfoca en el crecimiento económico e impulsa el progreso. (Naciones Unidas, 2019)

La meta 8.2 del objetivo 8 incita al uso de tecnología e innovación para obtener un nivel más alto de productividad, dicha meta hace referencia al éxito de los proyectos en el sector tecnológico de software, con el cual se busca la innovación y transformación digital.

El objetivo 9: Industria, innovación e infraestructuras, la cual promueve la innovación tecnológica que junto con la infraestructura da riendas sueltas a la generación de empleo e ingresos, un punto clave para el desarrollo. (Naciones Unidas, 2019) .

No obstante, la meta 9.4 del objetivo 9 se aplica en este estudio, ya que se busca adoptar nuevas tecnologías y procesos de manera formal para la gestión de proyectos de software, de esta manera aprovechar los recursos tecnológicos de mejor manera.

Justificación Tecnológica. - La tasa de fracasos de proyectos de software va en aumento, el uso empírico sobre herramientas que ayudan a la gestión de proyectos de software como Microsoft DevOps hacen que la mayoría de los proyectos que salen a flote sean producto de una causalidad, para lo cual es necesario implementar una guía metodológica para el correcto manejo de Microsoft DevOps y así reducción la causalidad de los proyectos de software y disminuyendo la tasa de fracasos de dichos proyectos (Nayebi et al., 2016).

Justificación Metodológica. – Se validará la guía metodológica para la gestión de desarrollo de proyectos de software utilizando SCRUM como marco de trabajo para el desarrollo, además validaremos la guía de gestión de software aplicando el estándar ISO/IEC 25000 enfocado en las medidas de completitud funcional.

Riesgos

El presente proyecto se enfoca principalmente en el trabajo en la nube, es así como por medio de internet tenemos acceso a los servicios que ofrece Microsoft DevOps Services, que gestiona nuestro proyecto de software.

Inaccesibilidad al servicio Web. Los servicios de Azure se encuentran alojados en la nube, para la cual el acceso al servicio web es de suma importancia, por lo cual un plan de internet sería la mejor opción para restaurar la conexión al servicio.

Mantenimiento de los Servicios de Azure. La plataforma aloja los servicios de Azure suele realizar mantenimientos constantes, la posibilidad de que el servicio este fuera de funcionamiento es muy baja, en caso de que llegue a suceder podemos conectarnos con servicios que ofrece Microsoft como: SharePoint

Cobro por suscripción mensual. Los proyectos que demandan mayor número de personas suelen tener el costo de una suscripción, si el proyecto demanda muchas personas

para el equipo lo que se puede hacer es acceder como usuarios invitados, los cuales no tienen que pagar suscripción por el uso del servicio.

Pérdida de información. En ocasiones la información no se la maneja de forma segura, Azure tiene un sistema de respaldo que permite recuperar nuestra información, a través de SharePoint de Microsoft.

Prohibición de usuarios externos. Si los usuarios son externos a la organización es necesario darse de alta en el servicio, una manera de solventar este riesgo es crear un perfil a través de Azure DevOps, el cual no tiene costo.

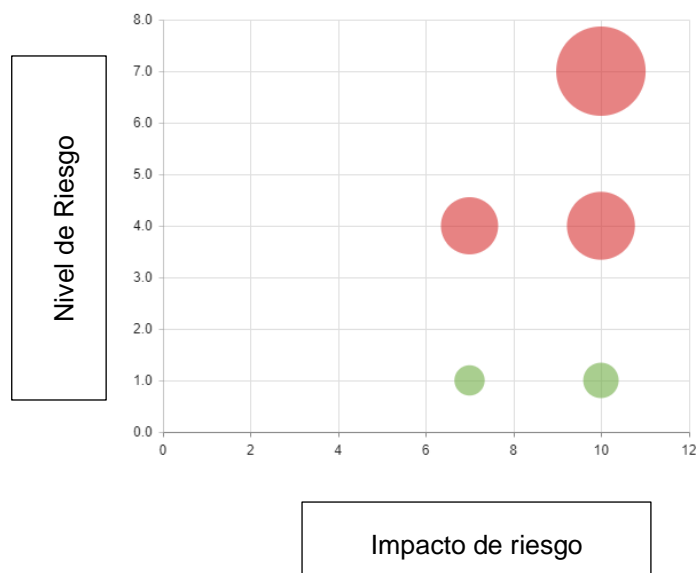


Fig. 5. Nivel de riesgo y su impacto
Fuente: Propia

CAPÍTULO I

1 Marco Teórico

En este capítulo se estableció un marco de trabajo, el cual sirvió para realizar la fundamentación teórica de los aspectos para la Elaboración de una guía metodológica para la gestión de proyectos de software utilizando Microsoft Azure DevOps y el marco de trabajo SCRUM. A continuación, en la Fig. 6 se muestra la estructura de este capítulo.

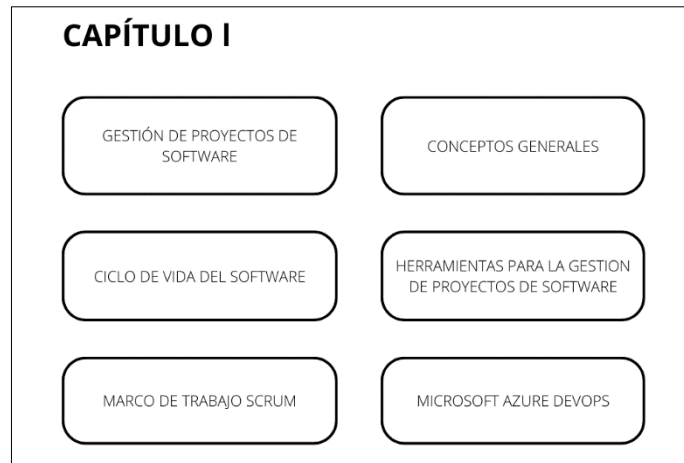


Fig. 6. Estructura Capítulo I.
Fuente: Propia

1.1 Gestión de proyectos de software

La gestión de proyectos se puede definir como el inicio de una actividad para alcanzar objetivos establecidos mediante la aplicación de conocimientos, habilidades, herramientas y técnicas en las diferentes actividades de un proyecto (Lopez, 2014). La gestión de proyectos involucra distintas áreas como la gestión de calidad, la gestión de tiempo o la gestión de comunicación, formando un ciclo dinámico que transcurre del planteamiento a la ejecución y control (Otalora-Luna et al., 2018).

La gestión de proyectos de software es un área de conocimiento de gran importancia, según los expertos, es necesario tomar en cuenta en el camino al éxito en el desarrollo de software (Calderón & Ruiz, 2015). Es importante tomar en cuenta que un proyecto de desarrollo de software está basado en los requisitos del cliente e implantación del producto de software. Por esta razón, los proyectos de software exigen no solo habilidades generales de gestión de proyectos, sino que también buenas habilidades de ingeniería de software. Un objetivo de cualquier gestión de proyectos de software es el desarrollo y el mantenimiento de un producto mediante la aplicación de buenos principios de gestión de proyectos, así como principios de ingeniería de software (García, 2014).

Comúnmente los procesos de gestión de proyectos deben incluir el inicio del proyecto, la planificación del proyecto, el seguimiento y control del proyecto y finalmente el cierre del proyecto. Sin embargo, los procesos de ingeniería de software deben incluir:

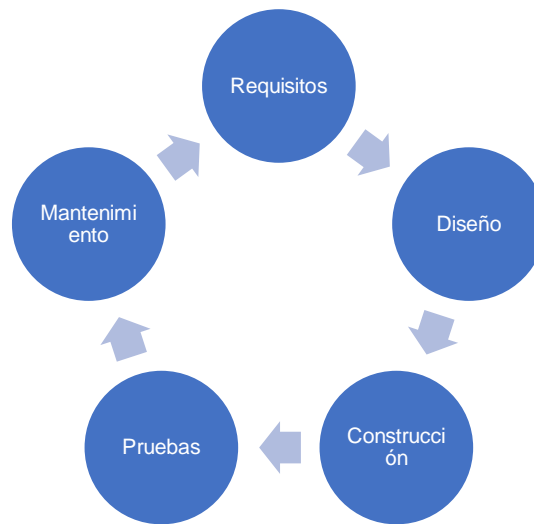


Fig. 7. Procesos de ingeniería de software.
Fuente: Propia

De esta manera se puede definir la gestión de proyectos de software como la aplicación de la gestión de proyectos y métodos de ingeniería de software para desarrollar y mantener un producto de software de modo que el objetivo de desarrollo y mantenimiento de un producto de software sea posible utilizando los mínimos recursos posibles y dinero en el mínimo tiempo posible. El éxito de un proyecto de software radica en la adecuada coordinación de los involucrados en el proceso del desarrollo del software (Lopez, 2014).

1.2 Conceptos generales

1.1.1 Proceso de software

En la actualidad el desarrollo de software aporta una gran parte al crecimiento de la industria tecnológica, pues la creación de estos cada día se vuelve más sofisticada y de gran complejidad, para ello las empresas encargadas de desarrollar software tienden a realizar procesos para su desarrollo. El proceso de desarrollo de software consiste en una serie de actividades relacionadas que conducen a la creación de un producto de software (Sommerville, 2017).

A continuación, se listan las cuatro actividades principales que deben tener para la ingeniería de software:

- *Especificación:* En la cual se debe definir tanto la funcionalidad del software como las restricciones de su operación.

- *Diseño e implementación:* En esta actividad el software debe desarrollarse para cumplir con las especificaciones.
- *Validación:* Es importante validar el software para asegurarse de que cumpla con los requisitos del cliente.
- *Evolución:* El software tiene que evolucionar para satisfacer las necesidades cambiantes del cliente (Sommerville, 2017).

1.2.1 Metodología de desarrollo

Según (Maida & Pacienza, 2018), una metodología consiste en un conjunto integrado de técnicas y métodos, la cuales nos permiten abordar las actividades para el ciclo de vida de un proyecto.

Por otro lado, una metodología para el desarrollo de software conlleva una manera sistemática de implementar, administrar el proyecto para tener altas probabilidades de éxito en completarlo y satisfacer las necesidades del y el objetivo para el cual fue creado.

Una metodología de desarrollo de software o metodología de desarrollo de sistemas en ingeniería de software es un marco de trabajo que se usa para estructurar, planificar y controlar el proceso de desarrollo de un sistema de información (Maida & Pacienza, 2018).

Las metodologías se dividen en dos grupos importantes, los cuales se listan a continuación:

Metodología Tradicional

Muchas veces denominadas como metodologías pesadas, estas centran su atención en la documentación exhaustiva del proyecto, la planificación y control de este, en especificaciones precisas de requisitos y modelado y en cumplir con un plan de trabajo, definido todo esto, en la fase inicial del desarrollo del proyecto (Maida & Pacienza, 2018). Cabe recalcar que este tipo de metodologías requiere de cierta disciplina para ser ejecutada, de este modo obtener un producto más eficiente.

Metodología Ágil

Las metodologías ágiles fueron creadas a partir de las metodologías tradicionales, enfocándose en resolver los problemas que pueden ocasionar dichas metodologías, basando su funcionalidad en la adaptabilidad de los procesos para el desarrollo de un producto.

Según (Cadavid, A. N., Martínez, J. D. F., & Vélez, 2013), las metodologías ágiles son flexibles, pueden ser modificadas para que se ajusten a la realidad de cada equipo

y proyecto. Los proyectos que se realizan con metodología ágil se caracterizan por subdividir los procesos, haciéndolos más pequeños.

1.2.2 DevOps

El término *DevOps* tiene origen de la combinación de las abreviaturas Dev (desarrollo) y Ops (Operaciones), lo cual se puede definir como un marco de trabajo en el que existe una correlación de ideas y aportes entre el departamento de desarrollo y el departamento de operaciones, con el fin de mejorar el proceso de desarrollo de software y obtener un producto de mayor calidad (Autentia, n.d.).

Según (Ebert et al., 2016), DevOps consiste de desarrollo y aprovisionamiento rápidos y flexibles procesos de negocios. Integra e implementa el desarrollo, entrega, y operaciones, facilitando así una conexión ágil y fluida de estos silos tradicionalmente separados. Este marco de trabajo está distribuido en fases importantes las cuales se pueden visualizar en la Fig. 8 y se listan a continuación:

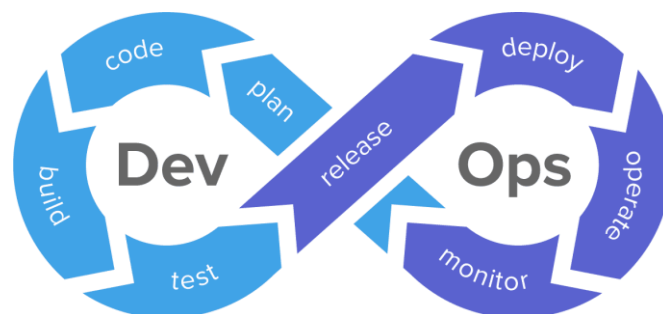


Fig. 8. DevOps.

Fuente: <https://www.inbest.cloud/comunidad/azure-devops-para-automatizar-tus-%C3%A1reas-y-procesos-internos>

Fases DevOps.

- *Planeación*: Esta fase está orientada a identificación de los requisitos por parte del cliente, dándose como producto de esta fase la una ruta de trabajo enfocada al desarrollo del producto.
- *Codificación*: Enfocada en el diseño y codificación, además de la definición de las herramientas utilizadas para el desarrollo del producto.
- *Compilación*: En esta etapa las DevOps toman importancia ya que, si se sujeta alguna actualización del código fuente, es necesario unir todos los componentes para una revisión para la temprana identificación de problemas.
- *Pruebas*: Después de concluir la etapa de compilación, es necesario realizar las pruebas dentro de un entorno real, dichas pruebas deben ser manuales y

automáticas. Estas pruebas dependen de la organización que se realiza durante la planificación.

- *Liberación:* También conocida como el lanzamiento del producto, en esta etapa la compilación esta lista y ha sido sometida a pruebas con resultados satisfactorios, en esta etapa se puede optar por automatizar ciertas compilaciones.
- *Despliegue:* En esta fase, el producto es lanzado a producción. Si es detectado algún inconveniente con la nueva compilación, simplemente se pueden dirigir las solicitudes al entorno anterior mientras se da solución (Felipe & Núñez, 2022).
- *Operación:* La versión del producto se encuentra trabajando y se espera que todo funcione sin inconvenientes. Considerando que el cliente es el mejor equipo de pruebas, es vital conocer su retroalimentación para dar forma al futuro desarrollo del producto (Felipe & Núñez, 2022).
- *Monitoreo:* En esta etapa se realiza una recopilación de los datos y funcionamiento para solventar posibles errores del cliente mediante el uso del producto.

1.2 Ciclo de Vida del Software

El desarrollo de software es un proceso altamente complejo que exige su elaboración sea llevada a cabo en grupo, requiere un conjunto de capacidades que, por lo general, no se encuentran en una sola persona y por ello es necesario contar con un equipo de trabajo que cubra las necesidades requeridas. El Ciclo de Vida del Software, es también conocido como SDLC, por sus siglas en inglés, Systems Development Life Cycle (Ashfaque Ahmed, 2011).

El ciclo de vida de un proyecto se divide en varias fases necesarias para el análisis, la especificación del producto final y la validación de requisitos del software (García, 2014). Lo que garantiza el cumplimiento de los requisitos funcionales que debe cumplir el software, además de la verificación del desarrollo que asegura la calidad de la metodología y si la misma es adecuada.

1.2.1 Etapas del ciclo de vida

Según (Cantone, 2006), el ciclo de vida de un software contiene tres etapas, las cuales se detallan a continuación:

- *Planificación:* En esta etapa se idealiza el planteamiento detallado del proyecto para la gestión del proyecto.
- *Implementación:* Se realizan las actividades que conllevan el desarrollo del producto para su puesta en producción.

- Producción: El proyecto entra en una etapa de definición, en donde se puede presentar el producto al cliente final, una vez se hayan realizado las respectivas pruebas que verifiquen que el producto funciones correctamente. Dicha etapa es de suma importancia debido a las múltiples dificultades que suele presentar en la práctica, alargándose excesivamente y provocando costos no previstos (Cantone, 2006).

Además de dichas etapas mencionadas anteriormente, es importante enunciar dos etapas más debido a su diferenciación e importancia requeridas.

- Inicio: Considerada como la concepción de la idea, esta etapa trata de habilitar los mecanismos necesarios para que las ideas del usuario puedan tomar forma y de este modo sean transformadas en requerimientos (Cantone, 2006). Esta etapa define un porcentaje de éxito en el proyecto.
- Control de Producción: El producto está en constante observación, con la finalidad de saber si los requerimientos iniciales cumplen o se alejan del objetivo. Además se incluyen factores como el liderazgo, documentación, capacitación proporcionada a los usuarios finales (Cantone, 2006).

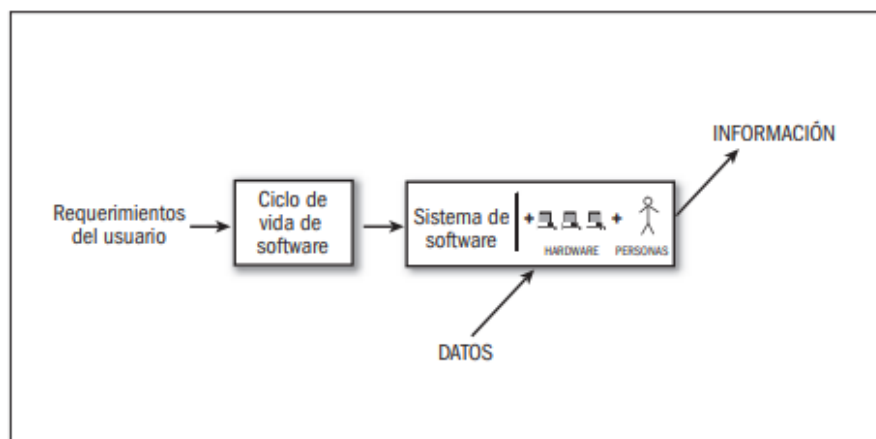


Fig. 9. Ciclo de vida de software fuera del ámbito productivo.
Fuente: (Cantone, 2006)

1.2.2 Modelos del ciclo de vida

Los diferentes modelos de ciclo de vida para la construcción de software se dividen en tres visiones fundamentales:

- Alcance del ciclo de vida: Depende hasta donde se requiere llegar con el proyecto.
- Calidad y cantidad de etapas: Se dividen según el ciclo de vida y el proyecto que se adopte.
- Estructura y sucesión de etapas: Si existe retroalimentación entre etapas y posibilidad de realizar iteraciones.

Según (Cantone, 2006), en los distintos modelos de ciclo de vida se menciona el riesgo que conlleva elegir uno de ellos. En lo referente al riesgo, es necesario saber si se puede retomar un punto de partida y evitar la pérdida de tiempo, dinero y esfuerzo.

TABLA 1. Modelos del ciclo de vida

MODELO	CARACTERÍSTICAS
Lineal	<ul style="list-style-type: none"> • Es el modelo más sencillo de todos. • Descompone cada actividad en etapas separadas para ser realizadas de manera lineal. • Es necesario que se conozca desde el primer momento lo que va a ocurrir en las distintas etapas antes de realizarlas
Cascada	<ul style="list-style-type: none"> • Es un modelo que admite interacciones, se caracteriza por realizar una o varias revisiones para pasar a la siguiente etapa • Los resultados se visualizan al final de cada etapa, lo que conlleva a la identificar errores con retardo, por ende, aumentando el costo, tiempo en el desarrollo. • Es un modelo teórico ya que el usuario pocas veces mantiene los requerimientos iniciales.
En V	<ul style="list-style-type: none"> • Similar al ciclo de vida en cascada. A diferencia del modelo en cascada, en este ciclo se agregaron subáreas de retroalimentación entre las etapas de análisis y mantenimiento y en el diseño y debugging.
Iterativo	<ul style="list-style-type: none"> • Este modelo busca reducir el riesgo entre las necesidades del usuario y el producto final. • Se basa en iteraciones, en las cuales el cliente evalúa el proyecto después de cada iteración hasta satisfacer sus necesidades. • Se puede optar por este modelo cuando los requerimientos del usuario no están claros.
Evolutivo	<ul style="list-style-type: none"> • En este modelo los requerimientos del usuario son susceptibles al cambio.

Incremental	<ul style="list-style-type: none"> • Los problemas sobre requerimientos se afrontan en una iteración de ciclos <i>requerimientos-desarrollo-evaluación</i>. • Su principal objetivo es la construcción del producto por módulos, de este modo ir aumentando las capacidades del software gradualmente. • Es una repetición del ciclo de vida en cascada aplicada a cada funcionalidad del programa.
Espiral	<ul style="list-style-type: none"> • Consiste en la ejecución de una serie de ciclos repetitivos, con la finalidad de ganar madurez hasta obtener el producto final. • Este modelo se caracteriza por sus las actividades como: <i>Planificación, Análisis de riesgo, Implementación, Evaluación</i> las mismas que se envuelven en cada etapa del desarrollo. • Algunas de las desventajas incluyen el costo temporal de cada vuelta de la espiral, dificultad de evaluar riesgos y necesidad continua con el cliente.

Fuente: (Cantone, 2006)

1.3 Herramientas para gestión de proyectos de software

Las herramientas para la gestión de proyectos nos permiten realizar la planificación de proyectos, además del manejo y control de presupuesto, asignación de recursos, los cuales son usados para manejar la complejidad que conlleva un proyecto (Bazán et al., 2012).

El objetivo principal de estas herramientas radica en la planificación estratégica del proyecto, de esta manera permitiéndonos trazar una ruta a seguir durante la realización del proyecto. Según (Bazán et al., 2012), dichas herramientas se pueden presentar dependiendo sus características o puede estar basadas en tecnologías web, para así permitirnos un trabajo colaborativo, el tipo de diagramación que puedan hacer o si permiten llevar el manejo de riesgos entre otras funciones que suelen ofrecerse.

A continuación, en la TABLA 2, se describen las herramientas para la gestión de proyectos de software más populares:

TABLA 2. Herramientas para la gestión

HERRAMIENTA PARA LA GESTIÓN	CARACTERÍSTICAS
MS Project Profesional	<ul style="list-style-type: none"> • Es una excelente herramienta diseñada por Microsoft para llevar a cabo la planificación de un proyecto, permite la integración con otros programas de Office y con herramientas colaborativas como Microsoft SharePoint. • Por el contrario, cabe destacar que la gestión documental no resulta ser muy útil, así como la colaboración e intercambio de información en tiempo real.
SINNAPS	<ul style="list-style-type: none"> • Creada por Sinnaps, está basada en la gestión de proyectos y carteras, control de presupuesto, gestión de roles y permisos y gestión de recursos con el objetivo de maximizar estos últimos y optimizar tiempo (Soto, 2017). • Es una herramienta bastante completa que nos permite gestionar las tareas, tiempo, recursos y presupuesto del proyecto, con una ayuda visual muy clara de toda la información.
WRIKE	<ul style="list-style-type: none"> • Es una herramienta que permite a los usuarios realizar una buena planificación, además de lograr colaboración de los miembros del equipo en tiempo real. • Nos permite gestionar diferentes proyectos a través de una misma herramienta. • No obstante, Wrike no está diseñada para la visualización de relación de la

	<p>duración planificada frente a la duración real.</p>
TRELLO	<ul style="list-style-type: none"> • Trello es un programa para la gestión de proyectos que se caracteriza por la utilización del sistema Kanban, tableros o sistemas de tarjetas para su visibilidad y mejorar el flujo de trabajo entre los miembros del equipo. • Se puede integrar con otras herramientas como Google Drive, Dropbox, Box y OneDrive. • Sin embargo, al ser una herramienta principalmente con finalidad colaborativa carece de varias funciones características de la gestión de proyectos, como la gestión de la carga de trabajo o gestión del presupuesto (Soto, 2017).
JIRA	<ul style="list-style-type: none"> • Es la herramienta ágil ocupada principalmente por los equipos de desarrollo de software TI. • Está diseñada para la gestión de procesos, el seguimiento y estado de las actividades y su visualización.
Asana	<ul style="list-style-type: none"> • Herramienta creada por el co-fundador de Facebook para mejorar la productividad y facilitando funciones dentro de la gestión del proyecto. • Asana permite la integración con una gran variedad de herramientas, tales como Google Drive, Dropbox, Google Sheets, y con otras herramientas de gestión como: Jira y Trello

Podio

- Citrix Podio es un programa para llevar a cabo la gestión de proyectos, se caracteriza por mejorar la organización, comunicación y seguimiento de las tareas.
- Se puede integrar con herramientas como Dropbox, Google Drive, Google Calendar.
- Se caracteriza por su personalización total que ayuda a definir claramente los roles de cada trabajador según el sistema de trabajo del equipo, logrando mejores tiempos de entrega, incrementando la eficacia y las relaciones (Soto, 2017).
- Sin embargo, Podio no nos permite realizar una comparación entre la planificación ejecutada y por hacer, de este modo no es posible saber con claridad las desviaciones de tiempo y presupuesto planificado.

Fuente: (Soto, 2017)

Es importante recalcar que la elección de una herramienta para la gestión de proyectos de software dependerá de la persona a cargo del proyecto o gestor, ya que lógicamente todas las herramientas tienen su enfoque en la gestión de tareas, recursos.

No obstante, la rapidez y cambios en tiempo real juegan un papel muy importante a la hora de gestionar un proyecto, es por esto por lo que muchas de las herramientas se basan en la nube, de esta manera siendo mucho más reactivas y en tiempo real.

1.4 Marco de trabajo SCRUM

Se trata de una metodología de desarrollo ágil, uno de los modelos de ciclo de vida del desarrollo del software más populares y recientes (Urteaga Pecharromán, 2015). El término métodos ágiles, nace como una alternativa de las metodologías formales. El desarrollo y perfeccionamiento de esta metodología se resume en cuatro postulados que se han denominado Manifiesto Ágil, que son los valores sobre los que se asientan estos métodos.

A continuación, se describen estos valores del manifiesto ágil:

- a) A los individuos y su interacción, por encima de los procesos y las herramientas
- b) Más un software funcional, por encima de la documentación exhaustiva
- c) La colaboración con el cliente más que la negociación contractual
- d) La respuesta al cambio más que el seguimiento de un plan

En el marco es posible realizar diversos procesos, técnicas y métodos envolviéndolas en prácticas existentes o las haciéndolas innecesarias. SCRUM hace visible la eficacia relativa de la gestión actual, el entorno y las técnicas de trabajo, de modo que se pueden realizar mejoras (Schwaber & Sutherland, 2020).

SCRUM es un modelo de desarrollo ágil caracterizado por:

- Equipos autónomos y autogestionados que comparten su conocimiento de forma abierta y aprenden juntos.
- Una estrategia de desarrollo incremental mediante iteraciones, en lugar de la planificación completa del producto.
- La calidad del resultado se basa en el conocimiento tácito de las personas y su creatividad, en lugar de la calidad de los procesos empleados.
- A diferencia de la metodología de cascada, se solapan las diferentes fases del desarrollo, en lugar de realizarlas una tras otra.

1.5.1 Artefactos SCRUM

Son las herramientas del SCRUM, es decir, sus bloques de construcción elementales, las cuales ayudan a los roles durante los eventos.

- **Pila del producto (Product Backlog)**

Es el registro de los requisitos del cliente. Se realiza una visión inicial del producto y evoluciona conforme se desarrolla el trabajo. Los requisitos suelen denominarse historias de usuario, que se descomponen en tareas de menor tamaño.

- **La pila del Sprint (Sprint Backlog)**

Se trata de una lista de los trabajos que se realizarán durante un sprint para generar el incremento establecido. Las tareas en el sprint backlog nunca son asignadas, son tomadas por los miembros del equipo del modo que les parezca oportuno (Aruquipa, 2016).

- **Incremento**

Es el resultado de cada sprint, completamente terminado y listo para ser usado.

1.5.2 Eventos SCRUM

En este apartado se detallan las prácticas y actividades que constituyen la rutina de trabajo en SCRUM

- **Sprint**

Es el núcleo fundamental de scrum, en torno al que se organizan todos los demás. A veces se llama también iteración. Son eventos de longitud fija de un mes o menos para crear consistencia. Un nuevo Sprint comienza inmediatamente después de la conclusión del Sprint anterior (Schwaber & Sutherland, 2020).

- **SCRUM diario (Daily SCRUM)**

Es una reunión diaria breve en la que el equipo hace un repaso al avance de cada tarea y se confirma el avance en un ritmo adecuado.

Los Scrums diarios (Daily Scrum) mejoran la comunicación, identifican impedimentos, promueven una rápida para la toma de decisiones, y en consecuencia, eliminan la necesidad de otras reuniones (Schwaber & Sutherland, 2020).

En esta reunión, se tratan los siguientes puntos:

- a) El trabajo que se ha realizado desde la última reunión
- b) Las actividades que se realizarán hasta la próxima reunión de seguimiento
- c) Los impedimentos para que se pueda buscar una solución de forma inmediata

- **Planificación de Sprint**

Marca el inicio de cada sprint, es la reunión de trabajo previa al inicio de cada sprint en la que se determina cuál va a ser el objetivo del sprint y las tareas necesarias para conseguirlo (Palacio, 2015).

- **Revisión del sprint (Sprint Review)**

Análisis e inspección del incremento generado y adaptación de la pila del producto si resulta necesario. Es una reunión donde se presentan los resultados.

- **Retrospectiva sprint (Sprint Retrospective)**

Consiste en la revisión de lo sucedido durante el Sprint. Esta es la reunión en la que el equipo analiza aspectos operativos de la forma de trabajo y crea un plan de mejoras para aplicar en el próximo sprint (Palacio, 2015).

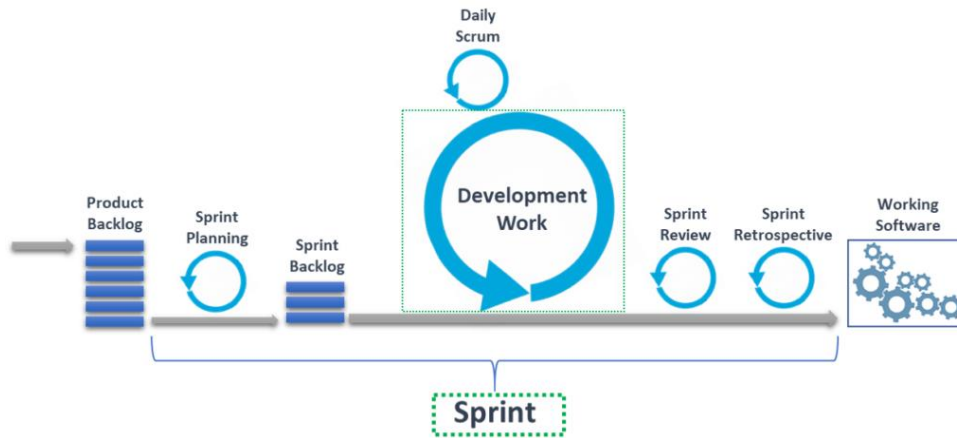


Fig. 10. Eventos SCRUM.

Fuente: <https://jorgeruizagile.com/2019/01/11/lo-que-sabemos-y-no-del-sprint-en-scrum/>

1.5.3 Roles SCRUM

- **Desarrolladores**

Es un equipo de trabajo conformado por miembros multifuncionales de preferencia de tres a nueve miembros. Estos son los encargados de realizar el incremento de cada sprint.

- **Propietario del producto (Product Owner)**

El Propietario del Producto es responsable de maximizar el valor del producto resultante del trabajo del equipo de SCRUM (Schwaber & Sutherland, 2020). Además, es quien toma las decisiones del cliente, por ejemplo, puede ser el responsable de una empresa.

- **Scrum Master**

Es el responsable de las reglas del marco de scrum. Se asegura de que estas sean entendidas por la organización y de que se trabaje conforme a ellas. También se encarga de moderar las reuniones de scrum diarias, gestionar dificultades de dinámica de grupo que puedan surgir y solucionar los impedimentos detectados durante el scrum diario para que el sprint siga avanzando. El Scrum Master actúa como enlace entre el Product Owner y el equipo (Urteaga Pecharromán, 2015).

1.5.4 Ciclo de vida SCRUM

Se encuentra basado en lo que es el desarrollo incremental, es decir, conforme pasen las fases y las iteraciones, va creciendo el tamaño del proyecto que se esté desarrollando, por lo que uno de los principales requisitos para llevarlo a cabo es un equipo de desarrollo de calidad.

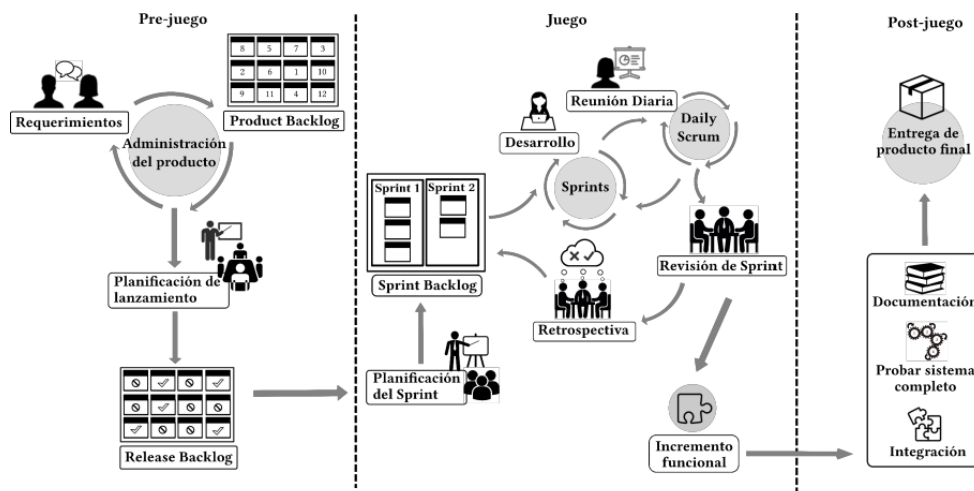


Fig. 11. Ciclo de vida SCRUM.
Fuente: (Urbina Delgadillo et al., 2016)

SCRUM está compuesta por varias fases, que se describen a continuación:

Pre-juego

En esta etapa se especifican las iteraciones, además de la prioridad con la que se van a tratar. Dentro de esta fase podemos encontrar las siguientes tareas:

- Planeación: Definición de una nueva versión basada en la pila actual, junto con una estimación de coste y agenda (Aruquipa, 2016). Las actividades son: escritura de la visión, el presupuesto, el sprint backlog del producto inicial (de alto nivel de abstracción) y los ítems estimados, también la arquitectura de alto nivel, el diseño exploratorio y prototipos (Rivadeneira, 2013).
- Montaje: Se identifican más requerimiento del backlog y se priorizan las tareas para el desarrollo del primer sprint.

Juego

- Planeación Sprint: Esta fase consiste en el desarrollo de un ciclo repetitivo. La gestión determina el cumplimiento de los tiempos, funcionalidad y calidad (Aruquipa, 2016).
- Desarrollo del Sprint: Se enfoca en el desarrollo de las actividades declaradas en el Sprint, según se vayan desarrollando, el producto va a ir evolucionando a lo largo de una serie de iteraciones de desarrollo o sprints.
- Revisión del Sprint: El tiempo estimado para el desarrollo del sprint establece su velocidad e intensidad. El riesgo se evalúa de forma continua a través de las respuestas a los controles adecuados establecidos (Aruquipa, 2016).

Post-juego

Esta fase tiene como objetivo realizar un despliegue operacional del producto, lo que engloban actividades como: documentación, entrenamiento, mercadeo y venta.

1.5 Microsoft Azure DevOps

Azure DevOps es una plataforma de acceso libre, que nos ayuda a la integración de procesos DevOps en diferentes segmentos de TI. Además, esta herramienta admite la integración con varias herramientas, para el análisis de código, verificar la calidad del código, herramientas de seguridad para escanear las vulnerabilidades en el código, herramientas de aprovisionamiento de infraestructura para aprovisionar automáticamente componentes de infraestructura, etc. (K K, 2020).

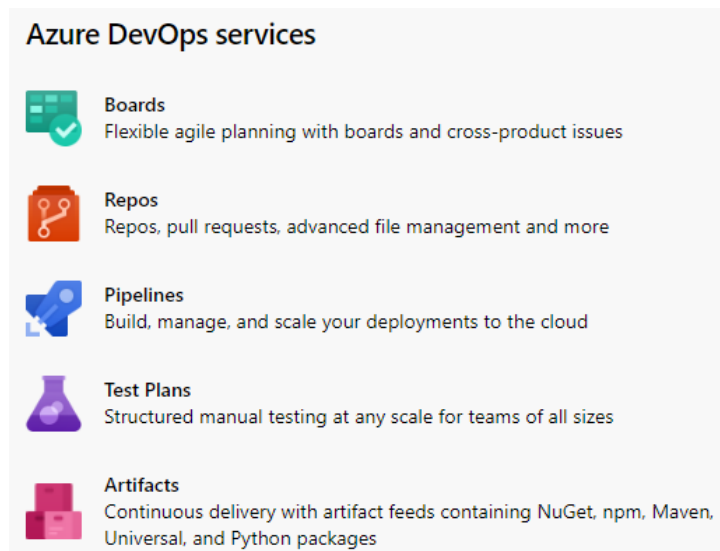


Fig. 12. Servicios de Azure DevOps.
Fuente: Configuraciones de Azure DevOps

El servicio de Azure DevOps nos proporciona herramientas que nos ayudan a realizar la integración continua de nuestro proyecto entre las cuales se encuentran las siguientes:

1.5.1 Azure DevOps Boards

Los tableros de Azure brindan un entorno visual en el que se puede organizar el trabajo a realizar para el desarrollo de software, de esta manera se obtiene un entorno de colaboración que satisface las necesidades del equipo de acuerdo con una metodología de trabajo ágil o DevOps.

Además, los tableros nos permiten realizar fácilmente el seguimiento de tareas, descripciones y posibles errores (Epics, Tasks o Issues) en el proyecto a través de Backlogs o Sprint Backlogs (Ocaña, 2019).

1.5.2 Azure DevOps Repos

Los repositorios de Azure DevOps permiten realizar la administración del código fuente a través del control de versiones en cualquier lenguaje de programación y plataforma de desarrollo. Por ello es posible configurar varios repositorios en un proyecto, de esta manera administrar los diferentes códigos fuente relacionados con el desarrollo de software (Ocaña, 2019).

Para realizar el control de versiones Azure DevOps Repos tiene dos sistemas de control de versiones los cuales son:

- GIT (Control de versiones distribuido)
- Team Foundation Version Control (Control de versiones de Team Foundation Server)

1.5.3 Azure DevOps Pipelines

Azure DevOps Pipelines definen los procesos de compilación y lanzamiento de cualquier plataforma o herramienta asociadas a la integración continua (IC) y despliegue continuo (CD). Los Pipelines nos permiten realizar compilaciones automáticas desde cualquier repositorio de código externo como GitLab (Ocaña, 2019).

Azure Pipelines posee tareas ya definidas que permiten compilar y desplegar el software, esta herramienta puede sustituir el uso de Jenkins, XebiaLabs, TeamCity entre otros que existen en el mercado.

1.5.4 Azure DevOps Test Plans

Azure DevOps Test Plans nos permite realizar definiciones de pruebas como pruebas de carga o pruebas de aceptación, además de agrupar los casos de prueba en función de las características priorizadas con un área específica (K K, 2020).

Para llevar un seguimiento de las pruebas Test Plans nos brinda distintos tableros con información sobre las pruebas que se están llevando a cabo.

1.5.5 Azure DevOps Artifacts

Azure DevOps Artifacts nos permite llevar una gestión de los paquetes utilizados para la implementación del software, mostrándonos así el almacenamiento utilizado por diferentes artefactos, como los artefactos de canalización, los símbolos utilizados para la depuración, las fuentes de paquetes NuGet, NPM, etc. (K K, 2020).

CAPÍTULO II

2 Metodología y Desarrollo

En este capítulo se estableció un marco de trabajo, el cual sirvió para realizar la estructura, diseño y metodología para la elaboración de la guía metodológica para la gestión de proyectos de software utilizando Microsoft Azure DevOps y el marco de trabajo SCRUM. A continuación, en la Fig. 13 se muestra la estructura de este capítulo.

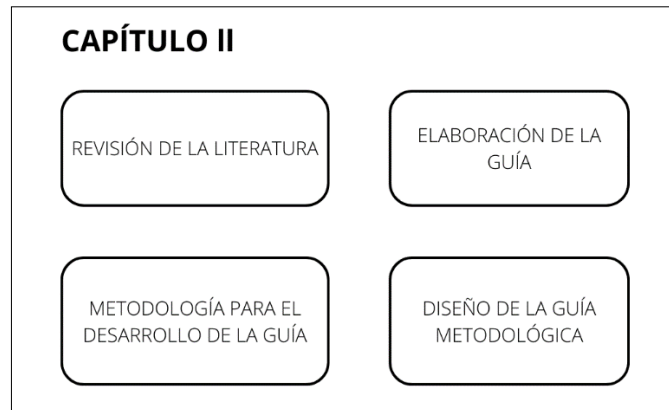


Fig. 13. Estructura Capítulo II.
Fuente: Propia

2.1 Revisión de la Literatura

Es importante recalcar que existe cierta dificultad en encontrar una guía metodológica de buena calidad y actualizada referente a un tema específico, ya que los autores de dichas guías pueden basarse en ejemplos publicados anteriormente (Lockwood & Tricco, 2020) . Por ello, no todos los ejemplos o métodos publicados suelen estar completos. Según (Lockwood & Tricco, 2020) mencionan que una guía clara, detallada, actualizada y de alta calidad debe establecer un alcance de acuerdo con el contexto en donde va a ser utilizada. Tomando en cuentas dichas consideraciones podemos definir que una guía metodología es un documento técnico físico o digital que está conformado por un conjunto de normas e instrucciones a seguir, que buscan la sistematización de un proceso, metodología o práctica (Meza, 2014).

Para ello, en este estudio se realizó una búsqueda de información en las bases de datos bibliográficos SCOPUS, SCIENCEDIRECT, SPRINGER, IEEE XPLORE y otros gestores de contenido, todo esto con el objetivo de encontrar modelos, guías o estándares que permitan establecer un proceso para el desarrollo de una guía metodológica. La cadena de búsqueda utilizada fue "Guide AND software AND development". El resultado de la búsqueda realizada mostró 143 resultados entre libros, artículos científicos, etc. De los cuales, se analizaron y escogieron siete estudios del

2012 en adelante, y que mostraban un proceso, estructura y metodología detallada que especificaba como desarrollar una guía metodológica. La TABLA 3, muestra los estudios escogidos del análisis de la revisión de la literatura.

TABLA 3. Documentos totales en base a parámetros establecidos.

	SCOPUS	SCIENCE DIRECT	SPRINGER	IEEE XPLORE	OTROS
Estructura	0	0	(K K, 2020), (Rossberg, 2019)	0	(Ocaña, 2019), (Halvorsen, 2020), (Ahmed, 2012)
Metodología	0	0	0	0	(Meza, 2014), (FOMIN, s.f.),

Fuente: Propia

2.1.1 Fases para el desarrollo de una guía metodológica.

De los trabajos mostrados en la **¡Error! No se encuentra el origen de la referencia.**, se tomó como base el trabajo propuesto por (Meza, 2014), debido a que en su estudio muestra una estructura y metodología a seguir para desarrollar una guía metodológica. Por este motivo, nuestro estudio tomará como base las etapas propuestas por (Meza, 2014) para crear la “Guía de desarrollo de software basada en SCRUM y Azure”. A continuación, se describe la estructura propuesta de dicha propuesta, en donde el autor se enfoca en describir que las guías metodológicas deben cumplir un alcance y objetivo en específico, tomando en cuenta la audiencia a la que va a ser dirigida la guía metodológica y la expectativa que se obtendrá de los lectores. El desarrollo de una guía metodológica según (Meza, 2014), se conforman de varios componentes los cuales se muestran el siguiente gráfico:

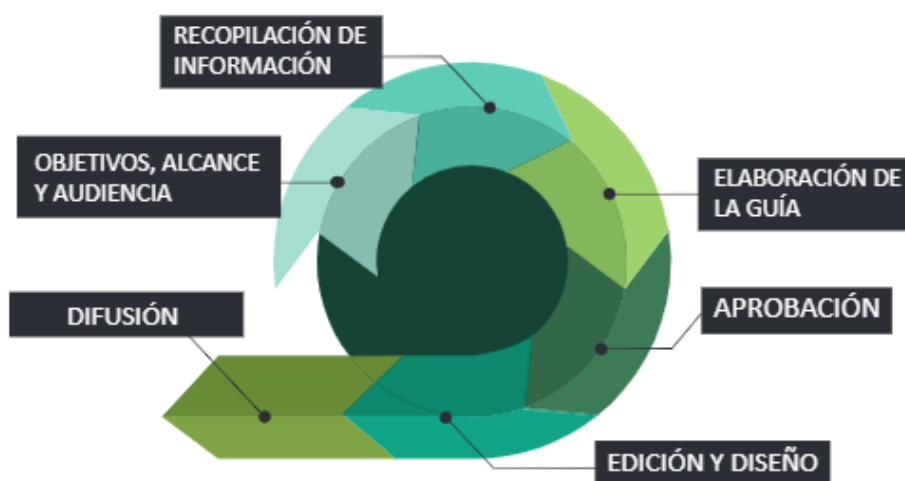


Fig. 14. Fases para el desarrollo de una guía.
Fuente: (Meza, 2014)

Las fases para el desarrollo de una guía metodológica propuesta por (Meza, 2014), consta de seis partes. A continuación, se describen las fases de desarrollo de una guía:

2.1.2 Objetivos, Alcance y Audiencia

Esta etapa hace énfasis al tema del cual tratará la guía, para ello se debe considerar contestar las siguientes preguntas:

➤ ¿Qué se quiere realizar con esta guía?

Es esencial precisar objetivo principal a cumplir, un ejemplo de ello es el documentar la gestión del desarrollo de software utilizando la herramienta Azure DevOps y basándonos en el marco de trabajo SCRUM.

➤ ¿A qué audiencia se va a dirigir?

Es importante saber a qué público se dirige la guía y la respuesta que se espera por parte de ellos. Por ejemplo: a partir de este trabajo las personas que tienen cierto dominio en las Tics, pueda solucionar un problema relacionado al desarrollo de software.

➤ ¿Cuál es el conocimiento que queremos sistematizar?

Es necesario identificar a alto nivel los contenidos que vamos a incluir en la guía metodológica. Dicho sea, el caso; el uso de herramientas tecnológicas basadas en la nube para gestión de software (Meza, 2014).

2.1.3 Recopilación de Información

El objetivo principal de esta fase consiste en la recopilación y análisis de la información obtenida, dicha información será de utilidad para el desarrollo de la guía metodológica. Un punto inicial para realizar esta fase consiste en la revisión de trabajos como:

- Documentación de proyectos, informes de misión, informes de consultores, informes de seguimiento de proyectos, información de sistemas de seguimientos, evaluaciones, registros de proyectos, etc.
- Trabajos realizados anteriormente.
- Documentos relevantes sobre la temática que puedan aportar información para el desarrollo de la guía.

2.2 Elaboración de la guía

Por un lado (Meza, 2014) nos dice que la estructura de una guía puede variar según el objetivo planteado, alcance y público al que está dirigido, pero de manera general se deben incluir los siguientes apartados:

- **Resumen ejecutivo:** Este apartado tiene como objetivo dar al lector una síntesis de manera clara sobre la guía que se realizó.
- **Introducción:** Este apartado presenta una descripción breve y clara acerca del contexto, los antecedentes, las problemáticas del proyecto, también los objetivos y como se realizaron. Finalmente, los resultados que se alcanzaron tras la validación de la guía.
- **Desarrollo o cuerpo de la guía:** Es la sección principal de una guía, ya que se debe describir de manera ordenada y detallada los pasos que se llevaron a cabo durante el proceso planteado.
- **Conclusiones:** En este apartado se destacan los puntos claves de la guía, el mismo modo las recomendaciones para realizar satisfactoriamente la implementación del proceso o metodología tratada en la guía.

Por otro lado, enfocando en la gestión del desarrollo del software (Ahmed, 2012) en su libro *Software Project Management: A Process-Driven Approach* nos da un a conocer modelos estructurados, los cuales nos brindan información acerca de la manera en que podemos realizar guías para la gestión de proyectos, a través del *Work Breakdown Structure (WBS)* que en español se traduce como *Estructura de desglose del trabajo (EDT)*, dicha estructura consta de algunas pautas para la creación y mantenimiento de un proyecto de software, dichas pautas son las siguientes:

- Un proyecto WBS se puede personalizar y diseñar para reflejar las características únicas de los esfuerzos del producto según sea necesario.
- La WBS es una lista estructurada en árbol y orientada a actividades de toda la tarea necesaria para guardar con los requisitos del proyecto.
 - a. La WBS está organizada por producto o servicio, por lo que las actividades de un producto a menudo se agrupan en una parte de la WBS.
 - b. Maneja todo el trabajo requerido, incluida la organización, planificación, seguimiento, control e informes sobre el estado de todos los elementos de trabajo a lo largo del proyecto.

- c. La WBS se utiliza como un conjunto de actividades planificadas en el proyecto. La estimación de los recursos necesarios a menudo se basa en el trabajo definido en la WBS. Si tiene trabajo que no está en su WBS, es posible que no tenga un horario o recursos. Pueden ocurrir fluctuaciones de horarios y costo (Ahmed, 2012).
 - d. La lista de verificación contiene una lista de las actividades más importantes que el proyecto debe realizar. Incluye un diccionario WBS que facilita la selección del elemento correcto.
- Aunque la WBS se basa en un producto o servicio, se debe recordar incluir las actividades y procesos de gestión (informes, incorporación, formación, gestión de requisitos, etc.) que deben realizarse para cumplir con los requisitos del proyecto (Ahmed, 2012).
 - Ordenar los elementos de trabajo específicos que desea realizar en elementos de trabajo más pequeños en etapas de la siguiente manera:
 - a. Se hace referencia a la subdivisión (o descomposición) del elemento de trabajo desde un nivel de perspectiva.
 - b. Los niveles más altos suelen reflejar espacios de trabajo clave o hitos de entrega (Ahmed, 2012).

2.2.1 Aprobación.

Una vez finalizada la elaboración de la guía metodológica, esta ingresa en un proceso de aprobación por parte de la organización beneficiada, en la cual dicha organización puede sugerir cambios en cuanto a la estructura, la metodología, pasos o elementos de la guía, según crea pertinente la organización beneficiada, esta etapa puede llegar a repetirse hasta alcanzar cierta aceptación (Meza, 2014). Dicho esto, es importante recalcar que el contenido de la guía, formato o metodología puede variar según el tipo de organización al que está dirigida la guía.

2.2.2 Edición y Diseño

Obtenida la aprobación de la guía, esta fase consiste en la revisión ante posible errores ortográficos y gramaticales, además de agregar un diseño según el validador de la guía considere.

2.2.3 Difusión

Finalmente, la difusión trata de hacer llegar el contenido realizado a lo largo de estos pasos al público que fue considerado para el desarrollo de la guía, esta etapa puede realizarse por diferentes formas de comunicación, por ejemplo: publicaciones en revistas científicas, talleres prácticos, etc.

2.3 Metodología para el desarrollo de la guía

El objetivo principal del presente tema de investigación consiste en realizar una guía metodológica para el desarrollo de proyectos de software utilizando las herramientas de Microsoft Azure DevOps para la gestión del desarrollo de software. Por lo cual, la metodología establecida para el desarrollo de la guía se basó en aplicar el marco de trabajo SCRUM como base procedimental, la cual operativizará cada una de sus fases: pre-juego, juego, post-juego con las herramientas de la plataforma Microsoft DevOps. La Fig. 15, muestra las fases planteadas para el desarrollo de la guía metodológica

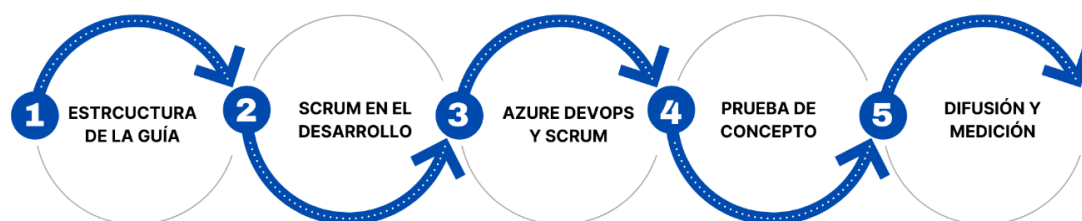


Fig. 15. Metodología para el desarrollo de la guía metodológica.
Fuente: Propia

La guía metodológica se divide en 5 etapas que se enfocan en la integración de herramientas DevOps con el marco de trabajo SCRUM. Las etapas propuestas en el desarrollo de la metodología tienen como fundamento las fases propuestas por (Meza, 2014). A continuación, se describen las fases para el desarrollo de la guía metodológica son las siguientes:

2.3.1 Estructura de la guía

Como resultado de la investigación realizada en la sección 2.1 (Revisión de la literatura) de esta manera, se pudo llegar al análisis de que estructura de la guía metodológica puede basarse en las fases propuestas por el modelo de (Meza, 2014), los componentes de dicho modelo constan de: Resumen, Introducción, Desarrollo y Conclusión. No obstante, haciendo énfasis en el desarrollo de la guía se utilizaron las pautas planteadas por (Ahmed, 2012), las cuales nos dan a conocer pautas que hablan acerca de las EDT (WBS por sus siglas en inglés de Work Breakdown Structure) o

conocidas como Estructuras de Desglose, dichas pautas resultan ser de gran utilidad al momento de realizar las descripciones de las actividades del proyecto.

2.3.2 SCRUM en el desarrollo

La guía metodológica está principalmente centrada en la aplicación de SCRUM como marco de trabajo, para ello las actividades llevadas a cabo para la gestión del proceso de desarrollo de software fueron distribuidas en base a las fases de SCRUM Pre-juego, Juego y Post-juego, dichas etapas fases se describen a continuación:

- Pre-Juego: (Planteamiento) Se establece la visión, las expectativas y el presupuesto para el proyecto. (Rivadeneira, 2013) Nos da a conocer las actividades dentro de esta etapa, entre las que se listan: la visión, el presupuesto, el sprint backlog inicial también como la configuración, además de arquitectura, diseño y prototipos.
- Pre-Juego: (Montaje) En esta etapa se establecen las tareas y requerimientos principales para la ejecución del primer sprint.
- Juego: También conocida como la fase del desarrollo, la cual consta de una serie de iteraciones conocidas como sprints. Las actividades son un encuentro de planeamiento de sprints en cada iteración, la definición del product backlog, los estimados y los encuentros diarios (Rivadeneira, 2013).
- Post-Juego: Conocida también como Liberación, en la cual se realiza el despliegue e implantación del producto, esta etapa va de la mano con la documentación, mantenimiento.

2.3.3 Azure DevOps y SCRUM

Azure DevOps nos brinda servicios para llevar a cabo la gestión del ciclo de vida de las aplicaciones (ALM), es decir, que estos servicios contemplan todos los aspectos del desarrollo de software, desde la planificación, los requisitos, la codificación, las pruebas, la implementación y el mantenimiento (Halvorsen, 2020).

Para llevar a cabo la planificación del proyecto de software, será utilizaron las herramientas de Azure DevOps Boards a través de esta se pudieron observar los trabajos pendientes y los tableros de tareas de Sprint, proporcionándonos una vista filtrada de los elementos de trabajo que un equipo ha asignado a una ruta de iteración específica o Sprint (K K, 2020).

En cuanto al desarrollo de software, la integración continua del proyecto se realizó en base a la herramienta Azure DevOps Repos, para la cual utilizamos GIT, de esta manera se pudo llevar un control de versiones distribuido (DVCS) que utiliza un repositorio local para rastrear y versionar archivos. Los cambios se comparten con otros

desarrolladores empujando y extrayendo cambios a través de un repositorio compartido remoto (Halvorsen, 2020).

Adicionalmente para generar un ambiente adecuado pero enfocado en la integración continua se utilizaron las herramientas de Azure DevOps Pipelines, Azure DevOps Artifacts, Azure DevOps TestPlans lo cual nos permitió automatizar las fases de desarrollo, compilación, pruebas y despliegue del proyecto de software.

2.3.4 Prueba de concepto

Para llevar a cabo la prueba de concepto se realizó una página web, la cual consiste en el desarrollo de un sitio web, dicho sitio web presenta información que describe la experiencia, trabajos, tecnologías de un individuo (portafolio web). El desarrollo de este producto de software se realizó utilizando las herramientas de Azure DevOps mencionadas en la sección anterior.

Principalmente se realizó un enfoque en la aplicación de la guía metodológica para el desarrollo de esta prueba de concepto, posteriormente procedimos a evaluar la guía metodológica en base a la prueba de concepto realizada.

2.3.5 Difusión y Medición

Una vez finalizada la guía metodológica, el siguiente paso consistió en difundir dicha guía, para ello se realizó un taller práctico en donde los estudiantes de la carrera de Software de la Universidad Técnica del Norte utilizaron la guía metodológica en conjunto con las herramientas de Azure DevOps, de esta manera desarrollaron un proyecto de software utilizando el marco de trabajo SCRUM.

Esta fase tiene como principal objetivo la medición de la calidad de uso sobre la guía, para ello se utilizaron las características del estándar ISO/IEC 25000 enfocado en las medidas de completitud funcional. Dicha normativa nos permitió medir la satisfacción, comodidad y utilidad que tienen los lectores al utilizar la guía metodológica planteada, a través de parámetros de satisfacción realizadas a los beneficiarios y profesionales en el área de las Tics.

2.4 Diseño de la guía metodológica

La guía metodológica propuesta está planteada en base a las fases del marco de trabajo SCRUM, las cuales serán gestionadas por la herramienta Azure DevOps. A continuación, la siguiente Fig. 16 muestra el diseño establecido para el desarrollo de esta guía, posteriormente en la TABLA 4 podemos observar la relación que tendrán las fases de SCRUM y como son abordadas con las herramientas de Azure DevOps.

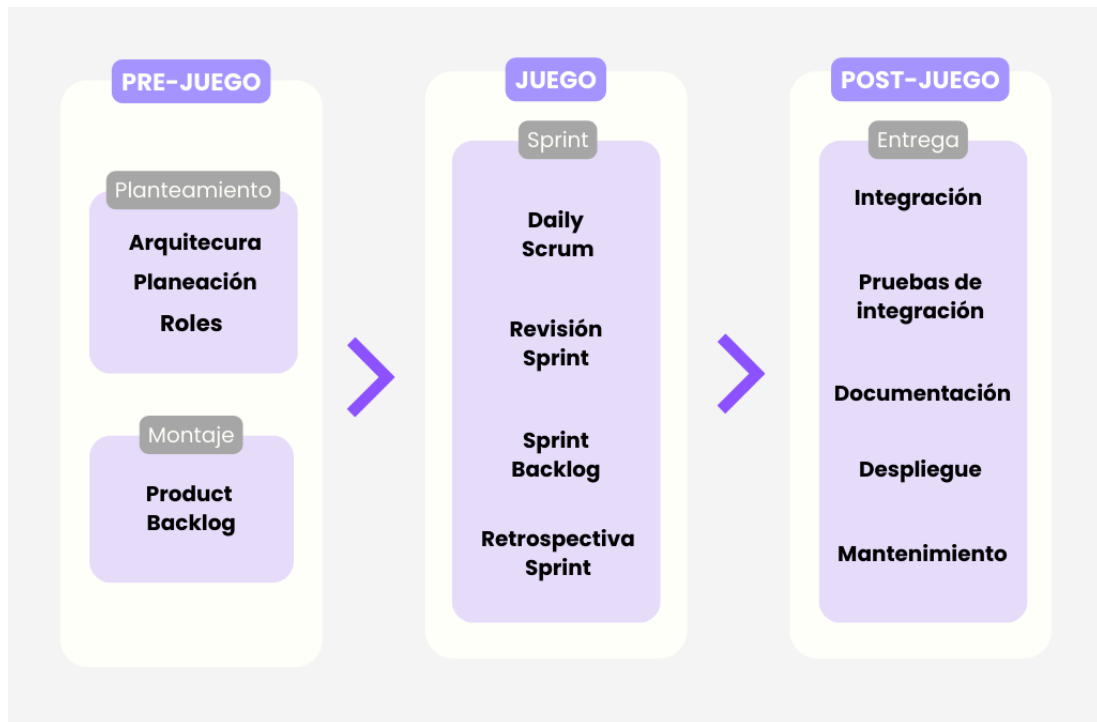


Fig. 16. Diseño de la guía metodológica.
Fuente: Propia

TABLA 4. SCRUM y las herramientas de Azure DevOps.

Fase	Actividad Scrum	Tarea Scrum	Funcionalidad Azure DevOps
Pre-juego	Planeamiento	Planeación Roles	Azure Overview Project Settings / General
	Montaje	Product Backlog	Azure Boards / Sprints Azure Boards / Boards
Juego	Sprint	Daily Scrum	Azure Boards
		Revisión Sprint	Azure Boards / Sprints
		Sprint Backlog	Azure Repos Azure Pipelines
Post-juego	Entrega	Retrospectiva Sprint	Azure Boards Azure Repos
		Integración	Azure Repos
		Pruebas de integración	Azure Test Plans
		Despliegue	Azure Repos
		Mantenimiento	Azure Pipelines

Fuente: Propia

A continuación, se describen las fases de SCRUM para el desarrollo de la guía metodológica:

2.4.1 Pre-juego

Esta fase se compone de dos partes, las cuales consisten en el planteamiento y el montaje del proyecto, estas partes se realizan antes de la asignación de las tareas, las partes que conforman el pre-juego se describen a continuación:

➤ Planteamiento

El planteamiento en la fase del Pre-juego trata principalmente de la visión, las expectativas y desarrollo del proyecto (Rivadeneira, 2013). También es conocida como el Sprint 0 ya que es la fase inicial del proyecto desde una perspectiva tecnológica, metodológica y organizativa (Urteaga Pecharromán, 2015). Todo ello consiste básicamente en la arquitectura que llevara el proyecto, la planeación del proyecto en el enfoque organizativo y la asignación de roles.

Azure DevOps nos permite realizar esta fase en el segmento general, dicho segmento brinda opciones al usuario para la configuración del equipo de trabajo, integrantes, permisos y roles en el equipo.

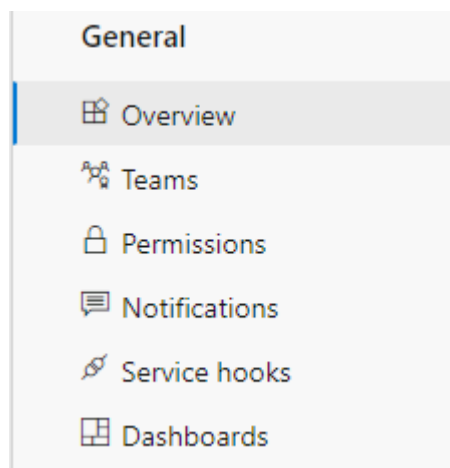


Fig. 17. Planeación en Azure DevOps.
Fuente: Propia

➤ Montaje

Esta fase tiene como énfasis la construcción del Product Backlog, el cual consta de las historias de usuario o elementos para ser desarrollados con una fecha de inicio y fin (Urteaga Pecharromán, 2015). Además de obtener como producto de esta etapa el Product Backlog, se toman en cuenta las actividades para llevar a cabo en el primer sprint.

Es posible crear el Product Backlog en Azure DevOps en la sección *Boards*, apartado Backlogs de un proyecto creado, el contenido en esta sección consiste en la creación de ítems con un título, descripción y orden. Estas tareas posteriormente son asignadas a un Sprint, los cuales se describen en la siguiente fase.

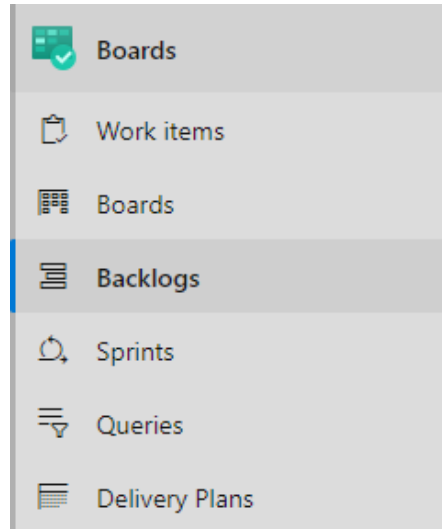


Fig. 18. Backlogs en Azure DevOps.
Fuente: Propia.

2.4.2 Juego

Una vez finalizada la asignación de roles, arquitectura, especialmente el Sprint Backlog del proyecto con una fecha de inicio y culminación, se procede a realizar la fase del juego, en la que se toma énfasis en el desarrollo del software, el cual se divide por Sprints de cierto periodo de tiempo. El equipo de trabajo considera los Sprints que sean necesarios, según la planificación previa.

La planificación establecida debe basarse en la planificación del proyecto, cada iteración cuenta con su respectivo detalle de tareas para lograr el cumplimiento de los requerimientos asignados (Valeria et al., n.d.).

Las etapas que se deben cumplir en esta fase son las siguientes: Daily SCRUM, Revisión Sprint, Sprint Backlog, Retrospectiva Sprint. La etapa del juego tiene como principal requerimiento el Sprint Backlog, la misma que se obtiene como producto de la fase del Pre-juego. Estas etapas se contemplan en actividades distribuidas en Sprint Backlog.

Los Azure Boards, nos permiten crear los Sprints que consideremos apropiados, de este modo a través de ítems de trabajo, asignamos tareas a los Sprints.

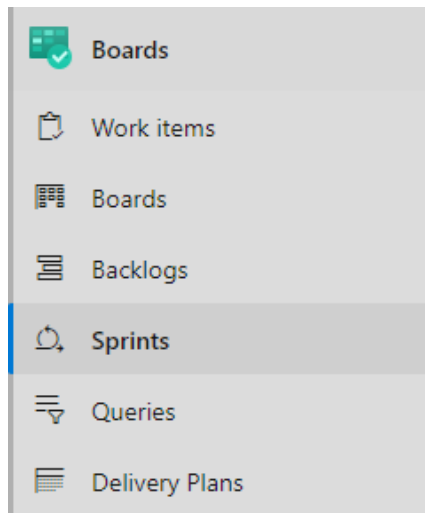


Fig. 19. Sprints en Azure DevOps.
Fuente: Propia.

Las actividades asignadas en el Sprint son divididas en tres estados to Do, In Progress y Done, cada miembro de equipo puede cambiar el estado de su tarea, según la actividad que realice. De este modo dar paso a las actividades de: Daily SCRUM, Revisión Sprint, Retrospectiva Sprint.

Por otro lado, para realizar actividades enfocadas al desarrollo de software, el uso de repositorios para el manejo de versiones del código fuente es de suma importancia, para solventar este requerimiento Azure Repos nos brinda la posibilidad de trabajar con GIT o Team Foundation Version Control (Ocaña, 2019).

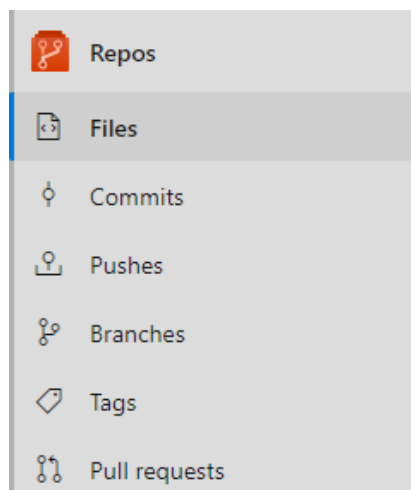


Fig. 20. Azure DevOps Repos.
Fuente: Propia.

2.4.3 Post-juego

Esta etapa inicia una vez que el desarrollo del software ha finalizado, por lo cual se llevan a cabo actividades referentes a la integración y mantenimiento del producto de

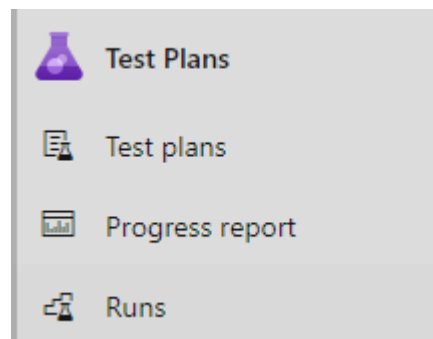
software. A través de las herramientas de Azure DevOps podemos realizar las siguientes actividades:

➤ Integración

Esta etapa se enfoca en la integración del código fuente realizada en la fase del Juego, a través de Azure Repos podemos realizar esta fase, para ello es de importancia haber realizado previamente una estandarización para la gestión del código fuente como: Branches, commits, tags, pushes, según crea necesario el líder del equipo designado a la codificación.

➤ Pruebas de Integración

Esta fase tiene como finalidad la calidad del producto de software, a través de Azure podemos analizar los casos de prueba, los cuales son los elementos de prueba básicos para validar la funcionalidad de una aplicación o los requisitos no funcionales asociados con una aplicación.



*Fig. 21. Azure DevOps Test Plans.
Fuente: Propia.*

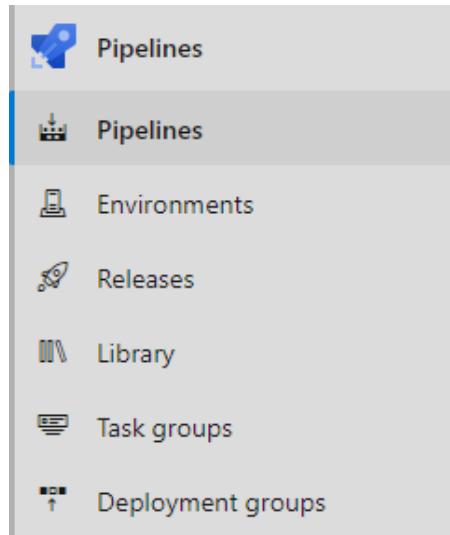
En la gestión de pruebas, los casos de prueba se clasifican en función de su uso (K K, 2020). Los siguientes son algunos de ellos:

- Caso de prueba de unidad: Esto será utilizado por los desarrolladores para verificar la integridad de su implementación. Hay muchas herramientas y tecnologías disponibles asociadas con cada una de las tecnologías de implementación para automatizar los casos de prueba unitaria (K K, 2020).
- Caso de prueba funcional: Este es un caso de prueba utilizado para validar la implementación funcional del sistema.
- Caso de prueba de seguridad: Esto verifica las restricciones de seguridad con el sistema.
- Caso de prueba de rendimiento: verifica el rendimiento del sistema con diferentes parámetros, como el volumen de datos, la carga de usuarios, las pruebas de estrés, etc.

- Prueba de aceptación del usuario: esta es la prueba de aceptación por parte de las partes interesadas del negocio.

➤ **Despliegue**

Una vez que la compilación está lista, debe implementarse en un entorno para verificar su funcionalidad. Si hay una canalización automatizada configurada desde el entorno de desarrollo hasta la implementación de producción, lo llamamos implementación continua (K K, 2020).



*Fig. 22. Azure DevOps Pipelines.
Fuente: Propia*

Las Pipelines definen los procesos de compilación y lanzamiento y las integraciones asociadas al desarrollo del proyecto de software.

➤ **Mantenimiento**

Orientada al mantenimiento del producto de software, consiste en conservar y mejorar el software para enfrentar ciertos errores descubiertos y nuevos requisitos del cliente.

CAPÍTULO III

3 DESARROLLO DE LA GUÍA METODOLÓGICA

La elaboración de la presente guía de desarrollo surgió de la necesidad solventar una un problema de gran relevancia en el campo de la gestión de proyectos de software. Se sabe que con el surgimiento de las nuevas tecnologías basadas en la nube el trabajo y la gestión de los proyectos de software tienden a mejorar la tasa de éxito en proyecto de software, en este documento se hace enfoque al uso de la herramienta Microsoft Azure DevOps. Esta herramienta nos permite gestionar del software, en base a un conjunto de herramientas integradas. Sin embargo, este servicio no cuenta con una guía metodológica que permita llevar a cabo la gestión de proyectos de software de principio a fin, basándose en un marco de trabajo como SCRUM y sus fases para el desarrollo.

En este capítulo, se llevó a cabo el desarrollo de una guía metodológica utilizando SCRUM como marco de trabajo y las herramientas para la gestión que engloba Azure DevOps, la cual concretamente tiene los siguientes contenidos:

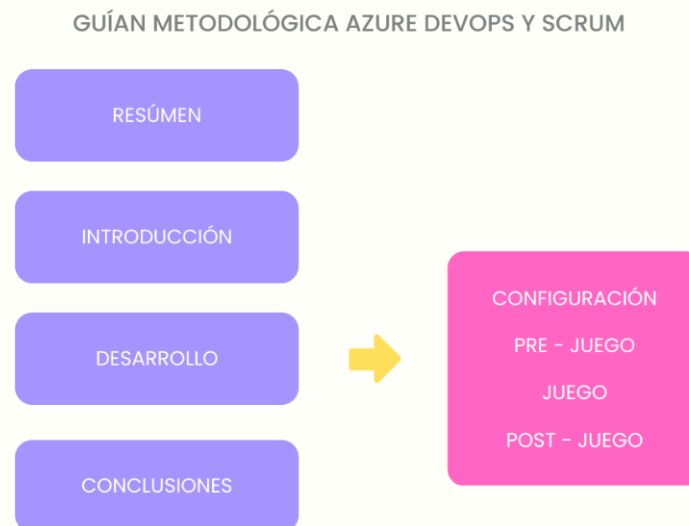


Fig. 23. Estructura Guía Azure DevOps y SCRUM.
Fuente: Propia

3.1 Introducción

Este documento tiene como objetivo principal orientar a los lectores que se enfrentan a la tarea de gestionar el desarrollo de un producto de software con herramientas basadas en la nube.

La guía metodológica a continuación tiene su enfoque en la gestión de proyectos de software basada en el marco de trabajo SCRUM y utilizando la herramienta de Microsoft

Azure DevOps, es el resultado de la investigación de conceptos, métodos y herramientas utilizadas en la gestión y el desarrollo del software.

A lo largo del documento se explican los pasos a seguir, partiendo desde la configuración e inicio de un proyecto, el desarrollo del producto de software y culminando en la entrega de este.

3.2 Desarrollo de la Guía

La guía desarrollada a continuación se ha dividido en cuatro secciones basadas en el marco de trabajo SCRUM. La primera sección, denominada “Configuración”, incluye los pasos para crear un proyecto en Microsoft Azure DevOps, invitación a los miembros del equipo, creación de roles y permisos.

La segunda sección: “Pre-Juego”, es el conjunto de pasos necesarios para realizar la gestión de equipos de trabajo y la distribución de las tareas a través de Sprints, las cuales se realizan por los miembros del equipo.

La tercera sección, llamada “Juego”, está diseñada para realizar las tareas asignadas en los Sprints. Además, describir el uso de las herramientas para gestión de código fuente con las herramientas de Azure DevOps.

Por último, la cuarta sección “Post-Juego”, describe los pasos para realizar pruebas del producto desarrollado para posteriormente realizar el despliegue del aplicativo.

3.2.1 Configuración del proyecto

En esta sección se crean y configuran los componentes principales para la gestión de un proyecto enfocado al desarrollo de software, tales como: Creación de proyecto, configuración de proyecto. La configuración del proyecto en base al marco de trabajo SCRUM, se considera como el Sprint Cero.

La gestión del proyecto con las herramientas de Microsoft Azure DevOps se realiza de la siguiente manera:

3.2.1.1 Creación del proyecto (Microsoft Azure DevOps):

- a) Ingrese al sitio web: <https://azure.microsoft.com/es-es/services/devops/>
- b) Seleccione la opción “Comenzar gratis” para activar una instancia gratuita de Microsoft Azure DevOps, como se muestra en la Fig. 24 a continuación.

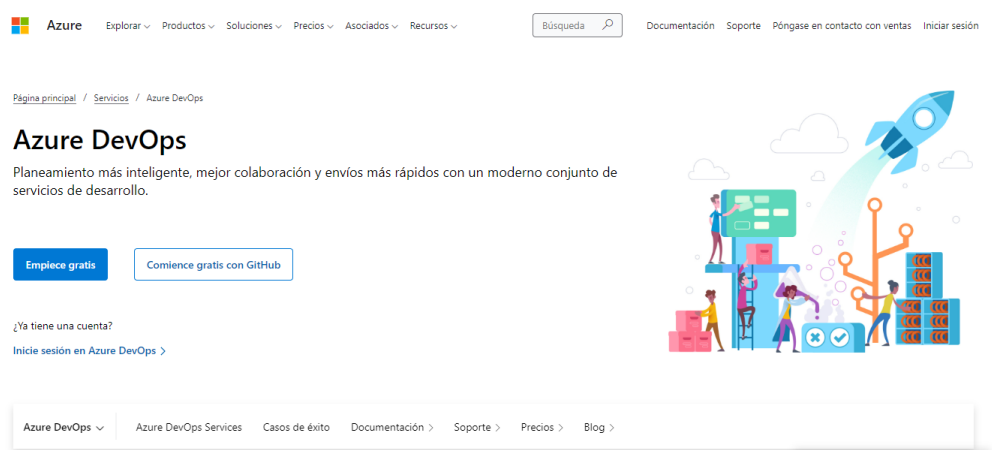


Fig. 24. Página principal Azure DevOps.
Fuente: Propia

- c) Proporcione el nombre de una organización para continuar con la configuración del proyecto.
- d) Cree un nuevo proyecto con los detalles mínimos incluyendo su privacidad (público o privado).

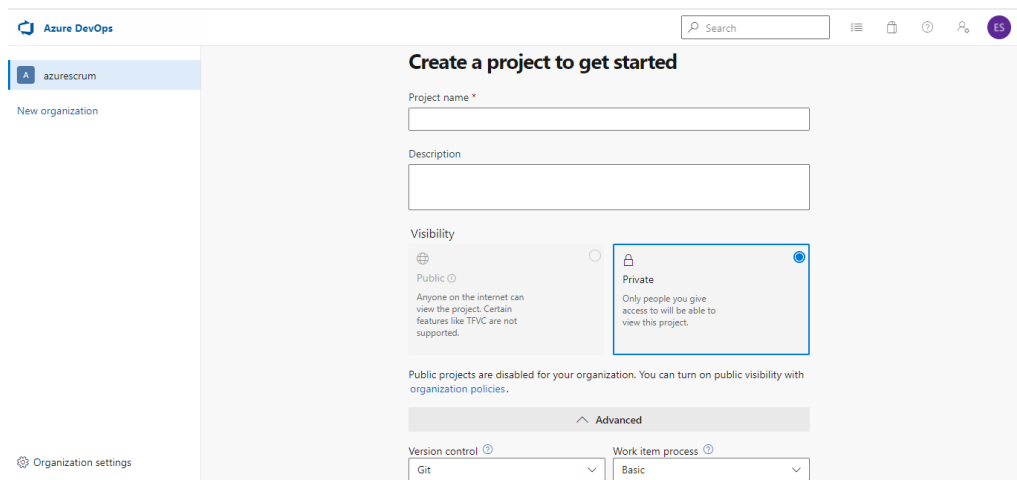


Fig. 25. Crear proyecto en Azure DevOps.
Fuente: Propia.

- e) Seleccione el lenguaje para el control de versiones (Git o Team Foundation Version Control) y por último la metodología de trabajo para ser utilizada.

Antes de iniciar con la ejecución de las etapas de SCRUM, es necesario especificar ciertos puntos de partida para el mejor entendimiento de la Guía Metodológica propuesta. A continuación, en la Fig. 26 se muestra la pantalla general que tiene un proyecto creado en Azure DevOps.

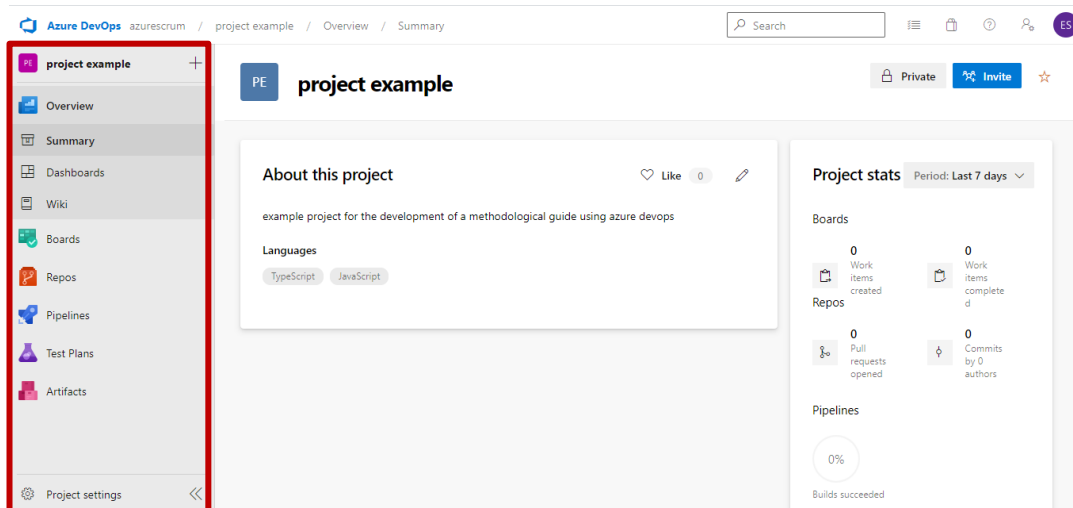


Fig. 26. Vista principal de un proyecto en Azure DevOps.
Fuente: Propia

En la parte izquierda de la fig. se visualizan las herramientas de Azure DevOps, mirar la Fig. 27.

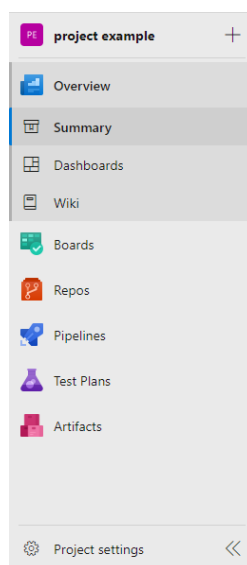


Fig. 27. Suite de herramientas de Azure DevOps.
Fuente: Propia

Partiendo de esta pantalla la guía metodológica propuesta especificará las distintas configuraciones y acciones para la gestión de un proyecto de software.

3.2.2 Pre-juego

La fase de Pre-juego está contemplada en dos etapas Planteamiento y Montaje, las cuales están distribuidas en base a su actividad y resultado, dichas etapas se describen a continuación:

3.2.2.1 Planteamiento

Es necesario recalcar que el planteamiento de esta fase se trata principalmente de la visión, expectativas y desarrollo del proyecto (Rivadeneira, 2013), es decir, la gestión del proyecto de desarrollo de software, sin tomar en cuenta particularidades tecnológicas como la arquitectura o lenguajes de programación.

Antes de realizar el planteamiento, debemos agregar miembros a nuestro equipo de trabajo, para ellos se deben seguir los siguientes pasos:

- Diríjase a las configuraciones (*Project Settings*) del proyecto creado en la sección 3.2.1.1, ingrese a la sección *Teams*.
- Por defecto Azure Creará un equipo de trabajo con el nombre del proyecto, elija el equipo respectivo.
- Al ingresar al equipo se mostrarán detalles de este, como se muestra en la Fig. 28 pulse el botón *Add*.

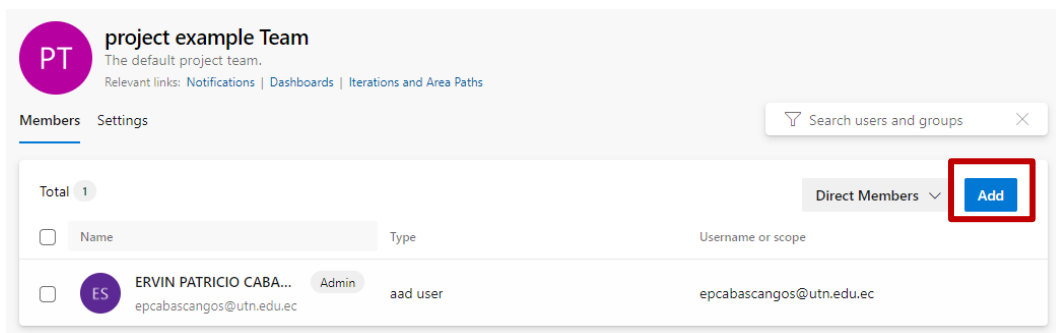


Fig. 28. Invitar miembros de equipo.
Fuente: Propia

- A continuación, ingrese una dirección de correo electrónico válida. Puede agregar varios miembros de equipo, esta invitación llegará al destinatario de la dirección de correo electrónico.

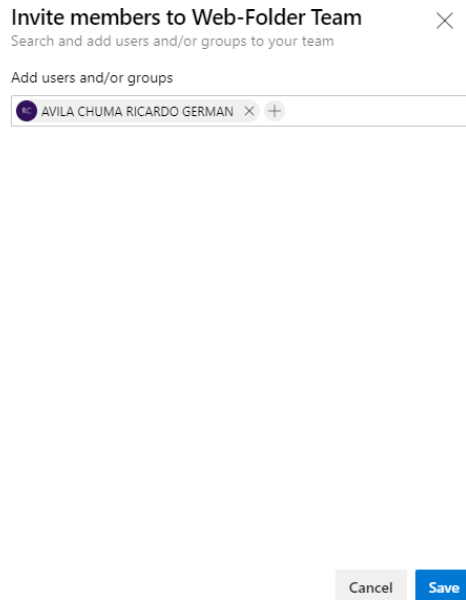


Fig. 29. Agregar miembro al proyecto.
Fuente: Propia

Planeación

- Para realizar esta sección, en la pantalla principal de Azure DevOps empiece creando un equipo de trabajo. Para aquello diríjase al apartado *Project Settings/Teams*.
- En este apartado se mostrarán los equipos creados para el proyecto, por defecto Azure DevOps crea un equipo con el nombre del proyecto.
- A continuación, pulse el botón *New Team* en la parte superior, esto desplegará el siguiente modal:

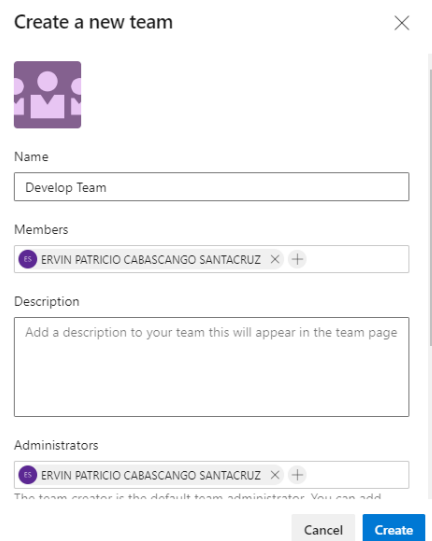


Fig. 30. Crear equipo Azure DevOps.
Fuente: Propia

- d) A continuación, se debe completar la información requerida como: Nombre del equipo, Miembros del equipo, Descripción, Administradores del equipo. Esta información es requerida. Los miembros y administradores de equipo pueden agregarse en cantidad a través de una dirección de correo electrónico.
- e) Una vez creado el equipo podrá ver la información de su equipo como se muestra en la Fig. 31

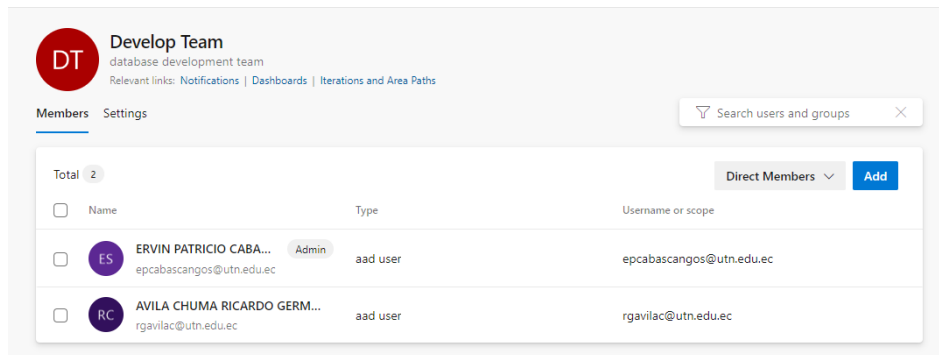


Fig. 31. Usuarios Equipo Azure DevOps.
Fuente: Propia.

Permisos por usuario

La configuración de servicios nos permite dar a los usuarios cierto nivel de acceso a funciones específicas de acuerdo con el rol que cumplen dentro del equipo de trabajo, estos permisos se dividen en secciones tales como: General, Boards, Analytics, Test Plans. Para realizar una configuración de roles por usuario, siga los siguientes pasos:

- a) Diríjase al apartado *Settings/Permissions/Users*. A continuación, la Fig. 32 muestra los usuarios agregados al equipo de trabajo.

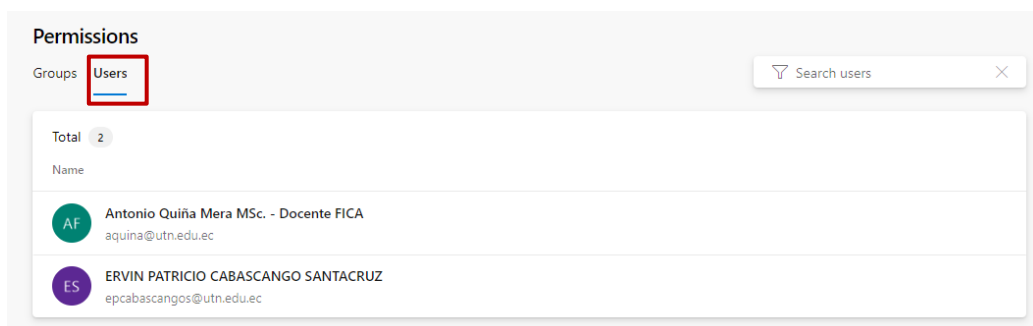


Fig. 32. Permisos Azure DevOps.
Fuente: Propia

- b) Posteriormente, elija un usuario al cual desea realizar configuraciones en sus permisos.
- c) A continuación, en la Fig. 33 muestra la configuración por defecto que Azure DevOps crea para cada usuario.

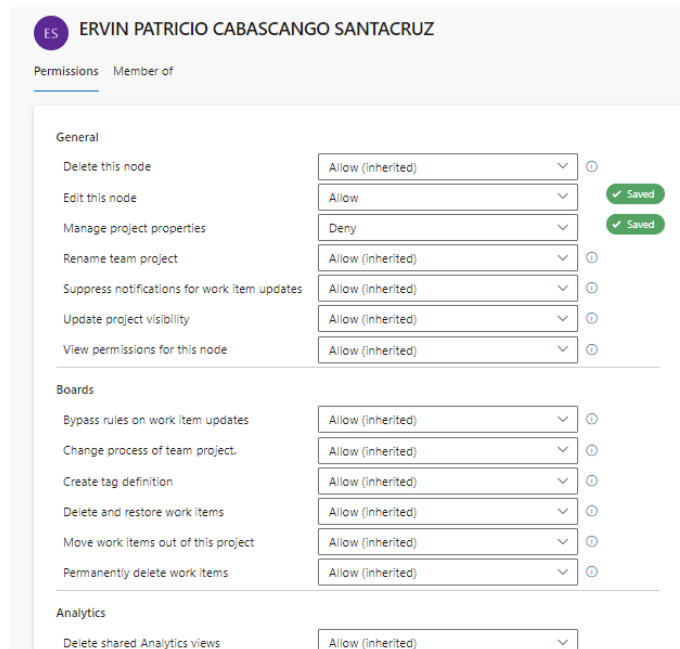


Fig. 33. Clasificación de Permisos Azure DevOps.
Fuente: Propia

- d) Las opciones mencionadas anteriormente permiten al usuario un nivel de acceso a las secciones de Azure DevOps, cada una de ellas puede asignarse a los siguientes estados: Not set, Allow, Deny.

Permisos por Equipo

Los permisos por equipo se realizan de manera similar a los permisos por usuario, A continuación, se detallan los pasos para realizar la configuración:

- a) Diríjase al apartado Settings/Permissions/Groups, dentro de esta sección encontrará contenido como se muestra en la Fig. 34.

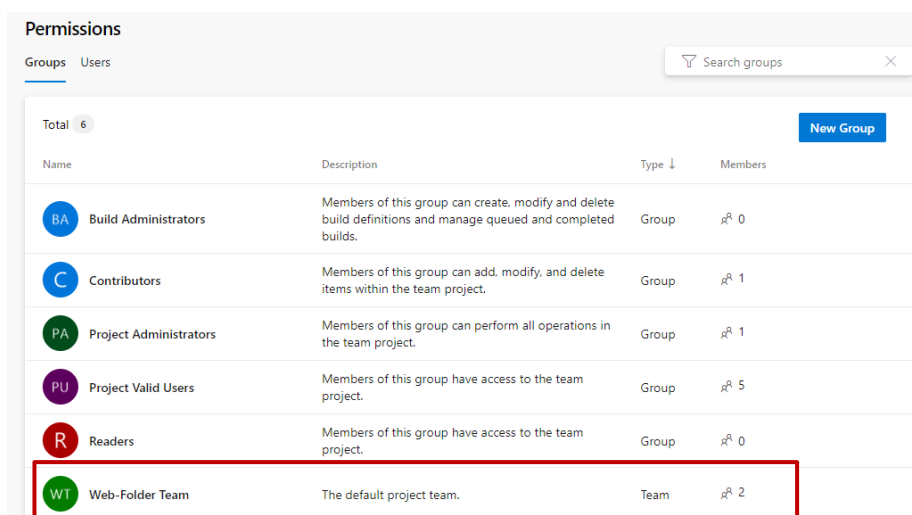


Fig. 34. Permisos Equipo Azure DevOps
Fuente: Propia

- b) Elija el equipo que tenga el nombre del proyecto creado, en este caso *Web-Folder Team*. Las opciones que se muestra son grupos que Azure DevOps crea por defecto.
- c) Una vez dentro, podrá visualizar diferentes paneles que le brindan más información sobre el equipo como se muestra en la Fig. 35.

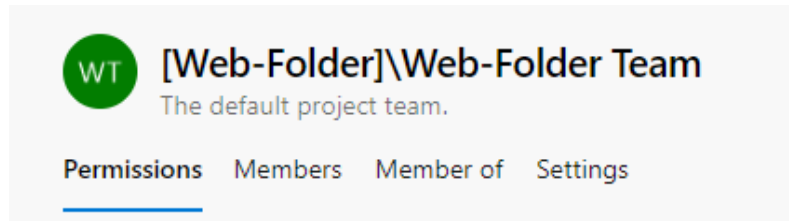


Fig. 35. Opciones de permiso de equipo.
Fuente: Propia

- d) Finalmente, configure los permisos que requiera el equipo similar a la sección **Permisos por usuario**. Es importante mencionar que no todos los equipos tendrán el mismo nivel de acceso a funciones y configuraciones disponibles en Azure DevOps.

3.2.2.2 Montaje

Esta etapa tiene como objetivo crear el Product Backlog, para definir los requisitos de software de acuerdo con las especificaciones de SCRUM.

Definición del Product Backlog

Azure DevOps permite crear una lista de tareas denominadas Product Backlog, las cuales, son los requerimientos iniciales del cliente.

- **Product Backlog**
 - a) Agregue una pila de Backlogs dirigiéndose al apartado *Boards/Backlogs*.
 - b) Seleccione el tablero elegido para agregar ítems como se muestra en la Fig. 36.

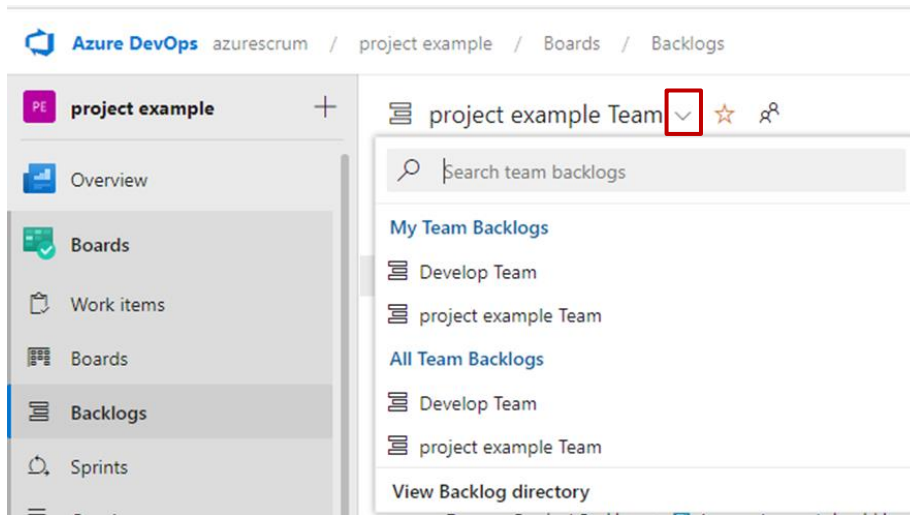


Fig. 36. Tableros de Backlog Azure DevOps.
Fuente: Propia

- c) Después, agregue un *New Work Item* y complete la información como se muestra en la Fig. 37.

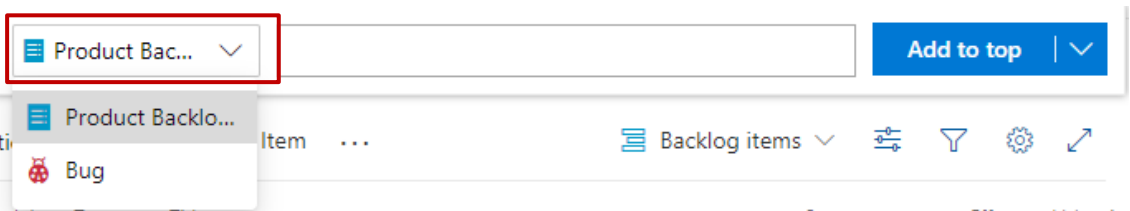


Fig. 37. Tableros de Backlog Azure DevOps.
Fuente: Propia

- d) Finalmente se visualizan las actividades agregadas, en base a las necesidades del proyecto.

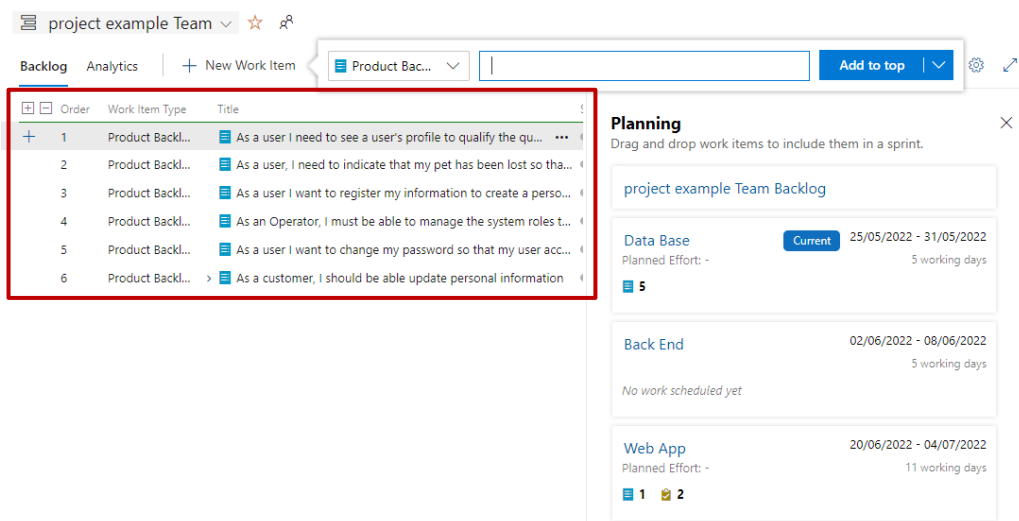


Fig. 38. Backlog Azure DevOps.
Fuente: Propia.

- e) Visualice el contenido del backlog a través de los tableros de Azure DevOps en el apartado de *Boards/boards*.

- f) Configure cada ítem mostrado en el tablero, puede agregar configuraciones tales como: tags, descripción y el estado de la actividad entre *New*, *Approved*, *Committed*.

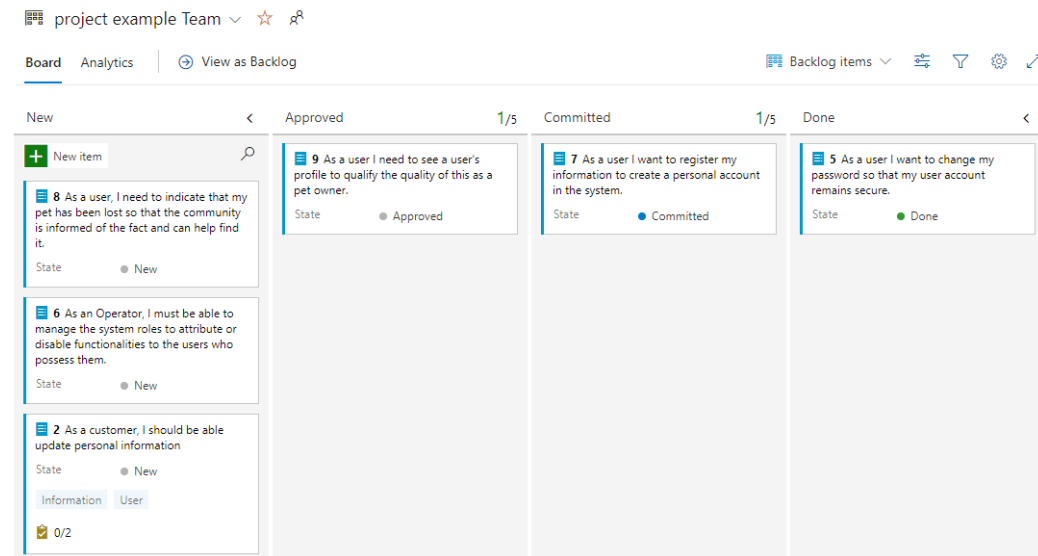


Fig. 39. Tableros con ítems de trabajo Azure DevOps.
Fuente: Propia

- g) Posteriormente configure el ítem del tablero, agregue miembros del equipo, comentarios, descripción del equipo, entre otros como se muestra en la siguiente Fig. 40.

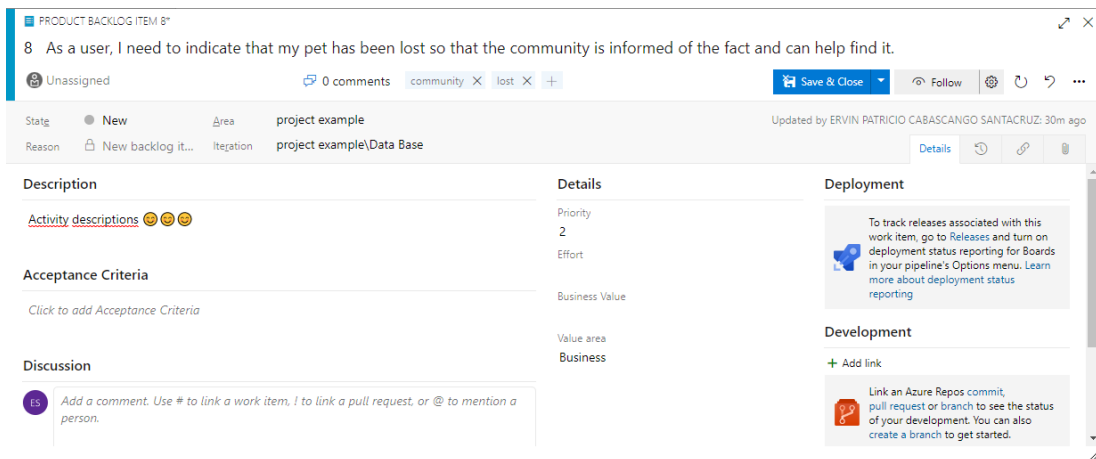


Fig. 40. Editar ítem de trabajo Tableros.
Fuente: Propia

Configuración Sprints

Dentro de la configuración Sprints, se ordenan las actividades creadas en la sección anterior (Product Backlog), para ello Azure DevOps nos permite realizar esta gestión de la siguiente manera:

- **Tableros – Sprints**

- a) Para configurar los Sprints por defecto dirijase al apartado *General/Boards/Project configuration*, se visualizará el contenido de la Fig. 41.

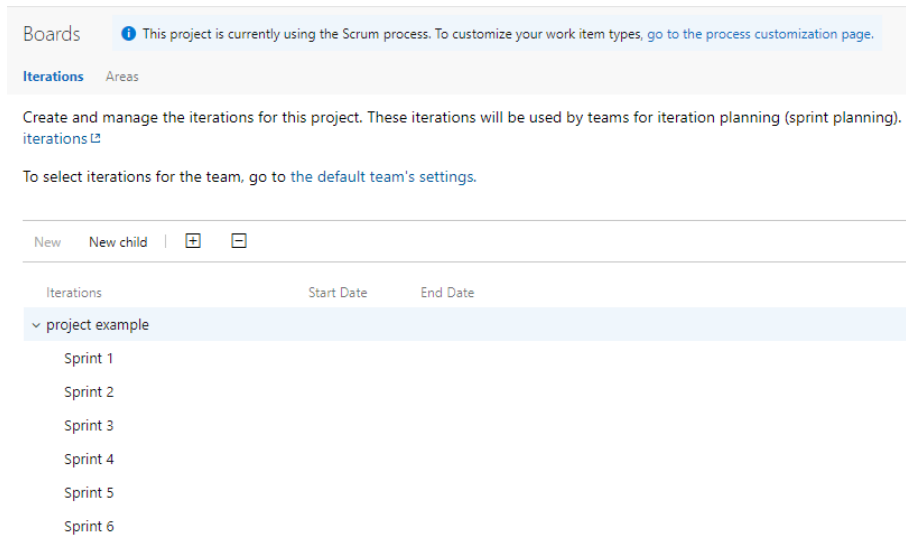


Fig. 41. Sprints del proyecto Azure DevOps.
Fuente: Propia

- b) Configure la información como fecha inicio, fecha fin, nombre de los Sprint.
c) Si desea, puede crear subniveles para distribuir de mejor manera el Sprint. Para ello seleccione el Sprint y elija la opción *New Child*.
d) Configure la información de la nueva iteración. Se mostrará información como en la siguiente Fig. 42 .

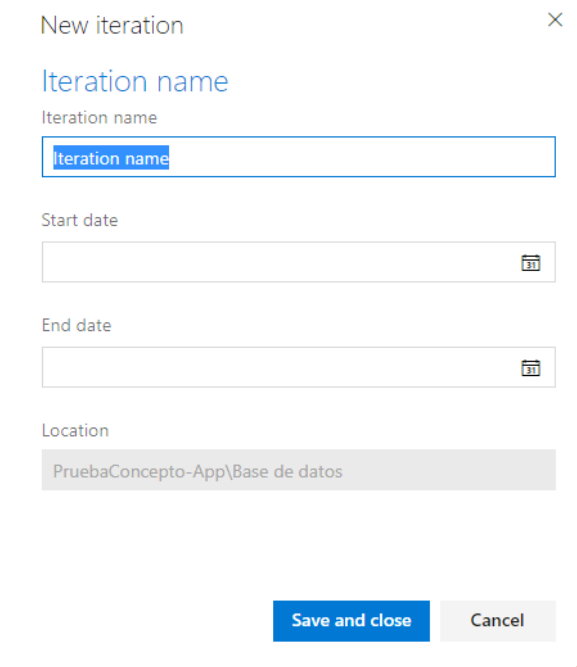


Fig. 42. Agregar iteración.
Fuente: Propia

- e) Finalmente, distribuya los Sprint de acuerdo con las necesidades del proyecto.

Iterations	Start Date	End Date
▼ project example	...	
▼ Data Base	25/05/2022	31/05/2022
Design	25/05/2022	27/05/2022
▼ Back End	02/06/2022	08/06/2022
Database connection	02/06/2022	04/06/2022
Development	06/06/2022	17/06/2022
▼ Web App	20/06/2022	04/07/2022
Desing	20/06/2022	24/06/2022
Development	26/06/2022	04/07/2022
Mobile App	05/07/2022	26/07/2022

Fig. 43. Distribución Sprint Azure DevOps.
Fuente: Propia

Asignación de actividades al Sprint.

El producto Backlog es la parte fundamental de esta sección, una vez se hayan creado correctamente esta lista de requerimiento, estos se mostrarán en la sección de *Boards/Backlogs*. Inicialmente se mostrarán todas las tareas creadas como se muestra en la Fig. 44.

The screenshot shows the Azure DevOps interface. On the left, a backlog board is visible with a table of work items:

Order	Work Item Type	Title
1	Bug	Review task 3
2	Bug	Bug Login
3	Product Backl...	As a user I need to see a user's profile to qualify the quality ...
4	Product Backl...	As a user, I need to indicate that my pet has been lost s...
5	Product Backl...	As a user I want to register my information to create a perso...
6	Product Backl...	As an Operator, I must be able to manage the system r...
7	Product Backl...	As a customer, I should be able update personal information
8	Bug	Bug seccion 3

On the right, a 'Planning' panel is open, showing a 'Current' sprint from 27/07/2022 to 08/08/2022 with 9 working days. Below the sprint, it says 'No work scheduled yet'. There is a '+ New Sprint' button at the bottom of the panel.

Fig. 44. Pila Backlog Boards.
Fuente: Propia

Una forma rápida de agregar un backlog a un sprint es la siguiente:

- En la parte izquierda de la pantalla se muestra la lista backlog y en la parte derecha los Sprints configurados en la sección *Configuración Sprints*, selecciones un ítem backlog.
- Arrastre un ítem a un Sprint disponible como se muestra en la Fig. 45. Si un Sprint no se visualiza en la sección de *Plannig*, probablemente la fecha de ese Sprint ya pasó.

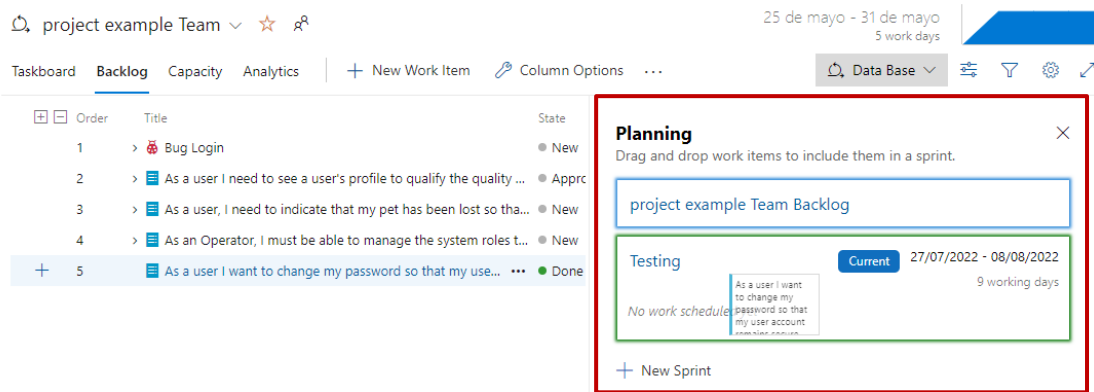


Fig. 45. Asignar Backlog a un Sprint.
Fuente: Propia

Azure nos permite crear un ítem backlog directamente desde un Sprint, para ello realice los siguientes pasos:

- a) Diríjase al apartado de *Boards/Sprints*. Por defecto se situará en el Sprint que se encuentre en ejecución de acuerdo con la configuración de fecha realizada en la sección anterior *Configuración Sprints*.
- b) Si desea crear un backlog en otro Sprint, pulse el botón en la parte superior derecha que tendrá el Sprint actual como se mira en la Fig. 46.

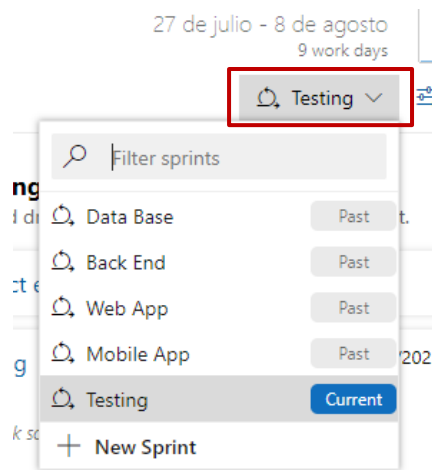


Fig. 46. Seleccionar Sprint.
Fuente: Propia

- c) Una vez seleccionado el Sprint, agregue un nuevo ítem backlog en la parte superior pulse el botón *New Work Item*, ver Fig. 47.

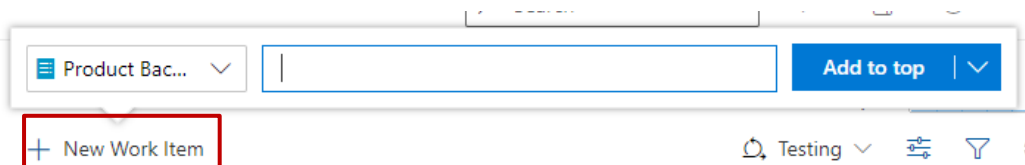


Fig. 47. Agregar ítem Product Backlog.
Fuente: Propia

d) Finalmente, el nuevo ítem creado se mostrará en la lista de backlog.

3.2.3 Juego

Esta fase de SCRUM está enfocada a la ejecución de los Sprints planificados en la fase de Pre-juego, de la cual se obtuvo el Product Backlog que fue distribuido acorde a las necesidades del proyecto. En esta sección se abarca el desarrollo del producto de software, lo que conlleva la creación de repositorios para gestión de código fuente, la creación de entornos de desarrollo y la descripción de cada una de estas, en base a las herramientas de Azure DevOps.

3.2.3.1 Sprints (Boards/Sprints)

La sección Sprints proporciona una vista de los elementos de trabajo planificados para una iteración o sprint, la duración del sprint, el progreso de la finalización del trabajo, el trabajo en equipo en el sprint y otra información relacionada con el sprint, como se muestra en la Fig. 48.

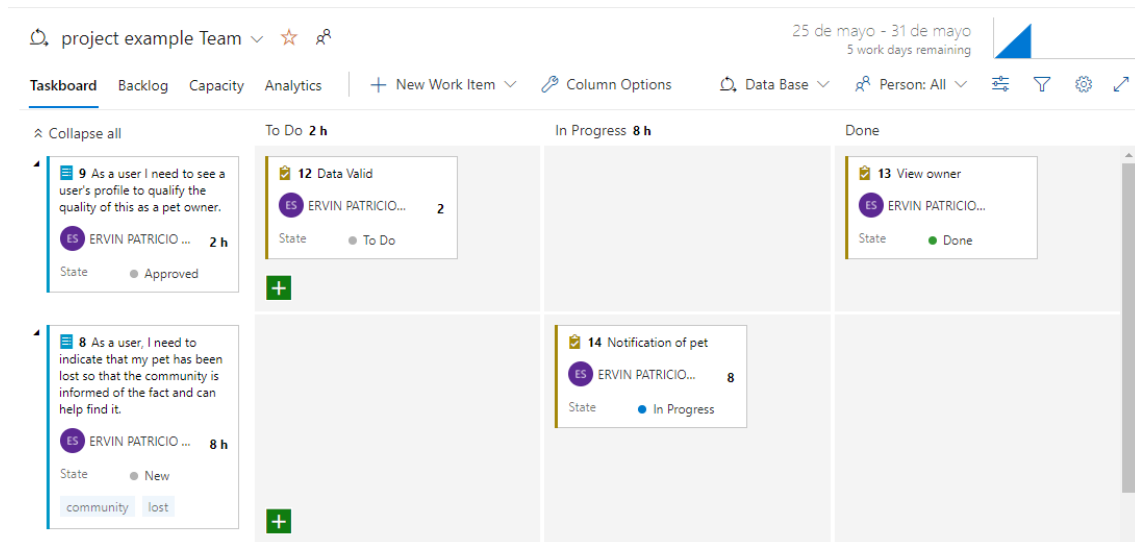


Fig. 48. Sprints Azure DevOps.
Fuente: Propia

Crear Tareas

- Seleccione el Sprint en donde quiere se requiere crear la nueva tarea. Cada Sprint contiene tareas diferentes de acuerdo con la distribución realizada anteriormente. A continuación, se muestra la representación en la Fig. 49.

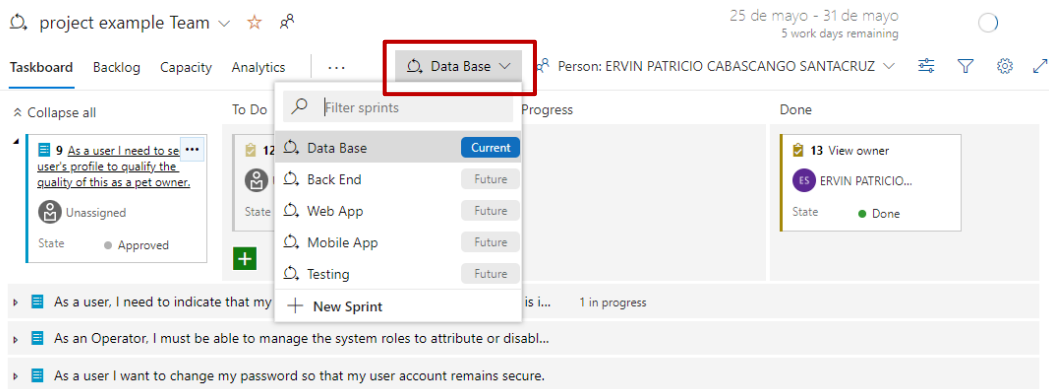


Fig. 49. Sprint por equipo.
Fuente: Propia

- b) Posiciónese en la historia de usuario ha de realizar
- c) Cree una tarea haciendo clic en la opción *New Work Item*.
- d) Seleccione el tipo de tarea.

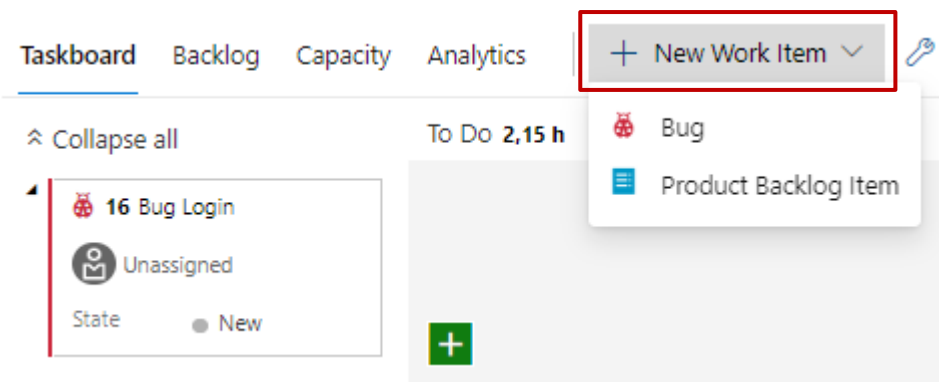


Fig. 50. Nuevo ítem en Sprint.
Fuente: Propia

- e) Adicional configure las características de la tarea como se muestra a continuación en la Fig. 51.

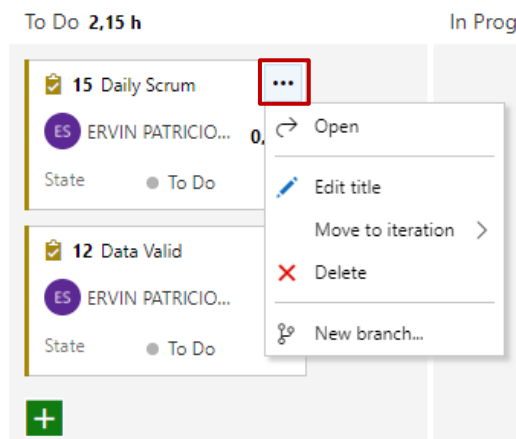


Fig. 51. Ítem de trabajo Sprint.
Fuente: Propia

- f) Puede asignar usuarios a la tarea y agregar información adicional

3.2.3.2 Revisión Sprint

- Asegúrese de estar en el Sprint correspondiente
- Arrastre la tarea a uno de los siguientes estados a la tarea: To Do, In Progress, Done.
- En caso de presentar un inconveniente puede crear una tarea nueva, para ello dirigirse al apartado **Crear tarea**.

3.2.3.3 Sprint Backlog

Creación repositorio de código

- Diríjase al apartado de Repos
- Seleccione la primera opción *Files*. A continuación, se mostrará contenido como en la en la Fig. 52. El contenido mostrado en la ilustración muestra las configuraciones necesarias para subir un repositorio nuevo a uno existente.

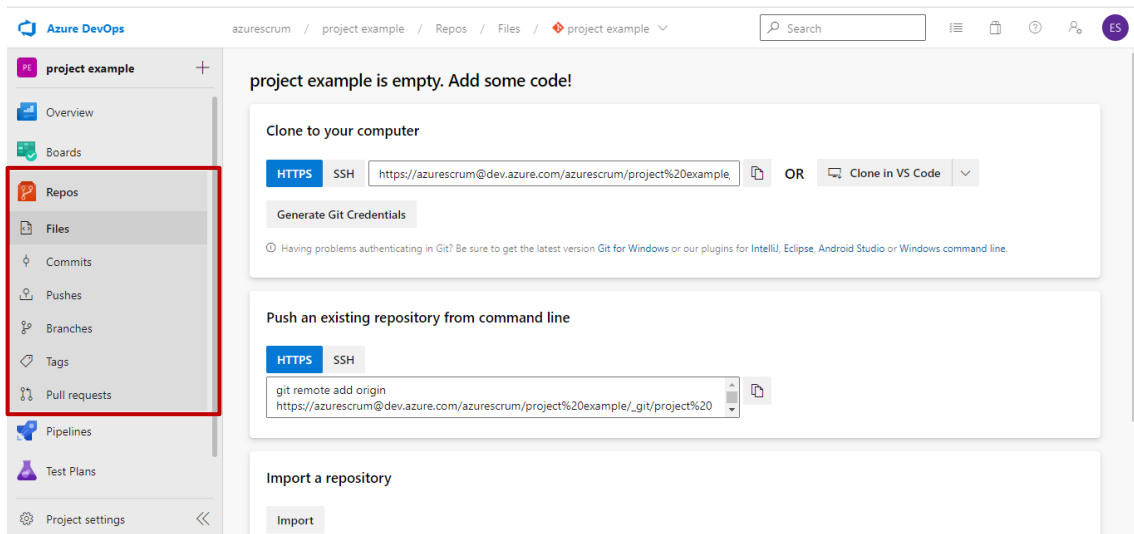


Fig. 52. Repos Azure DevOps.
Fuente: Propia

c) La siguiente Fig. 53, muestra un proyecto subido en el repositorio de Azure DevOps

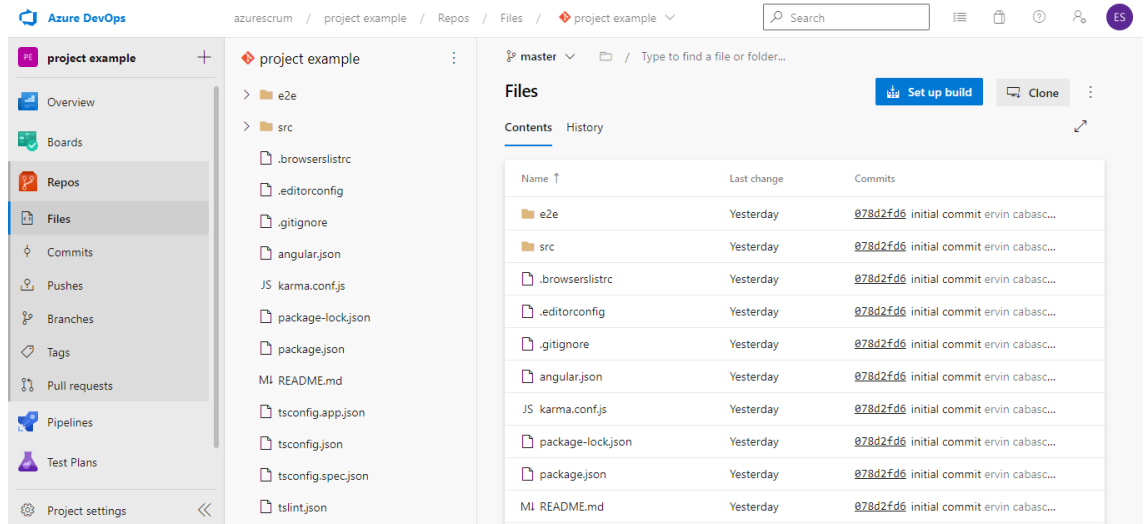


Fig. 53. Azure Repos Files.
Fuentes: Propia

Branches

- Diríjase a la opción *Branches* dentro del menú de Repos en Azure DevOps
- Pulse el botón *New Branch*. A continuación, se desplegará un formulario como se muestra en la Fig. 54.

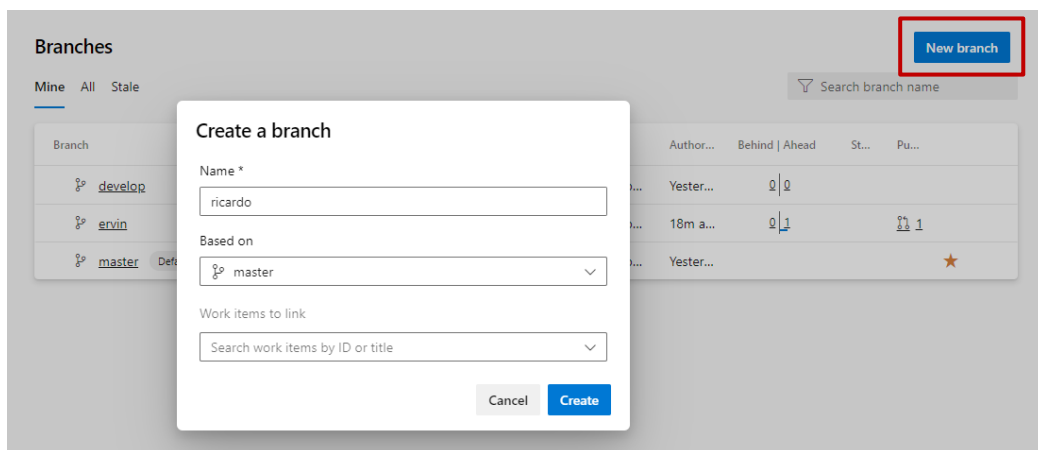


Fig. 54. Crear Rama Azure DevOps.
Fuente: Propia

- Complete la información del formulario, se debe tomar en cuenta la base de la cual se creará dicha rama, por defecto Azure DevOps crea una rama llamada master.
- Finalmente, en el apartado *Branches* se visualizarán las ramas creadas como se muestra en la Fig. 55 a continuación.

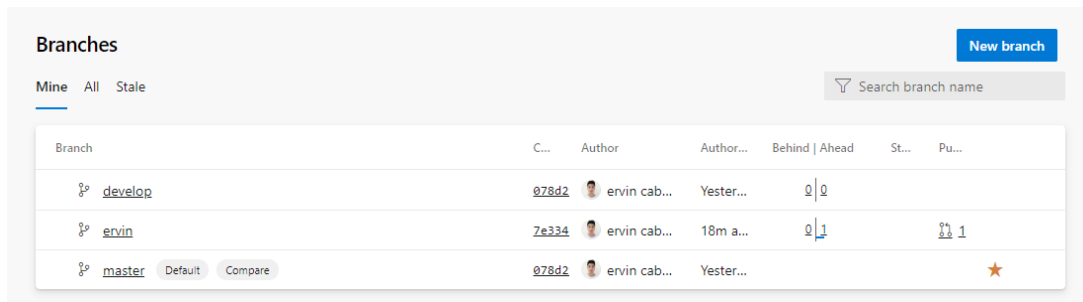


Fig. 55. Ramas en Azure Repos.
Fuente: Propia

Commits

- Diríjase a la opción *Commits* dentro del menú de Repos en Azure DevOps
- Esta sección nos permite visualizar los cambios que se han realizado por parte del equipo de desarrollo como se muestra en la Fig. 56.

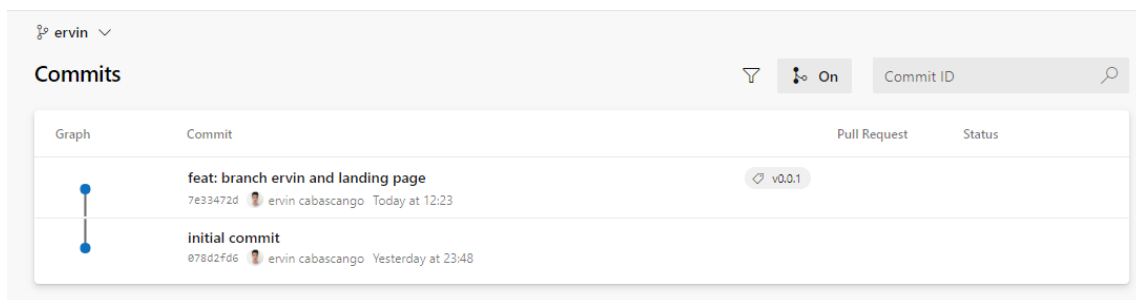


Fig. 56. Commits Azure Repos.
Fuente: Propia

- Se pueden mirar los commits realizados por las diferentes ramas, para ello diríjase a la parte superior izquierda y seleccione una rama creada.

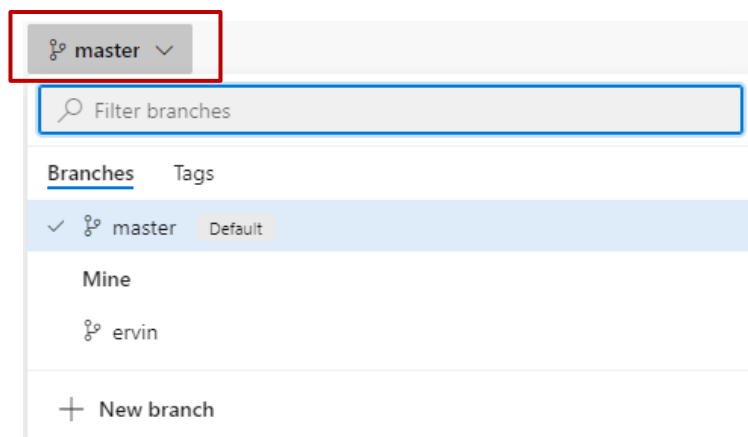


Fig. 57. Commits por rama Azure Repos.
Fuente: Propia

- El contenido de la lista cambiará según la rama que se haya seleccionado.

Pushes

- Diríjase a la opción *Pushes* dentro del menú de Repos en Azure DevOps

- b) La sección Pushes nos muestra la lista de confirmaciones enviadas al repositorio remoto como se muestra en la Fig. 58.

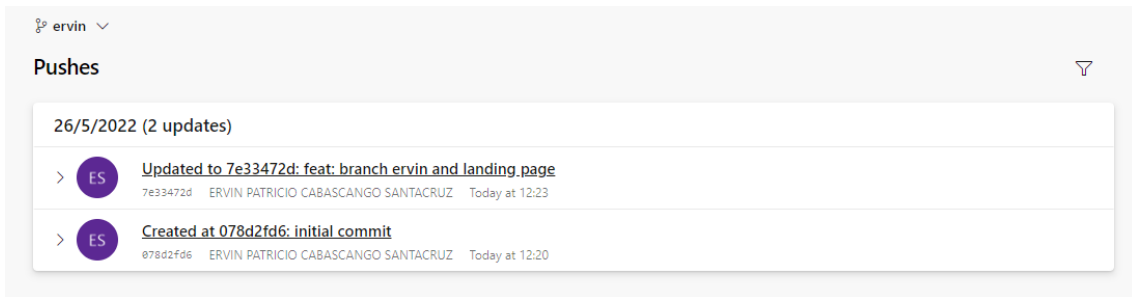


Fig. 58. Pushes Azure DevOps.
Fuente: Propia

- c) Puede dar seguimiento a los pushes realizados por las diferentes ramas, para ello diríjase a la parte superior izquierda y seleccione una rama creada.
- d) Seleccione un ítem para ver a detalle los cambios realizados.
- e) Los cambios realizados se mostrarán en base a los cambios anteriores, como se muestra a continuación:

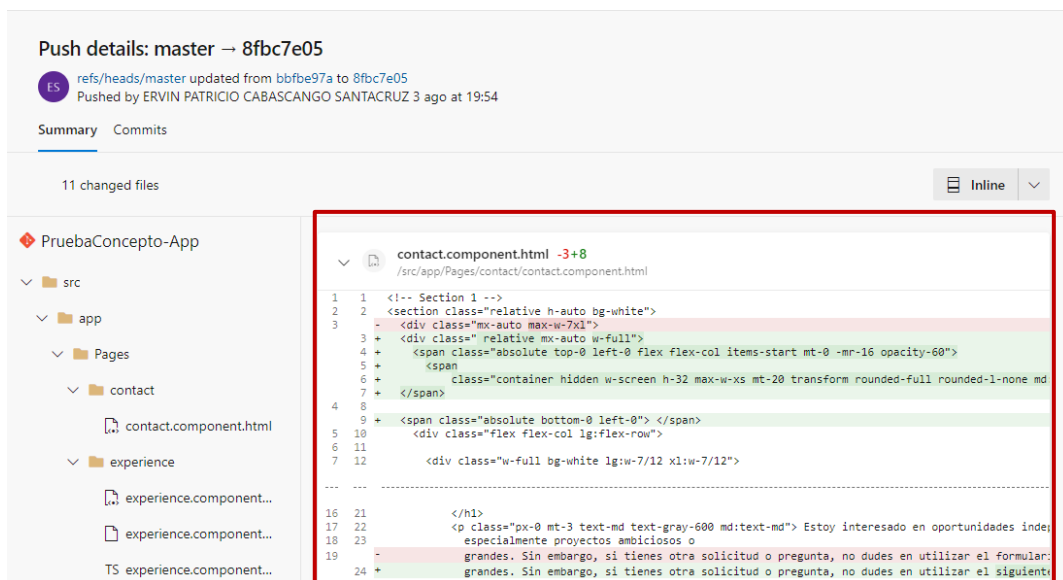


Fig. 59. Detalles de Push Azure DevOps.
Fuente: Propia

Tags

- a) Vaya a la opción *Tags* dentro del menú de Repos en Azure DevOps, os tags nos ayudan a establecer una versión del producto desarrollado.
- b) En la parte superior derecha, pulsamos el botón *New tag*, esto desplegará un formulario para la creación de un nuevo tag, ver imagen Fig. 60.

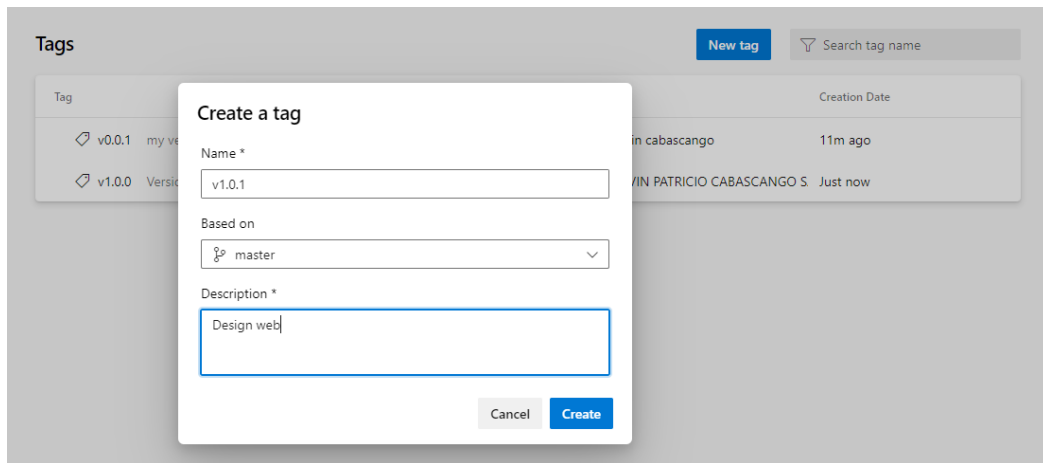


Fig. 60. Crear un Tag Azure Repos.
Fuente: Propia

- c) Complete la información del formulario, debe tener en cuenta que el tag será creado en base a la rama que seleccione.
- d) A continuación, los tags realizados del proyecto se mostrarán con en la Fig. 61.

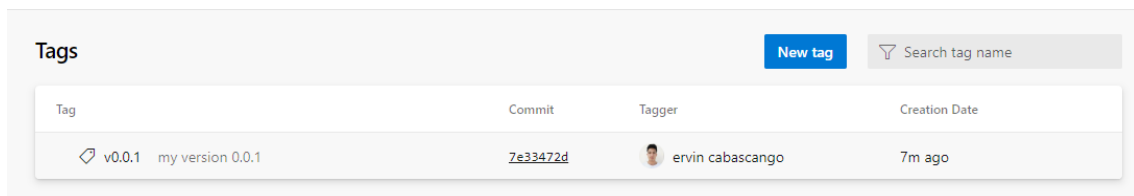


Fig. 61. Tags Azure Repos.
Fuente: Propia.

3.2.3.4 Retrospectiva Sprint

- a) Cree una tarea nueva, para ello dirigirse al apartado **Crear tarea**.
- b) Puede evidenciar la sección Sprints/Analytics, a través de esta pestaña podemos ver el tiempo de horas que debe cumplir el equipo de trabajo y cuál es el rendimiento ideal que podría tener el equipo como se muestra en la Fig. 62.

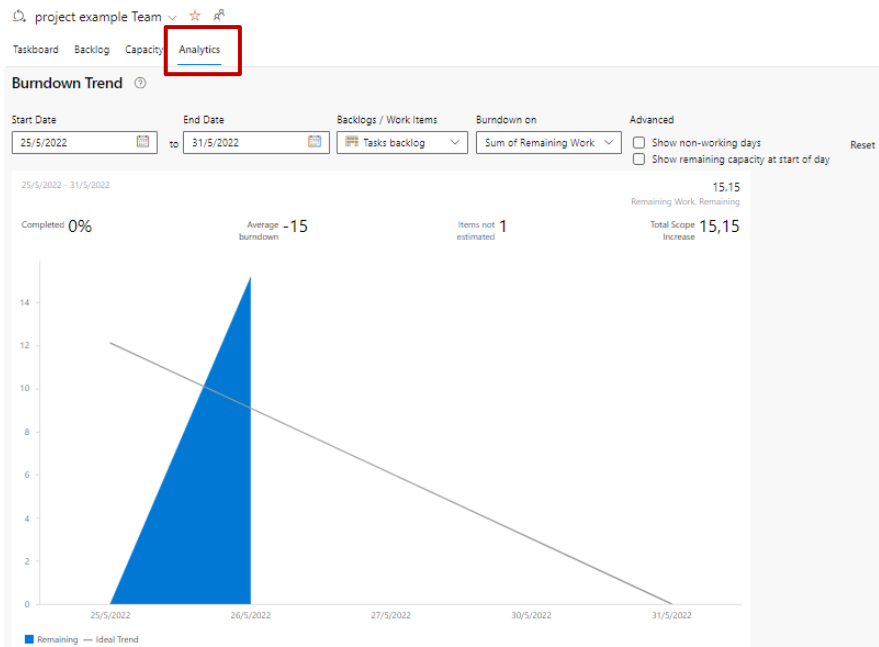


Fig. 62. Retrospectiva Sprint Azure Boards.
Fuente: Propia

3.2.4 Post-juego

3.2.4.1 Integración

- Realiza la integración de cambios de las diferentes ramas en el apartado de *Pull requests*.
- A continuación, seleccione un ítem creado de petición de cambios
- Complete la información de la petición con comentarios, nombre, además de agregar miembros de equipo según sea necesario.
- Apruebe o desapruebe los cambios después de su respectiva revisión, para ello haga clic en el botón *Approve*, mirar Fig. 63.

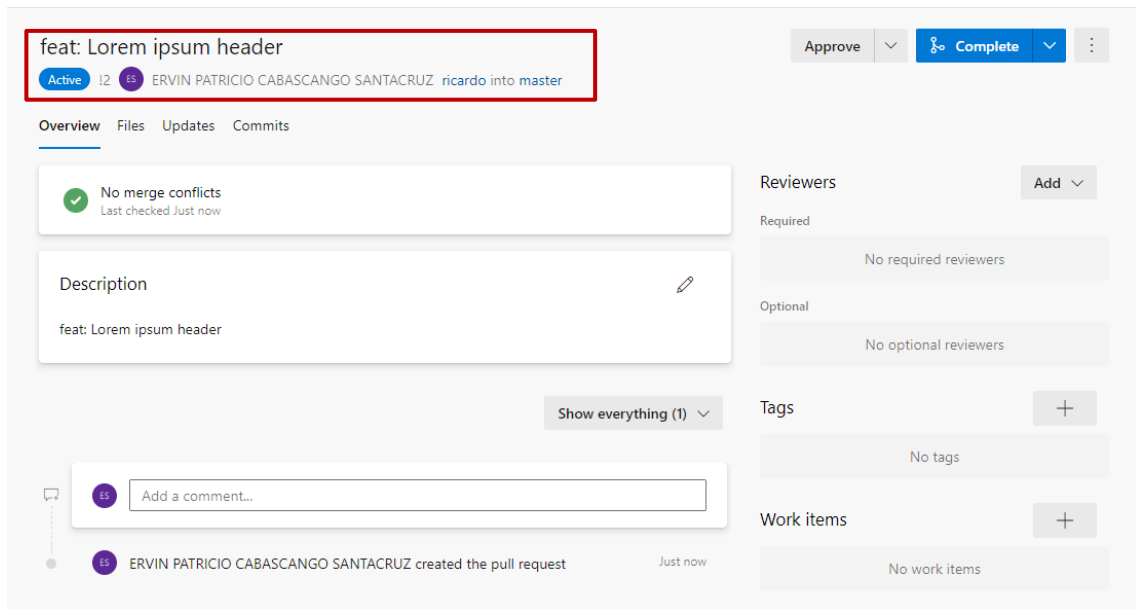


Fig. 63. Fusionar ramas Azure Repos.
Fuente: Propia

e) Las *Pull requests* se pueden crear desde Azure DevOps o por terminal a través de Git

3.2.4.2 Pruebas de Integración

- Cree una prueba directamente desde una actividad creada en los tableros de Azure Boards.
- Haga clic el icono de menú en el arte superior derecha de la actividad, ver imagen Fig. 64.
- Elija la opción *Add Task* de la lista de opciones.

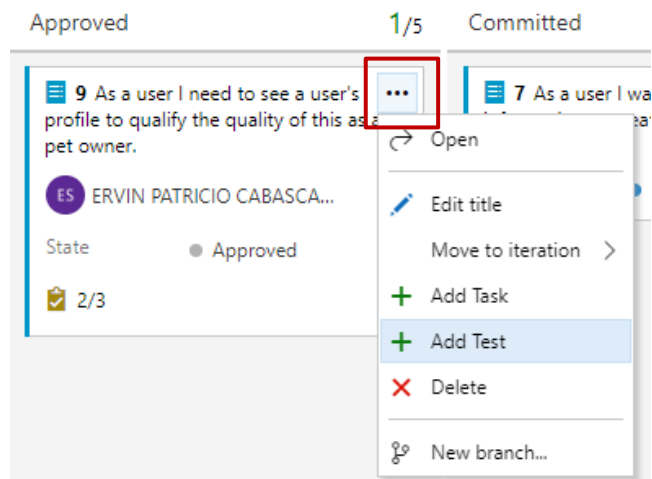


Fig. 64. Agregar Prueba Azure Boards.
Fuente: Propia.

- d) A continuación, se debe definir paso y lo que se espera obtener de dicha prueba, además del estado en el que se encuentra la prueba como se mira a continuación en la Fig. 65.

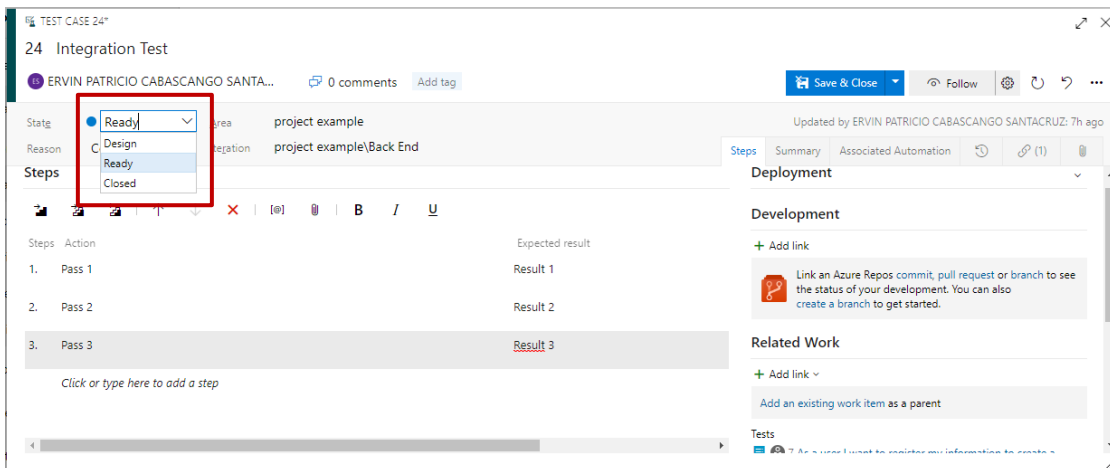


Fig. 65. Configurar una prueba.
Fuente: Propia

- e) Una vez finalizada la configuración de la prueba, procedemos a realizar dicha prueba.
- f) Puede ejecutar la prueba directamente desde las tareas del Backlog, de esta manera podemos verificar el estado de estas con las opciones que se muestran en la siguiente Fig. 66.

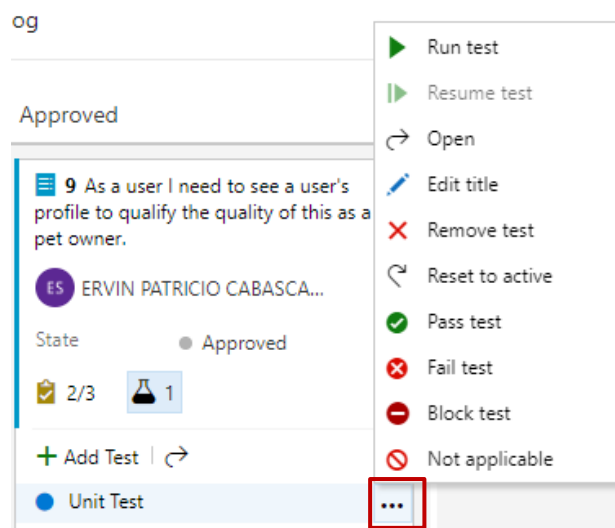


Fig. 66. Pruebas desde ítem de trabajo.
Fuente: Propia

- g) Al ejecutar una prueba, esta desplegará una ventana emergente, en la cual se muestra la configuración realizada anteriormente de nuestra prueba como los parámetros y actividades que debe cumplir dicha prueba, mirar la Fig. 67.
- h) Verifique que las pruebas hayan cumplido con satisfacción, para ello marque con un aprobado o desaprobado paso a paso de la prueba.

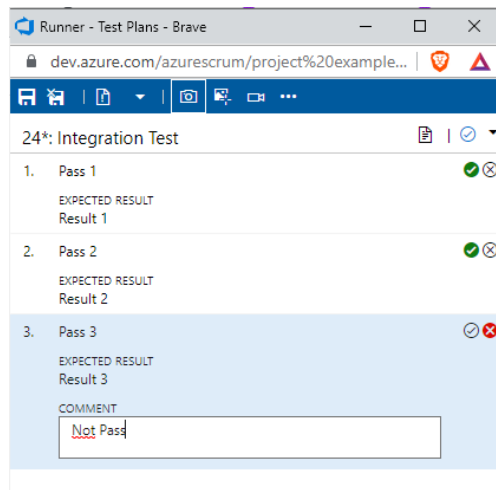


Fig. 67. Caso de Prueba.
Fuente: Propia

- i) Para observar las pruebas asignadas al equipo pueden observarse en el apartado de *Tests Plans* de la herramienta Azure DevOps.

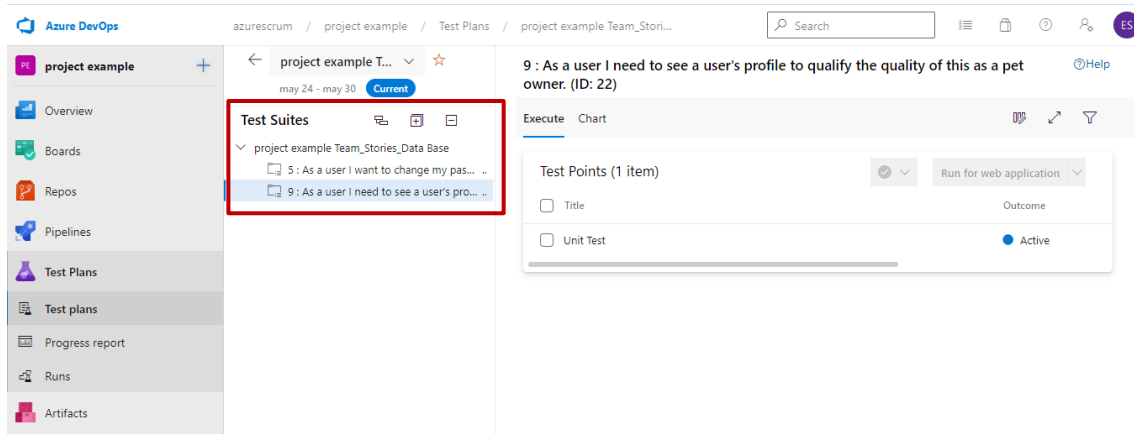


Fig. 68. Tests Plans Azure DevOps.
Fuente: Propia

Las pruebas realizadas se pueden evidenciar en el apartado de *Test Plans/Runs*, las pruebas se distribuyen de acuerdo a su estado. Además, de obtener una visión general de las pruebas a través de gráficos estadísticos como se muestra en la Fig. 70.

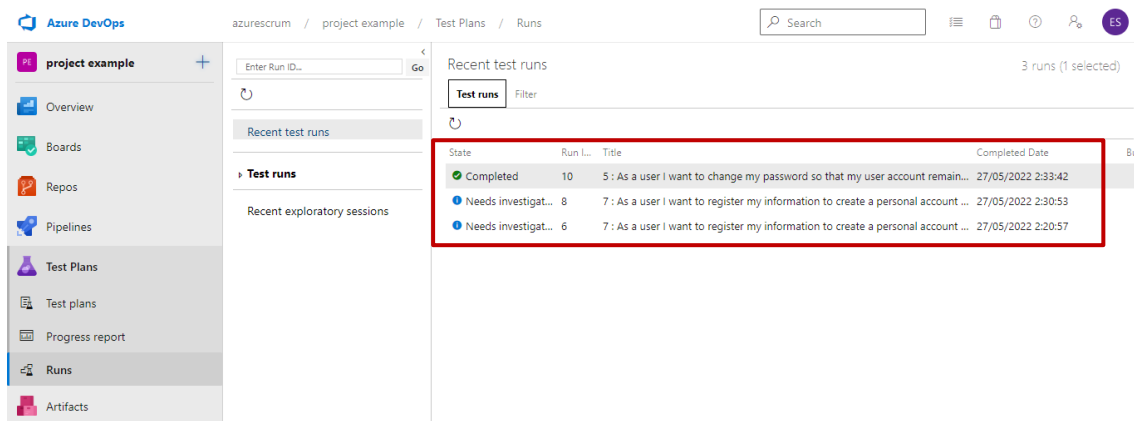


Fig. 69. Test Runs Azure DevOps.

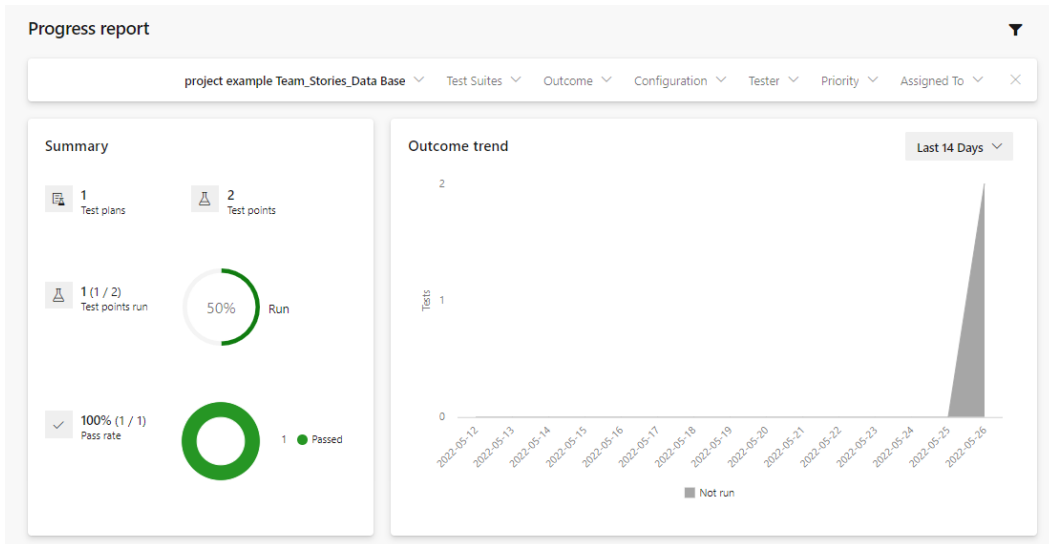


Fig. 70. Reporte de progreso.
Fuente: Propia

3.2.4.3 Despliegue

Los Pipelines definen los procesos de compilación y lanzamiento e integraciones asociadas.

- Cree un pipeline en la sección *Pipelines*, después de clic en el botón *New pipeline*.
- A continuación, haremos la configuración de cuatro pasos: conexión, selección, configuración y revisión. Estos pasos definen a un pipeline.
- Configure la conexión seleccionando un repositorio de código o un servidor externo, como se muestra en la Fig. 71.

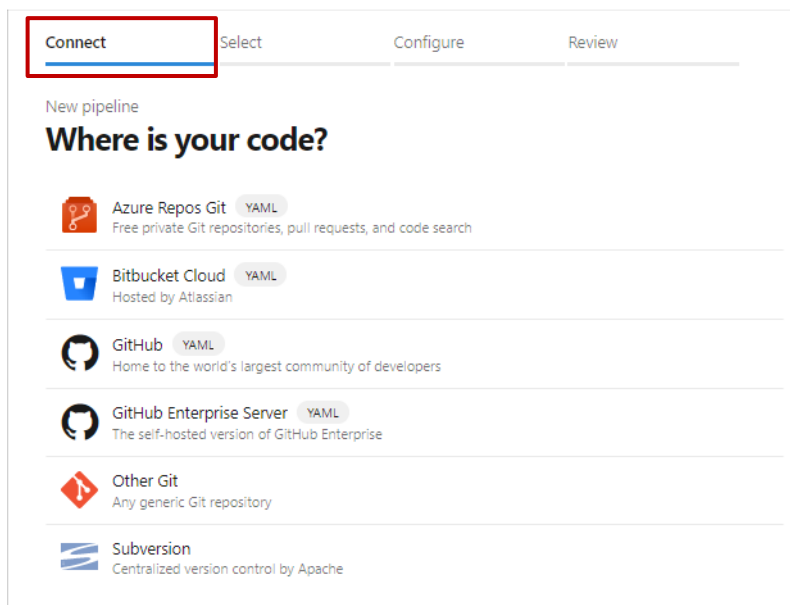


Fig. 71. Conexión de un Pipeline.
Fuente: Propia

- d) Una vez seleccionada una conexión, esta dependerá del proveedor seleccionado. Si selecciona otro repositorio que no sea Azure Repos Git, Azure lo redireccionará a la página de inicio de sesión de su proveedor, después enumerará los repositorios existentes, como se muestra en la Fig. 72.

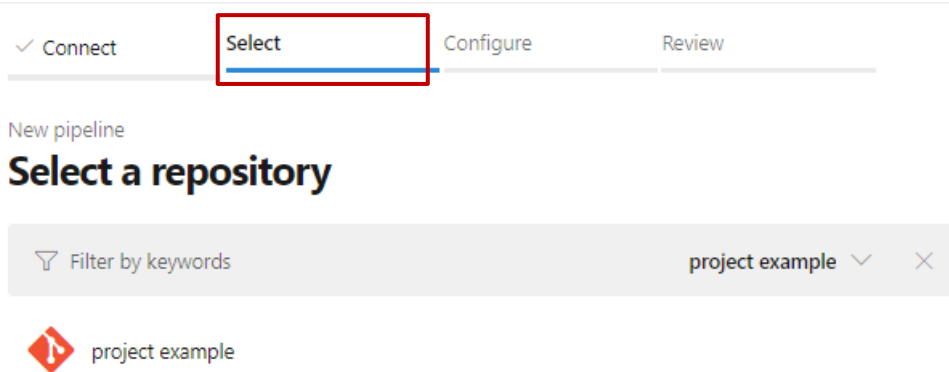


Fig. 72. Selección repositorio en Pipeline.
Fuente: Propia

- e) Seleccionamos el repositorio para su configuración. A continuación, seleccionamos una de las plantillas predeterminadas como se muestra en la Fig. 73, para crear la aplicación.

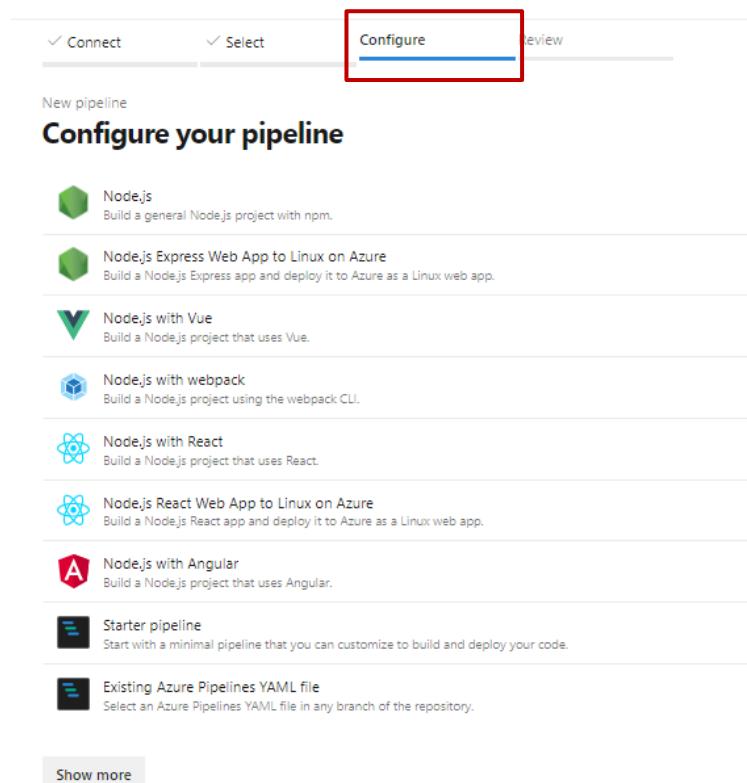


Fig. 73. Configuración Pipeline.
Fuente: Propia.

- f) Finalizada la configuración, revise los detalles como: el repositorio, la tecnología, entre otros. Los detalles se observan en la Fig. 74.

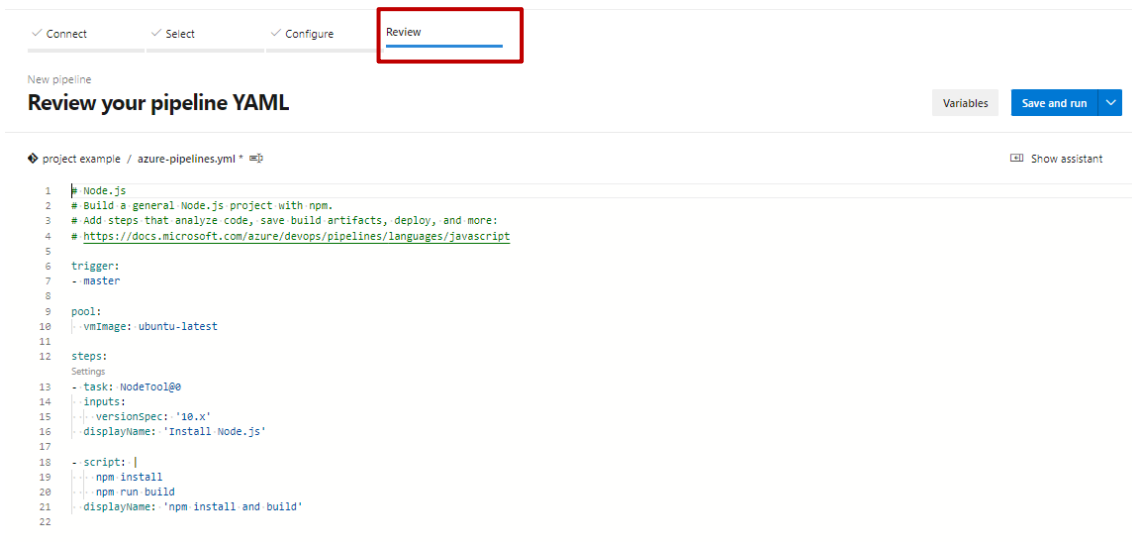


Fig. 74. Revisión de Pipeline.
Fuente: Propia.

3.3 Conclusiones de la elaboración de la guía

En base a la investigación realizada previamente sobre la elaboración de guías metodológicas, se pudo concluir que, la elaboración de una guía metodológica ordenada y detallada resultaría ser una herramienta de gran utilidad para las personas que se encuentran iniciando en la gestión del desarrollo de software utilizando herramientas para la gestión como Azure DevOps. Esta guía beneficiará a los diferentes equipos de trabajo dentro del desarrollo del proyecto de software.

Por otro lado, es recomendable tener una buena distribución de equipos de trabajo, para que de este modo la configuración de un proyecto se pueda realizar de manera más óptima.

Finalmente, la guía metodológica está desarrollada en base al marco de trabajo Scrum, para lo cual hay que tomar en cuenta si se desea trabajar con otro tipo de metodología, pero con las herramientas de Azure DevOps.

CAPÍTULO IV

4 VALIDACIÓN DE RESULTADOS

En este capítulo, se puso a prueba la guía metodológica desarrollada con Azure DevOps y el marco de trabajo SCRUM, mediante el desarrollo de una prueba de concepto la cual será validada con las métricas de completitud funcional basadas de la norma ISO/IEC 25023. A continuación, en la Fig. 75. se muestra la estructura del capítulo.

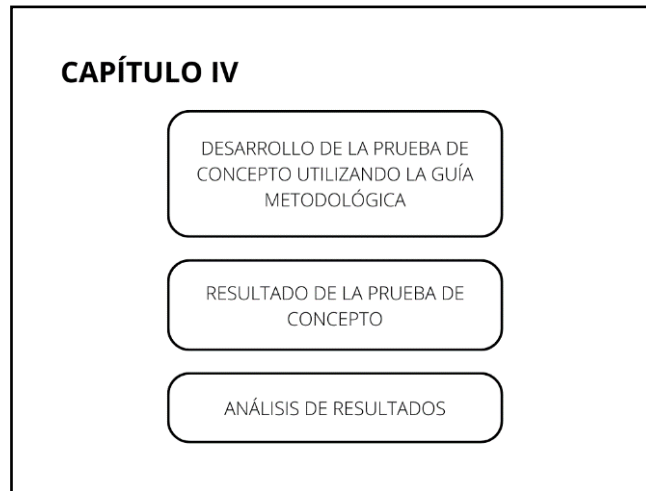


Fig. 75. Estructura Capítulo IV
Fuente: Propia

4.1 Desarrollo de prueba de concepto usando la guía metodológica

4.1.1 Configuración Proyecto

Creamos un nuevo proyecto en Azure DevOps, el proyecto tiene como nombre **Web-Folder**. Para la creación de este proyecto se utilizó el correo electrónico educativo, en este caso con la extensión (@utn.edu.ec).



Fig. 76. Creación de proyecto Prueba de concepto.
Fuente: Propia

4.1.2 Fase 1. Pre-Juego

Planteamiento.

A continuación, se lista el equipo de trabajo creado para la ejecución de concepto realizada.

The screenshot shows a user interface for managing team members. At the top left, it says 'Total 2'. On the right, there is a dropdown menu labeled 'Direct Members' and a blue 'Add' button. Below this is a table with three columns: 'Name', 'Type', and 'Username or scope'. The first row shows a member with a green profile picture (AF), name 'Antonio Quiña Mera MSc. - Do...', type 'aad user', and email 'aquina@utn.edu.ec'. The second row shows a member with a purple profile picture (ES), name 'ERVIN PATRICIO CABA...', type 'aad user', and email 'epcabascangos@utn.edu.ec'. There is an 'Admin' badge next to the second member's name.



<input type="checkbox"/>	Name	Type	Username or scope
<input type="checkbox"/>	 Antonio Quiña Mera MSc. - Do... aquina@utn.edu.ec	aad user	aquina@utn.edu.ec
<input type="checkbox"/>	 ERVIN PATRICIO CABA... epcabascangos@utn.edu.ec	aad user	epcabascangos@utn.edu.ec

Fig. 77. Integrantes de Equipo.
Fuente: Propia.

TABLA 5. Roles Del Proyecto

Rol	Nombre
Product Owner	MSc. Antonio Quiña.
Scrum Master	Sr. Ervin Cabascango
Development Team	Sr. Ervin Cabascango

Fuente: Propia

Montaje.

Una vez realizada la planeación se procede a crear la planificación de los Sprints en los cuales se distribuirán las etapas del proyecto.

Iterations	Start Date	End Date
▼ PruebaConcepto-App	***	
▼ Base de datos	30/05/2022	06/06/2022
Diseño	30/05/2022	31/05/2022
Desarrollo	01/06/2022	03/06/2022
Despliegue	06/06/2022	06/06/2022
▼ Aplicativo BackEnd	07/06/2022	20/06/2022
Diseño	07/06/2022	08/06/2022
Desarrollo	09/06/2022	17/06/2022
Despliegue	20/06/2022	20/06/2022
▼ Aplicativo FrontEnd	21/06/2022	11/07/2022
Diseño	21/06/2022	24/06/2022
Desarrollo	27/06/2022	08/07/2022
Despliegue	11/07/2022	11/07/2022
▼ Entrega	12/07/2022	19/07/2022
Pruebas Integración	12/07/2022	14/07/2022
Despliegue	15/07/2022	19/07/2022

Fig. 78. Sprint de Prueba de concepto.
Fuente: Propia

Montaje - Product Backlog

Creamos el Product Backlog del la prueba de concepto, estos son los requerimientos iniciales del proyecto.

Order	Work Item Type	Title
1	Product Backl...	Inicio de sesión usuario
2	Product Backl...	Creación de proyectos
3	Product Backl...	Creación ítems experiencia
4	Product Backl...	Creación ítems certificaciones
5	Product Backl...	Envío de mensajes por correo

Fig. 79. Product Backlog Prueba de Concepto.
Fuente: Propia

Montaje – Tableros

Después de crear el Product Backlog, es posible evidenciar las tareas en los tableros de Azure, adicional se muestran tareas correspondientes a los Sprint, según se vayan agregando.

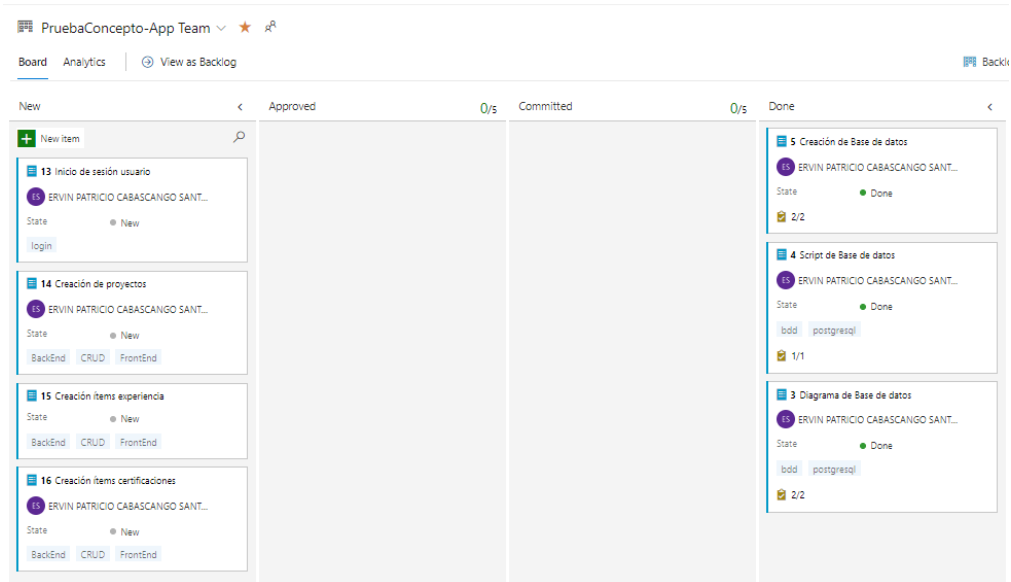


Fig. 80. Tablero Product Backlog prueba de concepto.
Fuente: Propia

Sprints

Esta sección se establece en base al montaje que configuramos anteriormente. Por lo cual, se distribuyó en Base de Datos, Aplicativo BackEnd, Aplicativo FrontEnd y Entrega.

- **Sprint 1 - Base de Datos.**

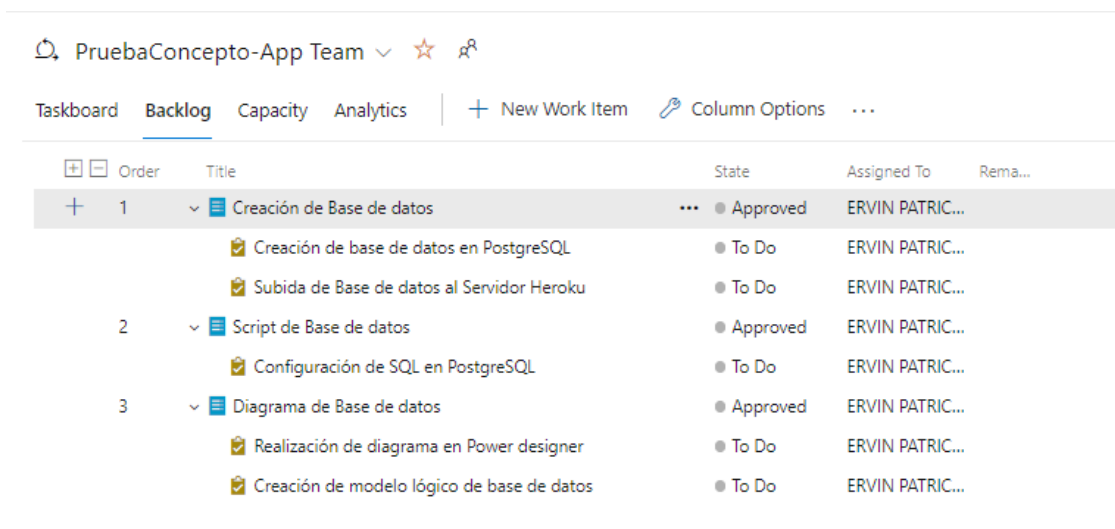


Fig. 81. Sprint 1 Base de datos.
Fuente: Propia

- **Sprint 2 - Aplicativo BackEnd**

PruebaConcepto-App Team

Taskboard Backlog Capacity Analytics + New Work Item Column Options ...

Order	Title	State	Assigned To	Rema...
1	Despliegue BackEnd	Committed	ERVIN PATRIC...	2
	Configuración y despliegue	To Do	ERVIN PATRIC...	2
2	CRUD Certificaciones	Committed	ERVIN PATRIC...	4
	API Certificaciones	To Do	ERVIN PATRIC...	3
	Pruebas	To Do	ERVIN PATRIC...	1
3	CRUD Experiencia	Committed	ERVIN PATRIC...	4
	API Experiencia	To Do	ERVIN PATRIC...	3
	Pruebas	To Do	ERVIN PATRIC...	1
4	CRUD Habilidades	Committed	ERVIN PATRIC...	4
	API Habilidades	To Do	ERVIN PATRIC...	3
	Pruebas	To Do	ERVIN PATRIC...	1
5	Seguridad APIS	Committed	ERVIN PATRIC...	4
	Inicio de sesión	To Do	ERVIN PATRIC...	3
	Implementación JWT	To Do	ERVIN PATRIC...	1
6	Configuración BackEnd	Committed	ERVIN PATRIC...	2
	Conexión a BDD	To Do	ERVIN PATRIC...	1
	Arquitectura BackEnd	To Do	ERVIN PATRIC...	1

Fig. 82. Sprint 2 BackEnd.
Fuente: Propia

- **Sprint 3 - Apicativo FrontEnd**

PruebaConcepto-App Team

Taskboard Backlog Capacity Analytics | + New Work Item Column Options ...

Order	Title	State	Assigned To	Rema...
1	Despliegue FrontEnd	Committed	ERVIN PATRIC...	3
	Configuración y despliegue	To Do	ERVIN PATRIC...	3
2	CRUD Habilidades	Committed	ERVIN PATRIC...	6
	Desarrollo página y CRUD Habilidades	To Do	ERVIN PATRIC...	5
	Pruebas	To Do	ERVIN PATRIC...	1
3	CRUD Experiencia	Committed	ERVIN PATRIC...	6
	Desarrollo página y CRUD experiencia	To Do	ERVIN PATRIC...	5
	Pruebas	To Do	ERVIN PATRIC...	1
4	CRUD Certificaciones	Committed	ERVIN PATRIC...	6
	Desarrollo página y CRUD certificaciones	To Do	ERVIN PATRIC...	5
	Pruebas	To Do	ERVIN PATRIC...	1
5	Seguridad FrontEnd	Committed	ERVIN PATRIC...	4
	Inicio de sesión	To Do	ERVIN PATRIC...	3
	Pruebas	To Do	ERVIN PATRIC...	1
6	Visualización Inicial y páginas.	Committed	ERVIN PATRIC...	2
	Creación de páginas del aplicativo	To Do	ERVIN PATRIC...	2
7	Configuración FrontEnd	Committed	ERVIN PATRIC...	3
	Creación y configuración de repositorio git	To Do	ERVIN PATRIC...	1
	Conexión a BackEnd	To Do	ERVIN PATRIC...	1
	Arquitectura FrontEnd	To Do	ERVIN PATRIC...	1

Fig. 83. Sprint 3 FrontEnd.
Fuente: Propia

- Sprint 4 - Entrega**

PruebaConcepto-App Team

Taskboard Backlog Capacity Analytics | + New Work Item Column Options ...

Order	Title	State	Assigned To	Remaining ...
1	Pruebas de Integración	New	ERVIN PATRICIO CAB...	4
	Pruebas de Diseño	To Do	ERVIN PATRICIO CAB...	2
	Pruebas de Funcionalidad	To Do	ERVIN PATRICIO CAB...	2
2	Despliegue Aplicativo	New	ERVIN PATRICIO CAB...	3
	Dominio configurado	To Do	ERVIN PATRICIO CAB...	2
	Pruebas de respuesta	To Do	ERVIN PATRICIO CAB...	1

Fig. 84. Sprint 4 Entrega.
Fuente: Propia

4.1.3 Fase 2. Juego

SPRINT 1 – Base de Datos

El siguiente sprint muestra las actividades realizadas para llevar a cabo el modelado y construcción de la base de datos utilizada para el portafolio personal, a continuación, se detallan las tareas realizadas y detalladas en Azure Boards.

TABLA 6. Prototipo Sprint 1

SPRINT 1 (30-05-2022 / 06-06-2022)		
Backlog	Tarea	Horas
Creación de Base de datos	Creación de base de datos en PostgreSQL	4h
	Despliegue de base de datos al servidor	2h
	Total	6h

Fuente: Propia

The screenshot shows the Azure Boards interface for a backlog titled 'Base de datos'. It lists several tasks under the category 'Creación de Base de datos', all of which are marked as 'Done' and assigned to 'ERVIN PATRICIO CABASCANGO SANTACRUZ'. The tasks include: 'Creación de base de datos en PostgreSQL', 'Subida de Base de datos al Servidor Heroku', 'Script de Base de datos', 'Configuración de SQL en PostgreSQL', 'Diagrama de Base de datos', 'Realización de diagrama en Power designer', and 'Creación de modelo lógico de base de datos'.

Order	Title	State	Assigned To
1	Creación de Base de datos	Done	ERVIN PATRICIO CABASCANGO SANTACRUZ
	Creación de base de datos en PostgreSQL	Done	ERVIN PATRICIO CABASCANGO SANTACRUZ
	Subida de Base de datos al Servidor Heroku	Done	ERVIN PATRICIO CABASCANGO SANTACRUZ
2	Script de Base de datos	Done	ERVIN PATRICIO CABASCANGO SANTACRUZ
	Configuración de SQL en PostgreSQL	Done	ERVIN PATRICIO CABASCANGO SANTACRUZ
3	Diagrama de Base de datos	Done	ERVIN PATRICIO CABASCANGO SANTACRUZ
	Realización de diagrama en Power designer	Done	ERVIN PATRICIO CABASCANGO SANTACRUZ
	Creación de modelo lógico de base de datos	Done	ERVIN PATRICIO CABASCANGO SANTACRUZ

Fig. 85. Sprint 1 Prueba de Concepto. Fuente: Propia

SPRINT 2 – Aplicativo Backend

TABLA 7. Prototipo Sprint 2

SPRINT 2 (07-06-2022 / 20-06-2022)		
Backlog	Tarea	Horas
Creación BackEnd	Arquitectura y despliegue de Backend al servidor	4h
	CRUD de tabla certificaciones	6h
	CRUD de tabla experiencias	6h
	CRUD de tabla habilidades	6h
	Implementación de seguridad en APIS	4h
	Envío de correos	4h
	Total	30h

Fuente: Propia

Order	Title	State	Assigned To
1	Enviar mensajes de correo	Done	ERVIN PATRICIO CABASCANGO SANTACRUZ
	Conexión Google Apis	Done	ERVIN PATRICIO CABASCANGO SANTACRUZ
	Conexión Nodemailer	Done	ERVIN PATRICIO CABASCANGO SANTACRUZ
	Pruebas	Done	ERVIN PATRICIO CABASCANGO SANTACRUZ
2	Configuración BackEnd	Done	ERVIN PATRICIO CABASCANGO SANTACRUZ
	Conexión a BDD	Done	ERVIN PATRICIO CABASCANGO SANTACRUZ
	Arquitectura BackEnd	Done	ERVIN PATRICIO CABASCANGO SANTACRUZ
3	CRUD Certificaciones	Done	ERVIN PATRICIO CABASCANGO SANTACRUZ
	API Certificaciones	Done	ERVIN PATRICIO CABASCANGO SANTACRUZ
	Pruebas	Done	ERVIN PATRICIO CABASCANGO SANTACRUZ
4	CRUD Experiencia	Done	ERVIN PATRICIO CABASCANGO SANTACRUZ
	API Experiencia	Done	ERVIN PATRICIO CABASCANGO SANTACRUZ
	Pruebas	Done	ERVIN PATRICIO CABASCANGO SANTACRUZ
5	CRUD Certificaciones	Done	ERVIN PATRICIO CABASCANGO SANTACRUZ
	Desarrollo página y CRUD certificaciones	Done	ERVIN PATRICIO CABASCANGO SANTACRUZ
6	CRUD Habilidades	Done	ERVIN PATRICIO CABASCANGO SANTACRUZ
	API Habilidades	Done	ERVIN PATRICIO CABASCANGO SANTACRUZ
	Pruebas	Done	ERVIN PATRICIO CABASCANGO SANTACRUZ
7	Seguridad APIS	Done	ERVIN PATRICIO CABASCANGO SANTACRUZ
	Inicio de sesión	Done	ERVIN PATRICIO CABASCANGO SANTACRUZ
	Implementación JWT	Done	ERVIN PATRICIO CABASCANGO SANTACRUZ
8	Despliegue BackEnd	Done	ERVIN PATRICIO CABASCANGO SANTACRUZ
	Configuración y despliegue	Done	ERVIN PATRICIO CABASCANGO SANTACRUZ

Fig. 86. Sprint 2 Prueba de Concepto.
Fuente: Propia

SPRINT 3 – Aplicativo Frontend

TABLA 8. Prototipo Sprint 3

SPRINT 2 (21-06-2022 / 11-07-2022)		
Backlog	Tarea	Horas
Creación FrontEnd	CRUD y vistas certificaciones	8h
	CRUD y vistas de tabla experiencias	8h
	CRUD y vistas de tabla habilidades	8h
	Implementación de seguridad en consumo de APIS	4h
	Envío de correos	2h
	Total	28h

Fuente: Propia

Order	Title	State	Assigned To
1	Despliegue FrontEnd	Done	ERVIN PATRICIO CABASCANGO SANTACRUZ
	Configuración y despliegue	Done	ERVIN PATRICIO CABASCANGO SANTACRUZ
2	CRUD Habilidades	Done	ERVIN PATRICIO CABASCANGO SANTACRUZ
	Desarrollo página y CRUD Habilidades	Done	ERVIN PATRICIO CABASCANGO SANTACRUZ
	Pruebas	Done	ERVIN PATRICIO CABASCANGO SANTACRUZ
3	CRUD Experiencia	Done	ERVIN PATRICIO CABASCANGO SANTACRUZ
	Desarrollo página y CRUD experiencia	Done	ERVIN PATRICIO CABASCANGO SANTACRUZ
	Pruebas	Done	ERVIN PATRICIO CABASCANGO SANTACRUZ
4	CRUD Certificaciones	Done	ERVIN PATRICIO CABASCANGO SANTACRUZ
	Desarrollo página y CRUD certificaciones	Done	ERVIN PATRICIO CABASCANGO SANTACRUZ
	Pruebas	Done	ERVIN PATRICIO CABASCANGO SANTACRUZ
5	Seguridad FrontEnd	Done	ERVIN PATRICIO CABASCANGO SANTACRUZ
	Interceptores para peticiones https	Done	ERVIN PATRICIO CABASCANGO SANTACRUZ
	Inicio de sesión	Done	ERVIN PATRICIO CABASCANGO SANTACRUZ
	Pruebas	Done	ERVIN PATRICIO CABASCANGO SANTACRUZ
6	Visualización Inicial y páginas.	Done	ERVIN PATRICIO CABASCANGO SANTACRUZ
	Creación de páginas del aplicativo	Done	ERVIN PATRICIO CABASCANGO SANTACRUZ
7	Configuración FrontEnd	Done	ERVIN PATRICIO CABASCANGO SANTACRUZ
	Creación y configuración de repositorio git	Done	ERVIN PATRICIO CABASCANGO SANTACRUZ
	Conexión a BackEnd	Done	ERVIN PATRICIO CABASCANGO SANTACRUZ
	Arquitectura FrontEnd	Done	ERVIN PATRICIO CABASCANGO SANTACRUZ

Fig. 87. Sprint 3 Prueba de Concepto.
Fuente: Propia

- Sprint Backlog**

Creación del repositorio de código, el proyecto utilizó una rama llamada *ervin* para realizar las diferentes actividades asignadas en los sprints.

Name ↑	Last change	Commits
Folder e2e	11 jul	b6e8e8f8 feat: init proyect ervin cab...
Folder src	26 jul	924e731a feat: changes style ervin ca...
.browserslistrc	11 jul	b6e8e8f8 feat: init proyect ervin cab...
.editorconfig	11 jul	b6e8e8f8 feat: init proyect ervin cab...
.gitignore	11 jul	b6e8e8f8 feat: init proyect ervin cab...
angular.json	11 jul	b6e8e8f8 feat: init proyect ervin cab...
JS karma.conf.js	11 jul	b6e8e8f8 feat: init proyect ervin cab...
package-lock.json	11 jul	b6e8e8f8 feat: init proyect ervin cab...
package.json	11 jul	b6e8e8f8 feat: init proyect ervin cab...
MI README.md	11 jul	b6e8e8f8 feat: init proyect ervin cab...

Fig. 88. Repositorio Prueba de Concepto.
Fuente: Propia

A continuación, se describen actividades realizadas en base a las tareas del Sprint

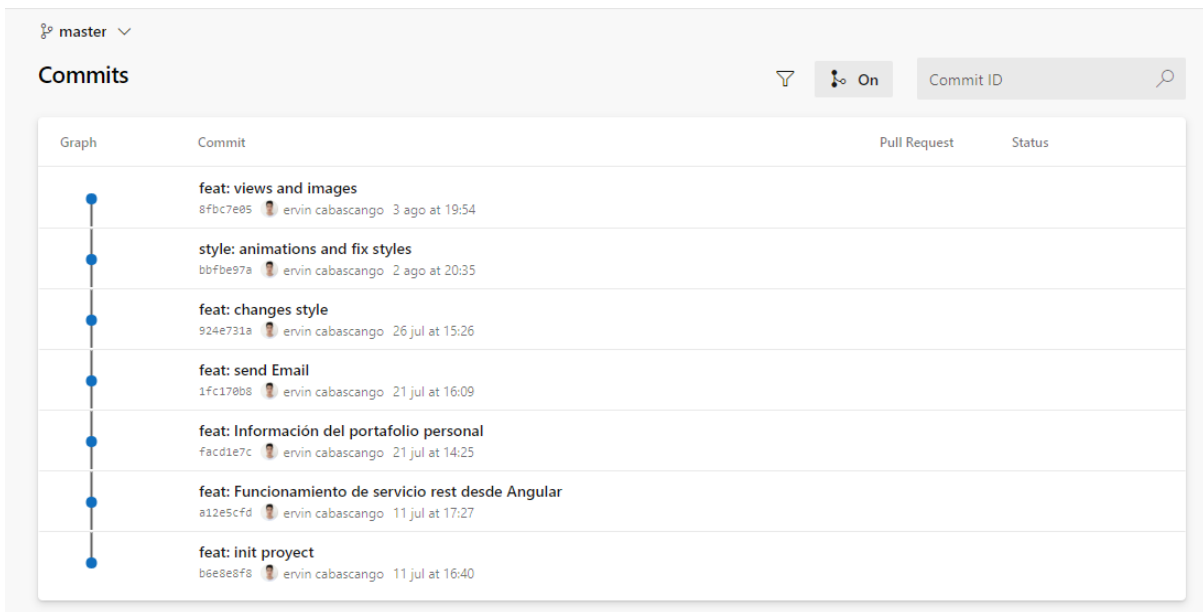


Fig. 89. Commits Portafolio web.
Fuente: Propia

SPRINT 4 – Entrega

TABLA 9. Prototipo Sprint 4

SPRINT 4 (12-07-2022 / 19-07-2022)		
Backlog	Tarea	Horas
Entrega	Pruebas de integración	4h
	Despliegue del aplicativo	3h
	Total	7h

Fuente: Propia

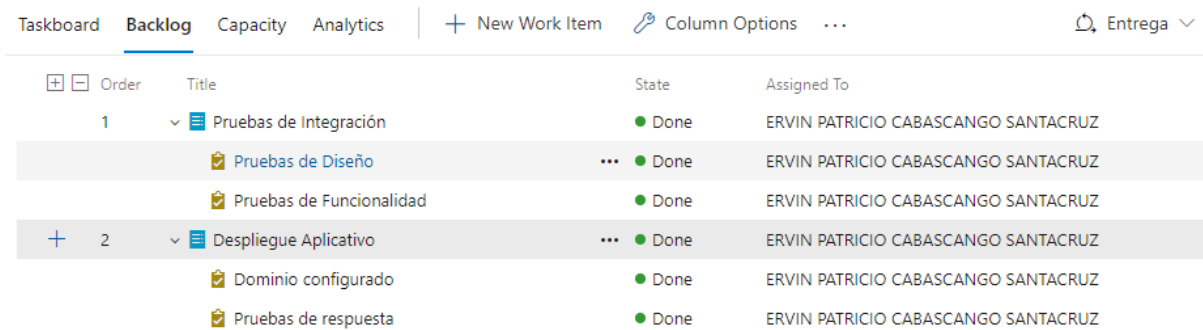


Fig. 90. Sprint 4 Prueba de Concepto.
Fuente: Propia

4.1.4 Fase 3. Post-Juego

En la siguiente imagen se muestra el repositorio de código integrado con los cambios realizados a lo largo de la planificación. El proyecto integrado es almacenado en la rama *master* del repositorio.

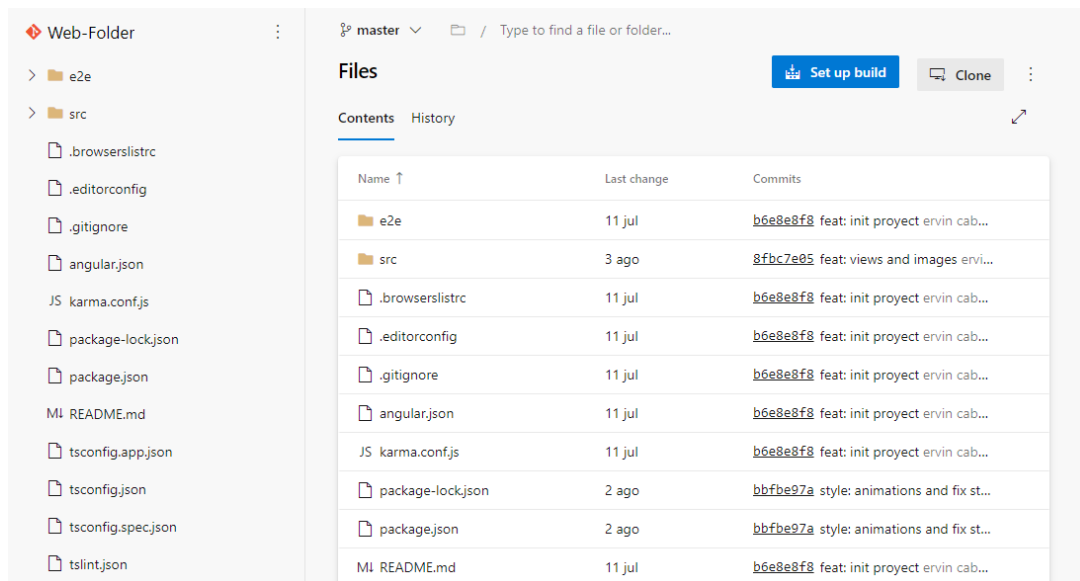


Fig. 91. Integración de actividades Prueba de Concepto.
Fuente: Propia

4.2 Resultados de la prueba de concepto

En esta sección se presenta la evaluación realizada a la guía metodológica para la gestión de proyectos de software usando Microsoft Azure DevOps y el marco de trabajo SCRUM. La evaluación de la presente guía se realizó en base a medidas de completitud funcional, para la cual se tomaron las etapas planteadas en la sección 3.2, los mismos que muestran cómo llevar a cabo la gestión de un proyecto de software, dividiéndose en tres etapas: pre-juego, juego, post-juego.

4.2.1 Análisis de resultados

4.2.1.1 Medida de completitud funcional

La medida de completitud funcional se utiliza para evaluar el grado en el cual un conjunto de funciones abarca diferentes tareas y objetivos especificados por el usuario. A continuación, en la siguiente TABLA 10 se describe el proceso de evaluación en base a la medida de completitud funcional.

TABLA 10. Medida de Completitud Funcional

ID	Nombre	Descripción	Función de medición
----	--------	-------------	---------------------

FCp-1	Cobertura funcional	¿Qué proporción de las funciones han sido implementadas?	$X = 1 - A / B$ A = Número de funciones ausentes B = Número de funciones establecidas
-------	---------------------	--	---

Nota 1. Las funciones pueden establecerse como una especificación de la guía de desarrollo.

Nota 2. Las funciones ausentes se detectan cuando el producto no tiene la capacidad de ejecutar una función establecida.

Fuente: Propia

4.2.1.2 Criterios de evaluación

La evaluación de la guía metodológica se realizó en base al desarrollo de una prueba de concepto, la cual se aplicaron actividades que fueron propuestas en la sección 4.1, a continuación, en la siguiente TABLA 11 se evidencian los resultados obtenidos:

TABLA 11. Evaluación Guía de Desarrollo

Fase	Actividad	Tarea SCRUM	Cumplimiento	Observación
	SCRUM			
PRE – JUEGO	Planteamiento	Planeación	SI	
		Roles	SI	
	Montaje	Product	SI	
		Backlog		
		Daily SCRUM	SI	Estas etapas pueden generarse en la creación de una tarea en el tablero de Azure Boards
Revisión Sprint	SI			
JUEGO	Sprint	Sprint Backlog	SI	
		Retrospectiva	SI	
		Sprint		
	Entrega	Integración	SI	Esta etapa se puede contemplar en la creación de una tarea asignándole un tipo en el tablero de Azure Boards.

POST - JUEGO	Pruebas de integración	SI (50%)	Las pruebas de integración no evalúan código.
	Despliegue	NO	
	Funciones especificadas	10	

Fuente: Propia

Las funciones especificadas para la evaluación de la guía metodológica planteada, en el mejor de los casos es de 10 funciones aprobadas para obtener una satisfacción del 100%, cada una de las funciones aprobadas suman un total de 8,5, sin embargo, ciertas funciones especificadas muestran limitaciones, por lo cual se toman en cuenta su valor decimal. A continuación, se obtiene el porcentaje de éxito en base a la función de medición:

- **X:** Éxito de la guía
- **A:** Número de funciones ausentes
- **B:** Número de funciones especificadas

$$x = 1 - \left(\frac{1,5}{10}\right)$$

De acuerdo con la fórmula, el porcentaje de éxito de cumplimiento de la guía con las funciones especificadas es del **85%**.

4.2.1.3 Escala de Evaluación

El análisis del resultado fue sustentando en base a una escalada de medición, la cual nos provee rangos de aceptación, entre los cuales van desde un rango Inaceptables hasta un excedente de requisitos como se muestra en la Fig. 92. La escala de evaluación de la guía metodológica desarrollada fue sustentada en base a la norma ISO/IEC 25040, dicha norma aplica métricas para cumplir requisitos de calidad del producto.

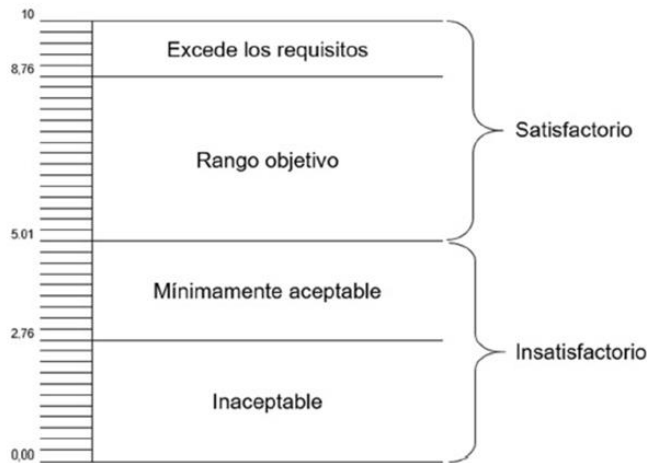


Fig. 92. Escala de evaluación
Fuente: Adaptada de (ISO/IEC 14598-1, 1999)

La evaluación de la guía metodológica para la gestión de proyectos de software usando Microsoft Azure DevOps y el marco de trabajo SCRUM, como se puede observar en la TABLA 11 nos muestra el éxito de cumplimiento de la guía metodológica, obteniendo un 8,5 (Rango objetivo) siendo satisfactoria en el cumplimiento de las funciones especificadas en la sección 4.2.1.2.

Conclusiones

Con el marco teórico y tecnológico establecido en el Capítulo 2, se fundamentó conceptualmente el desarrollo y evaluación de la propuesta metodológica “Guía para la gestión de proyectos de software con Microsoft Azure DevOps basado en marco de trabajo SCRUM”, como también, el desarrollo de la prueba de concepto utilizando la propuesta metodológica.

Con la elaboración de la Guía para la gestión de proyectos con Azure DevOps y el marco de trabajo SCRUM, se logró fundamentalmente integrar las actividades del marco de trabajo Scrum con las funcionalidades de Microsoft Azure DevOps, las cuales, operativizan la gestión de proyectos de desarrollo de software.

La prueba de concepto operativizada con Azure DevOps nos permitió validar las actividades propuestas por la guía metodológica, para lo cual se declararon 10 funciones distribuidas en todas las fases de SCRUM, teniendo como resultado un éxito 85% de aceptación de la guía metodológica.

La aplicación de la norma ISO/EC 25040, permitió evaluar los resultados obtenidos y se pudo determinar que tuvo un rango de objetivo (satisfactorio), en base a las funciones declaradas para la ejecución de la guía metodológica.

En base a la investigación realizada sobre la elaboración de guías metodológicas, se pudo concluir que, la elaboración de una guía metodológica ordenada y detallada resultaría ser una herramienta de gran utilidad para las personas que se encuentran iniciando en la gestión del desarrollo de software utilizando herramientas para la gestión como Azure DevOps.

Recomendaciones

La guía metodológica fue desarrollada en base al marco de trabajo SCRUM, por lo cual hay que tomar en cuenta este factor si se desea trabajar con otro tipo de metodología haciendo uso de las herramientas de Azure DevOps.

Se recomienda realizar una distribución adecuada de los roles de los equipos de trabajo en la configuración del proyecto para que los integrantes identifiquen de mejor manera las actividades que deben realizar y que haya una comunicación efectiva para llevar a cabo los pasos que se indican en la guía propuesta.

Es recomendable activar la opción de facturación en Azure para obtener el acceso a todas las características y beneficios ofrecidos por Microsoft Azure DevOps como añadir más de cinco miembros a los equipos de trabajo, mayor almacenamiento o despliegue del producto de software en la nube de Azure.

Se recomienda continuar con la investigación realizando unas guías metodológicas para la gestión de proyectos de software con otras herramientas tecnológicas como Jira, Asana, GitHub, Bitbucket como también, con otros marcos o modelos metodológicos ágiles como Extreme Programming o XP, Kanban, y Lean.

Referencias

- Ahmed, A. (2012). *Software Project Management: A Process-Driven Approach*. Auerbach Publications.
- Aruquipa, P. (2016). *SISTEMA DE REGISTRO Y ADMINISTRACIÓN DOCUMENTAL*. UNIVERSIDAD MAYOR DE SAN ANDRÉS.
- Ashfaqe Ahmed. (2011). *Software Project Management: A Process-Driven Approach 1st Edition*. In *CRC Press*. <http://dl.acm.org/citation.cfm?doid=2464526.2464537>
- Autentia. (n.d.). *Guía para directivos y técnicos GUÍA COMPLETA*. <https://www.autentia.com/libros/>
- Bazán, G., Romero, E., & Rivero, E. (2012). *Comparativa de Herramientas de Gestión de Software*. 6.
- Cadavid, A. N., Martínez, J. D. F., & Vélez, J. M. (2013). Revisión de metodologías ágiles para el desarrollo de software. *Prospectiva*, 11(2), 30–39. <http://www.redalyc.org/articulo.oa?id=496250736004%0ACómo>
- Calderón, A., & Ruiz, M. (2015). A systematic literature review on serious games evaluation: An application to software project management. *Computers and Education*, 87, 396–422. <https://doi.org/10.1016/j.compedu.2015.07.011>
- Cantone, D. (2006). *La Biblia Del Programador Implementacion Y Debugging*. 320.
- Ebert, C., Gallardo, G., Hernantes, J., & Serrano, N. (2016). DevOps. *IEEE Software*, 33(3), 94–100. <https://doi.org/10.1109/MS.2016.68>
- Felipe, A., & Núñez, F. D. J. (2022). DevOps: un vistazo rápido DevOps: a quick look. *Publicación Semestral*, 10(19), 35–40. <https://repository.uaeh.edu.mx/revistas/index.php/huejutla/issue/archive>
- García, R. I. (2014). *Guía Técnica de Gestión de Proyectos de Software*. Instituto Tecnológico de Ciudad Valles.
- Gallego, M. G., & Cáceres, J. H. (2015). Identificación de factores que permitan potencializar el éxito de proyectos de desarrollo de software. *Scientia Et Technica*, 20(1), 70–80.
- Girma, M., Garcia, N. M., & Kifle, M. (2019). Agile scrum scaling practices for large scale software development. *Proceedings - 2019 4th International Conference on Information Systems Engineering, ICISE 2019, Ld*, 34–38. <https://doi.org/10.1109/ICISE.2019.00014>
- Halvorsen, H.-P. (2020). *Software Development A Practical Approach!*

<https://www.halvorsen.blog>

- Harvie, D. P., & Agah, A. (2016). Targeted Scrum: Applying Mission Command to Agile Software Development. *IEEE Transactions on Software Engineering*, 42(5), 476–489. <https://doi.org/10.1109/TSE.2015.2489654>
- Hayat, F., Rehman, A. U., Arif, K. S., Wahab, K., & Abbas, M. (2019). The Influence of Agile Methodology (Scrum) on Software Project Management. *Proceedings - 20th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, SNPD 2019*, 145–149. <https://doi.org/10.1109/SNPD.2019.8935813>
- Ingrid Yamile Polo Caita, I. F. R. R. (2020). *gestión de personas en el desarrollo de proyectos de Factores de fracaso relacionados con la gestión de personas en el desarrollo de proyectos de software* Ingrid Yamile Polo Caita. TESIS Universidad EAN Facultad de Ingeniería Maestría En Gerencia de Proyectos Bogotá, Colombia.
- K K, A. (2020). *Azure DevOps for Web Developers*. In *Azure DevOps for Web Developers*. Apress. <https://doi.org/10.1007/978-1-4842-6412-6>
- Kurnia, R., Ferdiana, R., & Wibirama, S. (2018). Software metrics classification for agile scrum process: A literature review. *2018 International Seminar on Research of Information Technology and Intelligent Systems, ISRITI 2018*, 174–179. <https://doi.org/10.1109/ISRITI.2018.8864244>
- Lockwood, C., & Tricco, A. C. (2020). Preparing scoping reviews for publication using methodological guides and reporting standards. *Nursing and Health Sciences*, 22(1), 1–4. <https://doi.org/10.1111/nhs.12673>
- Lopez, S. (2014). *GESTION DE PROYECTOS DE SOFTWARE INGENIERIA*.
- Maida, E., & Pacienza, J. (2018). Metodologías de desarrollo de software. *Biblioteca Digital de La Universidad Católica Argentina*, 117. <http://bibliotecadigital.uca.edu.ar/repositorio/tesis/metodologias-desarrollo-software.pdf>
- Meza, E. (2014). *Propuesta de guía metodológica para la elaboración de trabajos especiales de grado, con el fin de facilitar el trabajo de investigación de los aspirantes a maestría en el Ecuador*. International Lifelong Learning University.
- Naciones Unidas. (2019). *Naciones Unidas*. Obtenido de Naciones Unidas: <https://www.un.org/sustainabledevelopment/es/>

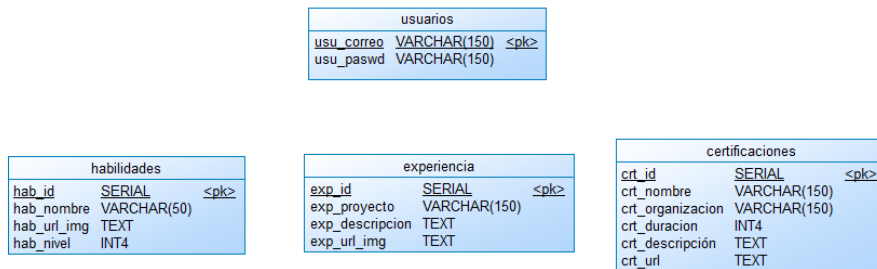
- Nayebi, M., Ruhe, G., Mota, R. C., & Mufti, M. (2016). Analytics for software project management - Where are we and where do we go? *Proceedings - 2015 30th IEEE/ACM International Conference on Automated Software Engineering Workshops, ASEW 2015*, 18–21. <https://doi.org/10.1109/ASEW.2015.28>
- Ocaña, A. (2019). *Herramientas para la gestión de un proyecto con Azure DevOps*. Universidad Mayor de San Simón.
- Otalora-Luna, J. E., Callejas-Cuervo, M., & Alarcon-Aldana, A. C. (2018). *Metamodelo de medicion de esfuerzo en proyectos de desarrollo de software*. Editorial UPTC.
- Palacio, J. (2015). Scrum Manager I. In *Scrum Manager* (Vol. 2).
- Pérez Velásquez, A. (Junio de 2017). Universidad Politécnica Salesiana. Obtenido de dspace.ups.edu.ec: <https://dspace.ups.edu.ec/bitstream/123456789/14417/1/UPS-CT007084.pdf>
- Rivadeneira, S. (2013). Metodologías ágiles enfocadas al modelo de requerimientos. *Informes Científicos - Técnicos UNPA*, 5(1), 1–29. <http://ict.unpa.edu.ar/journal/index.php/ICTUNPA/article/view/66>
- Robles, M. (2017). *Guía Metodológica. Qué es? Cómo se realiza? 1. Definición de objetivo, alcance y audiencia APROBACIÓN DIFUSIÓN EDICIÓN Y DISEÑO - PDF Descargar libre*.
- Rossberg, J. (2019). Agile Project Management with Azure DevOps. *Agile Project Management with Azure DevOps*, 37–66. <https://doi.org/10.1007/978-1-4842-4483-8>
- Schwaber, K., & Sutherland, J. (2020). *La Guía Scrum. La Guía Definitiva de Scrum: Las Reglas del Juego*. 17. <https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Spanish-European.pdf>
- Soto, B. (2017). *Análisis comparativo de las herramientas software para gestión de proyectos*. ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES.
- Sommerville, I. (2017). Procesos de Software. *Ingeniería de Software*, 36–38. <http://procesosdesoftware.blogspot.com/%0Ahttps://revistas.udistrital.edu.co/ojs/index.php/vinculos/article/view/4141/5806>
- Srivastava, A., Bhardwaj, S., & Saraswat, S. (2017). SCRUM model for agile methodology. *Proceeding - IEEE International Conference on Computing, Communication and Automation, ICCCA 2017, 2017-January*, 864–869.

<https://doi.org/10.1109/CCAA.2017.8229928>

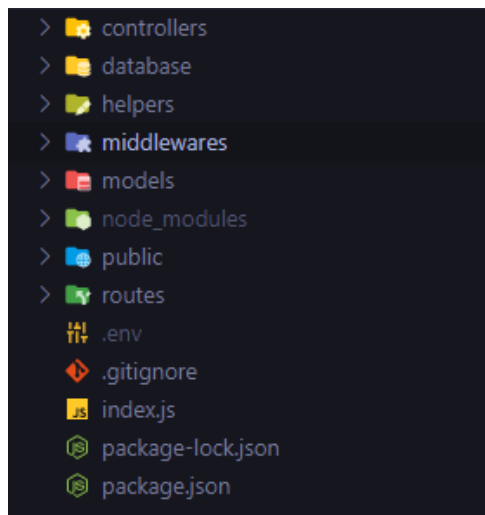
- Tavares, B. G., da Silva, C. E. S., & de Souza, A. D. (2019). Risk management analysis in Scrum software projects. *International Transactions in Operational Research*, 26(5), 1884–1905. <https://doi.org/10.1111/itor.12401>
- Techbizdesign*. (31 de Agosto de 2018). Obtenido de Techbizdesign: <https://www.techbizdesign.com/biz/fracaso-proyectos-software/>
- Urbina Delgadillo, M. L., Figueroa, M. A. A., Peláez Camarena, G., Alor Hernández, G., & Sánchez García, A. I. (2016). Mixing Scrum-PSP: Combinación de Scrum y PSP para mejorar la calidad del proceso de software. *Research in Computing Science*, 126(1), 19–29. <https://doi.org/10.13053/rcs-126-1-2>
- Urteaga Pecharromán, A. (2015). *Aplicación de la metodología de desarrollo ágil Scrum para el desarrollo de un sistema de gestión de empresas*. 131. <https://e-archivo.uc3m.es/handle/10016/23750>
- Valeria, E., Terán, F., & Julio, A. De. (n.d.). *Desarrollo e implementación de un sistema de información para la gestión del “Gremio de Maestros Mecánicos y Afines de Ibarra .”* 1–5.
- Verma, A., Malla, D., Choudhary, A. K., & Arora, V. (2019). A Detailed Study of Azure Platform Its Cognitive Services. *Proceedings of the International Conference on Machine Learning, Big Data, Cloud and Parallel Computing: Trends, Prespectives and Prospects, COMITCon 2019*, 129–134. <https://doi.org/10.1109/COMITCon.2019.8862178>
- Verner, J., Sampson, J., & Cerpa, N. (2008). What factors lead to software project failure? *Proceedings of the 2nd International Conference on Research Challenges in Information Science, RCIS 2008*, 71–79. <https://doi.org/10.1109/RCIS.2008.4632095>
- Zapata, M., Madrenas, J., Zapata, M., & Alvarez, J. (2019). *Advances in Artificial Intelligence, Software and Systems Engineering* (Vol. 787, Issue 1995). Springer International Publishing. <https://doi.org/10.1007/978-3-319-94229-2>
- Zavala, J. (2004). ¿ Por Qué Fracasan los Proyectos de Software ?; Un Enfoque Organizacional. *Congreso Nacional de Software Libre, November*, 1–21. <https://doi.org/10.13140/RG.2.1.4741.3206>

Anexos

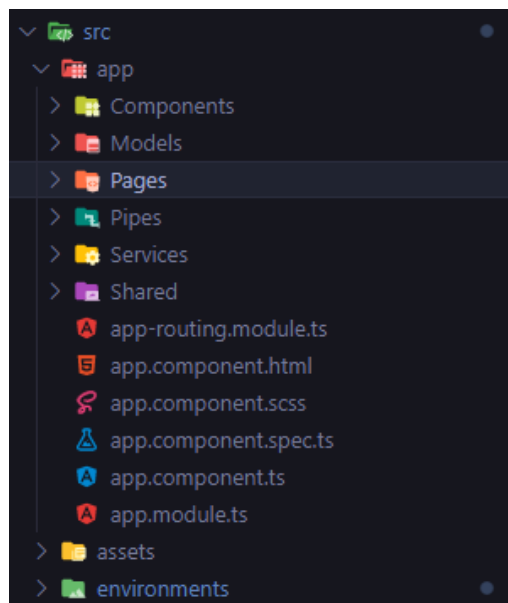
Anexo A: Diagrama de base de datos de la prueba de concepto.



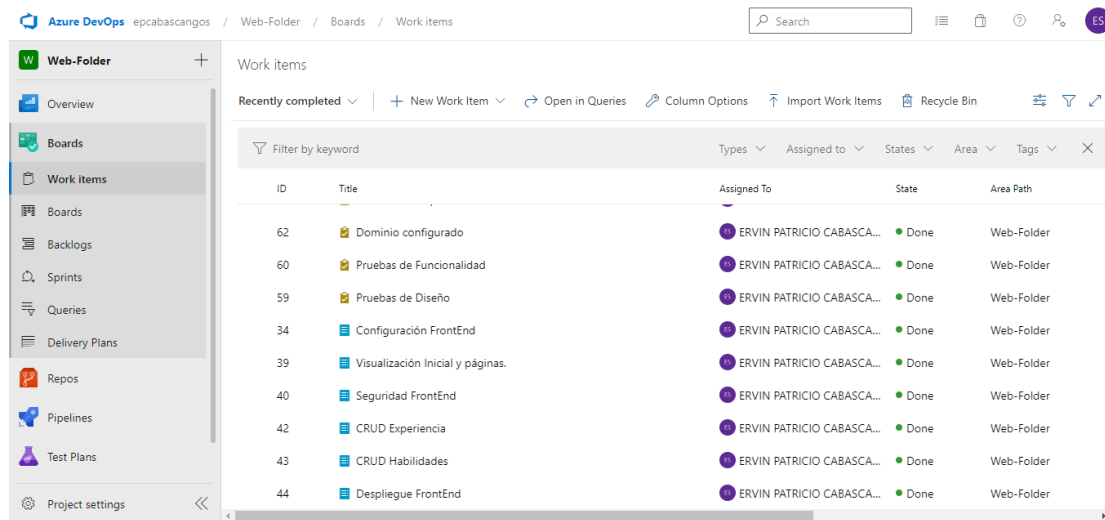
Anexo B: Arquitectura Backend aplicada en la prueba de concepto.



Anexo C: Arquitectura Frontend aplicada en la prueba de concepto.



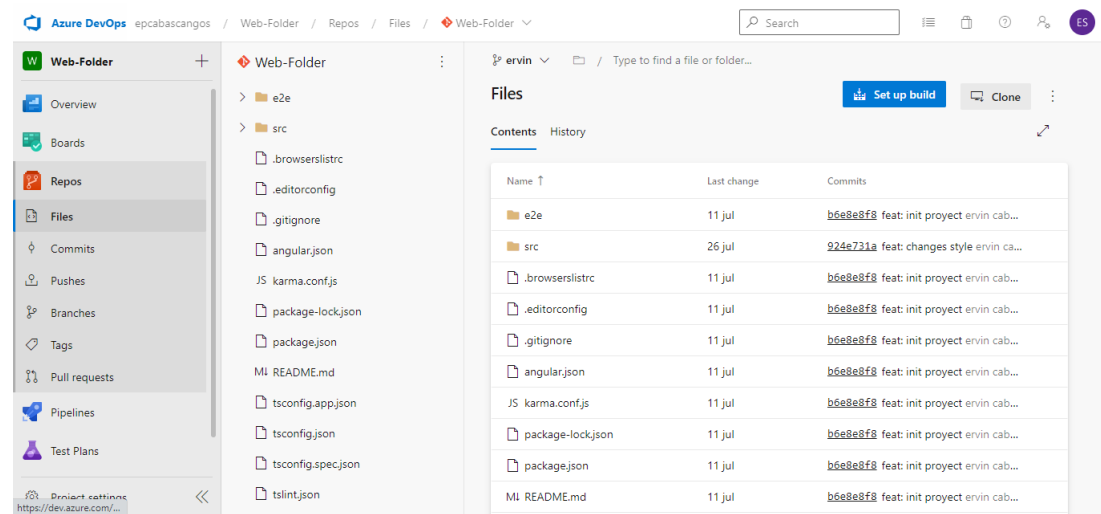
Anexo D: Azure DevOps Boards aplicado en la prueba de concepto.



The screenshot shows the Azure DevOps interface for a project named 'Web-Folder'. The 'Boards' section is active, displaying a Kanban board with a list of work items. The work items are all in the 'Done' state and are assigned to 'ERVIN PATRICIO CABASCA...'. The work items include tasks like 'Dominio configurado', 'Pruebas de Funcionalidad', 'Pruebas de Diseño', 'Configuración FrontEnd', 'Visualización Inicial y páginas.', 'Seguridad FrontEnd', 'CRUD Experiencia', 'CRUD Habilidades', and 'Despliegue FrontEnd'.

ID	Title	Assigned To	State	Area Path
62	Dominio configurado	ERVIN PATRICIO CABASCA...	Done	Web-Folder
60	Pruebas de Funcionalidad	ERVIN PATRICIO CABASCA...	Done	Web-Folder
59	Pruebas de Diseño	ERVIN PATRICIO CABASCA...	Done	Web-Folder
34	Configuración FrontEnd	ERVIN PATRICIO CABASCA...	Done	Web-Folder
39	Visualización Inicial y páginas.	ERVIN PATRICIO CABASCA...	Done	Web-Folder
40	Seguridad FrontEnd	ERVIN PATRICIO CABASCA...	Done	Web-Folder
42	CRUD Experiencia	ERVIN PATRICIO CABASCA...	Done	Web-Folder
43	CRUD Habilidades	ERVIN PATRICIO CABASCA...	Done	Web-Folder
44	Despliegue FrontEnd	ERVIN PATRICIO CABASCA...	Done	Web-Folder

Anexo E: Azure DevOps Repos aplicado en la prueba de concepto.



The screenshot shows the Azure DevOps interface for a repository named 'Web-Folder'. The 'Files' section is active, displaying a list of files and folders. The files include '.browserslistrc', '.editorconfig', '.gitignore', 'angular.json', 'JS karma.conf.js', 'package-lock.json', 'package.json', 'MI README.md', 'tsconfig.app.json', 'tsconfig.json', 'tsconfig.spec.json', and 'tslint.json'. The 'Contents' tab is selected, showing a table of files with their last change date and commit message.

Name	Last change	Commits
e2e	11 jul	b6e8e8f8 feat: init project ervin cab...
src	26 jul	924e731a feat: changes style ervin ca...
.browserslistrc	11 jul	b6e8e8f8 feat: init project ervin cab...
.editorconfig	11 jul	b6e8e8f8 feat: init project ervin cab...
.gitignore	11 jul	b6e8e8f8 feat: init project ervin cab...
angular.json	11 jul	b6e8e8f8 feat: init project ervin cab...
JS karma.conf.js	11 jul	b6e8e8f8 feat: init project ervin cab...
package-lock.json	11 jul	b6e8e8f8 feat: init project ervin cab...
package.json	11 jul	b6e8e8f8 feat: init project ervin cab...
MI README.md	11 jul	b6e8e8f8 feat: init project ervin cab...

Anexo F: Portafolio web desarrollado como prueba de concepto (Inicio).



Ervin Cabascango
Software Developer

Hola, soy Ervin



Soy un desarrollador front-end ubicado en Ecuador. Tengo una gran pasión por el desarrollo de aplicaciones web y móviles.

Soy una persona bien organizada, solucionador de problemas, empleado independiente con gran atención a los detalles. Aficionado a la sci-fi, las actividades al aire libre, las series de TV y la comida.

Estoy interesado en nuevos proyectos y retos. Trabajo en proyectos ambiciosos con personas positivas 🤝

Hagamos algo especial.

Mis habilidades.

Tecnologías para el desarrollo de software en las que he trabajado. Míralas a continuación 📌



Anexo G: Portafolio web desarrollado como prueba de concepto (Experiencia).

Trabajando en la próxima gran idea

Proyectos de clientes y otros trabajos profesionales 🤝



APRISA



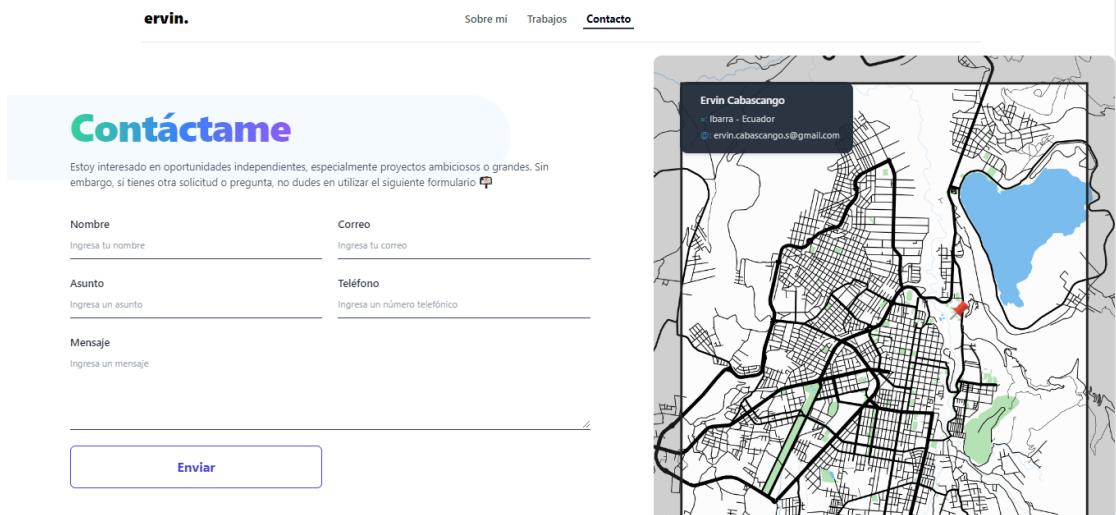
HEIMDALL



GLUK

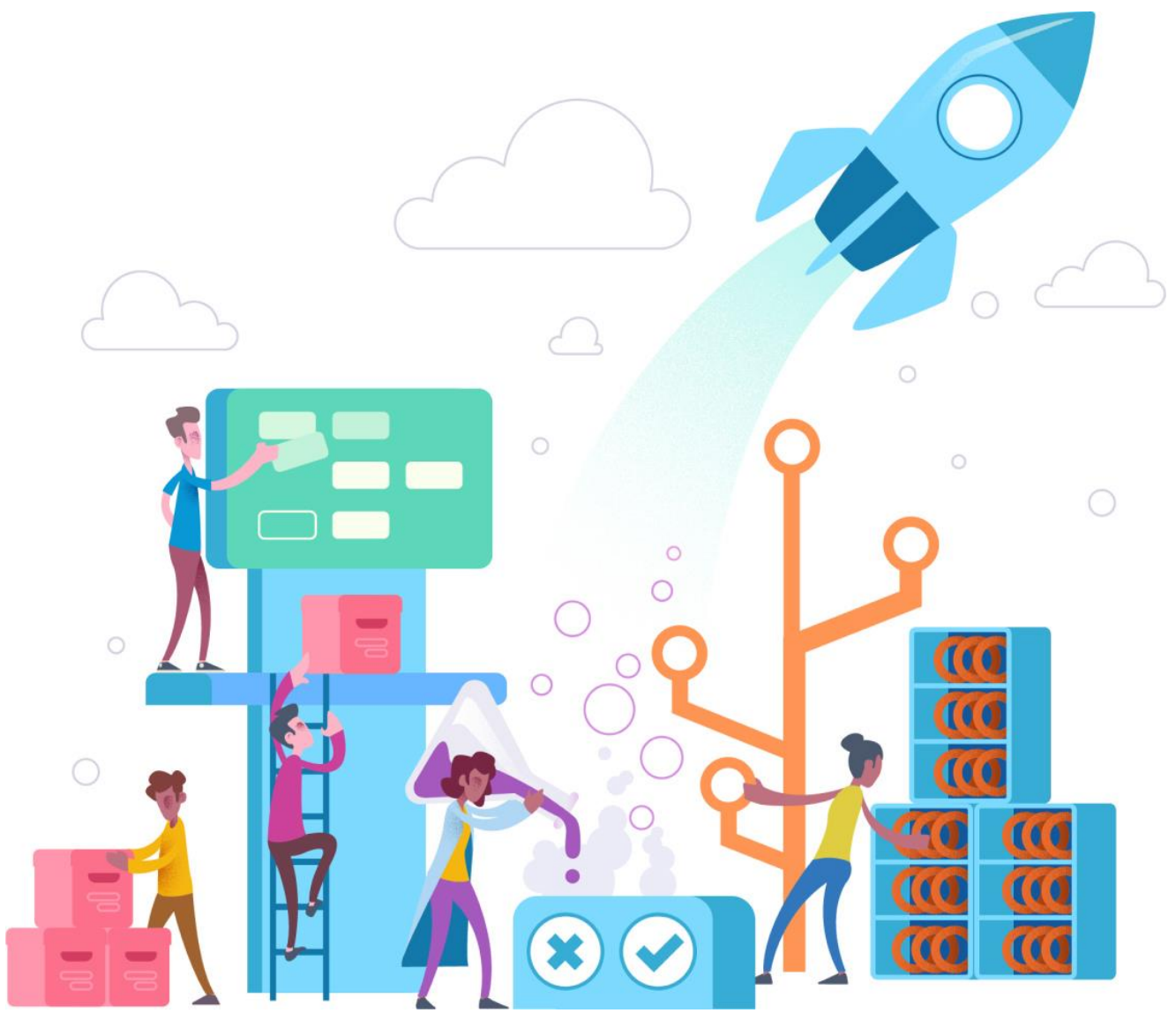


Anexo G: Portafolio web desarrollado como prueba de concepto (Contacto).



GUÍA PARA LA GESTIÓN DE PROYECTOS DE SOFTWARE CON MICROSOFT AZURE DEVOPS BASADO EN MARCO DE TRABAJO SCRUM

Ervin Cabascango / Antonio Quiña / 2022



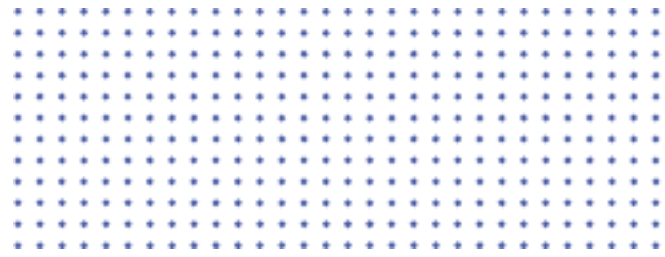


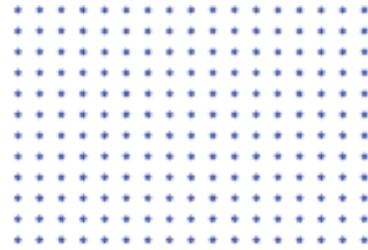
TABLA DE CONTENIDO

INTRODUCCIÓN74

DESARROLLO75

CONCLUSIONES100

INTRODUCCIÓN



Este documento tiene como objetivo principal orientar a los lectores que se enfrentan a la tarea de gestionar el desarrollo de un producto de software con herramientas basadas en la nube.

La guía metodológica a continuación tiene su enfoque en la gestión de proyectos de software basada en el marco de trabajo SCRUM y utilizando la herramienta de Microsoft Azure DevOps, es el resultado de la investigación de conceptos, métodos y herramientas utilizadas en la gestión y el desarrollo del software.

A lo largo del documento se explican los pasos a seguir, partiendo desde la configuración e inicio de un proyecto, el desarrollo del producto de software y culminando en la entrega de este.

Azure DevOps



Azure Boards



Azure Repos



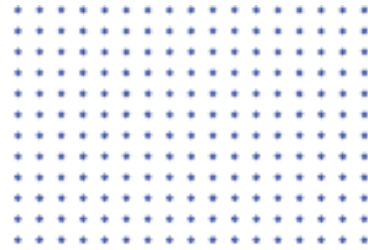
Azure Pipelines



Azure Test Plans



Azure Artifacts



La guía desarrollada a continuación se ha dividido en cuatro secciones basadas en el marco de trabajo SCRUM. La primera sección, denominada “Configuración”, incluye los pasos para crear un proyecto en Microsoft Azure DevOps, invitación a los miembros del equipo, creación de roles y permisos.

La segunda sección: “Pre-Juego”, es el conjunto de pasos necesarios para realizar la gestión de equipos de trabajo y la distribución de las tareas a través de Sprints, las cuales se realizan por los miembros del equipo.

La tercera sección, llamada “Juego”, está diseñada para realizar las tareas asignadas en los Sprints. A demás, describir el uso de las herramientas para gestión de código fuente con las herramientas de Azure DevOps.

Por último, la cuarta sección “Post-Juego”, describe los pasos para realizar pruebas del producto desarrollado para posteriormente realizar el despliegue del aplicativo.

CONFIGURACIÓN

En esta sección se crean y configuran los componentes principales para la gestión de un proyecto enfocado al desarrollo de software, tales como: Creación de proyecto, configuración de proyecto. La configuración del proyecto en base al marco de trabajo SCRUM, se considera como el Sprint Cero.

La gestión del proyecto con las herramientas de Microsoft Azure DevOps se realiza de la siguiente manera:

Creación del proyecto (Microsoft Azure DevOps):

- f) Ingrese al sitio web: <https://azure.microsoft.com/es-es/services/devops/>
- g) Seleccione la opción “Comenzar gratis” para activar una instancia gratuita de Microsoft Azure DevOps, como se muestra en la Fig. 24 a continuación.

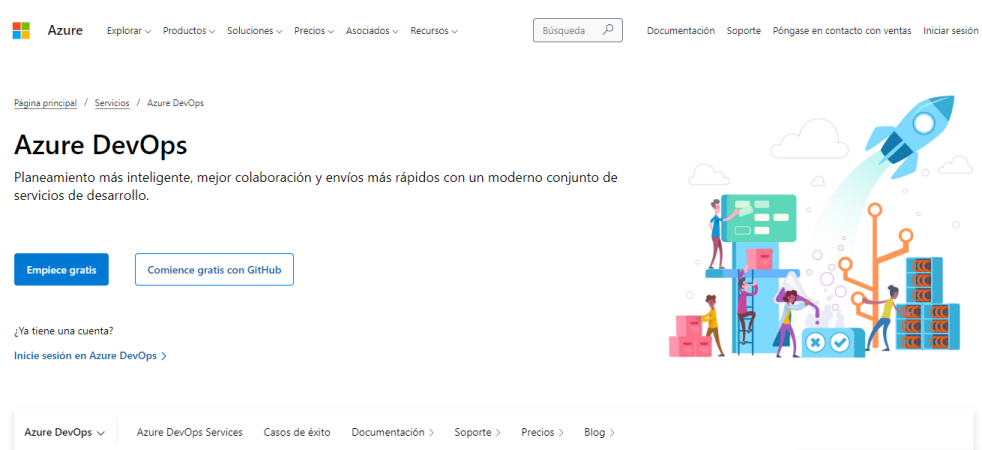


Fig. 93. Página principal Azure DevOps.
Fuente: Propia

- h) Proporcione el nombre de una organización para continuar con la configuración del proyecto.
- i) Cree un nuevo proyecto con los detalles mínimos incluyendo su privacidad (público o privado).

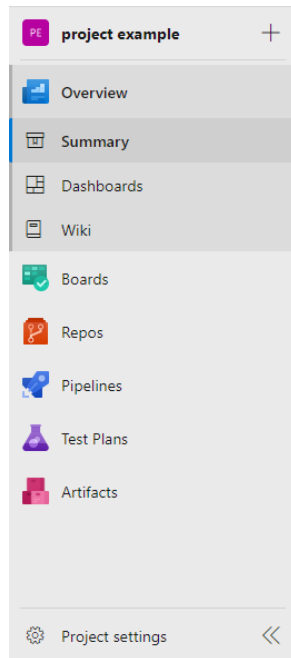


Fig. 96. Suite de herramientas de Azure DevOps.
Fuente: Propia

Partiendo de esta pantalla la guía metodológica propuesta especificará las distintas configuraciones y acciones para la gestión de un proyecto de software.

PRE - JUEGO

La fase de Pre-juego está contemplada en dos etapas Planteamiento y Montaje, las cuales están distribuidas en base a su actividad y resultado, dichas etapas se describen a continuación:

Planteamiento

Es necesario recalcar que el planteamiento de esta fase se trata principalmente de la visión, expectativas y desarrollo del proyecto (Rivadeneira, 2013), es decir, la gestión del proyecto de desarrollo de software, sin tomar en cuenta particularidades tecnológicas como la arquitectura o lenguajes de programación.

Antes de realizar el planteamiento, debemos agregar miembros a nuestro equipo de trabajo, para ellos se deben seguir los siguientes pasos:

- e) Diríjase a las configuraciones (*Project Settings*) del proyecto creado en la sección 3.2.1.1, ingrese a la sección *Teams*.
- f) Por defecto Azure Creará un equipo de trabajo con el nombre del proyecto, elija el equipo respectivo.
- g) Al ingresar al equipo se mostrarán detalles de este, como se muestra en la Fig. 28 pulse el botón *Add*.

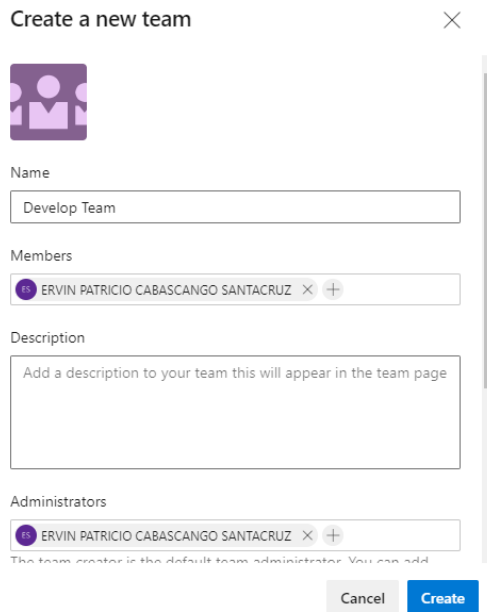


Fig. 99. Crear equipo Azure DevOps.
Fuente: Propia

- i) A continuación, se debe completar la información requerida como: Nombre del equipo, Miembros del equipo, Descripción, Administradores del equipo. Esta información es requerida. Los miembros y administradores de equipo pueden agregarse en cantidad a través de una dirección de correo electrónico.
- j) Una vez creado el equipo podrá ver la información de su equipo como se muestra en la Fig. 31

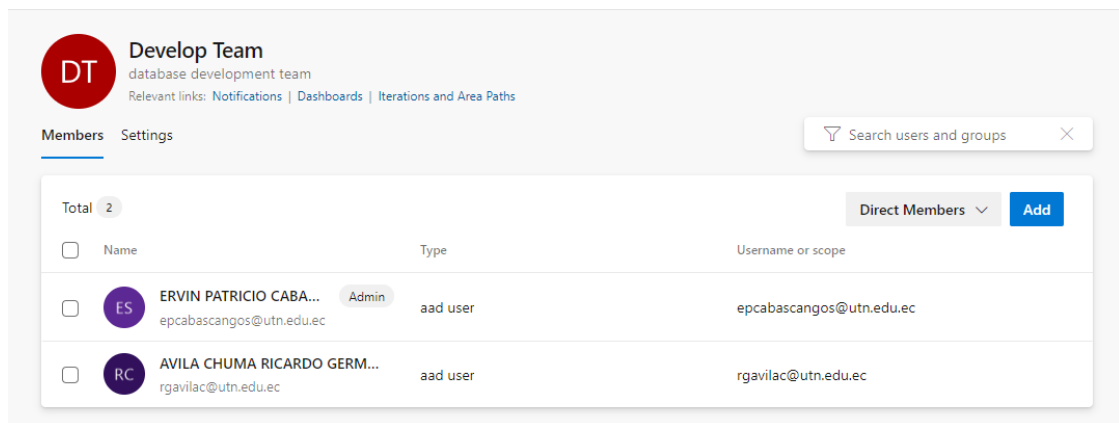


Fig. 100. Usuarios Equipo Azure DevOps.
Fuente: Propia.

Permisos por usuario

La configuración de servicios nos permite dar a los usuarios cierto nivel de acceso a funciones específicas de acuerdo con el rol que cumplen dentro del equipo de trabajo, estos permisos se dividen en secciones tales como: General, Boards, Analytics, Test Plans. Para realizar una configuración de roles por usuario, siga los siguientes pasos:

- e) Diríjase al apartado *Settings/Permissions/Users*. A continuación, la Fig. 32 muestra los usuarios agregados al equipo de trabajo.

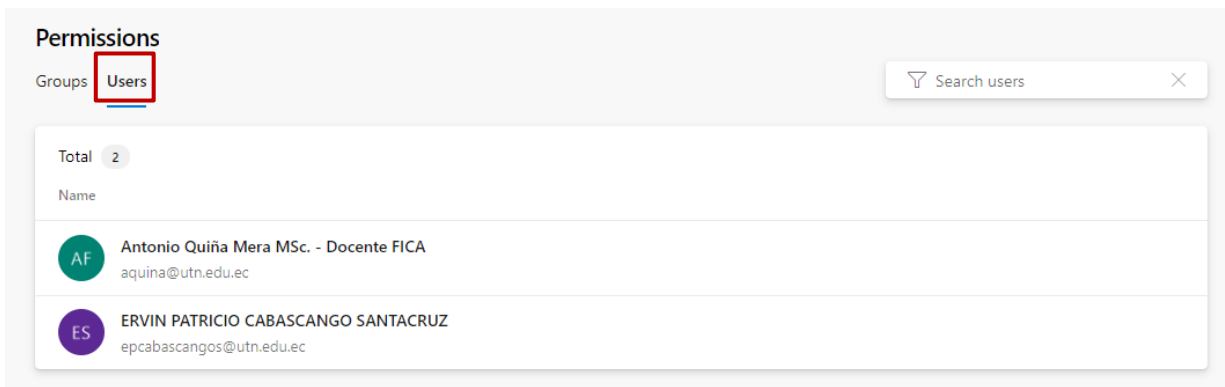


Fig. 101. Permisos Azure DevOps.
Fuente: Propia

- f) Posteriormente, elija un usuario al cual desea realizar configuraciones en sus permisos.
- g) A continuación, en la Fig. 33 muestra la configuración por defecto que Azure DevOps crea para cada usuario.

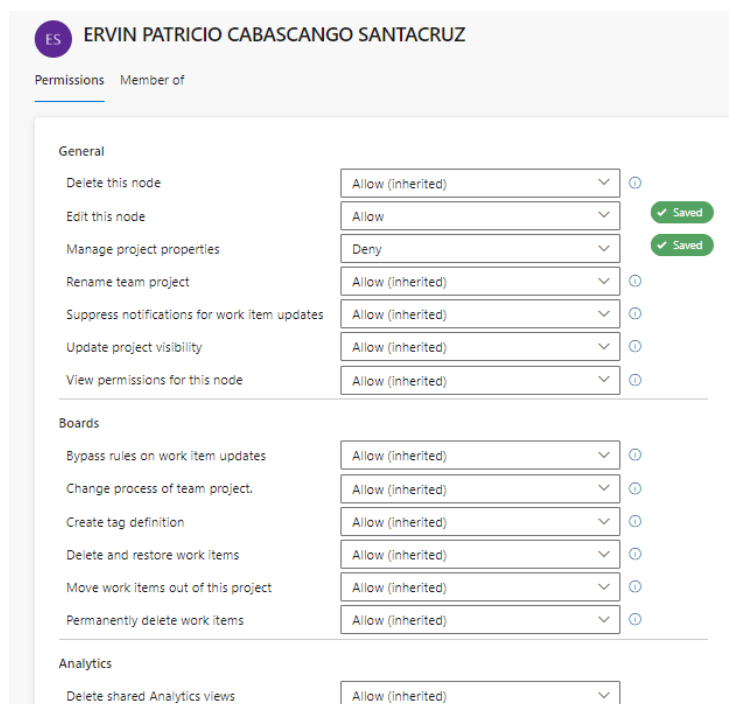


Fig. 102. Clasificación de Permisos Azure DevOps.
Fuente: Propia

- h) Las opciones mencionadas anteriormente permiten al usuario un nivel de acceso a las secciones de Azure DevOps, cada una de ellas puede asignarse a los siguientes estados: Not set, Allow, Deny.

Permisos por Equipo

Los permisos por equipo se realizan de manera similar a los permisos por usuario, A continuación, se detallan los pasos para realizar la configuración:

- e) Diríjase al apartado Settings/Permissions/Groups, dentro de esta sección encontrará contenido como se muestra en la Fig. 34.

Permissions

Groups Users

Total 6 [New Group](#)

Name	Description	Type ↓	Members
Build Administrators	Members of this group can create, modify and delete build definitions and manage queued and completed builds.	Group	0
Contributors	Members of this group can add, modify, and delete items within the team project.	Group	1
Project Administrators	Members of this group can perform all operations in the team project.	Group	1
Project Valid Users	Members of this group have access to the team project.	Group	5
Readers	Members of this group have access to the team project.	Group	0
Web-Folder Team	The default project team.	Team	2

Fig. 103. Permisos Equipo Azure DevOps
Fuente: Propia

- f) Elija el equipo que tenga el nombre del proyecto creado, en este caso *Web-Folder Team*. Las opciones que se muestra son grupos que Azure DevOps crea por defecto.
- g) Una vez dentro, podrá visualizar diferentes paneles que le brindan más información sobre el equipo como se muestra en la Fig. 35.

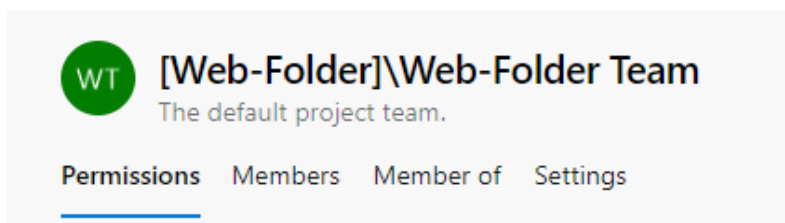


Fig. 104. Opciones de permiso de equipo.
Fuente: Propia

- h) Finalmente, configure los permisos que requiera el equipo similar a la sección **Permisos por usuario**. Es importante mencionar que no todos los equipos tendrán el mismo nivel de acceso a funciones y configuraciones disponibles en Azure DevOps.

Montaje

Esta etapa tiene como objetivo crear el Product Backlog, para definir los requisitos de software de acuerdo con las especificaciones de SCRUM.

Definición del Product Backlog

Azure DevOps permite crear una lista de tareas denominadas Product Backlog, las cuales, son los requerimientos iniciales del cliente.

Product Backlog

- h) Agregue una pila de Backlogs dirigiéndose al apartado *Boards/Backlogs*.
- i) Seleccione el tablero elegido para agregar ítems como se muestra en la Fig. 36.

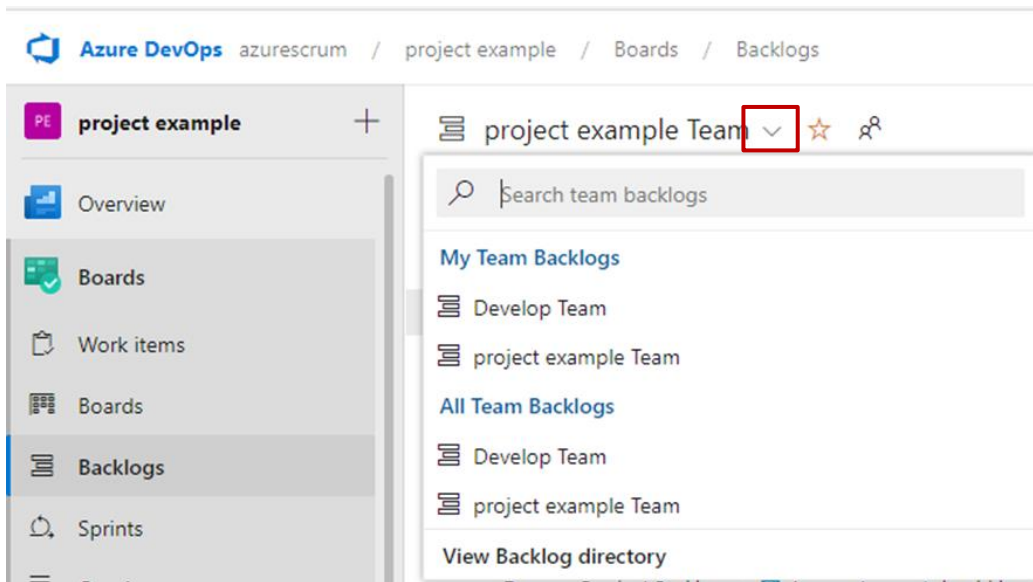


Fig. 105. Tableros de Backlog Azure DevOps.
Fuente: Propia

j) Después, agregue un *New Work Item* y complete la información como se muestra en la Fig. 37.

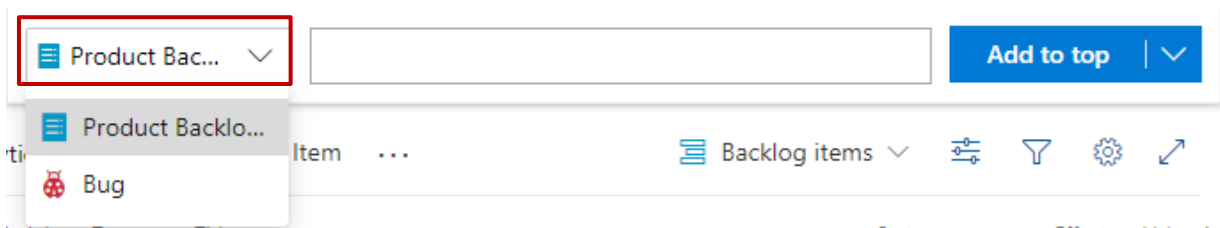


Fig. 106. Tableros de Backlog Azure DevOps.
Fuente: Propia

k) Finalmente se visualizan las actividades agregadas, en base a las necesidades del proyecto.

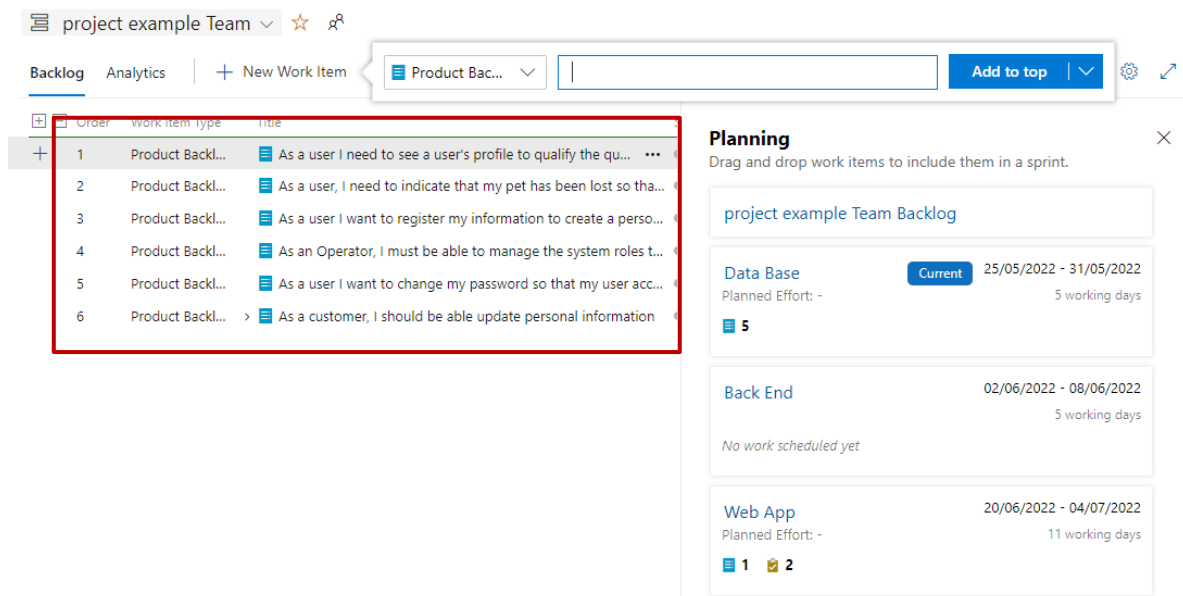


Fig. 107. Backlog Azure DevOps.
Fuente: Propia.

l) Visualice el contenido del backlog a través de los tableros de Azure DevOps en el apartado de *Boards/boards*.

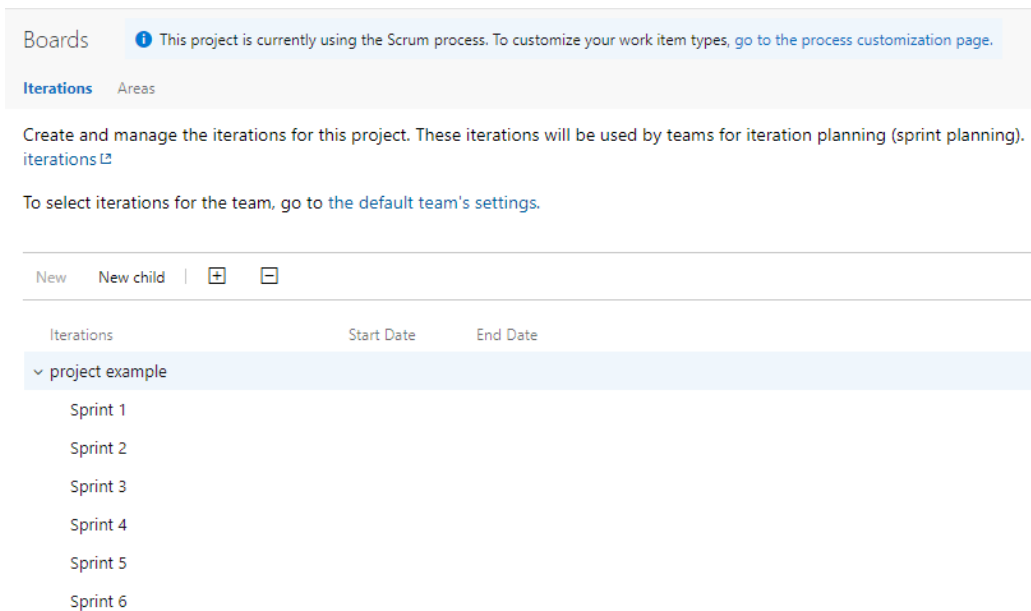


Fig. 110. Sprints del proyecto Azure DevOps.
Fuente: Propia

- g) Configure la información como fecha inicio, fecha fin, nombre de los Sprint.
- h) Si desea, puede crear subniveles para distribuir de mejor manera el Sprint. Para ello seleccione el Sprint y elija la opción *New Child*.
- i) Configure la información de la nueva iteración. Se mostrará información como en la siguiente Fig. 42 .

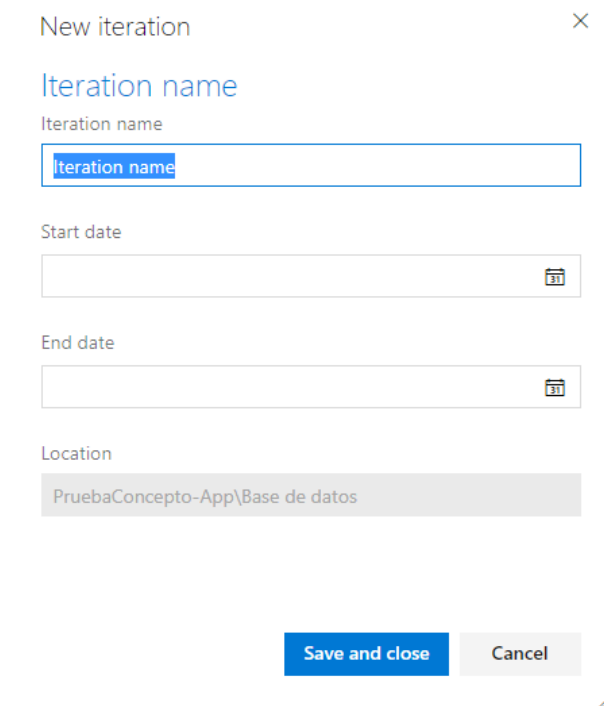


Fig. 111. Agregar iteración.
Fuente: Propia

- j) Finalmente, distribuya los Sprint de acuerdo con las necesidades del proyecto.

Iterations	Start Date	End Date
▼ project example	...	
▼ Data Base	25/05/2022	31/05/2022
Design	25/05/2022	27/05/2022
▼ Back End	02/06/2022	08/06/2022
Database connection	02/06/2022	04/06/2022
Development	06/06/2022	17/06/2022
▼ Web App	20/06/2022	04/07/2022
Desing	20/06/2022	24/06/2022
Development	26/06/2022	04/07/2022
Mobile App	05/07/2022	26/07/2022

Fig. 112. Distribución Sprint Azure DevOps.
Fuente: Propia

Asignación de actividades al Sprint.

El producto Backlog es la parte fundamental de esta sección, una vez se hayan creado correctamente esta lista de requerimiento, estos se mostrarán en la sección de *Boards/Backlogs*. Inicialmente se mostrarán todas las tareas creadas como se muestra en la Fig. 44.

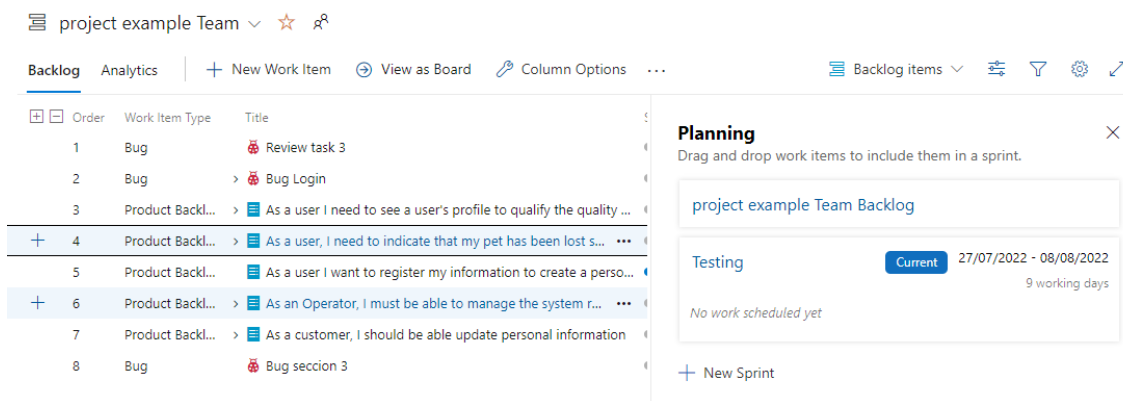


Fig. 113. Pila Backlog Boards.
Fuente: Propia

Una forma rápida de agregar un backlog a un sprint es la siguiente:

- En la parte izquierda de la pantalla se muestra la lista backlog y en la parte derecha los Sprints configurados en la sección *Configuración Sprints*, selecciones un ítem backlog.
- Arrastre un ítem a un Sprint disponible como se muestra en la Fig. 45. Si un Sprint no se visualiza en la sección de *Planning*, probablemente la fecha de ese Sprint ya pasó.

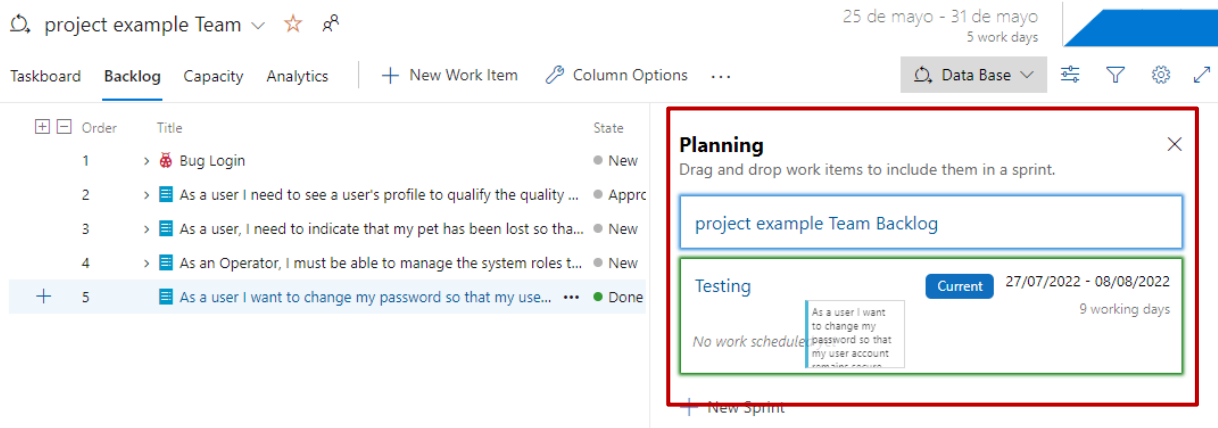


Fig. 114. Asignar Backlog a un Sprint.
Fuente: Propia

Azure nos permite crear un ítem backlog directamente desde un Sprint, para ello realice los siguientes pasos:

- e) Diríjase al apartado de *Boards/Sprints*. Por defecto se situará en el Sprint que se encuentre en ejecución de acuerdo con la configuración de fecha realizada en la sección anterior *Configuración Sprints*.
- f) Si desea crear un backlog en otro Sprint, pulse el botón en la parte superior derecha que tendrá el Sprint actual como se mira en la Fig. 46.

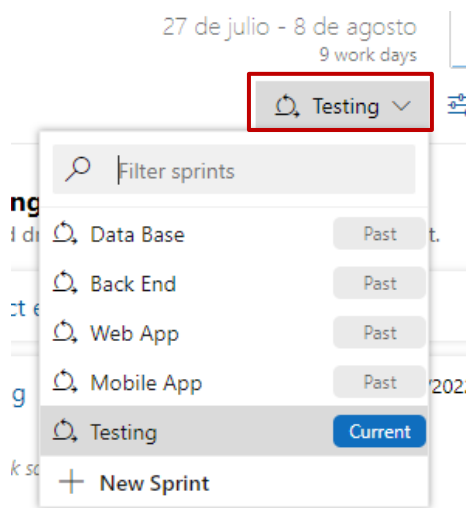


Fig. 115. Seleccionar Sprint.
Fuente: Propia

- g) Una vez seleccionado el Sprint, agregue un nuevo ítem backlog en la parte superior pulse el botón *New Work Item*, ver Fig. 47.

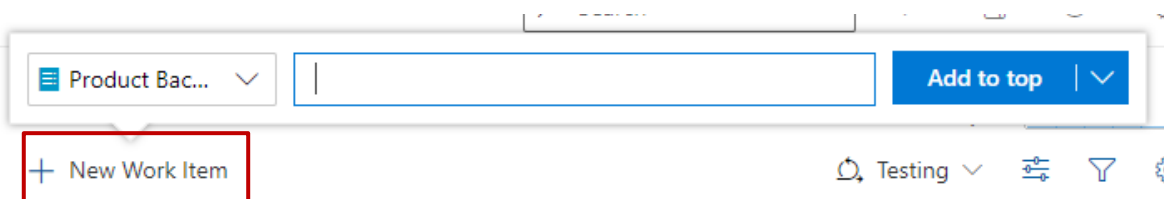


Fig. 116. Agregar ítem Product Backlog.
Fuente: Propia

- h) Finalmente, el nuevo ítem creado se mostrará en la lista de backlog.

JUEGO

Esta fase de SCRUM está enfocada a la ejecución de los Sprints planificados en la fase de **Pre-juego**, de la cual se obtuvo el Product Backlog que fue distribuido acorde a las necesidades del proyecto. En esta sección se abarca el desarrollo del producto de software, lo que conlleva la creación de repositorios para gestión de código fuente, la creación de entornos de desarrollo y la descripción de cada una de estas, en base a las herramientas de Azure DevOps.

Sprints (Boards/Sprints)

La sección Sprints proporciona una vista de los elementos de trabajo planificados para una iteración o sprint, la duración del sprint, el progreso de la finalización del trabajo, el trabajo en equipo en el sprint y otra información relacionada con el sprint, como se muestra en la Fig. 48.

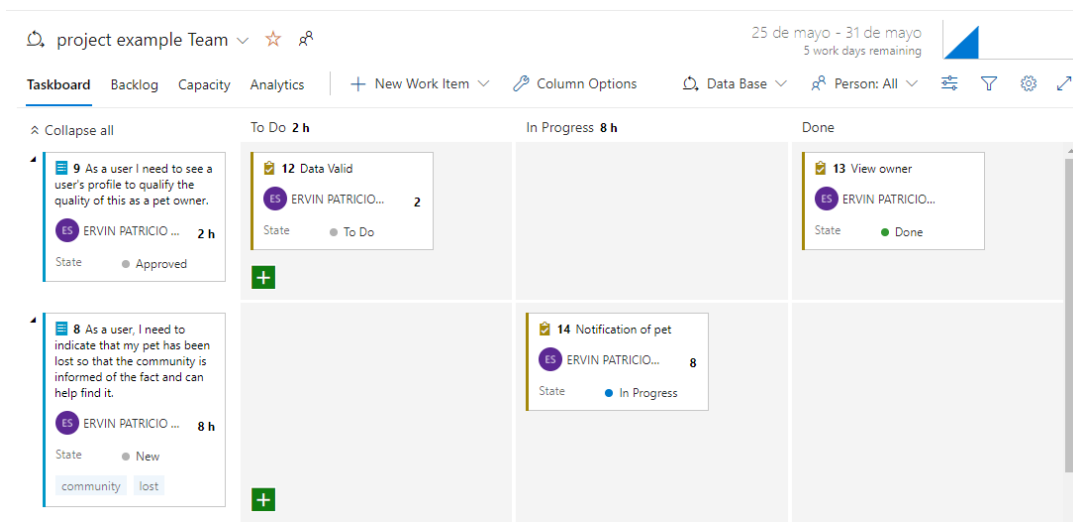


Fig. 117. Sprints Azure DevOps.
Fuente: Propia

Crear Tareas

- g) Seleccione el Sprint en donde quiere se requiere crear la nueva tarea. Cada Sprint contiene tareas diferentes de acuerdo con la distribución realizada anteriormente. A continuación, se muestra la representación en la Fig. 49.

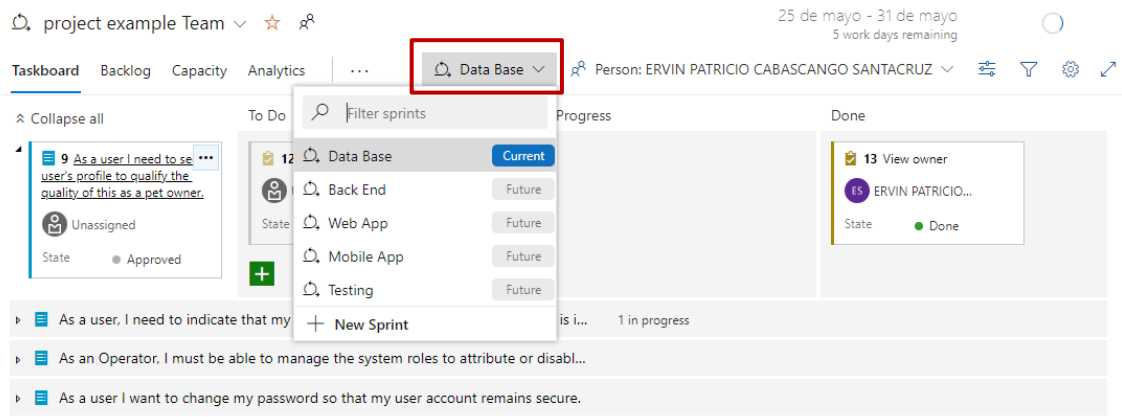


Fig. 118. Sprint por equipo.
Fuente: Propia

- h) Posiciónese en la historia de usuario ha de realizar
- i) Cree una tarea haciendo clic en la opción *New Work Item*.
- j) Seleccione el tipo de tarea.

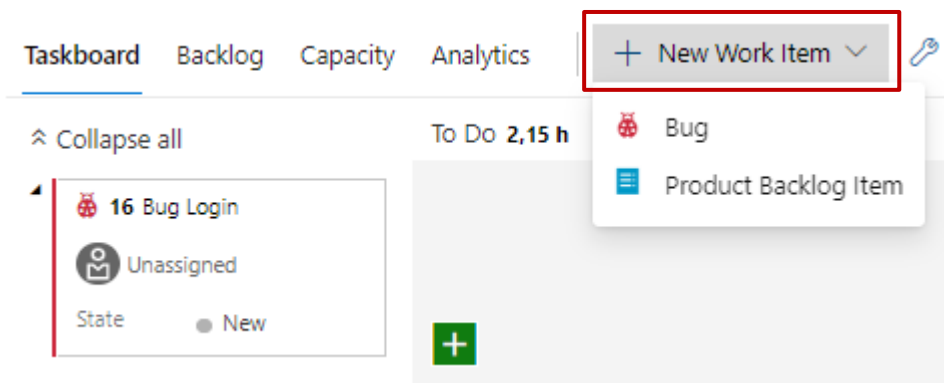


Fig. 119. Nuevo ítem en Sprint.
Fuente: Propia

- k) Adicional configure las características de la tarea como se muestra a continuación en la Fig. 51.

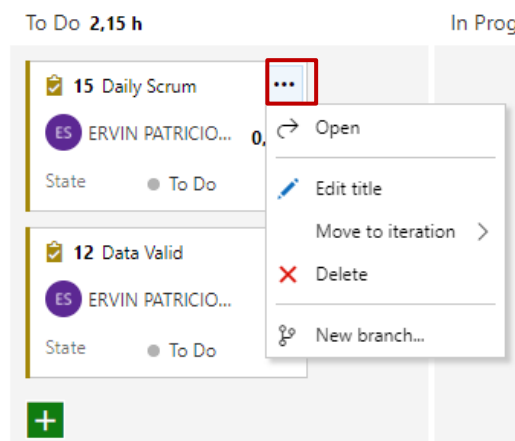


Fig. 120. Ítem de trabajo Sprint.
Fuente: Propia

- l) Puede asignar usuarios a la tarea y agregar información adicional

Revisión Sprint

- d) Asegúrese de estar en el Sprint correspondiente

- e) Arrastre la tarea a uno de los siguientes estados a la tarea: To Do, In Progress, Done.
- f) En caso de presentar un inconveniente puede crear una tarea nueva, para ello dirigirse al apartado **Crear tarea**.

Sprint Backlog

Creación repositorio de código

- d) Diríjase al apartado de Repos
- e) Seleccione la primera opción *Files*. A continuación, se mostrará contenido como en la en la Fig. 52. El contenido mostrado en la ilustración muestra las configuraciones necesarias para subir un repositorio nuevo a uno existente.

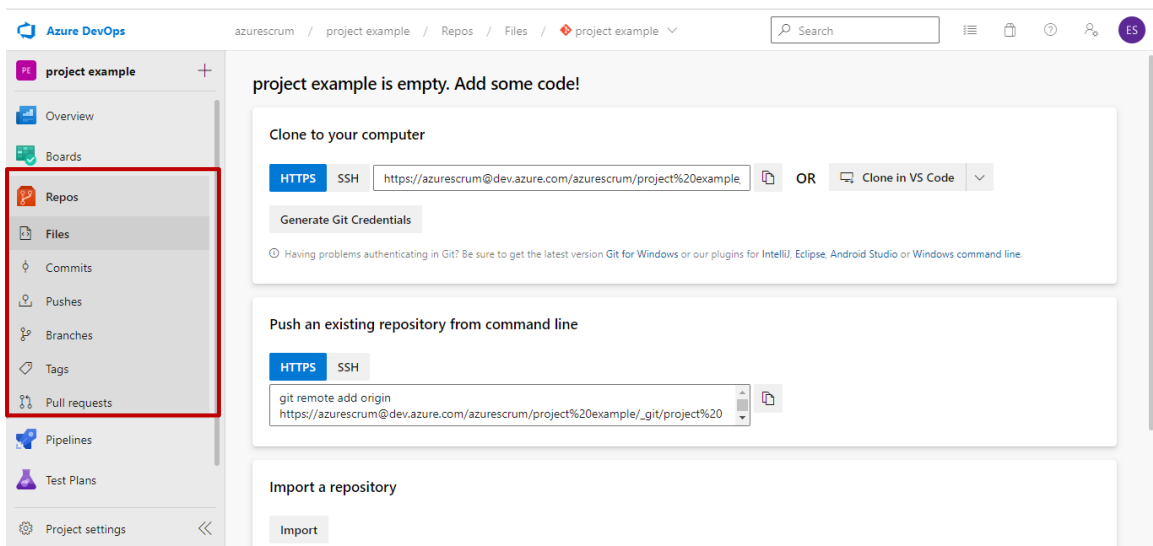


Fig. 121. Repos Azure DevOps.
Fuente: Propia

- f) La siguiente Fig. 53, muestra un proyecto subido en el repositorio de Azure DevOps

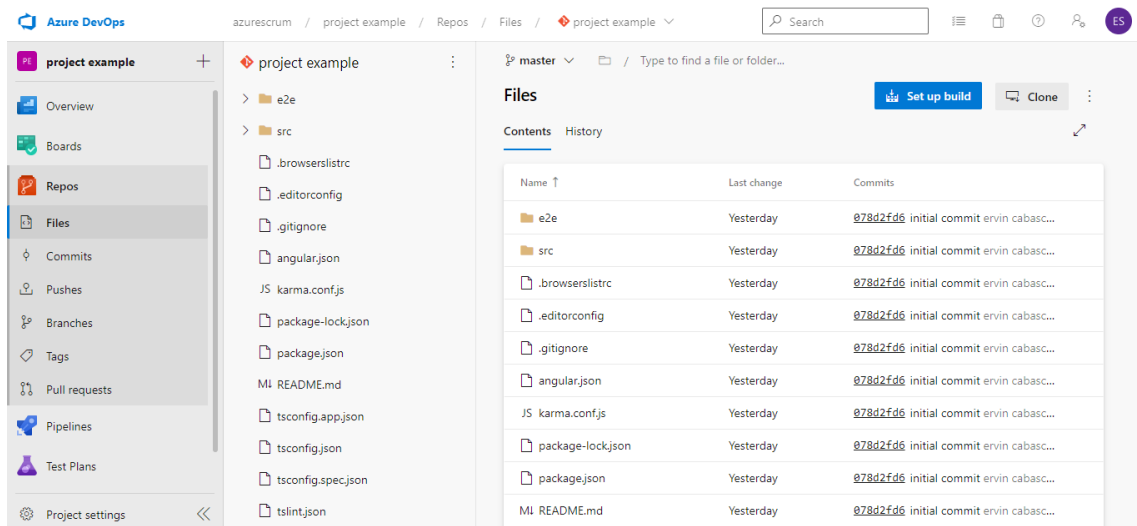


Fig. 122. Azure Repos Files.

Branches

- e) Diríjase a la opción *Branches* dentro del menú de Repos en Azure DevOps
- f) Pulse el botón *New Branch*. A continuación, se desplegará un formulario como se muestra en la Fig. 54.

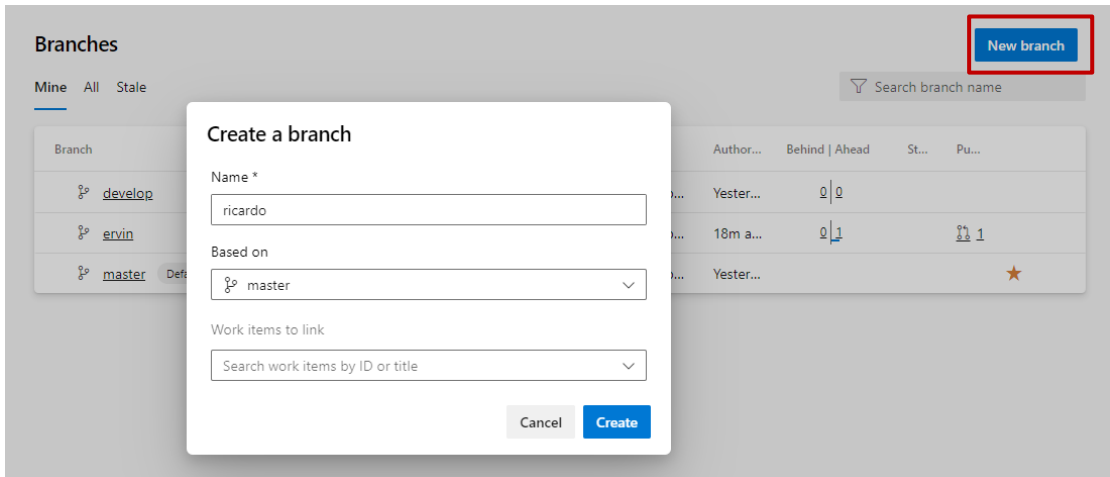


Fig. 123. Crear Rama Azure DevOps.
Fuente: Propia

- g) Complete la información del formulario, se debe tomar en cuenta la base de la cual se creará dicha rama, por defecto Azure DevOps crea una rama llamada master.
- h) Finalmente, en el apartado *Branches* se visualizarán las ramas creadas como se muestra en la Fig. 55 a continuación.

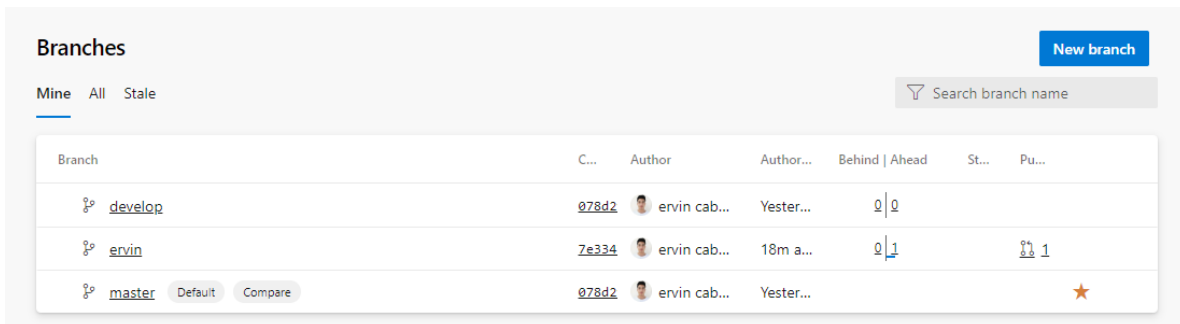


Fig. 124. Ramas en Azure Repos.
Fuente: Propia

Commits

- e) Diríjase a la opción *Commits* dentro del menú de Repos en Azure DevOps
- f) Esta sección nos permite visualizar los cambios que se han realizado por parte del equipo de desarrollo como se muestra en la Fig. 56.

- j) Los cambios realizados se mostrarán en base a los cambios anteriores, como se muestra a continuación:

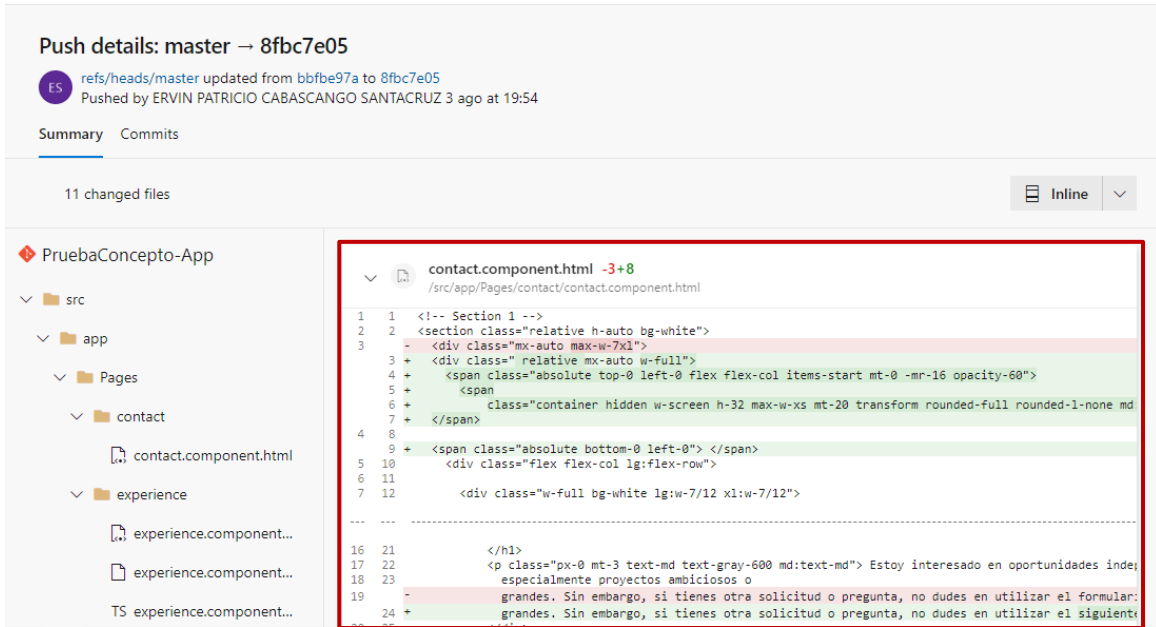


Fig. 128. Detalles de Push Azure DevOps.
Fuente: Propia

Tags

- e) Vaya a la opción *Tags* dentro del menú de Repos en Azure DevOps, os tags nos ayudan a establecer una versión del producto desarrollado.
- f) En la parte superior derecha, pulsamos el botón *New tag*, esto desplegará un formulario para la creación de un nuevo tag, ver imagen Fig. 60.

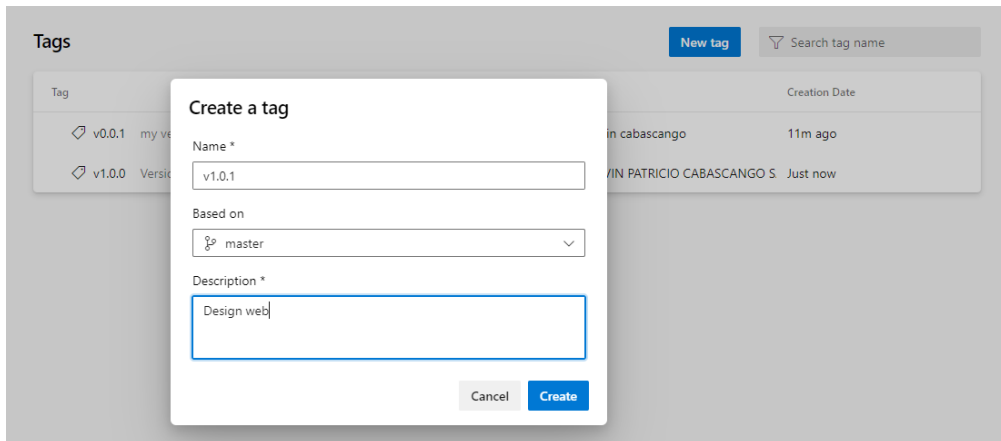


Fig. 129. Crear un Tag Azure Repos.
Fuente: Propia

- g) Complete la información del formulario, debe tener en cuenta que el tag será creado en base a la rama que seleccione.
- h) A continuación, los tags realizados del proyecto se mostrarán con en la Fig. 61.

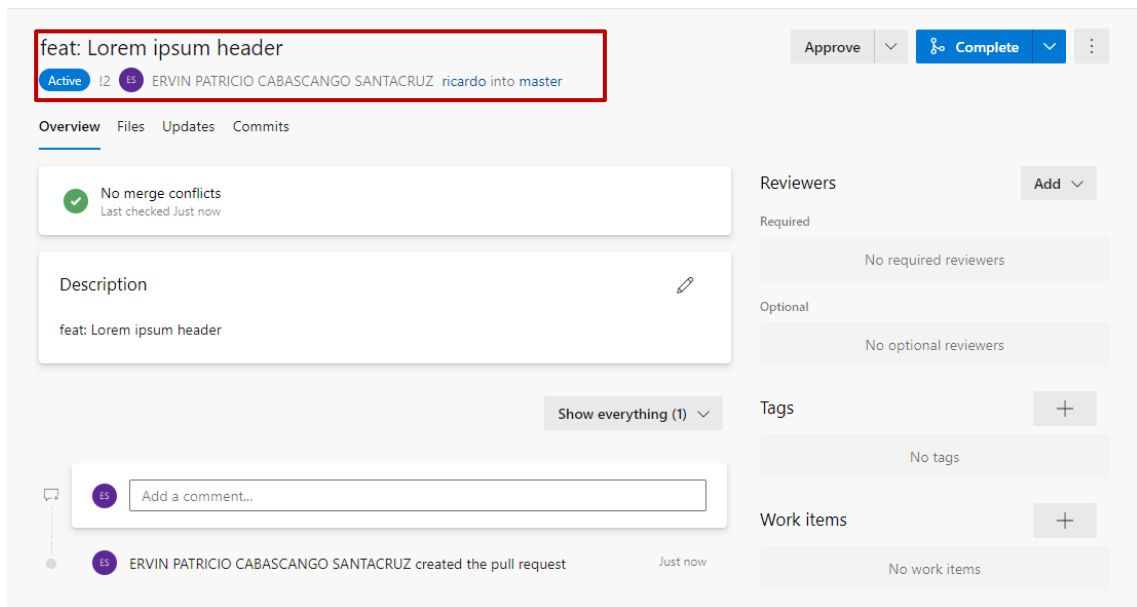


Fig. 132. Fusionar ramas Azure Repos.
Fuente: Propia

j) Las *Pull requests* se pueden crear desde Azure DevOps o por terminal a través de Git

Pruebas de Integración

- j) Cree una prueba directamente desde una actividad creada en los tableros de Azure Boards.
- k) Haga clic el icono de menú en el arte superior derecha de la actividad, ver imagen Fig. 64.
- l) Elija la opción *Add Task* de la lista de opciones.

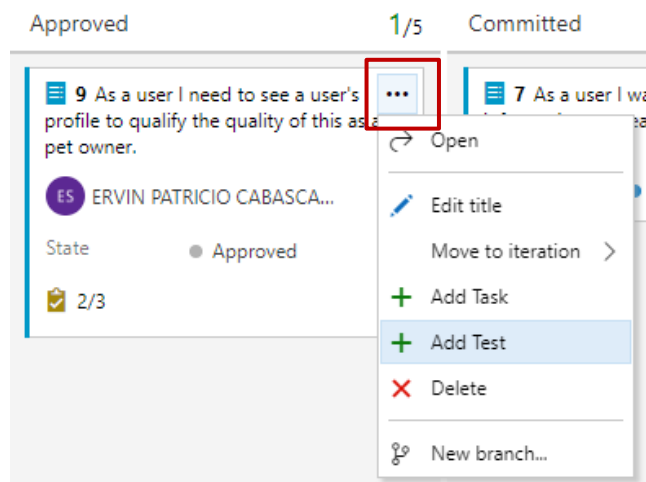


Fig. 133. Agregar Prueba Azure Boards.
Fuente: Propia.

m) A continuación, se debe definir paso y lo que se espera obtener de dicha prueba, además del estado en el que se encuentra la prueba como se mira a continuación en la Fig. 65.

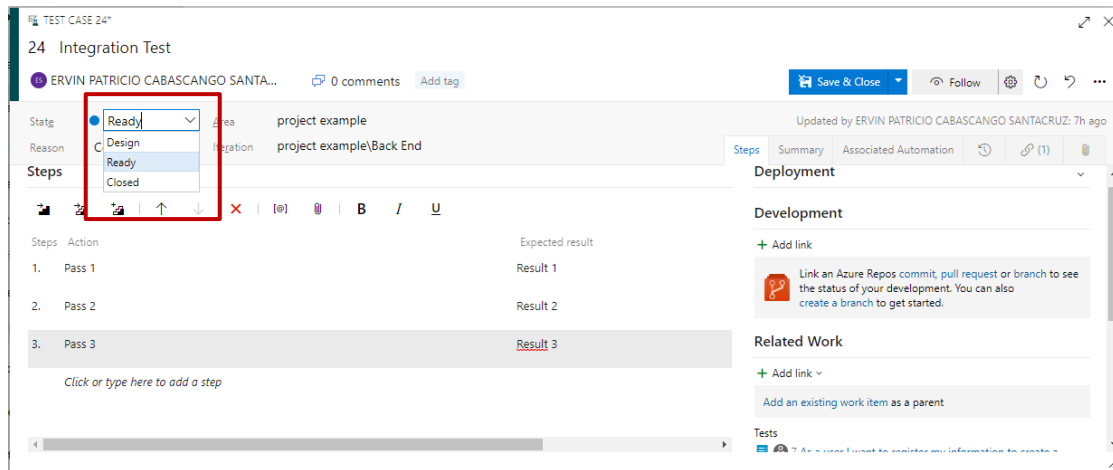


Fig. 134. Configurar una prueba.
Fuente: Propia

- n) Una vez finalizada la configuración de la prueba, procedemos a realizar dicha prueba.
- o) Puede ejecutar la prueba directamente desde las tareas del Backlog, de esta manera podemos verificar el estado de estas con las opciones que se muestran en la siguiente Fig. 66.

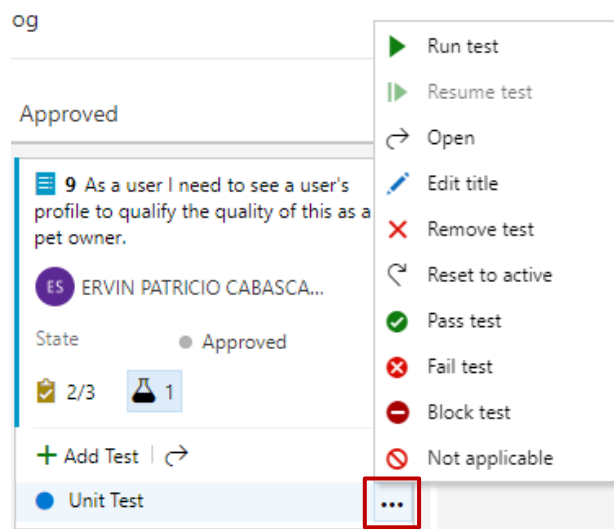


Fig. 135. Pruebas desde ítem de trabajo.
Fuente: Propia

- p) Al ejecutar una prueba, esta desplegará una ventana emergente, en la cual se muestra la configuración realizada anteriormente se nuestra prueba como los parámetros y actividades que debe cumplir dicha prueba, mirar la Fig. 67.
- q) Verifique que las pruebas hayan cumplido con satisfacción, para ello marque con un aprobado o desaprobado paso a paso de la prueba.

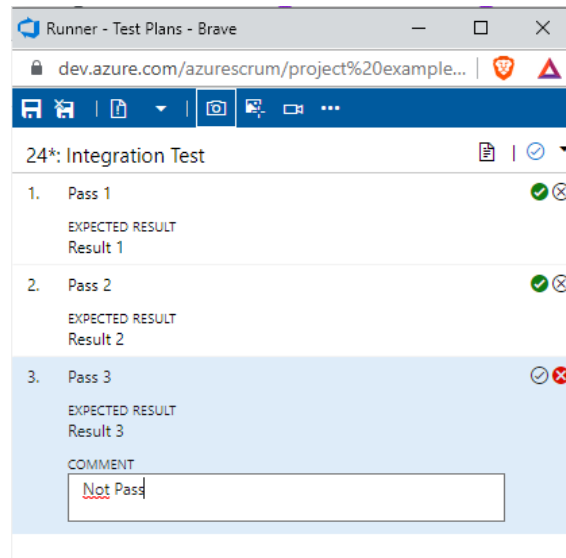


Fig. 136. Caso de Prueba.
Fuente: Propia

r) Para observar las pruebas asignadas al equipo pueden observarse en el apartado de *Tests Plans* de la herramienta Azure DevOps.

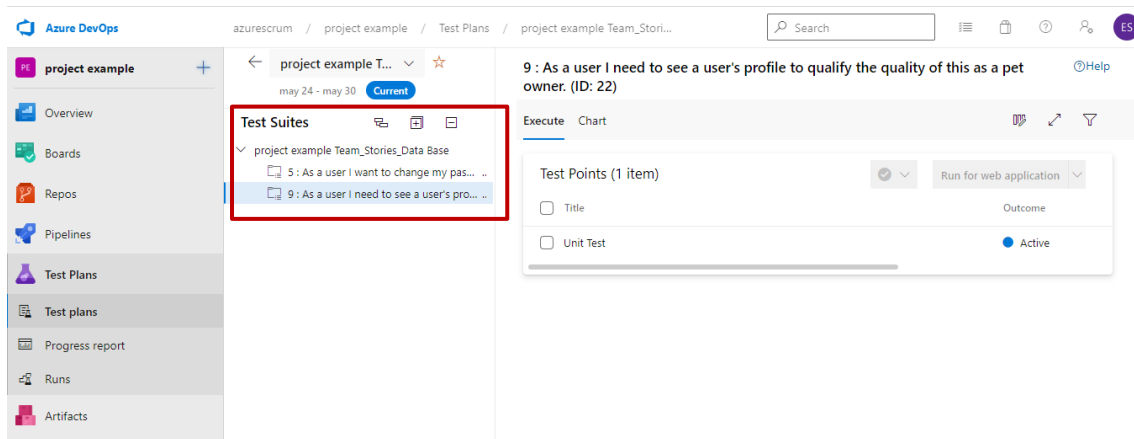


Fig. 137. Tests Plans Azure DevOps.
Fuente: Propia

Las pruebas realizadas se pueden evidenciar en el apartado de *Test Plans/Runs*, las pruebas se distribuyen de acuerdo a su estado. Además, de obtener una visión general de las pruebas a través de gráficos estadísticos como se muestra en la Fig. 70.

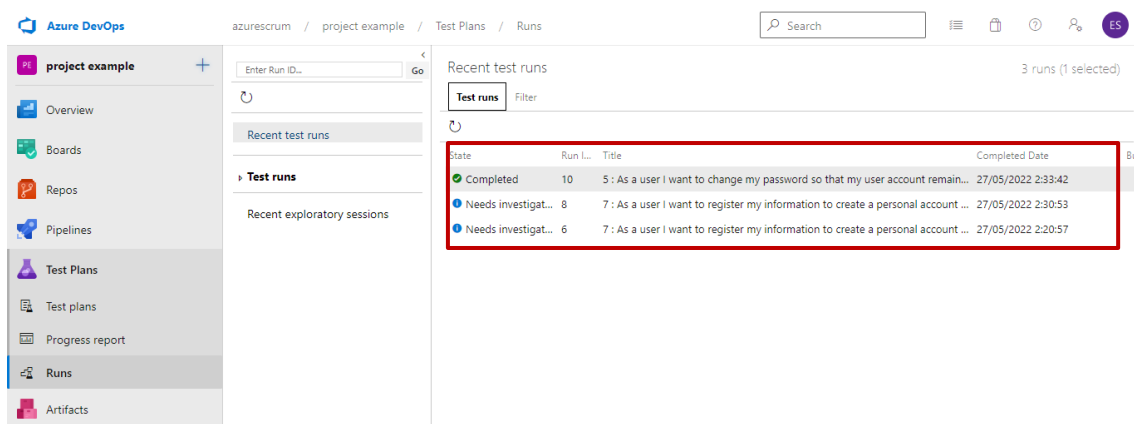


Fig. 138. Test Runs Azure DevOps.
Fuente: Propia

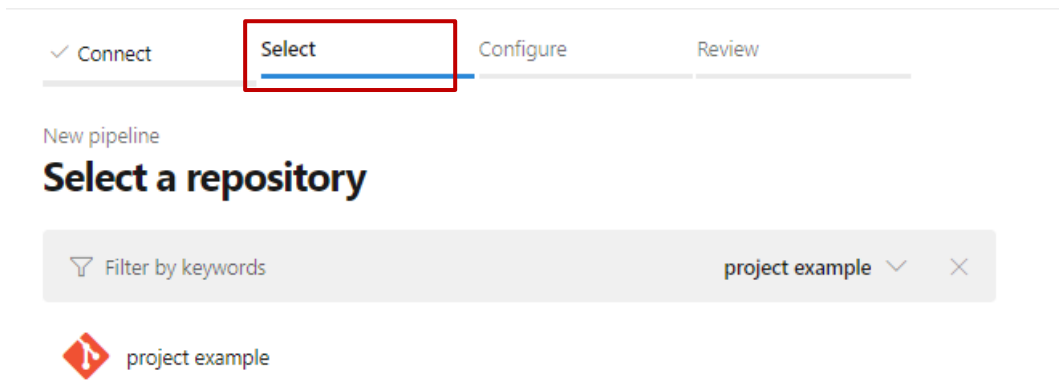


Fig. 141. Selección repositorio en Pipeline.
Fuente: Propia

- k) Seleccionamos el repositorio para su configuración. A continuación, seleccionamos una de las plantillas predeterminadas como se muestra en la Fig. 73, para crear la aplicación.

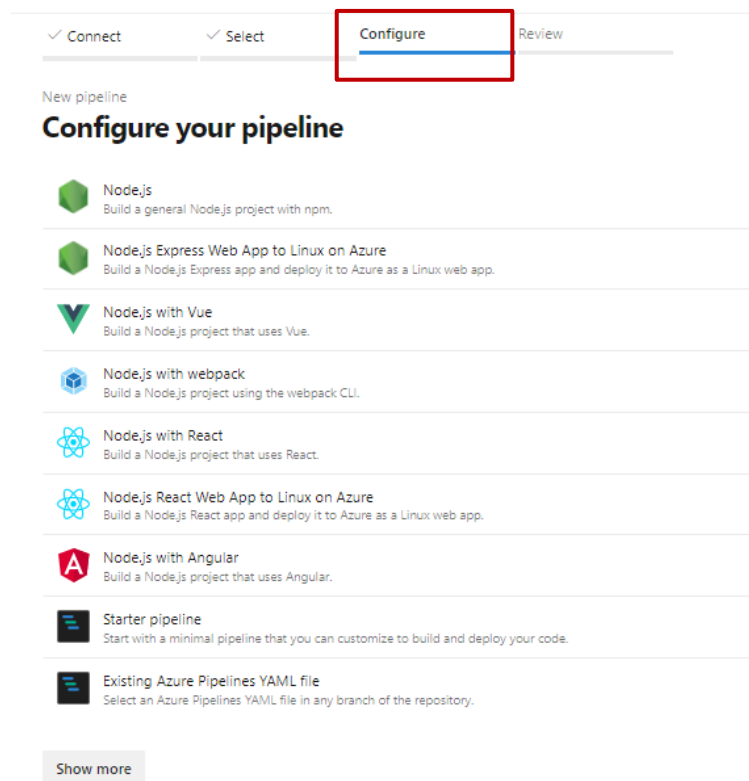


Fig. 142. Configuración Pipeline.
Fuente: Propia.

- l) Finalizada la configuración, revise los detalles como: el repositorio, la tecnología, entre otros. Los detalles se observan en la Fig. 74.

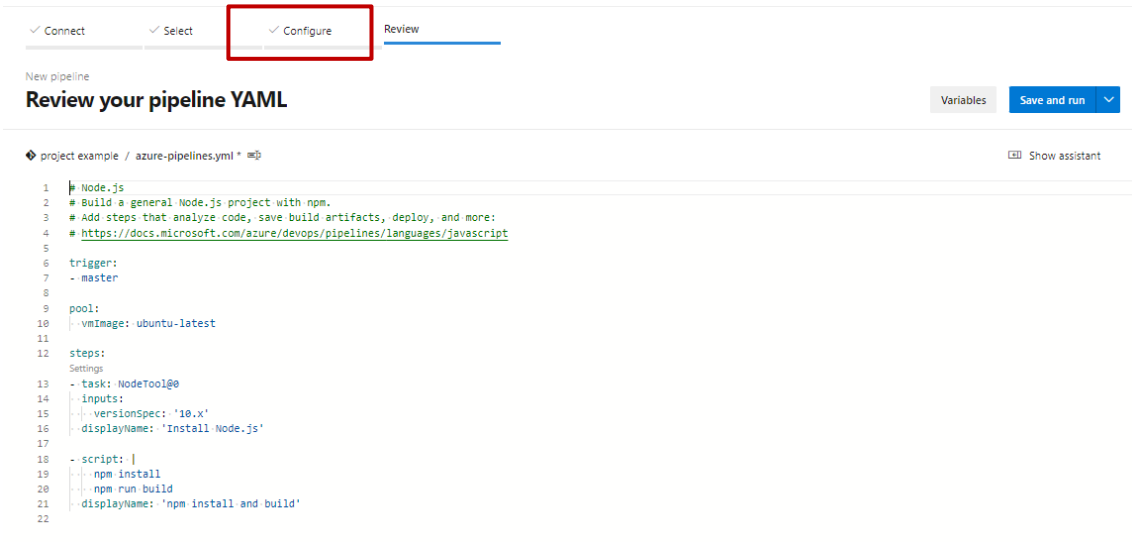
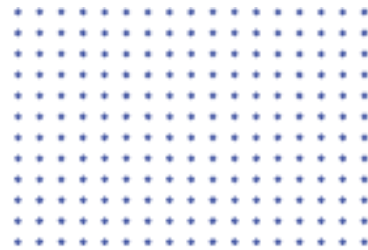


Fig. 143. Revisión de Pipeline.
Fuente: Propia.

CONCLUSIONES



En base a la investigación realizada previamente sobre la elaboración de guías metodológicas, se pudo concluir que, la elaboración de una guía metodológica ordenada y detallada resultaría ser una herramienta de gran utilidad para las personas que se encuentran iniciando en la gestión del desarrollo de software utilizando herramientas para la gestión como Azure DevOps. Esta guía beneficiará a los diferentes equipos de trabajo dentro del desarrollo del proyecto de software.

Por otro lado, es recomendable tener una buena distribución de equipos de trabajo, para que de este modo la configuración de un proyecto se pueda realizar de manera más óptima.

Finalmente, la guía metodológica está desarrollada en base al marco de trabajo Scrum, para lo cual hay que tomar en cuenta si se desea trabajar con otro tipo de metodología, pero con las herramientas de Azure DevOps.