FACULTAD DE POSTGRADOS

FORTALECIMIENTO DE LA FÁBRICA DE SOFTWARE IN HOUSE DE LA UNIVERSIDAD DE LAS AMÉRICAS

AUTORES

Cristóbal Geovanni Gómez Carrera
Diego Javier Viñamagua Quezada

AÑO

2022

FACULTAD DE POSTGRADOS

FORTALECIMIENTO DE LA FÁBRICA DE SOFTWARE IN HOUSE DE LA UNIVERSIDAD DE LAS AMÉRICAS

"Trabajo de Titulación presentado en conformidad con los requisitos establecidos para optar por el título de MÁSTER EN GERENCIA DE SISTEMAS Y TECNOLOGÍA EMPRESARIAL"

AUTORES

Cristóbal Geovanni Gómez Carrera

Diego Javier Viñamagua Quezada

AÑO

2022

DECLARACIÓN DEL PROFESOR GUÍA

"Declaro haber dirigido el trabajo, Fortalecimiento de la fábrica de software in house de la Universidad de Las Américas, a través de reuniones periódicas con los estudiantes Cristóbal Geovanni Gómez Carrera y Diego Javier Viñamagua Quezada, en los talleres del trabajo Capstone, orientando sus conocimientos y competencias para un eficiente desarrollo del tema escogido y dando cumplimiento a todas las disposiciones vigentes que regulan los Trabajos de Titulación".

_____

Germán Ernesto Pancho Carrera
061918253

DECLARACIÓN DE AUTORÍA

“Declaramos que este trabajo es original, de nuestra autoría, que se han citado las fuentes correspondientes y que en su ejecución se respetaron las disposiciones legales que protegen los derechos de autor vigentes.”

Cristóbal Geovanni Gómez Carrera
CI. 0502978323

Diego Javier Viñamagua Quezada
CI. 1717705451

AGRADECIMIENTOS

Queremos agradecer a nuestros estimados docentes que nos compartieron su conocimiento a lo largo de nuestros estudios de maestría.

Así mismo expresamos nuestro más sincero agradecimiento al director del Área de Tecnología de UDLA Lenin Landázuri por el apoyo y la confianza brindada para poder iniciar y culminar exitosamente este programa académico.

Se el cambio que deseas ver en el Mundo – Mahatma Gandhi.

DEDICATORIA

El presente trabajo es resultado de muchas horas de esfuerzo y sacrificio que no es solo mío, sino de las personas que están junto a mí. Quiero dedicar enteramente este proyecto a mi amada esposa quién estuvo apoyándome para no rendirme tras las largas jornadas laborales, mi hermana que con su conocimiento brindó grandes ideas, a mi mamá por siempre estar pendiente de mis estudios.

Finalmente, a la memoria de todos mis seres queridos que han partido de este mundo, en especial a mi querido abuelito don Cristóbal Gómez Bonilla.

Gracias a Dios por las bendiciones recibidas.

Cristóbal Gómez Carrera

DEDICATORIA

Este proyecto va dedicado a mis seres queridos que me han apoyado en todo momento y a lo largo de mis estudios.

Diego Viñamagua Quezada

RESUMEN

En el presente trabajo de titulación se realizará un análisis de la situación actual de la fábrica de *software* de la Universidad de Las Américas en referencia a los procesos llevados a cabo en el ciclo de vida de desarrollo *software*.

En la actualidad, el principal servicio que oferta la fábrica de *software* a los clientes internos es el desarrollo de soluciones tecnológicas a la medida que apalancan los procesos administrativos y académicos de UDLA, esto a través de un portafolio de aplicaciones informáticas que en algunos casos presentan retrasos en los tiempos de entrega, problemas al momento de su ejecución en los ambientes productivos, falta de herramientas para la medición de rendimiento, entre otros; lo que genera inconvenientes en la entrega de valor al cliente.

El principal objetivo del trabajo es identificar y proponer mejoras en los procesos junto con herramientas tecnológicas que contribuyan a optimizar el desarrollo de *software* y de esta manera solventar los inconvenientes previamente indicados.

Al tratarse de un problema empresarial complejo, durante el análisis se aplicará el método ADM de TOGAF para abordar sistemáticamente la identificación de problemas, los marcos de trabajo, buenas prácticas y referentes de la industria a ser usados.

ABSTRACT

In this degree work, an analysis of the current situation of the software factory of the Universidad de Las Américas will be performed in reference to the processes carried out in the software development life cycle.

Currently, the main service offered by the software factory to internal customers is the development of customized technological solutions that leverage the administrative and academic processes of UDLA, this through a portfolio of software applications that in some cases have delays in lead times, problems in productive environment at the moment of its execution, lack of tools for performance measurement, among others; which generates problems in delivering value to the customer.

The main objective of the work is to identify and propose improvements in the processes together with technological tools that contribute to optimize software development and thus solve the previously mentioned problems.

As this is a complex business problem, the TOGAF ADM method will be applied during the analysis to systematically address the identification of problems, frameworks, best practices and industry benchmarks to be used.

Contenido

Índice de tablas

Índice de Figuras

1 Fase Preliminar

1.1 Contexto e influenciadores

La Universidad de Las Américas es una entidad de educación superior que se encuentra ubicada en Quito. En su oferta académica contempla carreras de pregrado, posgrados y educación continua en modalidades presencial, virtual e híbrida. Lo que permite a los estudiantes a nivel nacional para que puedan iniciar, cursar y finalizar su formación académica.

Forma parte de un consorcio educativo conformado por tres universidades junto con la Universidad Latina y la Universidad Americana que se encuentran en Costa Rica. El objetivo de esta alianza estratégica es compartir modelos de operación, procesos y tecnología para fortalecer las actividades académicas.

1.1.1 Organigrama de UDLA

El modelo de gobierno de la Universidad de Las Américas plantea el siguiente organigrama:

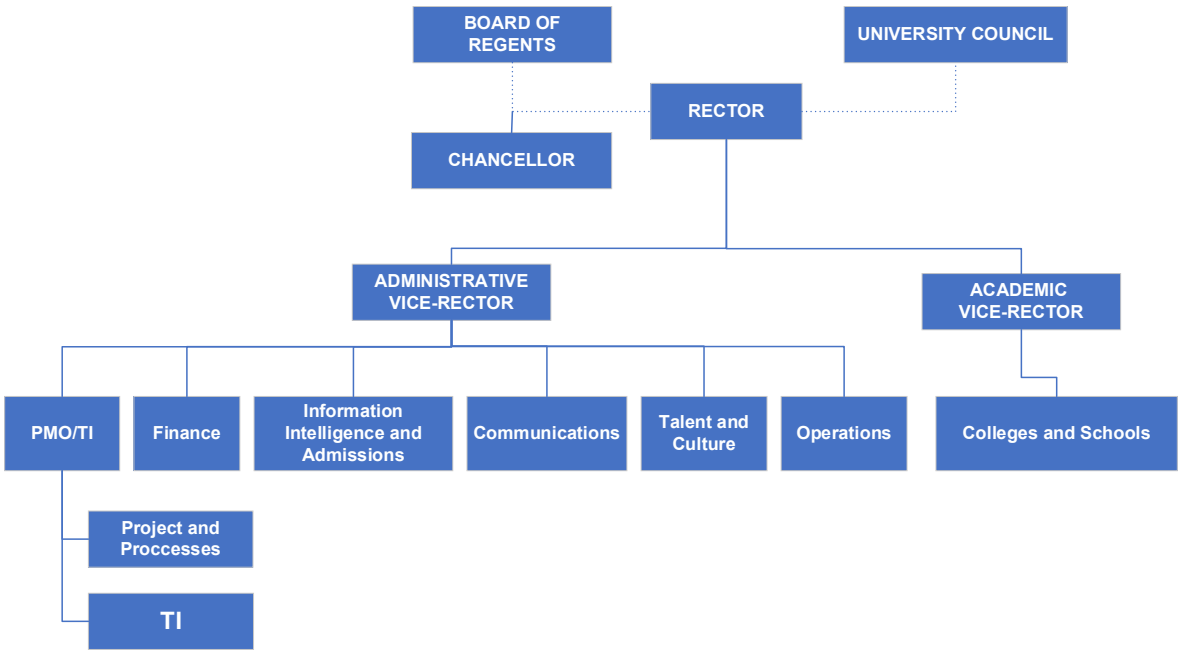


Figura 1.1 Organigrama UDLA (Universidad de Las Américas, 2020)

Como se muestra en la **Figura 1.1** la Universidad tiene como autoridad máxima a un órgano colegiado académico superior, denominado Consejo Universitario que es presidido por el Rector de la Universidad. Nuestras autoridades institucionales son:

- El Rector
- El Vicerrector Académico
- El Vicerrector Administrativo

(Universidad de Las Américas, 2020)

De acuerdo con la Visión 2020-2025 de la Universidad de Las Américas.

> Ser un modelo de referencia en la educación superior ecuatoriana, que sirva a un público amplio y diverso a través de la excelencia académica, la gestión de calidad y el servicio excepcional, con tecnología de vanguardia. Generar, principalmente, conocimiento relevante para el desarrollo del país. (Universidad de Las Américas, 2020)

De esto se denota que para UDLA la tecnología es una pieza fundamental para apalancar todos sus procesos, siendo de esta manera TI (Tecnología de Información) un área estratégica dentro de la organización.

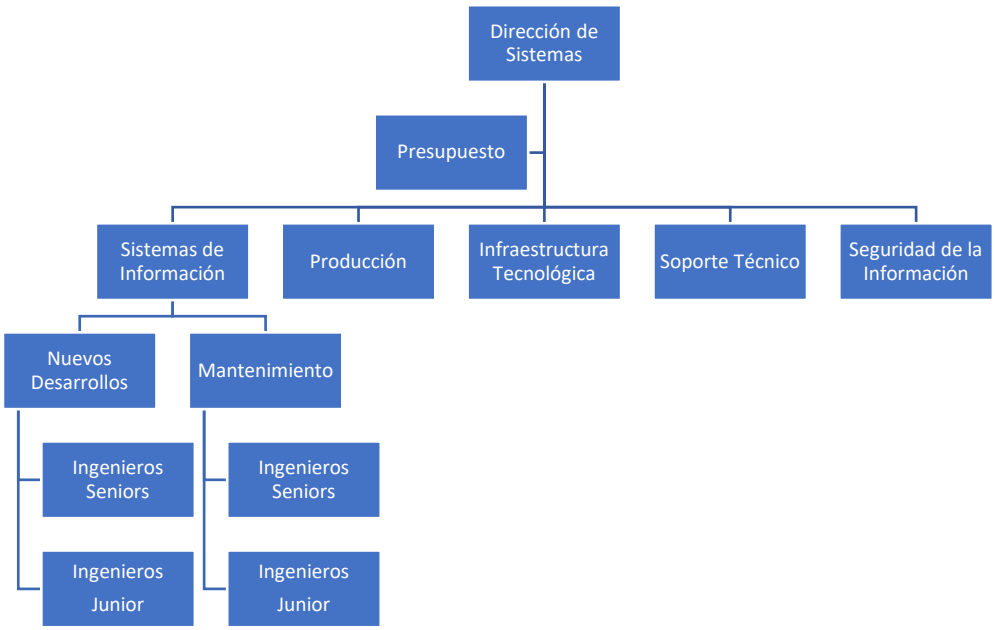


Figura 1.2 Organigrama TI – UDLA

Dentro del organigrama general se encuentra el área de Tecnología de la Información que está conformada como se muestra en la **Figura 1.2**

Donde cada área tiene las siguientes funciones:

- **Sistemas de Información:** Es la fábrica de *software* de UDLA en donde se realizan todos los desarrollos e implementaciones de aplicaciones que la universidad requiere.
- **Producción:** Se encarga de administrar y velar que las aplicaciones desplegadas en los ambientes productivos trabajen de manera eficiente.
- **Infraestructura Tecnológica:** Administra los elementos de *hardware*, *software* base, sistemas operativos y red que son necesarios para gestionar los entornos de TI.
- **Soporte Técnico:** Brindan asistencia relacionada a *hardware*, redes y aplicaciones informáticas a la comunidad universitaria.
- **Seguridad de la Información:** Protege a la organización de posibles ataques informáticos que pueden poner en riesgo a la operación de la organización.

En el área de Sistemas de Información al año 2022, trabajan 24 personas distribuidos en los siguientes roles y funciones:

- **Jefe de Sistemas de Información:** Es el líder del área y cumple funciones como administración, control y gestión de los recursos del área. Total: 1.
- **Coordinación:** Son líderes de las sub-áreas de Nuevos Requerimientos y Mantenimiento. Realizan funciones de arquitectura de soluciones. Total: 2.
- **Ingeniero senior:** Realizan actividades de documentación, análisis y desarrollo de *software*. Propone soluciones innovadoras basadas en tecnología y mejores prácticas. Total: 9.
- **Ingeniero junior:** Realizan actividades de análisis y desarrollo de *software*. Total: 7.
- **Pasantes:** Realizan actividades de desarrollo de *software* guiados por los otros roles. Total: 5.

Existen dos líneas de desarrollo dentro del área de Sistema de Información:

- **Nuevos desarrollos**: Su objetivo es trabajar en proyectos de desarrollo *software* cuya duración sea mayor a un mes.
- **Mantenimiento:** Se encarga de proyectos de hasta un mes de duración. Así como el desarrollo de nuevos requerimientos en sistemas existentes y resolución de *bugs*.

Durante el año 2021, el número de desarrollos realizados fueron los siguientes:

- **Proyectos nuevos:** 12
- **Requerimientos:** 136

Para los desarrollos, se cuentan con dos ambientes definidos para cada aplicación.

- **Pruebas:** Ambientes que se usan para validar los desarrollos realizados por los analistas. Dependiendo de la aplicación, puede existir más de un ambiente de pruebas debido a que varias personas pueden trabajar sobre diferentes requerimientos en la misma aplicación.
- **Producción:** Ambientes productivos en donde se despliegan las aplicaciones certificadas para que los usuarios del sistema puedan usar sus funcionalidades.

Actualmente, se cuenta con servidores *on-premises* que administra el área de Infraestructura Tecnológica y los ambientes de pruebas y/o producción se crean en estos servidores.

1.1.2 Problemática

En la fábrica de *software* de la UDLA se utiliza una metodología de desarrollo de aplicaciones que tiene un enfoque tradicional tipo cascada, en donde cada fase se ejecuta cuando la fase previa ha terminado. Esto se ilustra en la **Figura 1.3**.

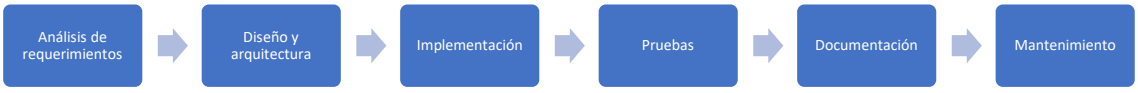


Figura 1.3 Proceso de desarrollo de software

Aunque esta es una metodología robusta y probada en la industria del *software*, existe un porcentaje de error en la entrega y despliegue de las aplicaciones desarrolladas debido a las siguientes falencias identificadas:

- Problemas en el análisis de requerimientos, en la definición del alcance y planificación de esfuerzos. Esto provoca una estimación errónea de tiempos y entregas no oportunas.
- Falta de buenas prácticas en el desarrollo dando como resultado problemas de rendimiento en los ambientes productivos, *software* de baja calidad y dificultad en el mantenimiento.
- Ausencia de un área de calidad de *software* que se enfoque a realizar pruebas de carga, rendimiento, funcionales y no funcionales de las aplicaciones creadas.
- Problemas de despliegue en los ambientes productivos de los desarrollos.
- Ausencia de métricas para valorar esfuerzos en las tareas de desarrollo de *software* por persona.
- Desconocimiento de plataformas de desarrollo *Low-Code*.

Lo anterior indicado ocasiona un alto porcentaje de aplicaciones con defectos que lleva a realizar correcciones y ajustes durante y después del lanzamiento del *software* al ambiente productivo. Adicional, se incumplen los tiempos de entrega generando insatisfacción y pérdida de credibilidad antes los usuarios. En función de esto, el proyecto *Capstone* plantea un fortalecimiento en la Fábrica de *Software* de la UDLA para subir sus estándares de desarrollo y posicionarse como un área diferenciadora dentro de la institución.

1.2 Organización impactada

En la **Tabla 1.1** se identifican las áreas que intervienen en la solución al *Concern* planteado junto con el nivel de impacto por cada una.

Tabla 1.1 Organización Impactada

Área	Nivel de impacto	Descripción
Dirección de TI	Alto	• Auspicio en el ejercicio de la arquitectura empresarial.
Sistemas de Información	Alto	• Uso de marcos de trabajo ágiles. Buenas prácticas para el desarrollo de *software*. Arquitectura de diseño en las aplicaciones. Inversión en capacitaciones para el equipo.
Producción	Medio	• Cambios en el proceso de despliegue de las aplicaciones en ambientes productivos.
Infraestructura Tecnológica	Bajo	• Mayores capacidades de infraestructura para soportar carga sobre las aplicaciones. • Implementación de *framework* para despliegue del *software*.
PMO	Medio	• Uso de marcos de trabajo ágiles.

1.3 *Stakeholders* y expectativas de valor

En la
Tabla 1.2 se describen las jefaturas de las áreas interesadas, las brechas existentes en la comprensión y compromiso de lo actual y lo requerido junto con las expectativas en relación con el problema.

Tabla 1.2 Stakeholders expectativas de valor

Unidad	Cargo	Comprensión actual	Comprensión Requerida	Compromiso actual	Compromiso requerido	Expectativas en relación al Concern
Dirección de TI	Director de TI	4	5	3	5	• Indicadores de cumplimiento. • Disminución en los indicadores de retrabajo. • Uso de plataformas Low-Code and Hi-Productivity.
Sistemas de Información	Jefe de Sistemas de Información	4	5	4	5	• Indicadores de cumplimiento. • Disminución en los indicadores de retrabajo. • Mayor rapidez y calidad en las entregas. • Uso de plataformas Low-Code and Hi-Productivity.
	Coordinador-Nuevos desarrollos	4	5	4	5	• Indicadores de cumplimiento. • Disminución en los indicadores de retrabajo. • Mayor rapidez y calidad en las entregas.
	Coordinador-Mantenimientos	4	5	4	5	• Indicadores de cumplimiento. • Disminución en los indicadores de retrabajo. • Mayor rapidez y calidad en las entregas.
Producción	Jefe de Producción	3	4	3	4	• Mayor rapidez y calidad en las entregas.
Infraestructura Tecnológica	Jefe de Infraestructura	2	3	3	4	• Mejora en la gestión de ambientes no productivos.
	Analista Líder de Infraestructura	2	3	3	4	• Mejora en la gestión de ambientes no productivos.
PMO	Jefe	3	4	3	4	• Marcos de trabajo ágiles. • Mayor rapidez y calidad en las entregas.

1.4 Gestión de *Stakeholders*

En la **Tabla 1.3** se identifican a los *Stakeholders* junto con su importancia en relación con el problema empresarial planteado.

Tabla 1.3 Gestión de Stakeholders

Unidad	Cargo	Poder	Nivel de interés	Estrategia	Cuadrante
Dirección de TI	Director de TI	Alto	Alto	Interesado clave	5
Sistemas de Información	Jefe de Sistemas de Información	Alto	Alto	Interesado clave	5
	Coordinador-Nuevos desarrollos	Medio	Alto	Mantenerlo satisfecho	4
	Coordinador-Mantenimientos	Medio	Alto	Mantenerlo satisfecho	4
Producción	Jefe de Producción TI	Alto	Medio	Mantenerlo satisfecho	4
Infraestructura Tecnológica	Jefe de Infraestructura	Alto	Medio	Mantenerlo informado	3
	Analista Líder de Infraestructura	Medio	Medio	Mantenerlo informado	3
PMO	Jefe	Alto	Medio	Mantenerlo satisfecho	3

1.5 Caracterización del problema

En función de los elementos que se han presentado en las secciones previas, para la caracterización del problema vamos a utilizar la herramienta *Business Motivation Model* como se ve en la **Figura 1.4** que proporciona un esquema para desarrollar, comunicar y administrar las motivaciones en relación con una iniciativa de manera organizada. Caracterizando la visión de tener una fábrica de *software* fortalecida y sus objetivos en función de las estrategias y tácticas que se requieren implementar.

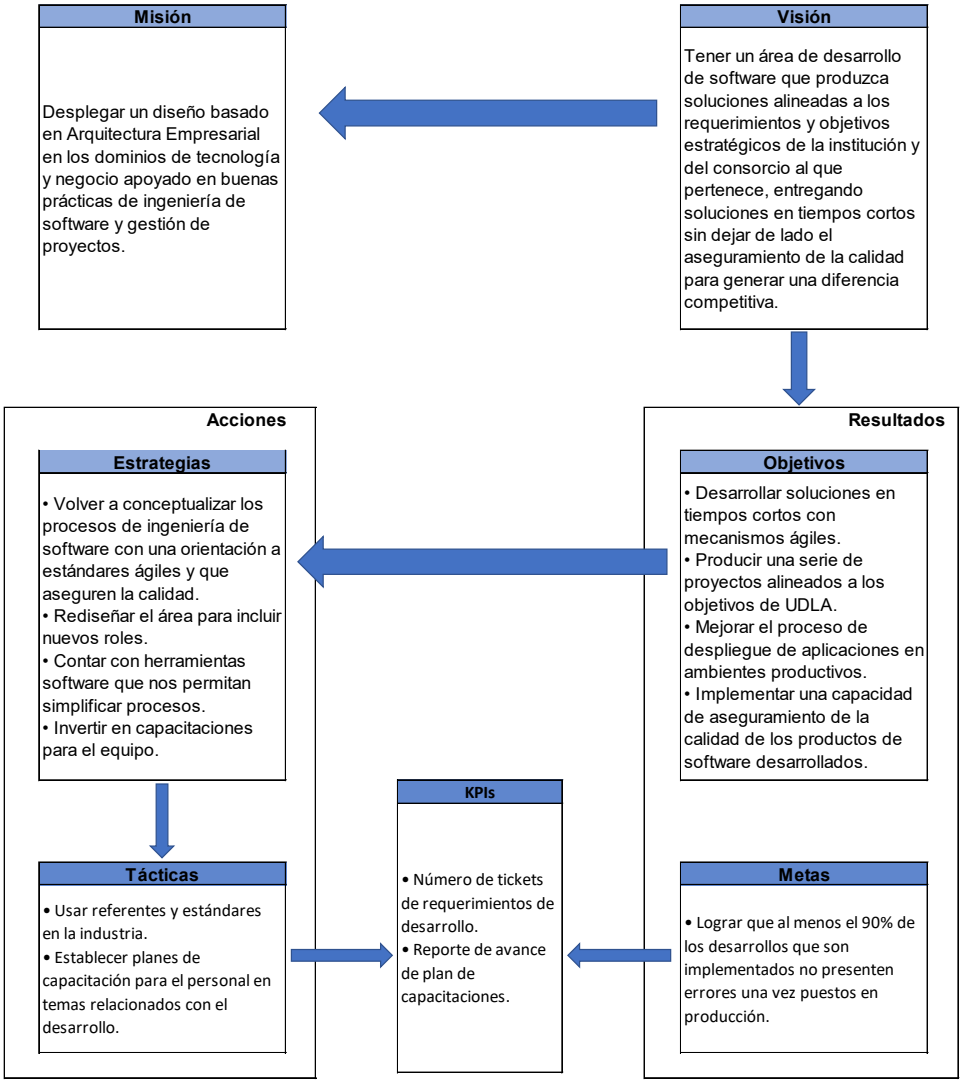
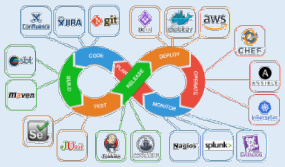


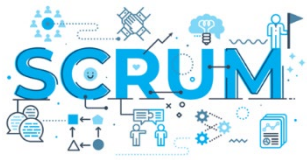
Figura 1.4 Business Motivation Model

1.6 Marcos de referencia

El área de sistemas de información deberá apoyarse en estándares, buenas prácticas y tendencias tecnológicas para solucionar el problema. En la Tabla 1.4 se describen cada uno de ellos en relación con las áreas.

Tabla 1.4 Marcos de referencia

Área	Referente	Descripción
Fábrica de *software*		*DevOps* es un marco de trabajo y una filosofía en constante evolución que promueve un mejor desarrollo de aplicaciones en menos tiempo y la rápida publicación de nuevas funciones de *software* o productos para los clientes. (Netapp, 2019)
	OWASP The Open Web Application Security Project	*The Open Web Application Security Project® (OWASP)*, es una fundación sin fines de lucro que trabaja para mejorar la seguridad del *software* a través de proyectos de *software* de código abierto liderados por la comunidad, cientos de capítulos locales en todo el mundo, decenas de miles de miembros y liderando conferencias educativas y de capacitación. *OWASP* es la fuente para que los desarrolladores y tecnólogos protejan la web. (Owasp, 2022)
	CMMI DEV/3	CMMI *Development* es un conjunto integrado de mejores prácticas que mejora el rendimiento y las capacidades clave para las organizaciones que desarrollan productos, componentes y servicios. (Cmmi Institute, 2022)
	SFIA FOUNDATION	SFIA define las habilidades y competencias requeridas por los profesionales que diseñan, desarrollan, implementan, administran y protegen los datos y la tecnología que impulsan el mundo digital. (Sfia, 2022)
PMO	PMI Project Management Institute	*Project Management Institute*, organización que se dedica al estudio y promoción de la

		dirección de proyectos, estableciendo un conjunto de directrices que orientan la dirección y gestión de proyectos. Propone una serie de procesos de gestión que se han identificado como los más habituales y que la práctica han demostrado que son efectivos y eficientes. (A. Perez, 2021)
PMO-Fábrica de *software*		*Scrum*, es un marco de trabajo para manejar problemas complejos y adaptativos, mientras que de forma creativa se entregan productos con un alto valor para las partes interesadas. Consta de equipos y sus roles, eventos, artefactos y reglas que tienen un propósito específico y esencial para el éxito del uso de *Scrum* en los proyectos. Tiene un enfoque iterativo incremental, se compone de grupos pequeños de personas que se caracterizan por ser muy flexible y adaptativos a los cambios.

1.7 Equipo de Arquitectura

En la **Figura 1.5** se establece el organigrama del equipo de arquitectura necesario para resolver el *Concern*. En la **Tabla 1.5** se describe las responsabilidades de cada rol de los miembros del equipo.

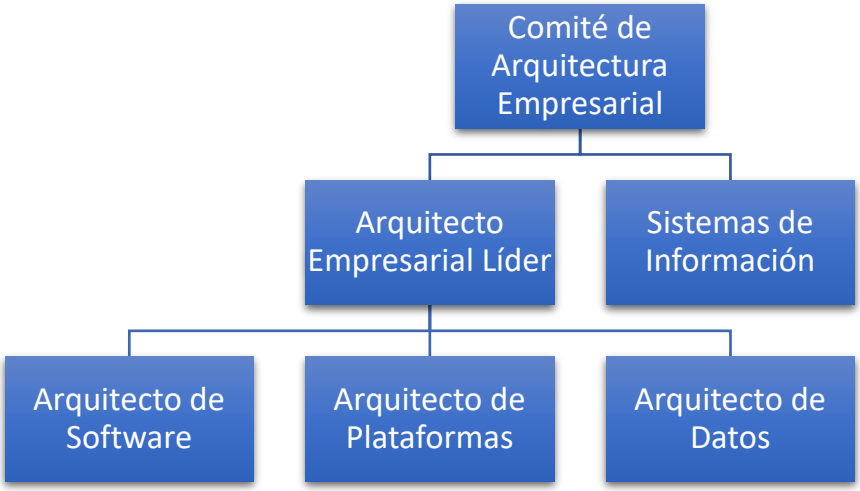


Figura 1.5 Equipo de arquitectura

Tabla 1.5 Equipo de arquitectura

Rol	Responsabilidad
Arquitecto Empresarial Líder	Es el encargado de guiar al equipo en las actividades para el desarrollo del ejercicio de arquitectura empresarial.
Arquitecto de *Software*	Es el responsable de establecer los estándares y buenas prácticas de programación. Así como determinar las versiones de *frameworks* adecuadas para los diferentes tipos de aplicaciones que se desarrollan en el departamento. Además, debe conocer las interacciones entre las aplicaciones para determinar la arquitectura más adecuada.
Arquitecto de Datos	Es el responsable de asegurar que el modelo de datos soporte las funcionalidades del negocio y sea escale en el tiempo, aportando en el ciclo de vida de los datos y su integridad.
Arquitecto de Plataformas	Gestiona la infraestructura tecnológica y sus componentes para determinar la mejor arquitectura basado en una estrategia de alta disponibilidad dependiendo de la criticidad de las aplicaciones.

1.8 Catálogo de Principios

Se ha determinado un catálogo de principios que se muestra en la **Tabla** 1.6 como definiciones iniciales para el desarrollo de este ejercicio de arquitectura empresarial. Son normas y directrices generales que informan y apoyan la forma en que la organización se enfoca a resolver el problema.

Tabla 1.6 Catálogo de principios

Código	Principio	Dominio	Definición	Motivación	Implicaciones
N01	Alineamiento de iniciativas de TI a los procesos de negocio.	Negocio	Los objetivos de TI deben de estar alineados con la estrategia de negocio, para lo cual se debe tener procedimientos de evaluación objetiva del Gobierno TI.	• Tener una congruencia entre la estrategia del negocio planteada por la alta dirección y los subsistemas. • Cumplir con los objetivos propuestos, optimizar el rendimiento y tener una ventaja competitiva sostenible en el largo plazo.	• Se debe tener un entendimiento claro de la estrategia del negocio. • Las áreas de la organización deben involucrarse activa y colectivamente para tomar decisiones acertadas que contribuyan de manera efectiva y eficiente a los lineamientos de la propuesta de valor.
A01	Desarrollo basado en estándares y tomando en cuenta el framework Net 6 o superior.	Aplicaciones	El desarrollo de aplicaciones software debe basarse en estándares, buenas prácticas, patrones y el framework de Microsoft Net 6 o superior.	• Obtener un software de calidad. • Estandarizar el uso del framework Net 6 para nuevas aplicaciones basándose en la adopción de las nuevas tendencias, así como en la escalabilidad que nos ofrece este framework de Microsoft. • Planificar la migración de las aplicaciones actuales desarrolladas en diferentes versiones de .Net Framework a Net 6.	• En la actualidad las aplicaciones están desplegadas en servidores on-premises. Se debe planificar la migración a la nube tanto de las aplicaciones como de la infraestructura. • El desarrollo de las aplicaciones con las nuevas versiones del framework de Microsoft, facilitarán la migración a la nube. • En caso de contar con proveedores, también se deben alinear a este principio.
A02	La arquitectura de las aplicaciones debe ser extensible, escalable y adaptable.		Las aplicaciones se deben basar en un marco arquitectónico flexible y que les permita añadir funcionalidades de una manera estandarizada.	• Contar con una arquitectura estándar para cada uno de los tipos de proyectos software en el que se inicie el desarrollo. • Se debe contemplar el agrupar aplicaciones con funcionalidad en común, no tener un universo amplio de aplicaciones independientes que cumplen funciones similares.	• Se deberá contar con personal capacitado para poder establecer esta arquitectura. • Se debe contar con una infraestructura tecnológica para que soporte la carga que pueden tener las aplicaciones.
D01	Seguridad en datos.	Datos	Se debe definir e implementar controles de seguridad para el acceso a la información a través de roles y perfiles en todas las aplicaciones.	• Salvaguardar la integridad de los datos que son procesados por las aplicaciones creadas. • Ofrecer seguridad y disponibilidad a los usuarios de acceder a los datos mediante las aplicaciones corporativas.	• Garantizar la integridad y tomar las precauciones necesarias para que la información no sea modificada o eliminada sin autorización.
T01	Uso de la nube	Infraestructura Tecnológica	Red mundial de servidores, cada uno con una función única. La nube no es una entidad física, sino una red enorme de servidores remotos de todo el mundo que están conectados para funcionar como un único ecosistema y que nos permiten utilizar servicios a través del internet.	• Reducir costos (Capex). • Evitar obsolescencia de los dispositivos físicos • Reducción de interrupciones y mejora de la estabilidad de TI • Escalado para satisfacer demandas geográficas • Seguridad mejorada	• Diseñar un plan de migración a la nube que incluya aplicaciones, bases de datos y sistemas centrales • Hacer ajustes en las aplicaciones para que puedan correr en la nube • Seleccionar una nube que se adapte a las necesidades organizacionales

2 Fase de Visionamiento

En esta fase se desarrolla un visionamiento de alto nivel de la solución al problema identificado en la fase preliminar tomando en cuenta estándares, buenas prácticas y casos de éxito en la industria del desarrollo de *software*. Esta solución se va a generar en los dominios de ADM de TOGAF:

- Negocio.
- Aplicaciones.
- Información.
- Tecnología base.

2.1 Requerimientos de alto nivel

A continuación, se describe los requerimientos de alto nivel para el fortalecimiento de la Fábrica de Software de La Universidad de Las Américas.

2.1.1 Negocio

- Implementar Ingeniería de Requisitos, para contar con una especificación correcta, completa y adecuada de las necesidades de los usuarios.
- Contar con un perfil especialista para organizar un área de Aseguramiento de la Calidad del Software que determine las mejores prácticas para realizar las pruebas funcionales y no funcionales en las aplicaciones nuevas, así como de los requerimientos de cambios, mejoras o correcciones de *bugs* en los sistemas existentes.
- Basarse en CMMI Dev para los desarrollos, con el fin de mejorar los procesos de creación de aplicaciones y obtener mejores productos y servicios.
- Para las dos líneas de desarrollo que existe en el área de Sistemas de Información, se debe usar marcos de trabajo ágiles como *Scrum* para la gestión de nuevos proyectos y para los requerimientos de cambios, mejoras o correcciones de *bugs* en los sistemas existentes.

2.1.2 Aplicaciones

- Mejorar los procesos de integración continua, despliegue continuo, automatización de pruebas mediante la implementación de DevOps.
- Usar una herramienta de planificación y seguimiento de proyectos.
- Contar con una plataforma de desarrollo de aplicaciones *Low-Code* que permitan ahorrar costos y tiempo junto con generar alta productividad.

2.1.3 Información

- Generar reportes, indicadores y KPIs de esfuerzos por desarrollador.
- Usar un repositorio para compartir estándares, buenas prácticas y una base de conocimientos al que tengan acceso los miembros del equipo.

2.1.4 Infraestructura Base

- Gestión de ambientes no productivos, de acuerdo con la aplicación que esté en proceso de desarrollo y al número de equipos que estén trabajando de manera concurrente sobre una misma aplicación. Para estos casos, se deben definir los siguientes ambientes:
    - Desarrollo.
    - Pruebas.

A continuación, se muestra un diagrama para relacionar los requerimientos de alto nivel con los objetivos identificados en el capítulo uno.

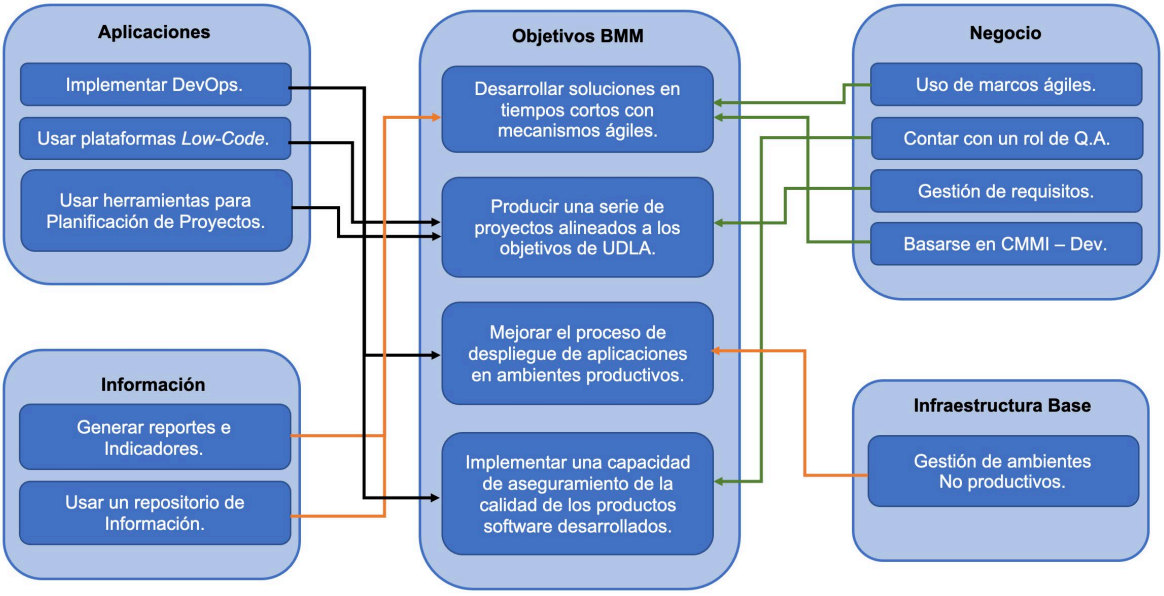


Figura 2.1 Diagrama conceptual de requerimientos de alto nivel

2.2 Visionamiento y escenarios de solución

Como estrategia para resolver el problema planteado en la fase preliminar, se recomienda basarse en estándares y buenas prácticas de la industria en lugar de improvisar. El presente análisis toma en cuenta definiciones como DevOps, agilismo, plataformas *Low-Code*, CMMI-Dev. Adicional, junto con esto se referencia casos de éxito en el área de desarrollo de software.

2.2.1 Estándares y tendencias de la industria

2.2.1.1 DevOps

Una tendencia que en los últimos años ha tenido gran acogida en las áreas de desarrollo de las fábricas de *software*, es DevOps, que proviene de unir los términos "*Development* (Desarrollo) y *Operations* (Operaciones)". Se puede definir como "Un marco de trabajo y una filosofía en constante evolución que promueve un mejor desarrollo de aplicaciones en menos tiempo y la rápida publicación de nuevas funciones de *software* o productos para los clientes" (NetApp, 2019). En otras palabras, buenas prácticas para hacer más ágil el desarrollo y despliegue de *software*.

Nace de la necesidad de mejorar la comunicación, colaboración e integración de las áreas de desarrollo de *software* y operaciones, así como automatizar tareas repetitivas, eliminar errores humanos en el desarrollo y despliegue de las aplicaciones.

Existe una relación entre DevOps con agilismo. “DevOps es una cultura que promueve la colaboración entre todos los roles implicados en el desarrollo y el mantenimiento de *software*” (Microsoft Azure, 2021). El enfoque ágil nos guía para mantener la productividad e impulsar el lanzamiento de versiones en proyectos con necesidades que cambian continuamente y que están en un entorno volátil. DevOps y el agilismo a menudo se practican juntos.

DevOps conjuga cultura, prácticas y herramientas para que los equipos alcancen la capacidad de responder mejor a los requerimientos de los clientes y lograr los objetivos empresariales en menos tiempo.

a) Cultura

Comprende un conjunto de valores que los miembros del equipo materializan mientras trabajan y colaboran en los proyectos. Esto implica:

- Ciclos de lanzamiento de productos más cortos.
- Colaboración, visibilidad y alineamiento.
- Rendición de cuentas.
- Aprendizaje continuo.

b) Prácticas

Se deben implementar durante todas las fases del ciclo de vida del *software* para automatizar, mejorar y acelerar dichas etapas. Se determinan las siguientes prácticas:

- Integración continua (CI - *Continuous Integration*) y Entrega continua (CD – *Continuous Delivery*).
- Control de Versiones.
- Desarrollo ágil de *software*.
- Infraestructura como código (IaC – *Infrastructure as Code*).

- Administración de configuración.
- Supervisión continua.
- Pruebas continuas.

c) Herramientas

En el mercado tecnológico, existe una amplia gama de herramientas DevOps que se pueden utilizar en las diferentes fases del ciclo de vida del *software*. Así mismo, las herramientas encajan con las prácticas de DevOps previamente indicadas.

- Automatización DevOps.
- DevOps *Pipeline* (CI/CD).
- Control de versiones DevOps.
- Gestión de la configuración de DevOps.
- Pruebas DevOps.
- Monitorización de DevOps.

En la **Figura 2.2** se ilustra el ciclo de entrega continua en DevOps.

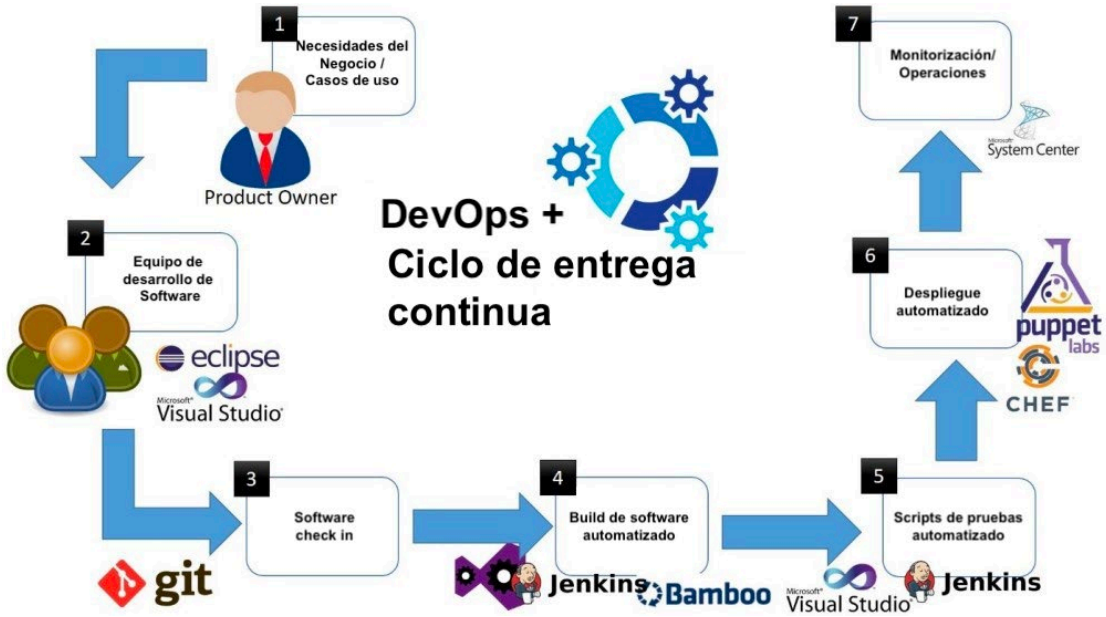


Figura 2.2 DevOps + Ciclo de entrega continua

(DevOps & Agile = Better Builds & Faster Releases, 2018)

En la **Figura 2.3** se ilustra la relación entre DevOps y el Agilismo. Tomando como base que DevOps es un concepto para administrar procesos de ingeniería de

extremo a extremo, mientras que ágil es el marco de trabajo usado para manejar proyectos complejos.

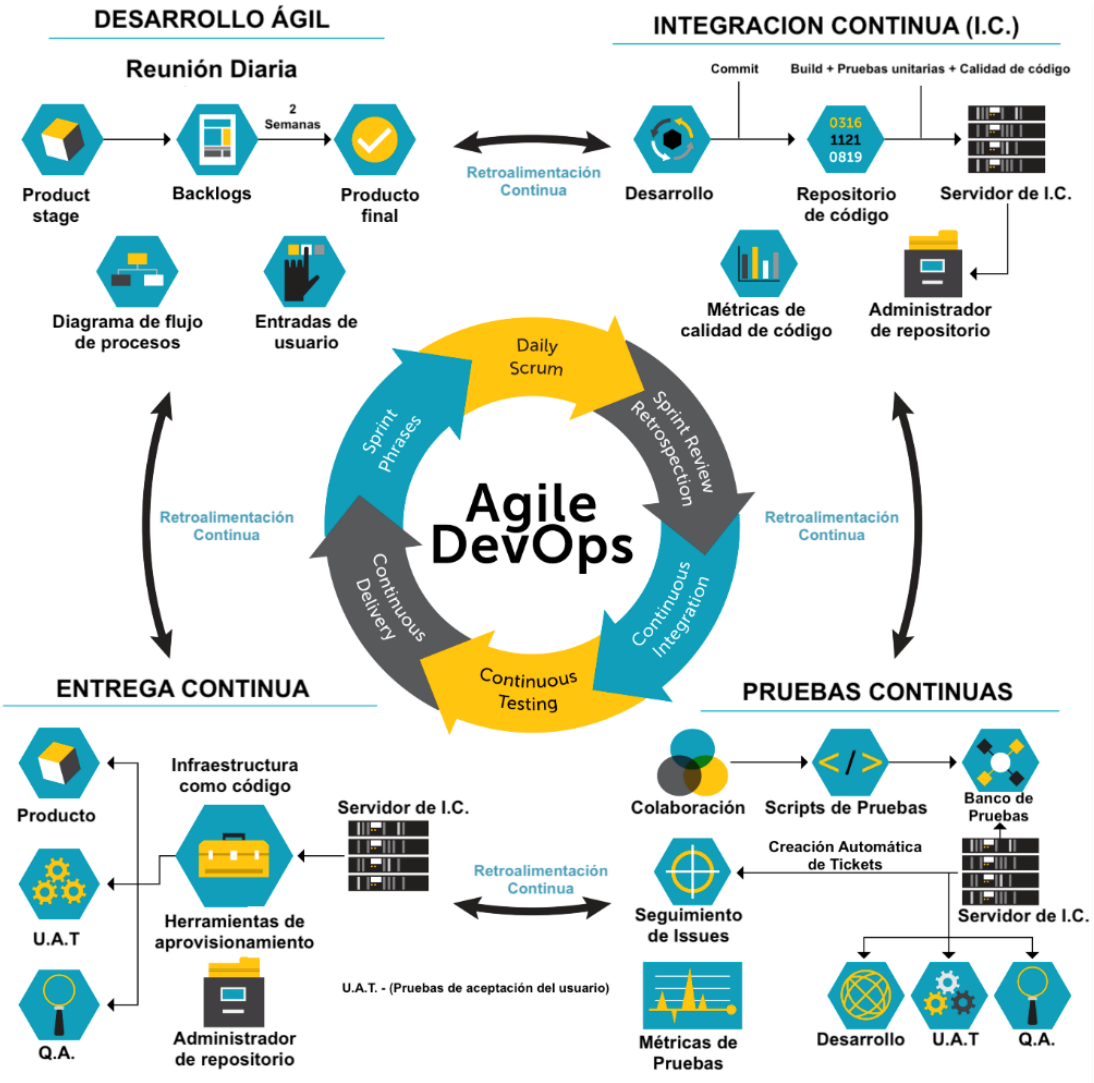


Figura 2.3 DevOps & Agile

(DevOps & Agile = Better Builds & Faster Releases, 2018)

2.2.1.2 CMMI

Es un modelo de calidad del *software* que ayuda a medir o establecer que tan madura es la forma de trabajo de una organización en sus prácticas. Está compuesto por cinco niveles:

- **Nivel 1 - No gestionado:** Todos los procesos son caóticos, los resultados obtenidos se dan por las habilidades de cada persona.

- **Nivel 2 - Gestionado:** Ya se tiene algo de gestión de los proyectos y se puede tomar decisiones.
- **Nivel 3 - Definido:** Se encuentran la mayor parte de las prácticas de ingeniería.
- **Nivel 4 - Medido:** Se toman decisiones en base a datos.
- **Nivel 5 - Optimizado:** La forma de trabajo es constantemente revisada y mejorada.

CMMI, es un modelo sólido, es una guía de buenas prácticas que no impone métodos. Es decir que indica que se debe hacer, pero no especifica cómo se debe hacer. La única condición que impone es incluir todas las prácticas debido a que son necesarias.
El enfoque de CMMI es crear una cultura donde todos dentro de la organización entiendan que el mejor camino es definir procesos y que para esto se requiere apoyo de gerencia, equipos entrenados y disponibles. CMMI abarca tres disciplinas superpuestas:

- Desarrollo de procesos y servicios.
- Gestión de servicios.
- Adquisición de productos y servicios.

Para el desarrollo de procesos y servicios se cuenta con CMMI DEV que tiene por objetivo ser una guía para mejorar el proceso de desarrollo de *software*. Se puede usar para mejorar ciertas actividades o áreas de procesos hasta alcanzar un cierto nivel esperado o mejorar un grupo establecido de actividades organizadas en áreas de procesos.

CMMI es un marco de trabajo que contiene todos los objetivos y prácticas que se utilizan en los distintos modelos. Los componentes del modelo se agrupan en tres categorías:

- **Requerido:** Estos son componentes esenciales para lograr mejoras de procesos en áreas específicas.
- **Esperado:** Son actividades importantes para lograr un componente CMMI requerido. Los componentes esperados en CMMI son las

practicas específicas y genéricas. Las metas se consideran satisfechas cuando se implementa las prácticas en la organización.

- **Informativos:** Estos ayudan a los usuarios a modelar, comprender los componentes requeridos y esperados de CMMI. Estos componentes pueden ser cuadros de ejemplo, explicaciones detalladas u otra información útil, notas, referencias, títulos de objetivos, títulos de prácticas, fuentes, los productos de trabajo de ejemplo y la elaboración de prácticas genéricas.

Las áreas de procesos son un grupo de prácticas relacionadas en un área que, cuando se implementan colectivamente, satisfacen un conjunto de objetivos considerados importantes para la organización. (Isolution, 2020). Estas se agrupan como se muestra en la **Figura 2.4**:

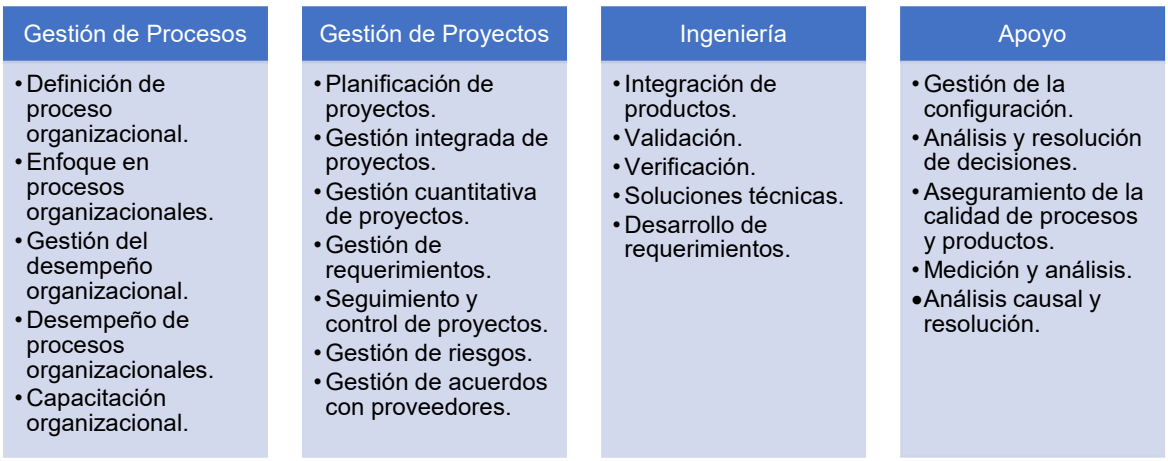


Figura 2.4 Áreas de procesos y practicas CMMI

2.2.1.3 Plataformas Low-Code

Conforme ha avanzado las nuevas tecnologías, se han planteado soluciones a las necesidades crecientes de las empresas por tener plataformas de desarrollo que implique crear aplicaciones de gran impacto, con poco código y que sean fáciles de construir. Estas herramientas no solamente ayudan con la codificación, también con las configuraciones e implementaciones rápidas.

Las plataformas *Low-Code* proveen una interfaz gráfica de usuario para la creación de este tipo de aplicaciones por lo que se desarrolla el código a un ritmo

muy rápido y reduce los esfuerzos de programación tradicionales. Adicional, permiten a sus usuarios poder crear aplicaciones con tan solo dibujar un diagrama de flujo o arrastrar y soltar componentes para que el código sea generado.

En este sentido, se introduce un nuevo término conocido como "*Citizen Developer*" que se define como los colaboradores o empleados de una organización. Toman gran relevancia en las aplicaciones *Low-Code* debido a que se vuelven protagonistas al poder usar dichas plataformas de una manera más natural por la fácil adopción de esta tecnología permitiéndoles solucionar problemas identificados en sus respectivas áreas de trabajo y crear aplicaciones de forma independiente.

Los beneficios de contar con una plataforma de este tipo son:

- Incrementar la productividad.
- Disminuir costos al desarrollar aplicaciones en menos tiempo.
- Acelerar la transformación e innovación digital.
- Reducir el backlog y aumentar la capacidad de responder a los requerimientos.
- Reducir la dependencia de habilidades técnicas difíciles de contratar.
- Se democratiza el desarrollo de aplicaciones al permitir a los "*Citizen Developers"* mejorar procesos internos.

Así mismo se identifican las siguientes limitaciones:

- Se requiere de un perfil técnico en caso de requerir una customización significativa.
- En ciertos casos se puede necesitar codificación por lo que no es ideal para todos los usuarios.

Forrester diferencia las plataformas *Low-Code* de las *No-Code* por sus usuarios finales objetivos. Las aplicaciones *Low-Code* a pesar de que pueden requerir algunas habilidades de codificación, atienden a desarrolladores profesionales, así como a los denominados "*Citizen developers*". Mientras que los productos

No-Code están enfocados específicamente para usuarios de negocio que no tengan conocimientos técnicos. (FlowForma, 2021).

Otra diferencia detectada, las aplicaciones *Low-Code* en referencia a las *No-Code*, ofrecen cierta flexibilidad para incluir un nivel de personalización al permitir incluir algún script para usarlo en soluciones más complejas.
Los casos de uso más comunes son:

- Incorporación de nuevos empleados para Recursos Humanos.
- Aplicación móvil para registrar personas a eventos.
- Flujo para proceso de compras.
- Gestionar solicitudes de trabajo.

Existen varias herramientas con presencia en el mercado que ofrecen funcionalidades similares entre ellas. Según el Cuadrante Mágico de Gartner para plataformas del tipo *Low-Code*, Microsoft se encuentra entre los líderes del mercado con su producto *Power Apps como se muestra en la* **Figura 2.5**.

Figura 2.5 Cuadrante mágico de Gartner

(Microsoft, 2021)

Power Apps, es la plataforma de Microsoft orientada a generar aplicaciones del tipo *Low-Code*.

> Según la descripción dada en la documentación de Microsoft (Microsoft, ¿Qué es Power Apps?, 2022), *Power Apps* es un conjunto de aplicaciones, servicios y conectores, así como una plataforma de datos que proporciona un entorno de desarrollo de aplicaciones ágil para crear aplicaciones personalizadas para las necesidades de su empresa.
>
> Las aplicaciones creadas ofrecen una completa lógica de negocios y capacidades de flujo de trabajo con el fin de transformar las operaciones empresariales manuales para procesos digitales y automatizados. Además, presentan un diseño dinámico y pueden ejecutarse sin problemas en un explorador y en dispositivos móviles (teléfono o tableta).
>
> *Power Apps* "democratiza" la experiencia de creación de aplicaciones empresariales personalizadas ya que permite a los usuarios crear dichas aplicaciones con múltiples características sin escribir código. También proporciona una plataforma extensible que permite a los desarrolladores profesionales interactuar mediante programación con datos y metadatos, aplicar lógica empresarial, crear conectores personalizados e integrarse con datos externos.

2.2.1.4 Perfiles de Q.A. - Testers

El acrónimo Q.A. proviene de las palabras “*Quality Assurance*” que traducido al español es “Aseguramiento de Calidad”. Comúnmente también es conocido como *Tester* o Pruebas y es el rol que en las fábricas de *software* se encarga de certificar la calidad de las aplicaciones o funcionalidades que el equipo de desarrollo crea. Las personas de este perfil deben contar con características como la proactividad, investigación y resiliencia para adaptarse a los cambios que implica trabajar en IT y sobre todo en conjunto con el área de desarrollo.

Las actividades de *testing*, no se quedan fuera del concepto de agilismo, más bien se vuelven parte de este. Los autores (Greaves & Laing, 2013), definieron el “Manifiesto para pruebas ágiles de *software*” que dicta los siguiente:

- *Testing* durante sobre *testing* al final.
- Prevenir bugs sobre encontrar bugs.
- Entender lo que se testea sobre verificar funcionalidad.
- Construir el sistema sobre destruir el sistema.
- Responsabilidad del equipo sobre responsabilidad del *tester*.

En palabras de los autores, lo anterior se resume en: “Mientras hay valor en los elementos de la derecha, valoramos más los de la izquierda” (Greaves & Laing, 2013).

En los equipos ágiles, cada uno de los miembros es responsable de entregar un producto de alta calidad que provea valor al negocio. Así mismo el rol de los *tester* en los equipos ágiles es asegurar que su equipo entregue la calidad que sus clientes necesitan.

El desarrollo ágil nos anima a resolver los problemas como equipo en donde cada uno de los roles es importante y esencial (personas del negocio, programadores, *tester*). Deciden juntos la manera óptima de mejorar sus productos, los *tester* trabajan en conjunto con el equipo de personas que se sienten responsables de brindar la mejor calidad posible y todos están involucrados en las pruebas.

El equipo de *testing* debe enfocarse en los siguientes tipos de pruebas que son inherentes a cualquier tipo de proyecto de desarrollo sea este usando marcos de desarrollo ágil o tradicional:

- Características y funcionalidades orientadas al negocio o deseadas por los expertos de negocio.
- Pruebas de unidad y de componente tales como funcionales, sistema, carga, rendimiento, seguridad, estrés, usabilidad, exploración, extremo a extremo y aceptación del usuario.

A continuación, se muestra en la **Figura 2.6** la interacción entre los roles:

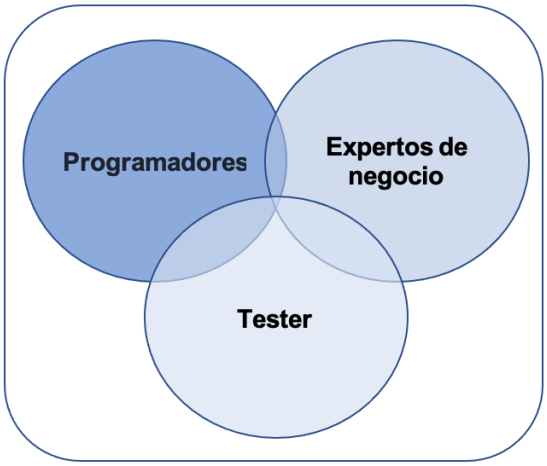


Figura 2.6 Equipo ágil

Junto con lo descrito en los párrafos anteriores, se complementa con una explicación sobre SFIA. Es un modelo de referencia que “define las habilidades y competencias requeridas por los profesionales que diseñan, desarrollan, implementan, administran y protegen los datos y la tecnología que impulsan el mundo digital” (SFIA 8, 2021). De esta manera, SFIA en su versión 8 define las responsabilidades de rol de *tester* de la siguiente manera:

> La planificación, el diseño, la administración, la ejecución y la presentación de informes de pruebas, utilizando herramientas y técnicas de prueba apropiadas y que se ajustan a los estándares de procesos acordados y las regulaciones específicas de la industria. El objetivo de las pruebas es asegurar que los sistemas, configuraciones, paquetes o servicios nuevos y modificados, junto con las interfaces, funcionen según lo especificado (incluidos los requisitos de seguridad) y que los riesgos asociados con la implementación se comprendan y documenten adecuadamente. Las pruebas incluyen el proceso de ingeniería, uso y mantenimiento de *software* de prueba (casos de prueba, scripts de prueba, informes de pruebas, planes de prueba, etc.) para medir y mejorar la calidad del *software* que se prueba.

SFIA tiene siete niveles de responsabilidad que van desde el nivel 1 que es el más bajo hasta el nivel 7, el más alto. Estos niveles describen comportamientos,

valores, conocimientos y características que una persona debe tener para ser competente en dichos niveles.

2.2.1.5 Fábricas de software

Es un servicio que consiste en el proceso de desarrollo de sistemas basados en una metodología ágil, enfocado en el aumento de la productividad y la eficiencia. En la actualidad, las empresas de todos los sectores intentan actualizar sus sistemas de producción a las nuevas tecnologías. Para este enfoque es esencial:

- Entorno DevOps, donde el desarrollo de *software* y las operaciones de TI colaboran conjuntamente.
- La automatización puede aplicarse a una serie de prácticas de desarrollo, como la integración/entrega continua y las pruebas automatizadas.

Según el portal Sigma (Sicma21, 2021), las fábricas de *software* cuentan con herramientas, procesos y componentes propios los cuales les ayudan a organizar y procesar fácilmente los requisitos para crear un programa con rapidez usando plantillas. Los componentes más comunes son:

- **Esquema de fábrica:** Es un documento que se utiliza para categorizar y resumir los activos utilizados para construir y mantener un sistema. Pueden ser documentos XML, modelos, etc. También define las relaciones entre los activos.
- **Implementación de referencia:** Es un ejemplo de un producto ya desarrollado que sirve de ayuda a otros desarrolladores.
- **Patrones de orientación de la arquitectura:** Definen las opciones de diseño de la aplicación y el motivo de dichas opciones.
- **Cómo se hace (*How-to*):** Contienen procedimientos e instrucciones detalladas para completar las tareas.
- **Recetas:** Son los procesos de automatización que ayudan a los desarrolladores de *software* a completar las tareas rutinarias habituales con poca o ninguna aportación.
- **Plantillas:** Elementos de desarrollo de aplicaciones listos para usar con marcadores de posición para los argumentos, normalmente utilizados para crear elementos iniciales del proyecto.

- **Diseñadores:** Los desarrolladores utilizan la información de los diseñadores para modelar aplicaciones con niveles de abstracción más altos.
- **Código reutilizable:** Son componentes que implementan mecanismos o funciones comunes. El código reutilizable en una fábrica de *software* anula la necesidad de escribir código manualmente en muchas áreas y fomenta la reutilización entre aplicaciones.

Una fábrica de *software* debe contar con las siguientes áreas:

- Servicio al cliente: Contacto inicial para entender el requisito.
- Departamento de planificación y control de la producción: Responsable de verificar la continuidad de los proyectos y la utilización de los recursos humanos y materiales.
- Unidad de producción: Equipo que actúa en el desarrollo del *software*.
- Gestión de calidad: Responsable de analizar si las herramientas están dentro de los patrones establecidos.
- Departamento de soporte: Ayuda al cliente a resolver eventuales problemas identificados en los sistemas y aplicaciones desarrollados.

En la medida en que una fábrica de *software* presenta una estructura de trabajo bien definida, mayores son las posibilidades de comprometer a los trabajadores y de prestar servicios de alta calidad a los usuarios.

2.2.2 Casos de éxito

2.2.2.1 Caso de éxito DevOps

Un caso de éxito referente en la adopción de DevOps es Netflix. Sus servicios eran muy costosos de operar y testear. Los principales problemas identificados fueron:

- Tiempos de validación de nuevas funcionalidades de una semana.
- Ningún equipo involucrado se responsabilizaba de los errores detectados.

Netflix decidió implementar la cultura, prácticas y herramientas DevOps en su modelo de desarrollo de *software* tomando en cuenta los siguientes aspectos:

- **Identificar el ciclo de vida interno del desarrollo de *software*.**
  Descubrieron que eran muy eficaces en cada una de sus áreas y muy ineficientes en el ciclo completo.
- **Inspirarse en el principio del movimiento DevOps.**
  Como resultado, identificaron que era muy eficiente que el área de desarrollo se encargara de mantener y operar el sistema. Esta estrategia les permite ser más ágiles para detectar y resolver incidencias. Fue un cambio de paradigma dentro del área debido a que antes tenían equipos especializados para operar distintos sistemas.

Después de dos años de trabajo y adopción de DevOps, lograron reducir a horas el proceso que tomaba días completos.

2.2.2.2 Caso de éxito Power Apps

Northwell Health, es uno de los proveedores de atención medica más grandes de Estados Unidos, tiene una cultura de innovación y por esto siempre están abiertos a nuevas tecnologías que les ayuden a mejorar su servicio. El objetivo es que, mediante el uso de herramientas tecnológicas, puedan optimizar la atención al paciente, reducir los costos y garantizar el cumplimiento normativo.

En este sentido uno de los médicos creo una aplicación denominada *Rounding App*, esta se desarrolló en pocas semanas y tiene como función principal ofrecer a los médicos, enfermeras y administradores, la visibilidad de las tareas que cada equipo debe completar, esto va desde los requerimientos más simples de los pacientes hasta los más complejos.

Un punto para tener en cuenta era la privacidad y la seguridad de los datos por esto es de gran utilidad que las aplicaciones de *Microsoft* administren estas funcionalidades sin mayor complejidad. “Cuando creamos aplicaciones personalizadas internas en el pasado, una cantidad significativa de nuestros recursos de desarrollo se destinó a problemas de seguridad y cumplimiento de normativos”, dice Anantraman. “Pero *Microsoft* ha sido excelente en la obtención

de certificaciones reglamentarias, lo que libera a nuestros desarrolladores para tareas más creativas". (Microsoft, 2020)

En las siguientes fases, la aplicación se ha mejorado incluyendo capacidades de Inteligencia Artificial con *Microsoft Bot Framework*, por lo que los médicos y administradores interactúan con ella para obtener respuestas rápidas a las preguntas sobre los pacientes. Otro beneficio que tiene es que cuentan con un modelo de datos en común y a futuro tienen planeado implementar realidad aumentada para emparejar a los socorristas en el campo con expertos en hospitales. Con aplicaciones basadas en *Microsoft Power Apps* y *Dynamics* 365 dentro de *Northwell* se han creado aplicaciones en corto tiempo y de gran aporte para el negocio.

2.2.2.3 Caso de éxito implementación de una fábrica de software

Micro Focus International fue fundada en 1976 es una empresa multinacional dedicada al negocio de las tecnologías de la información y el *software*. Esta empresa describe como construyeron su fábrica de *software*.

Modelo operativo ideal de la fábrica de *software*

Micro Focus uso la arquitectura de referencia *IT4IT* que se muestra en la **Figura 2.7** como marco para el modelo operativo.

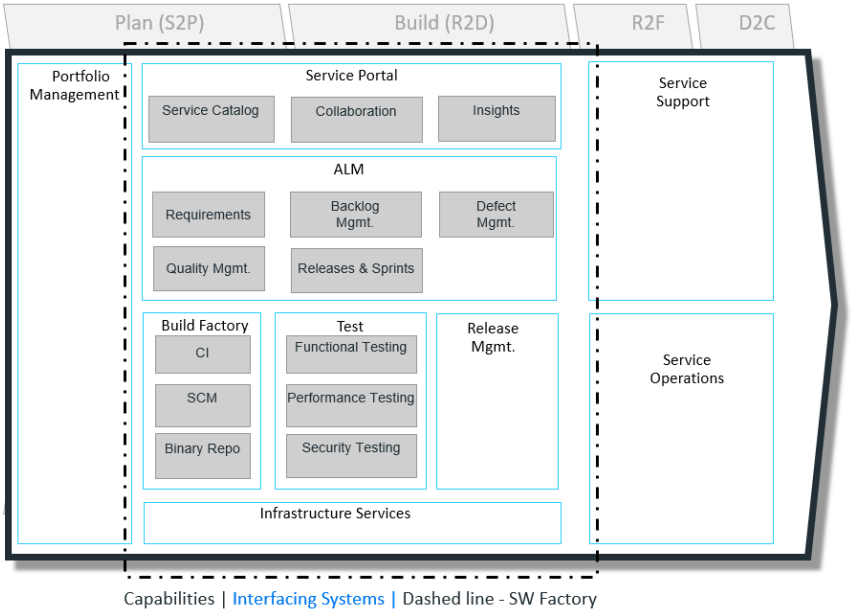


Figura 2.7 Arquitectura de referencia IT4IT

(Sayers, 2021)

En la figura se muestran los principales servicios que deberia tener una fabrica de *software*. En la parte superior se muestran los cuatro flujos de valor primarios, estos son:

- Planificar (estrategia para la cartera)
- Construir (requisito para implementar el ciclo de vida)
- Solicitud para cumplir (R2F)
- Ciclo de vida de detección para corregir (D2C)

A medida que los productos pasan por las actividades de una cadena, ganan valor en cada paso del camino. Debajo de los flujos de valor, las principales funciones y servicios aparecen dentro de cuadros y se agrupan en familias que representan dominios lógicos. La línea discontinua captura el alcance de la fábrica de *software*, con las principales interfaces a las funciones externas capturadas fuera de él.

a) Sistema de gestión de código fuente

SCM es un servicio clave porque posee la propiedad intelectual de su empresa. En este caso se pasó de SVN a Git para el control de versiones, lo que a su vez desencadenó nuestra transición de los repositorios monolíticos que se tenía en SVN a repositorios más pequeños, más ágiles y orientados a microservicios en Git. Esta herramienta complementa el SCM que tenemos implementado.

b) Integración continua

Garantiza que el flujo de código principal funcione continuamente. Si bien los equipos tienen políticas diferentes, cualquier compromiso de código desencadena un ciclo de I.C. que ejecuta un ciclo de prueba corto que verifica los problemas principales con los cambios de código mientras controla el tiempo requerido para no retrasar la canalización.

c) Repositorio binario

El repositorio binario es donde puede acceder a todos los artefactos de compilación intermedios y artefactos de compilación.

d) Pruebas

Para mejorar la agilidad y la calidad, se ampliaron las pruebas a lo largo de todo el ciclo de vida del desarrollo de aplicaciones. Se integró pruebas funcionales, de rendimiento y de seguridad.

e) Pruebas funcionales

Se requiere un sistema de prueba funcional capaz de cubrir un amplio conjunto de tecnologías en nuestra cartera, que incluye web, servicios web, dispositivos móviles, Java y .NET. También se implementó la automatización de pruebas para las aplicaciones.

f) Pruebas de rendimiento

Las pruebas de rendimiento incluyen pruebas de concurrencia, pruebas de longevidad, pruebas de estrés y pruebas de carga. Por la naturaleza de las aplicaciones era necesario cargar la prueba con millones de usuarios simultáneos a medida que avanzaba la confirmación en nuestra canalización de entrega.

g) Pruebas de seguridad

Se implemento un conjunto de prácticas de prueba de seguridad que brindan una cobertura de seguridad completa, incluido el análisis de código estático, el análisis de código dinámico, las verificaciones de dependencia de terceros mediante *OWASP Dependency Check* el escaneo de contenedores con *Clair* y el escaneo de infraestructura.

h) Gestión de la liberación

Es el proceso mediante el cual los productos y servicios están disponibles para el mercado. El sistema de gestión de lanzamientos está estructurado en capas en el sentido de que brinda visibilidad, a nivel de cartera, sobre la preparación de cada lanzamiento y, a nivel de equipo de producto, sobre el progreso del lanzamiento en nuestros entornos de compilación, prueba, preparación y producción.

i) Servicios de infraestructura

Los servicios de infraestructura impulsan los diversos casos de uso en la fábrica de *software*. Los procesos continuos de integración y construcción tienen picos que requieren una alta potencia informática.

j) Portal de servicio

Esta es una ventanilla única que brinda acceso, soporte y conocimiento para todos los servicios de la fábrica de *software*.

k) Catálogo de servicios

Se modelaron todos los servicios y se pusieron a disposición en un catálogo. Ahí es donde los ingenieros pueden obtener acceso al sistema de administración de código fuente.

l) Colaboración

Se promueve activamente la colaboración dentro y entre los equipos a través de un servicio wiki compartido que se usa para planificar actividades como la recopilación y definición de requisitos, el diseño y la arquitectura, y la planificación de versiones. Se crearon canales ad hoc sobre temas específicos, como la colaboración en equipos multifuncionales para resolver problemas de clientes o resolver defectos.

m) Perspectivas

Para la mejorara continua, se necesitaba un conjunto de KPI que pudieran ayudarnos a capturar nuestra línea de base actual con respecto a la agilidad, la productividad y la calidad para que se pueda medir el progreso, los cuellos de botella y las áreas que necesitan mejorar.

n) Proceso de incidentes de ciclo cerrado

Es un proceso y una integración entre la fábrica de *software* y el sistema de soporte que permite a los ingenieros de productos colaborar y compartir conocimientos de manera efectiva.

Para implementar con éxito una fábrica de *software*, debe seguir cuatro pasos:

- Obtener una visión de afuera hacia adentro de su operación para obtener una visión holística de los flujos de valor necesarios para impulsar el negocio, los procesos y las integraciones con funciones externas.
- Tomar una vista de adentro hacia afuera para capturar los principales servicios de ingeniería e integraciones que necesitará para esos flujos de valor.
- Operar de manera ágil, entregar un MVP y evolucionar continuamente en función de la demanda y los comentarios de los consumidores.
- Utilizar esto como una oportunidad para transformarse; vuelva a evaluar sus opciones de herramientas, modernice, consolide e integre su cadena de herramientas para entregar más rápido y con un mayor nivel de calidad.

2.2.3 Resultado del Visionamiento

Después de analizar estándares, buenas prácticas, herramientas y los casos de éxito, se determina que, para fortalecer la Fábrica de *Software* de la UDLA a nivel general, se debe hacer uso de:

- Agilismo mediante el marco de referencia Scrum para la gestión de proyectos y requerimientos, así como la adopción de la cultura, prácticas y herramientas determinadas por DevOps en cada una de las fases del desarrollo *software*. Estos dos enfoques trabajan en conjunto para apoyar activamente en las fábricas de *software*.

  Dentro de Scrum se hace referencia a componentes que son miembros de equipo y personas que están involucradas de forma directa e indirecta con el proyecto; tanto en el desarrollo de este, como en su difusión en el mercado como se muestra en **Figura 2.8**.

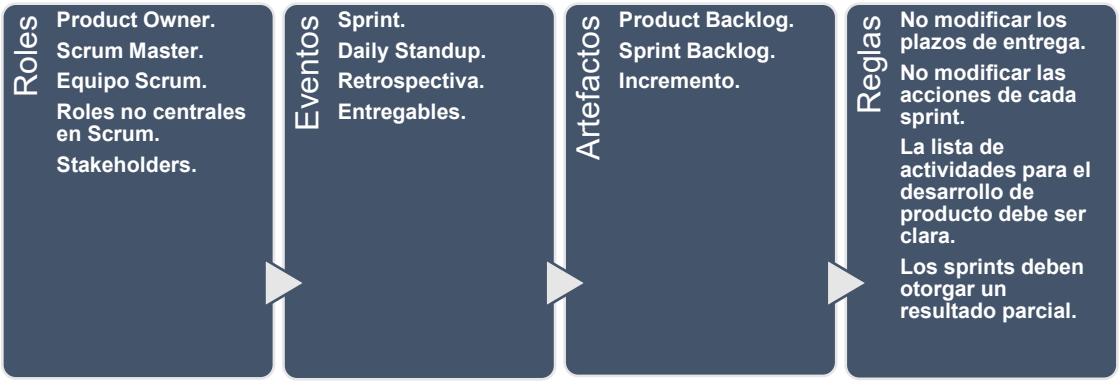


Figura 2.8 Componentes de Scrum

En referencia a la implementación de DevOps, se recomienda realizar una transición paso a paso, es decir no quitar todo lo antiguo y traer todo lo nuevo. En base a esto se define los siguientes pasos:

- Crear una hoja de ruta de transformación de DevOps.
- Definir las herramientas.
- Capacitación a los equipos de trabajo.
- Automatización de procesos.
- Obtener métricas.
- Ejecutar un proyecto piloto.
- Aprendizaje y mejora continua.

- Plataforma de *Low-Code* and *High Productivity*, tomando como referencia específica a *Power Apps* de *Microsoft*. Se recomienda los siguientes pasos para crear aplicaciones:
  - **Planificación:** Se debe identificar el problema a resolver, quién utilizará esta aplicación y qué finalidad tiene.
  - **Diseño:** Conceptual y arquitectónico.
  - **Fase de creación:** Creación de la aplicación como tal.
  - **Fase de pruebas:** Pruebas unitarias, de principio a fin, de aceptación de usuario.
  - **Implementación:** Tan pronto como su aplicación esté lista para usar, debe publicarla y compartirla.

- Se requiere contar con un perfil capaz de orquestar un equipo que realice actividades de *testing* sobre el software desarrollado. Este equipo deberá encargarse de realizar pruebas de negocio, funcionales, sistema, carga, rendimiento, seguridad, estrés, usabilidad, exploración, extremo a extremo y aceptación del usuario.

- CMMI-Dev. Para implementar CMMI-DEV se requiere del apoyo de un patrocinador el cual tiene las siguientes actividades:
    - Influir en la organización para adoptar CMMI.
    - Seleccionar las mejores personas para gestionar el esfuerzo de mejora de procesos.
    - Monitorizar personalmente el esfuerzo de mejora de procesos.
    - Ser un defensor y portavoz activo del esfuerzo de mejora de procesos.
    - Asegurar que estén disponibles los recursos adecuados para permitir que el esfuerzo de mejora de procesos tenga éxito.

  Una vez que se cuenta con el patrocinio se puede adoptar CMMI tomando como base el modelo IDEAL como se muestra en la **Figura 2.9**. El modelo que se muestra representa cinco fases de una iniciativa SPI (*Software Process Improvement*), que proporcionan un ciclo continuo.

  Es importante tener en cuenta que el tiempo que se toma en completar un ciclo a través del modelo IDEAL variará de una organización a otra dependiendo de los recursos que se dispongan ya que hay actividades que se pueden hacer de manera paralela. También es importante señalar que la infraestructura establecida para lograr el SPI desempeñará un papel importante en el éxito o el fracaso de esta iniciativa. El valor que aporta la infraestructura a una iniciativa de SPI, al comprender sus funciones y responsabilidades, no se puede subestimar.

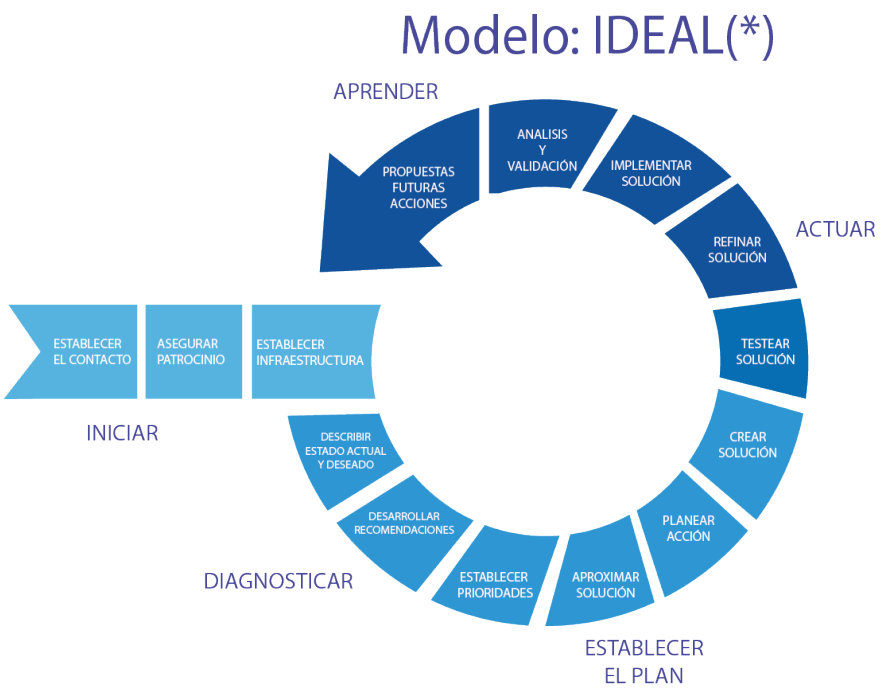


Figura 2.9 Modelo IDEAL CMMI

(McFeeley, 1996)

El modelo IDEAL recibe su nombre de las cinco fases por las que pasa una organización al realizar una iniciativa de cambio:

- ***Initiating* (Iniciar):** Sentar las bases para un esfuerzo de mejora exitoso
- ***Diagnosing* (Diagnosticar):** determine dónde se encuentra en relación con el lugar donde desea estar
- ***Establishing* (Establecer):** planifique los detalles de cómo alcanzará los objetivos:
- ***Acting (Actuar)*:** Hacer el trabajo de acuerdo con el plan. En la fase Actuar, se implementan las actividades planificadas.
- ***Learning (Aprender)*:** Aprender de la experiencia y mejorar la capacidad de adoptar nuevas mejoras en el futuro:

2.3 Análisis de Brechas

El siguiente análisis nos permite identificar y detallar brechas en base a una evaluación entre la situación actual y la objetivo teniendo en cuenta como

referente al estado del arte. Para esta valoración se establece una rúbrica de calificación fundamentada en CMMI (*Capability Maturity Model Integration*) de 6 niveles donde:

Tabla 2.1 Niveles CMMI

Nivel	Nombre	Descripción
0	Inmadura	Falta de cualquier capacidad básica.
1	Inicial	Proceso impredecible, mal controlada y reactivo. Logra más o menos su propósito a través de la aplicación de un conjunto de actividades incompletas que pueden caracterizarse como iniciales o intuitivas, no muy organizadas. (ISACA, 2021)
2	Administrado	Proceso que caracteriza a los proyectos y es a menudo reactivo. Logra su propósito a través de la aplicación de un conjunto de actividades básicas, pero completas, que pueden caracterizarse como realizadas. (ISACA, 2021)
3	Definido	Proceso caracterizado por la organización y es proactivo. Logra su propósito de forma mucho más organizada. Los procesos están, por lo general, bien definidos. (ISACA, 2021)
4	Administrado cuantitativamente	Proceso medido y controlado. El proceso logra su propósito, está bien definido, y su rendimiento se mide de forma cuantitativa. (ISACA, 2021)
5	Optimizado	Foco en mejora continua de procesos. Logra su propósito, está bien definido, su rendimiento se mide para mejorar el desempeño y se persigue la mejora continua. (ISACA, 2021)

Tabla 2.2 Análisis de Brechas

Dominio	Práctica	Actual	Objetivo	Referente	Brecha	Iniciativa de cierrre de brecha
Negocio	Uso de marcos ágiles	2	4	5	El área de Sistemas de Información cuenta con dos líneas de desarrollo: • Nuevos desarrollos. Se está aplicando Scrum, pero no en todos los proyectos. • Mantenimiento. No se aplica prácticas ágiles.	El uso de prácticas ágiles debería estar presente para las dos líneas de desarrollo que tiene el área.
	Contar con un rol de Q.A.	1	3	5	No existe perfiles especializados en la realización de pruebas sobre las aplicaciones. Actualmente las pruebas funcionales y no funcionales se realiza de manera artesanal, sin seguir estándares ni buenas prácticas. Más bien es un ejercicio empírico entre el desarrollador y el funcional a cargo del proceso.	Definir el perfil de un experto en Q.A. que aporte con las buenas prácticas y estándares para la ejecución de testing sobre las aplicaciones.
	Gestión de requisitos	3	4	5	• Nuevos desarrollos. Se apoya en la PMO para levantar requerimientos de proyectos. • Mantenimiento. Los requerimientos se levantan de manera empírica y dependiendo de la urgencia del cambio para la organización.	Implementar ingeniería de requisitos que abarque a las dos líneas de desarrollo existentes en el área.
	Basarse en CMMI - Dev	2	4	5	Los desarrollos no se basan en buenas prácticas y estándares provocando problemas funcionales, no funcionales y de rendimiento en las aplicaciones.	Aplicar las recomendaciones de marcos de referencia para generar aplicaciones con una mayor confiabilidad y mantenibilidad en concordancia con los requisitos exigidos.
Aplicaciones	Implementar DevOps	1	3	5	Los despliegues de aplicaciones son manuales y conllevan trabajo repetitivo.	Implementar capacidades de DevOps (cultura, prácticas, herramientas).
	Usar herramientas para planificación de proyectos	2	4	5	El control de los proyectos y requerimientos se basa en MS Project y herramientas creadasç en MS Excel.	Contar con una herramienta tecnológica que apoye de manera dinámica a la gestión de proyectos y requerimientos que se atienden en el área.
	Usar plataformas Low Code	1	4	5	Existe pocas iniciativas para el uso de plataformas low code.	Capacitar sobre el uso, la importancia y el impacto de estas plataformas en la organización para generar una alta productividad.
	Automatización de pruebas	1	3	5	El proceso de Q.A. sobre las aplicaciones, se realiza de manera artesanal, sin seguir estándares ni buenas prácticas. Más bien es un ejercicio empírico entre el desarrollador y el funcional a cargo del proceso.	Implementar herramientas de automatización de pruebas que aseguren la realización de pruebas continuas para garantizar la calidad de la programación. Se puede ejecutar las veces que sea necesario sin necesidad de una persona que las inicie. Adicional, son mucho más rápidas que las pruebas manuales.
Información	Generar reportes e indicadores	2	3	5	• No existe suficiente información para generar reportes sobre el esfuerzo por analista. • No se cuenta con una herramienta tecnológica para poder registrar data y obtener reportes de esfuerzos.	Contar con una herramienta tecnológica que apoye de manera dinámica a la gestión de proyectos y requerimientos que se atienden en el área.
	Usar un repositorio de información	2	3	5	Existe un repositorio digital que contiene las buenas prácticas y estándares de desarrollo definidos por los arquitectos del área, catálogo de ambientes no productivos disponibles y resolución a problemas más comunes.	Socializar sobre este repositorio digital e incentivar el uso y aporte entre los miembros del equipo.
Infraestructura Base	Gestión de ambientes no productivos	3	4	5	Actualmente se definen y se crean ambientes no productivos para las aplicaciones en coordinación con el área de Infraestructura Tecnológica.	Implementar capacidades de DevOps (cultura, prácticas, herramientas).
	Promedio	**1,82**	**3,55**			

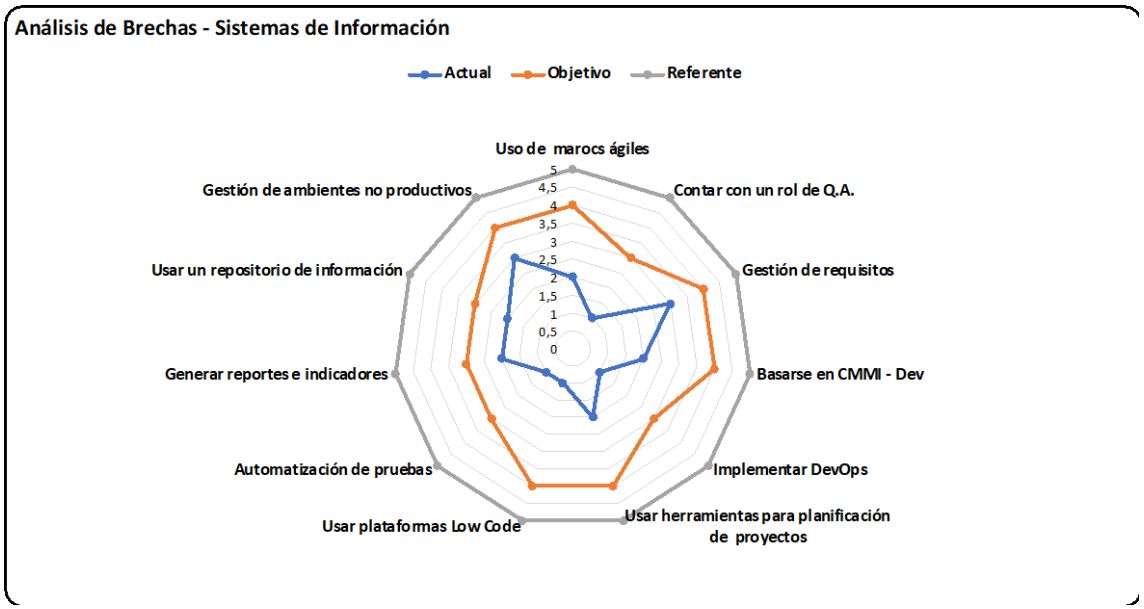


Figura 2.10 Análisis de brechas

De la

Tabla 2.2 se puede concluir los siguientes resultados:

- Promedio arquitectura actual: 1,82
- Promedio arquitectura objetivo: 3,55

Según la rúbrica utilizada, la arquitectura actual si bien tiene una valoración cercana a “Administrado”, ha sido suficiente para dar soporte a los requerimientos institucionales hasta el momento. Pero se evidencia una necesidad de implementar las iniciativas de cierres de brechas planteados en la

Tabla 2.2 para llegar en el mediano plazo a un nivel cercano a “Administrado cuantitativamente” lo que permitirá apuntalar las crecientes demandas tecnológicas de la organización.

2.4 Arquitectura objetivo

Esta es la arquitectura a la que se estima llegar basándose en los análisis anteriores.

2.4.1 Target de arquitectura de negocio

Después de determinar el problema y realizar el análisis de brechas se concluyó ciertas falencias en las que es necesario tomar acciones para crear procesos o mejorar los existentes. Dentro de la arquitectura de negocio se definirán procesos requeridos para realizar el fortalecimiento de la Fábrica de *Software* de la UDLA y estos se basarán en:

- CMMI DEV.
- Agile DevOps.
- Contar con un equipo de QA.

En la **Figura 2.11** se representa el proceso para el desarrollo basado en CMMI, Scrum y DevOps

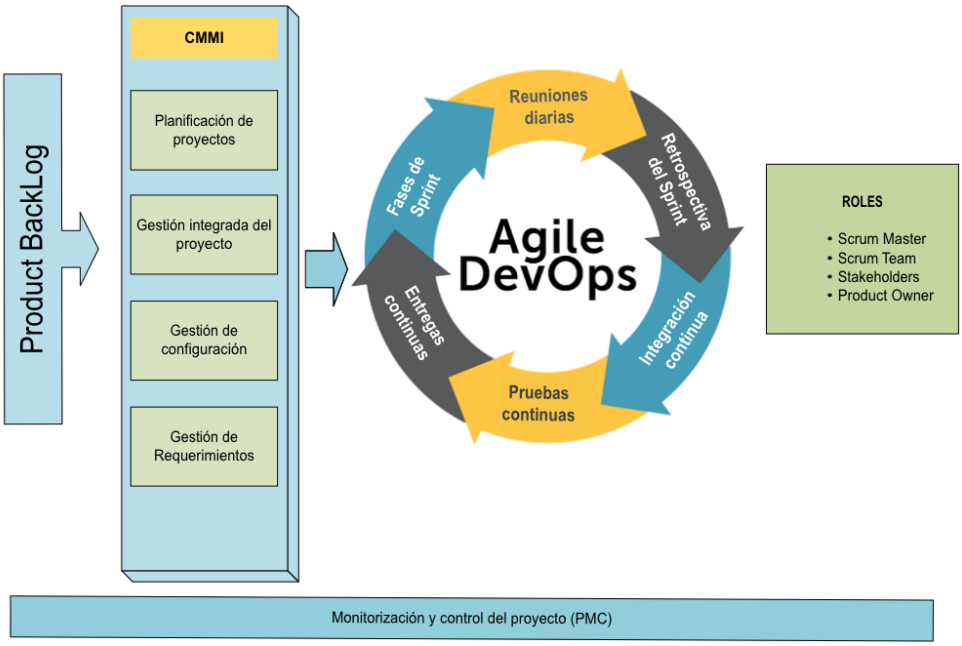


Figura 2.11 Modelo de negocio

2.4.2 Target de arquitectura de datos

En la arquitectura de datos para el objetivo que se persigue, se requieren tener siguientes características:

- Reportes, indicadores y KPIs de esfuerzos por analista.
- Repositorio para compartir estándares, buenas prácticas y una base de conocimientos al que tengan acceso los miembros del equipo.

Se deberá apoyar en la solución Jira de *Atlassian* que contiene módulos de reportería en base a la información de los proyectos que se manejen en dicha herramienta.

2.4.3 Target de arquitectura de aplicaciones

La arquitectura de aplicaciones describe las herramientas que se pueden utilizar o implementar para mejorar los procesos de desarrollo y despliegue. Se realiza una clasificación en subgrupos de aplicaciones y se diferencia entre aplicaciones existentes y por implementar como se muestra en la **Figura 2.12**.

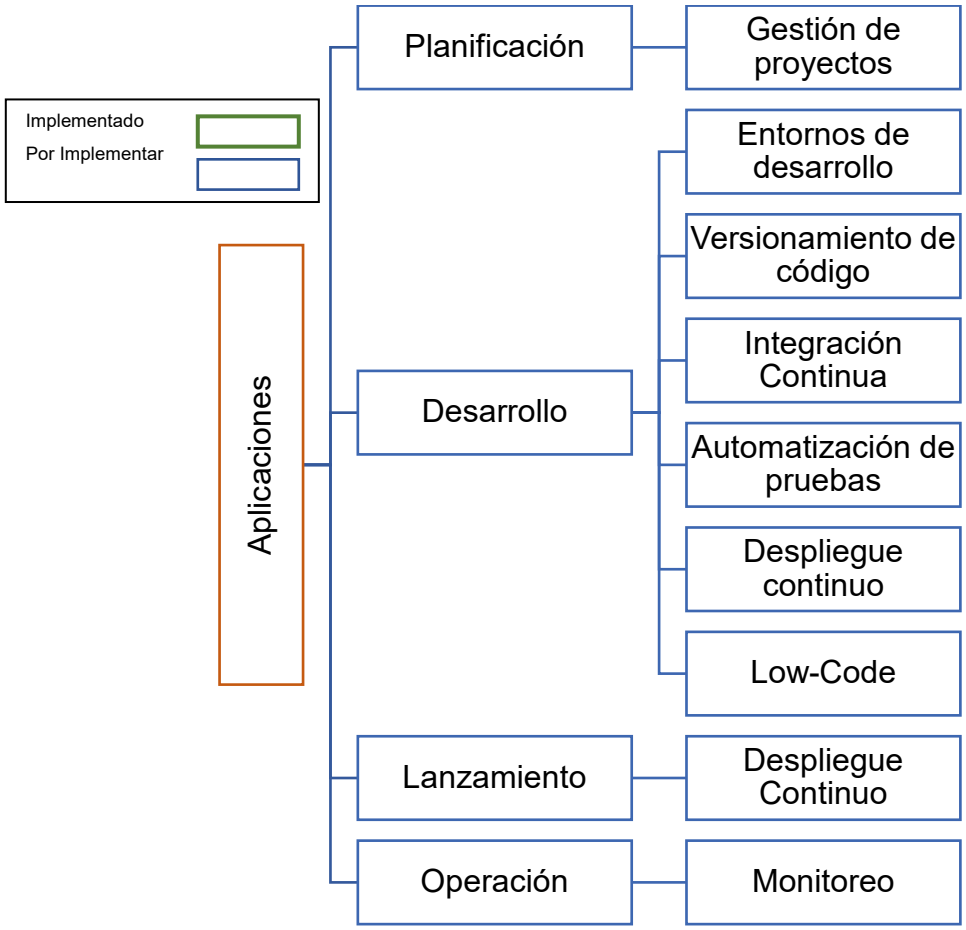


Figura 2.12 Arquitectura de aplicaciones

2.4.4 Target de arquitectura de infraestructura base

En esta arquitectura se describen los componentes base de hardware y *software* para soportar los requerimientos. En la actualidad, el área de Infraestructura Tecnológica es quien nos brinda el servicio para tener la infraestructura necesaria y poder desplegar las aplicaciones. Se cuenta con la siguiente arquitectura on-premise:

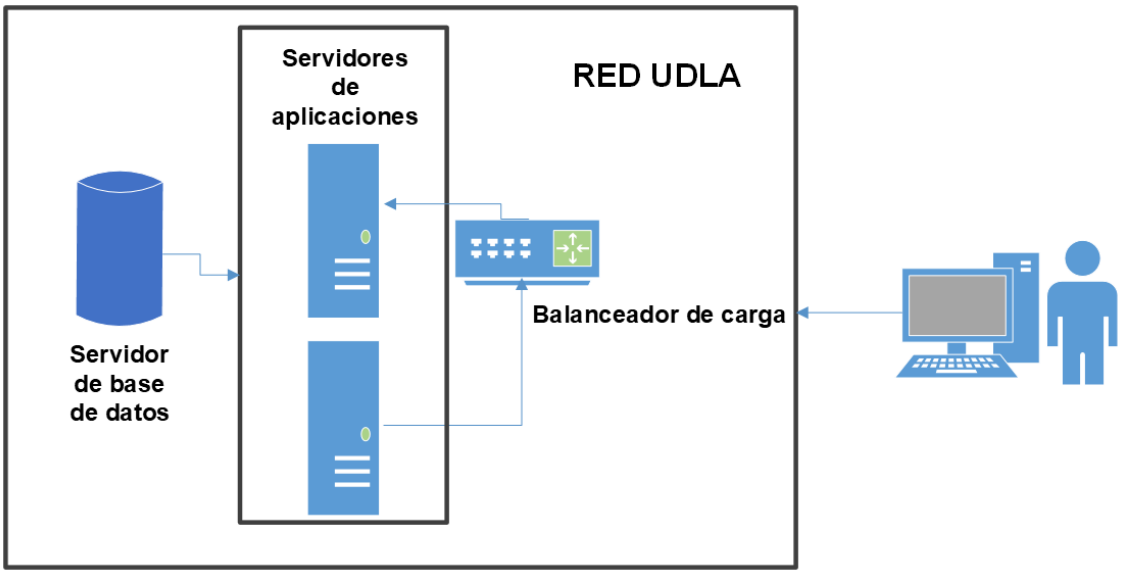


Figura 2.13 Arquitectura infraestructura para aplicaciones UDLA

Por lo tanto, la arquitectura de infraestructura ya está disponible, pero teniendo en cuenta el catálogo de principios establecidos para este ejercicio de Arquitectura Empresarial, las aplicaciones futuras deben ser orientadas a desplegarse en ambientes basado en la nube, de tal forma que los nuevos desarrollos deben ser realizados con frameworks diseñados para este fin. El área de Sistemas de Información de la UDLA utiliza frameworks de Microsoft para actividades de desarrollo. En este sentido se establece las siguientes actividades:

- Migración de aplicaciones basadas en Net Framework a Net 6. En caso de no poder realizarlo, se debe usar la última versión de Net Framework.
- Los nuevos desarrollos deben ser hechos en el Framework Net 6.

3 Arquitectura de Negocio

3.1 Business Model Canvas

3.1.1 Definición

Es un instrumento que proporciona una visión global para la innovación, desafíos de mercado, identificación de marca, proyectándose a corto y mediano plazo permitiendo flexibilidad visual de elementos. Esto permite dar valor a las ideas de negocio dándole entendimiento a las relaciones entre las áreas que intervienen.

3.1.2 Modelo Canvas de Sistema de Información

Se va a definir brevemente cada elemento con la intención de clarificar el modelo basado en las actividades que se van a desarrollar en la Fábrica de Software de la UDLA también conocido como Área de Sistemas de Información.

3.1.2.1 Propuesta de valor

La propuesta de valor que ofrece el Área de Sistemas de Información es el mejoramiento continuo de los procesos a través de soluciones tecnológicas que faciliten y automaticen los procesos operativos de las áreas de UDLA cumpliendo altos estándares de calidad que satisfagan los requerimientos planteados.

3.1.2.2 Segmento de Clientes

Los clientes objetivos a quienes Sistemas de Información se va a enfocar a atender y brindar servicios son las áreas administrativas y académicas de UDLA. Para esto se genera la Figura 3.1 Distribución de las áreas en UDLA**Figura 3.1** que indica la distribución de las áreas basado en un total de 29.

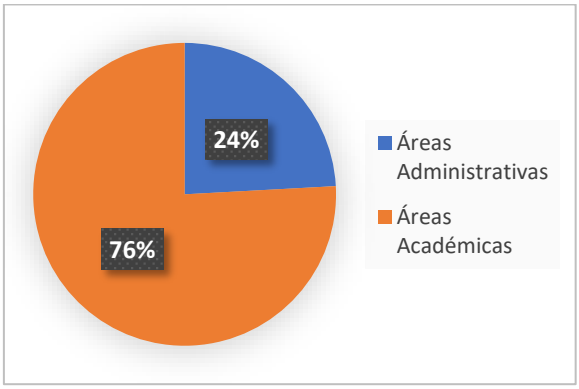


Figura 3.1 Distribución de las áreas en UDLA

3.1.2.3 Canales de distribución

Se indica la estructura de negocio que va desde el origen del producto hasta el consumidor. Una vez desarrollado un nuevo sistema, o posterior a realizar un cambio en un sistema existente, se debe pasar a los ambientes productivos para que esté disponible a los usuarios funcionales. El Comité de Paso a Producción es el encargado de analizar y determinar el impacto junto con los posibles riesgos que implican los cambios. Una vez aprobado por el comité, el Área de Producción toma posesión de los cambios y los administra en los ambientes productivos. En este sentido, se determina los siguientes canales de distribución:

- Comité de paso a producción
- Área de producción.

3.1.2.4 Relación con los clientes

En esta instancia se identifican los canales por los que llegan los requerimientos desde las áreas funcionales de la Universidad al Área de Sistemas de Información. Las áreas funcionales generan documentos de especificación de requerimientos a alto nivel que llegan a través de los siguientes canales:

- **SysAid**. Es un sistema del tipo ITSM (*Information Technology Service Management*) que principalmente se usa para la administración de la mesa de servicios tecnológicos que gestiona y formaliza los requerimientos de trabajo a nivel de software y hardware.
- **Correo electrónico institucional.** Mediante el envío de un correo electrónico que adjunte el documento de requerimiento, se formaliza la petición.

3.1.2.5 Ingresos

La fuente de ingresos económicos para el Área de Sistemas de Información se basa principalmente en el presupuesto que la universidad pueda asignar al departamento de TI como tal.

3.1.2.6 Recursos clave

Como recurso clave para el éxito de cualquier organización, se identifica principalmente al talento humano con el que se cuenta dentro del área. Aportan

conocimiento y experiencia en temas de tecnologías de la información. Se identifican los siguientes recursos clave:

- **Equipo de desarrollo y líderes de equipos:** los colaboradores que desempeñan las diferentes labores dentro de la Fábrica de Software que aportan valor en el proceso de desarrollo. Se hace énfasis del capital humano, debido a la gran importancia que tienen dentro de las organizaciones como un factor clave preponderante para el éxito de estas. Se requiere que los colaboradores del área cuenten con las denominadas habilidades duras que refuercen las capacidades técnicas que pueda ofrecer en su conjunto el área. Pero así mismo se requiere que cuenten con habilidades blandas como liderazgo, comunicación, inteligencia emocional, empatía, resiliencia, ética, entre otros.
- **Herramientas de desarrollo de software:** Estas herramientas deben estar basadas en los frameworks de Microsoft por estándares de desarrollo que se manejan en la Fábrica de Software.
- **Herramientas de integración y despliegue continuo:** Herramientas del tipo DevOps que permitan facilitar procesos de pruebas, compilación y despliegue continuo.
- **Herramienta de control de código fuente:** Herramientas para gestionar el código creado por los equipos de desarrollo.
- **Arquitectura de ambientes productivos y no productivos:** Ambientes necesarios para el correcto despliegue de las aplicaciones desarrolladas, se necesita contar con ambientes en donde los equipos de pruebas puedan validar las funcionalidades y finalmente los ambientes en donde se puedan desplegar para el uso de los usuarios finales.
- **Servicios en la Nube:** Contar con infraestructura en la nube del tipo software como servicio (SaaS), plataforma como servicio (PaaS) e infraestructura como servicio (IaaS). Esto se vuelve necesario para poder llevar todos los elementos de trabajo de la Fábrica de Software a la nube y aprovechar las ventajas que brinda la Nube.

3.1.2.7 Actividades clave

Como propuesta de valor que se puede brindar al cliente interno de UDLA se identifica:

- **Innovación.** Se cuenta con un equipo de trabajo dedicado a la investigación, desarrollo e innovación en temas tecnológicos. Esto ha permitirá que se realicen pruebas de concepto y productos mínimos viables con temas de actualidad que aporten valor a la institución.
- **Capacitación continua.** En un mundo tecnológico la constante es el cambio, por lo que es necesario que el talento humano del departamento continue capacitándose y aprendiendo nuevos temas que aporten directamente a las actividades diarias dentro del área.
- **Procesos estratégicos, misionales y de apoyo de la fábrica.** Procesos descritos en el mapa de procesos que son la base fundamentar del área.

3.1.2.8 Alianzas clave

Se identifica las siguientes alianzas clave:

- Proveedores de servicios de outsourcing.
- Proveedores de software como servicio.
- Proveedores de servicios tecnológicos.

3.1.2.9 Costos

Los costos principales dentro del área son:

- Capacitaciones.
- Sueldos y honorarios.
- Pagos a proveedores de servicios outsourcing, proveedores de software como servicio, proveedores de servicios tecnológicos.
- Pago de licenciamientos de herramientas software.

3.1.3 Representación gráfica del modelo Canvas del Área de Sistemas de Información

Una vez descritos los componentes del modelo Canvas, se muestra la **Figura 3.2** que sintetiza la información que se detalló en las secciones previas:

Figura 3.2 Aplicación gráfica del modelo Canvas

3.2 Arquitectura de negocio objetivo

3.2.1 Mapa de procesos

En la **Figura 3.3** se plantea el siguiente mapa de procesos para la fábrica de software UDLA:

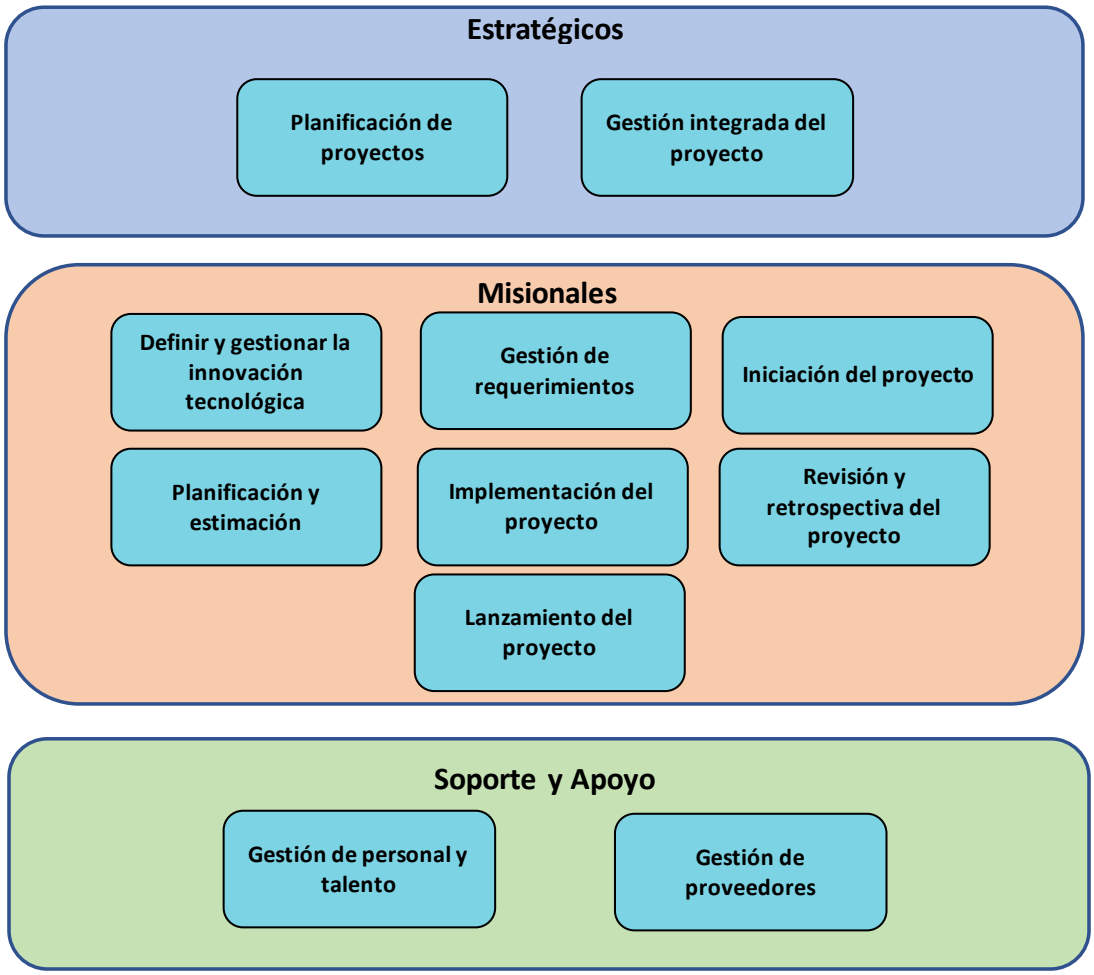


Figura 3.3 Mapa de procesos

3.2.2 Detalle de procesos

3.2.2.1 Planificación de proyectos

Una de las claves para la gestión eficaz de proyectos es la planificación. En la **Tabla 3.1** se muestra la descripción del proceso de planificación de proyectos.

Tabla 3.1 Proceso planificación de proyectos

Nombre del proceso:	**Planificación de proyectos**	
Objetivo:	Establecer y mantener planes que definan las actividades del proyecto.	
Responsable:	Product owner, stakeholders, scrum master	
Entradas	**Actividades**	**Salidas**
• Caso de negocio	Establecer las estimaciones de los atributos de los productos de trabajo y de las tareas	• Descripciones de las tareas
• Descripciones de las tareas.	Definir las fases del ciclo de vida del proyecto	• Estimaciones de los atributos
	Estimar el esfuerzo y el costo	• Fases del ciclo de vida del proyecto
	Desarrollar un plan de proyecto	• Estimaciones del esfuerzo del proyecto.
	Establecer el presupuesto y el calendario	• Calendarios del proyecto
• Proyecto	Identificar los riesgos del proyecto	• Impactos y probabilidad de ocurrencia de los riesgos
	Planificar los recursos del proyecto	• Paquetes de trabajo
	Planificar la involucración de las partes interesadas	• Plan para la involucración de las partes interesadas
	Establecer el plan de proyecto	• Plan global del proyecto
	Obtener el compromiso con el plan	• Registro de las revisiones de los planes que afectan al proyecto

Para establecer el plan de proyecto, puede ser necesario realizar iteraciones de estas actividades.

El plan de proyecto se modifica generalmente a medida que el proyecto progresa, para abordar los cambios en los requisitos y en los compromisos, las estimaciones inexactas, las acciones correctivas y los cambios a los procesos.

El plan de proyecto puede ser un documento independiente o puede estar distribuido en múltiples documentos.

3.2.2.2 Gestión integrada del proyecto

La gestión del esfuerzo, el costo, el cronograma, las personas, el riesgo y otros factores del proyecto está relacionada con las tareas en el proceso de gestión integrada del proyecto. En la

Tabla 3.2 se muestra el detalle del proceso.

Tabla 3.2 Proceso gestión integrada del proyecto

Nombre del proceso:	**Gestión integrada del proyecto**	
Objetivo:	Establecer y gestionar el proyecto y la involucración de las partes interesadas relevantes de acuerdo con un proceso integrado y definido, que se adapta a partir del conjunto de procesos estándar de la organización	
Responsable:	Product owner, stakeholders, scrum master	
Entradas	**Actividades**	**Salidas**
• Plan del proyecto	Establecer el proceso definido del proyecto	• Proceso definido del proyecto
	Utilizar los activos de proceso de la organización para planificar las actividades del proyecto	• Estimaciones de proyecto
• Requerimientos del proyecto	Establecer el entorno de trabajo del proyecto.	• Equipamiento y herramientas para el proyecto
	Establecer los equipos	• Lista de miembros asignados a cada equipo
	Gestionar la involucración de las partes interesadas.	• Agendas y calendarios para las actividades de colaboración
• Plan del proyecto	Resolver las cuestiones de coordinación	• Compromisos para tratar las dependencias críticas

El proceso definido para cada proyecto se adapta a partir del conjunto de procesos estándar de la organización, la variabilidad entre los proyectos normalmente se reduce y los proyectos pueden compartir fácilmente los activos de proceso, los datos y las lecciones aprendidas.

3.2.2.3 Gestión de requerimientos

El proceso de gestión de requisitos maneja todos los requisitos recibidos o generados por un proyecto, incluidos los requisitos técnicos y no técnicos, así como los requisitos que la organización define para el proyecto. En la **Tabla 3.3** se muestra el detalle del proceso de gestión de requerimientos.

Tabla 3.3 Proceso gestión de requerimientos

Nombre del proceso:	**Gestión de requerimientos**	
Objetivo:	Gestionar los requisitos de los productos y los componentes de producto del proyecto, asegurar la alineación entre esos requisitos, los planes y los productos de trabajo del proyecto	
Responsable:	Product owner, stakeholders, scrum master	
Entradas	**Actividades**	**Salidas**
• Solicitud de requerimientos	Comprender los requisitos.	• Criterios para la evaluación y la aceptación de los requisitos • Criterios para distinguir a los proveedores apropiados de requisitos
	Obtener el compromiso sobre los requisitos.	• Informes de impacto del cambio de requisitos
	Gestionar los cambios a los requisitos.	• Petición de cambio de requisitos
• Lista de requisitos	Asegurar el alineamiento entre el trabajo del proyecto y los requisitos.	• Documentación de inconsistencias entre los requisitos, los planes del proyecto y los productos de trabajo, incluyendo fuentes y condiciones

En este proceso se realizan los pasos apropiados para asegurar que el conjunto de requisitos aprobados se gestiona para dar soporte a las necesidades de planificación y de ejecución del proyecto.

Los proyectos gestionan el cambio a medida que evoluciona e identifican las discrepancias que surgen entre los planes, los productos de trabajo y los requisitos. En el caso de las actividades de mantenimiento, los cambios se basan en cambios en los requisitos, el diseño o la implementación existentes. Los cambios en los proyectos de creación de capacidad de productos también están relacionados con las necesidades cambiantes de los clientes, la madurez u obsolescencia de la tecnología y los estándares cambiantes.

3.2.2.4 Definir y gestionar la innovación tecnológica

Proceso para llevar a cabo una vigilancia tecnológica sobre las nuevas tendencias y tecnologías emergentes que puedan tener un impacto directo la organización. Sus principales objetivos son:

- Investigar sobre tecnologías disruptivas que puedan ayudar a los procesos académicos y administrativos.
- Gestionar pruebas de concepto, prototipos y productos mínimos viables que demuestren la aplicabilidad de las tecnologías investigadas.
- Realizar un traspaso de conocimientos a los equipos de desarrollo.

En la

Tabla 3.4 se describe el proceso.

Tabla 3.4 Proceso definir y gestionar la innovación tecnológica

Nombre del proceso:	**Definir y gestionar la innovación tecnológica**	
Objetivo:	Búsqueda de conocimiento original y desconocido hasta el momento, relacionado principalmente con la ciencia y la tecnología, con el objetivo de desarrollar o mejorar una solución a un problema existente.	
Responsable:	Equipo de desarrollo	
Entradas	**Actividades**	**Salidas**
• Lluvia de ideas	Establecer criterios de selección para iniciativas de investigación	• Pruebas de concepto
• Investigación básica	Analizar conceptos tecnológicos emergentes	• Prototipos
• Investigación aplicada	Identificar conceptos y capacidades tecnológicas	• Documentación de la investigación
• Necesidades del negocio	Ejecutar proyectos de investigación de TI	• Productos mínimos viables (MVP)
• Pruebas de concepto, Prototipos	Identificar y promover conceptos viables	
• Pruebas de concepto, Prototipos	Transferencia de conocimiento a equipos de desarrollo	• Sesiones de capacitaciones

3.2.2.5 Iniciación del proyecto

El objetivo principal es crear la visión del proyecto que sirve de enfoque y dirección. Se identifican roles claves del proyecto como el *Scrum Master*, *Product Owner*, *Stakeholders* y el *Scrum Team*. Se desarrollan las épicas tomando como base la visión del proyecto. Finalmente, se crea el *Product Backlog* que sirve de base para la elaboración del plan de lanzamiento y tamaño de cada *Sprint*.

Actividades clave

a) **Crear la visión del proyecto.**

Se revisa el caso de negocio del proyecto a fin de crear una declaración de la visión del proyecto, que servirá de inspiración y proporcionará un enfoque para todo el proyecto. En este proceso se identifica al *Product Owner*.

b) Identificar al *Scrum Master* y a los *stakeholders*.

Se identifica al *Scrum Master* y *stakeholders* utilizando criterios de selección específicos.

c) Formación del equipo *Scrum*.

Se identifican a los miembros del Equipo *Scrum*. El *Product Owner* en colaboración con el *Scrum Master* son los responsables de la selección de los miembros del equipo.

d) Desarrollo de épicas.

La declaración de visión del proyecto sirve como base para el desarrollo de épicas que se llevan a cabo reuniones de grupos de usuarios.

e) Creación del backlog.

Se refinan las épicas para priorizarlas y crear un *Backlog* priorizado del proyecto.

f) Realizar el plan de lanzamiento.

El equipo *Scrum* revisa las historias de usuario del *backlog* priorizado para desarrollar un cronograma de planificación del lanzamiento, en donde se especifica la implementación por fases que se puede compartir con los *stakeholders* del proyecto. (Alvarado, 2021).

En la

Tabla 3.5 se indica la descripción del proceso.

Tabla 3.5 Proceso iniciación del proyecto

Nombre del proceso:	Iniciación del proyecto	
Objetivo:	Crear la visión del proyecto, identificar roles claves del proyecto, desarrollar las épicas tomando como base la visión del proyecto y crea el Product Backlog	
Responsable:	Product owner, scrum master, Equipo de desarrollo, stakeholders	
Entradas	**Actividades**	**Salidas**
• Caso de negocio del proyecto	Crear la visión del proyecto	• Visión del proyecto • Se identifica al Product Owner
• Especificación de criterios de selección	Identificar al Scrum Master y a los stakeholders	• Se identifica al Scrum Master y stakeholders
	Formación del equipo Scrum	• Se identifican a los miembros del Equipo Scrum
• Declaración de visión del proyecto	Desarrollo de épicas	• Listado de épicas
• Listado de épicas	Creación del backlog	• Backlog priorizado del proyecto
• Backlog priorizado del proyecto	Realizar el plan de lanzamiento	• Cronograma de planificación del lanzamiento • Definición de duración del sprint

3.2.2.6 Planificación y estimación

Se definen los *Sprints,* así como las historias de usuarios que generan valor a la organización. Se hacen las estimaciones de tiempo y esfuerzo que se traducen en listas de tareas cuyos tiempos de desarrollo se definen en reuniones de equipo correspondientes y el proceso de definición del *Sprint Backlog* (Salazar, 2016).

Actividades clave

a) Elaborar historias de usuario.

Se crean las historias de usuario y los criterios de aceptación de estas (Alvarado, 2021). Generalmente las escribe el *Product Owner* y están diseñadas para asegurar que los requisitos del cliente estén claramente representados y puedan ser plenamente comprendidos por todos los *stakeholders*.

b) Estimar y asignar historias de usuarios.

El *Product Owner* aclara las historias de usuario para que el *Scrum Master* y el Equipo *Scrum* puedan estimar el esfuerzo necesario para desarrollar la funcionalidad descrita en cada una de ellas. Estas se asignan al Equipo *Scrum* y se compromete a entregar al *Product Owner* las historias de usuario aprobadas para un *sprint*. El resultado de este proceso serán las historias de usuario comprometidas. (Alvarado, 2021)

c) Elaboración de tareas.

Las historias de usuario comprometidas se detallan en tareas específicas y se registran en una lista de tareas.

d) Estimar tareas.

El equipo *Scrum* estima el esfuerzo necesario para cumplir con cada tarea de la lista de tareas. El resultado de este proceso es un *Effort Estimated Task List*.

e) Elaboración de la lista de pendientes del *Sprint*.

El equipo *Scrum* elabora un *Sprint Backlog* que contiene todas las tareas a ser completadas en un *sprint* como parte de la Reunión de Planificación del *Sprint*.

En la

Tabla 3.6 se indica la descripción del proceso.

Tabla 3.6 Proceso planificación y estimación

Nombre del proceso:	**Planificación y Estimación**	
Objetivo:	Definir los Sprints y las historias de usuarios.	
Responsable:	Scrum master, equipo de desarrollo	
Entradas	**Actividades**	**Salidas**
• Requerimientos del cliente expresados por el product owner	Elaborar historias de usuario	• Historias de usuario • Criterios de aceptación
• Lista de las historias de usuario	Estimar y asignar historias de usuarios	• Estimación del esfuerzo de las historias de usuario
• Lista de las historias de usuario	Elaboración de tareas	• Lista de tareas por historia de usuario
• Lista de tareas por historia de usuario	Estimar tareas	• Lista del esfuerzo estimado por tareas
• Lista del esfuerzo estimado por tareas	Crear el Sprint Backlog	• Sprint Backlog

3.2.2.7 Implementación del proyecto

Se trabaja en las tareas del *Sprint Backlog* para crear *Sprint Deliverables*. Se realizan los *Daily Standup Meeting* y su avance se registra en tableros.

Actividades clave

a) **Crear entregables.**

En esta actividad el Equipo *Scrum* trabaja en las tareas en el *Sprint Backlog* para crear los entregables del *sprint*. Se utiliza tableros para dar seguimiento a las actividades que se llevan a cabo clasificándolas en: pendientes, en curso y hechas.

b) **Llevar a cabo el *standup* diario.**

Reuniones diarias altamente focalizadas y conocida también como *Daily Standup*. Aquí es donde los miembros del Equipo *Scrum* se actualizan el uno al otro referente a sus progresos y sobre los impedimentos que pudieran enfrentar. Se responde principalmente a tres preguntas: ¿Qué hiciste ayer?, ¿Qué vas a hacer hoy?, ¿Qué problemas tienes?

c) **Mantenimiento de la lista priorizada de pendientes del producto.**

En esta actividad se trata de ir actualizando y refinando constantemente el *Backlog* Priorizado del Producto en base a un análisis de cualquier cambio o actualización al *backlog* para incorporar actividades según sea necesario.

En la

Tabla 3.7 se indica la descripción del proceso.

Tabla 3.7 Implementación del proyecto

Nombre del proceso:	**Implementación del proyecto**	
Objetivo:	Trabaja en las tareas del Sprint Backlog para crear Sprint Deliverables.	
Responsable:	Product owner, equipo de desarrollo, analista de calidad	
Entradas	**Actividades**	**Salidas**
• Sprint Backlog	Crear entregables	• Entregables del sprint
• Retroalimentación del equipo	Llevar a cabo el standup diario	• Seguimiento diario
• Sprint Backlog	Mantenimiento de la lista priorizada de pendientes del producto	• Sprint Backlog

3.2.2.8 Revisión y retrospectiva del proyecto

En esta etapa se llevan a cabo reuniones de revisión del producto a entregar y retrospectiva del proyecto.

Actividades clave

a) **Demostración y validación del *Sprint*.**

El Equipo *Scrum* muestra los entregables funcionales del *sprint* al *Product Owner* y a los *stakeholders* relevantes en una Reunión de Revisión del *Sprint*. El objetivo principal de esta reunión es asegurar que se obtenga la aprobación y aceptación por parte del *Product Owner* de los entregables elaborados en el *sprint*.

b) Retrospectiva de Sprint.

Reunión en la que el *Scrum Master* y el Equipo *Scrum* analizan lo que se hizo bien y mal durante todo el *Sprint*. Esta información se documenta en forma lecciones aprendidas que se pueden aplicar a futuros *Sprints*.

En la **Tabla 3.8** se indica la descripción del proceso.

Tabla 3.8 Revisión y retrospectiva del proyecto

Nombre del proceso:	**Revisión y retrospectiva del proyecto**	
Objetivo:	Mostrar al propietario del producto y a los socios relevantes el resultado del trabajo del sprint.	
Responsable:	Product owner, equipo de desarrollo, stakeholders, analista de calidad	
Entradas	**Actividades**	**Salidas**
• Entregables del sprint	Demostración y validación del Sprint.	• Aprobación y aceptación del Product Owner
• Análisis de los temas positivos y negativos durante el sprint	Retrospectiva de Sprint	• Documento de lecciones aprendidas

3.2.2.9 Lanzamiento del proyecto

En este proceso se combina e involucra prácticas y herramientas DevOps para facilitar y automatizar el paso de un software que esté en etapa de desarrollo a los diferentes ambientes no productivos y productivos. Se usan herramientas para gestionar, programar y automatizar las tareas de paso a producción de las aplicaciones.

Actividades clave

a) Integración continua

Es una técnica que pretende detectar posibles problemas que pueda tener el software de forma temprana permitiendo solucionarlo antes que sea demasiado tarde. Se realiza de forma periódica (varias veces al día) compilaciones de código, ejecución de pruebas unitarias, revisión de la calidad del código y detección de vulnerabilidades.

b) Pruebas continuas

Establece la capacidad de realizar pruebas automatizadas y programadas previamente para que se realicen mientras el código de la aplicación se está creando o modificando.

c) **Entregas continuas**

La entrega continua está relacionada con el despliegue automatizado de versiones de software en el entorno de producción elegido.

En la

Tabla 3.9 se indica la descripción del proceso.

Tabla 3.9 Proceso lanzamiento del proyecto

Nombre del proceso:	**Lanzamiento del proyecto**	
Objetivo:	Lanzar a los ambientes productivos los entregables aprobados por el product owner.	
Responsable:	Equipo de desarrollo, analista de calidad	
Entradas	**Actividades**	**Salidas**
• Entregables del sprint	Integración continua	• Código compilado y probado
• Código compilado y probado	Pruebas continuas	• Código compilado y probado
• Código compilado y probado	Entregas continuas	• Despliegue de aplicaciones en ambientes

3.2.2.10 Gestión de personal y talento

Este proceso está diseñado para garantizar que las personas con las habilidades adecuadas proporcionen el servicio que se necesita dentro del área. Se busca reducir los retrasos, mejorar la calidad, eliminar la repetición del trabajo por defectos, acortar el tiempo de entrega, abordar las brechas de conocimiento y habilidades. Adquirir el talento adecuado se está volviendo crítico a medida que las organizaciones transforman sus prácticas y capacidades organizacionales y de automatización para respaldar la economía digital y aumentar la velocidad de comercialización. En la **Tabla 3.10** se indica la descripción del proceso.

Tabla 3.10 Proceso gestión de personal y talento

Nombre del proceso:	**Gestión de personal y talento**	
Objetivo:	Garantizar que la organización cuente con las personas adecuadas con las habilidades y el conocimiento adecuados y en los roles correctos para respaldar sus objetivos comerciales.	
Responsable:	Scrum master	
Entradas	**Actividades**	**Salidas**
• Estructura del área	Traducir la estrategia y los objetivos de la organización en las capacidades organizativas deseadas y luego en competencias y roles	• Perfil de personal
• Diagnóstico de necesidad de capacitación	Capacitación dirigida y oportunidades de aprendizaje experiencial utilizando varios métodos formales y no formales	• Personal capacitado
	Autocapacitación haciendo uso por parte de un empleado de los roles laborales publicados y los marcos de competencia para planificar de manera proactiva el crecimiento y el avance personal	• Personal capacitado
• Necesidad de reemplazo	Actividades formales de planificación de sucesión, participación y tutoría proporcionadas por el liderazgo	• Personal capacitado

3.2.2.11 Gestión de proveedores

Este proceso busca gestionar las relaciones con los proveedores de servicios en los que confían las organizaciones de TI. Su principal objetivo es conseguir la máxima calidad a un precio razonable. Para ello, y teniendo siempre en cuenta las directrices de la estrategia de servicio.

Tabla 3.11 Gestión de proveedores

Nombre del proceso:	Gestión de Proveedores	
Objetivo:	Gestionar los requisitos de los productos y los componentes de producto del proyecto, asegurar	
Responsable:	Scrum master	
Entradas	**Actividades**	**Salidas**
• Requerimiento de área o usuarios	Identificar las necesidades de los clientes	• Requerimiento para adquisición de software o servicio
• Requerimiento para adquisición de software o servicio	Evaluar proveedores	• Lista de posibles proveedores
• Lista de posibles proveedores	Reclutar nuevos proveedores	• Contacto con proveedores
• Lista de actividades a cumplir	Gestionar rendimiento de los proveedores	• Evaluación a proveedores
• Lista de problemas reportados	Gestionar problemas con proveedores	• Solución de inconvenientes

3.3 Estructura organizativa

En la arquitectura objetivo, se plantea tener los siguientes roles que serán distribuidos al personal actual que es parte del área y de UDLA. La estructura organizativa actual para el Área de Sistemas de Información se indica en el primer capítulo en la **Figura 1.2 Organigrama TI – UDLA,** mientras que la estructura organizativa propuesta se muestra en la **Figura 3.4.**

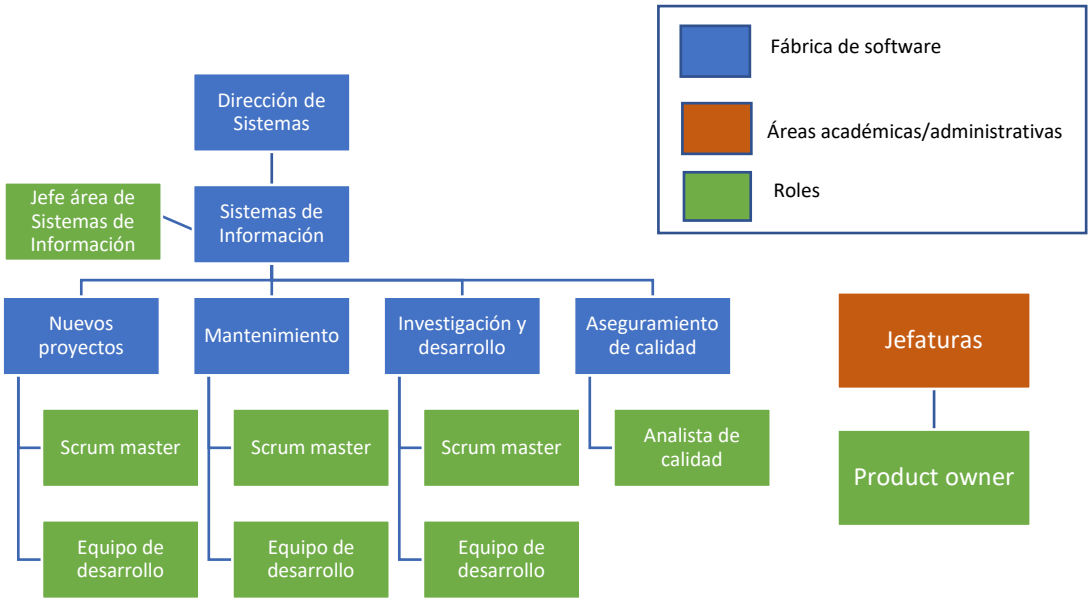


Figura 3.4 Organigrama TI Objetivo

En la **Tabla 3.12** se estable una matriz que relaciona los roles propuestos junto con los procesos en los que participarían cada uno.

Tabla 3.12 Mapeo de roles y procesos

Procesos \ Roles	Jefe área de sistemas de información	Stakeholder	Scrum Master	Product Owner	Equipo de desarrollo	Analista de calidad
Planificación de proyectos	A	R	R	R		
Gestión integrada del proyecto	A	R	R	R		
Definir y gestionar la innnovación tecnológica			A		R	
Iniciación del proyecto	A	R	R	R	R	
Planificación y estimación			R	A	R	
Implementación del proyecto			A	R	R	R
Revisión y retrospectiva del proyecto		R	A	R	R	R
Lanzamiento del proyecto			A		R	R
Gestión de personal y talento	A		R			
Gestión de proveedores	A		R			

A continuación, se describe los roles centrales de Scrum como los roles complementarios que son necesarios para el correcto funcionamiento de la fábrica de desarrollo de software:

3.3.1 Jefe Área de Sistema de Información

Es el responsable de organizar y gestionar el Área de Sistemas de Información procurando el uso óptimo y eficiente de recursos, líder el área proyectos de desarrollo de sistemas para lograr resultados esperados en tiempo, costos y forma.

Entre sus competencias está: Conducir las actividades diarias del equipo, ejerciendo un control sobre resultados, plazos y calidad. Mantener la relación con usuarios y clientes, motivando y brindando apoyo a los integrantes del equipo y gestionando los recursos necesarios, tomando decisiones operativas para mantener los proyectos en tiempo, alcances y costo.

3.3.2 Stakeholders

También conocidos como “la parte interesada” debido a que incluye a clientes, usuarios y patrocinadores que interactúan con el equipo *scrum*. Tiene influencia durante todo el proceso de desarrollo del proyecto ya que se benefician directamente de los resultados de este.

3.3.3 Scrum Master

Es un facilitador del trabajo y líder servicial que guía y facilita para asegurar que el equipo *scrum* cuente con un ambiente propicio para completar con éxito el proyecto. Elimina impedimentos que enfrenta el equipo y se encarga de asegurar que el equipo siga los procesos ágiles recomendados. No se debe confundir con un jefe que da órdenes o controla al equipo.

3.3.4 Product Owner

Es la persona que representa la voz del cliente y está encargada de lograr obtener el máximo valor para el proyecto. También es responsable de la articulación de los requerimientos por parte de los clientes (*Product Backlog*) y de mantener la justificación del negocio. Comprende las necesidades de los *stakeholders*.

3.3.5 Scrum Team

Equipo multidisciplinario y autoorganizado que está conformado por personas responsables de entender los requerimientos especificados por el *Product Owner*. Deciden la cantidad de trabajo a realizar mediante la estimación de las historias de usuarios que se van a abordar en cada *sprint* de trabajo. Deben tener habilidades de desarrollar, realizar pruebas y garantizar la calidad del producto final (Q.A.). En su conjunto son la base para cualquier proyecto y aquí radica la importancia de contar con los miembros adecuados.

3.3.6 Analista de Calidad

Se encarga de garantizar que los procesos y actividades de aseguramiento de la calidad sobre los componentes software desarrollados sean sólidos y se basen en prácticas estándar de la industria. Sus habilidades son aplicables a todas las metodologías de prueba que se pueden entregar utilizando enfoques predictivos basados en planes y enfoques iterativos o ágiles.

Las actividades pueden incluir planificación, diseño, gestión, ejecución y presentación de informes de pruebas. Pruebas funcionales de capacidades o características de los productos. Pruebas no funcionales de cualidades tales

como rendimiento, seguridad, acceso, copia de seguridad y recuperación, archivo y retención, solidez, disponibilidad, capacidad, escalabilidad, fiabilidad, rendimiento, estrés, volumen, mantenibilidad y portabilidad.

3.4 Análisis de Brechas

A continuación, se realiza una valoración de las brechas existentes en referencia a la arquitectura de negocio actual y la objetivo. El análisis se realiza en base a los procesos definidos para CMMI, Agile DevOps y en los roles necesarios en la arquitectura objetivo.

En la

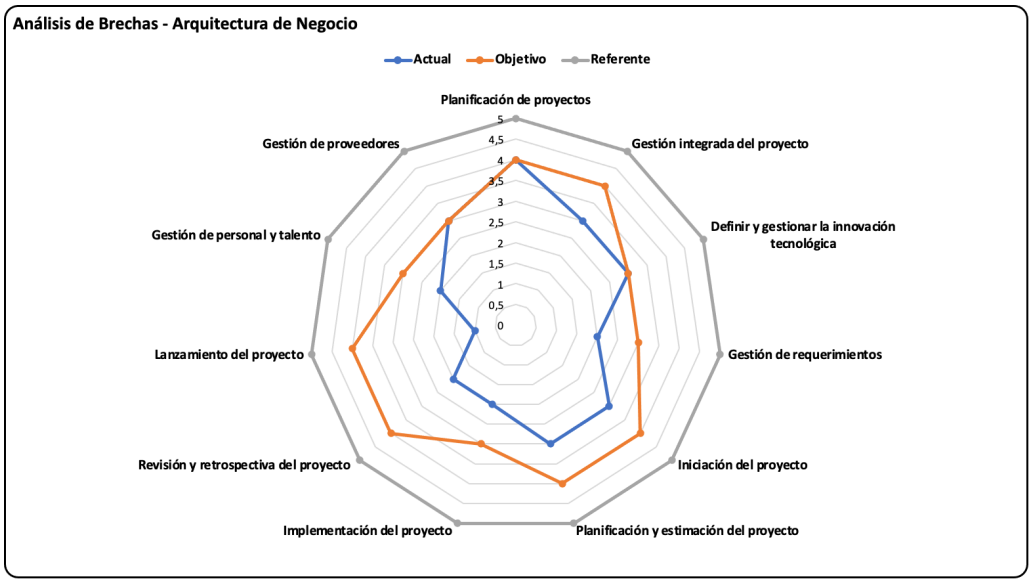


Figura 3.5 Gráfico radar de análisis de brechas

Tabla 3.13 se detalla las brechas de la arquitectura actual y objetivo.

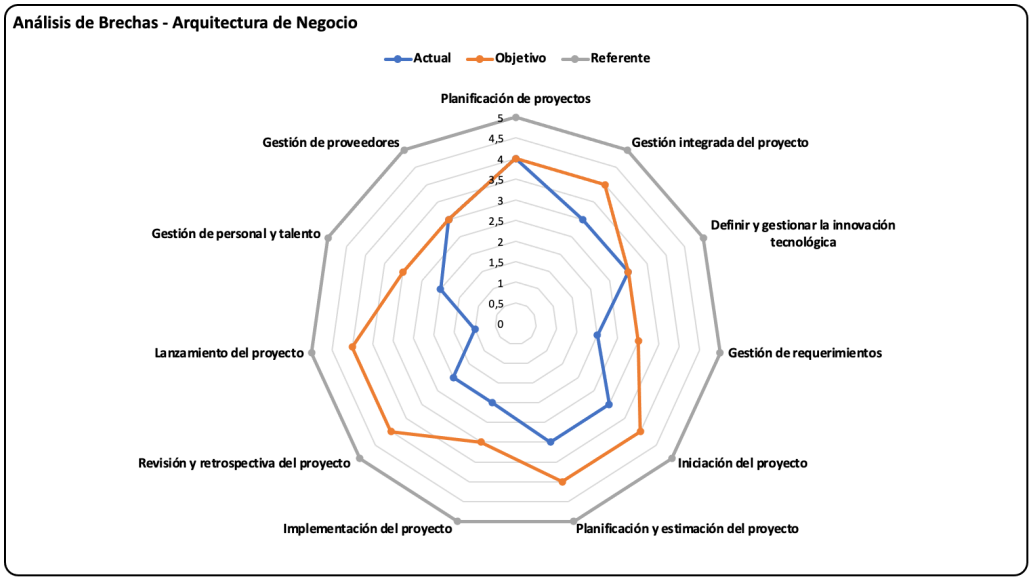


Figura 3.5 Gráfico radar de análisis de brechas

Tabla 3.13 Análisis de brechas de arquitectura actual y objetivo

Proceso	Actual	Objetivo	Referente	Brecha
Planificación de proyectos	4	4	5	• No existe brecha.
Gestión integrada del proyecto	3	4	5	• Falta de una adecuada interacción con los involucrados.
Definir y gestionar la innovación tecnológica	3	3	5	• No existe brecha.
Gestión de requerimientos	2	3	5	• Falta una adecuada documentación para los criterios de aceptación de los requerimientos
Iniciación del proyecto	3	4	5	• Deficiencias en la administración de los backlogs. • Stakeholders no se involucran como parte del equipo durante el sprint. • No está definido las personas que son scrum master y no tienen una capacitación formal. • Al scrum team le falta un perfil de Aseguramiento de calidad.
Planificación y estimación del proyecto	3	4	5	• Product Owner no se involucra como parte del equipo durante el sprint. • No se realiza una correcta estimación de los proyectos y requerimientos.
Implementación del proyecto	2	3	5	• No se aplican correctamente los sprints en los proyectos y requerimientos de desarrollo. '• No se establecen en todos los proyectos y requerimientos de desarrollo las reuniones de seguimiento diarias.
Revisión y retrospectiva del proyecto	2	4	5	• Casi nunca se realiza la reunión de retrospectiva en un sprint.
Lanzamiento del proyecto	1	4	5	• No se han implementado herramientas DevOps que faciliten la integración continua. • No se han implementado herramientas DevOps que faciliten las pruebas continuas. • No se han implementado herramientas DevOps que faciliten las entregas continuas.
Gestión de personal y talento	2	3	5	• El áre a de RRHH lleva el proceso de contratación pero quien selecciona a la persona adecuada es el jefe del área de desarrollo. No se aplica pruebas técnicas específicas para validar los conocimientos.
Gestión de proveedores	3	3	5	• No existe brecha.
	2,55	**3,55**		

De los resultados obtenidos en la

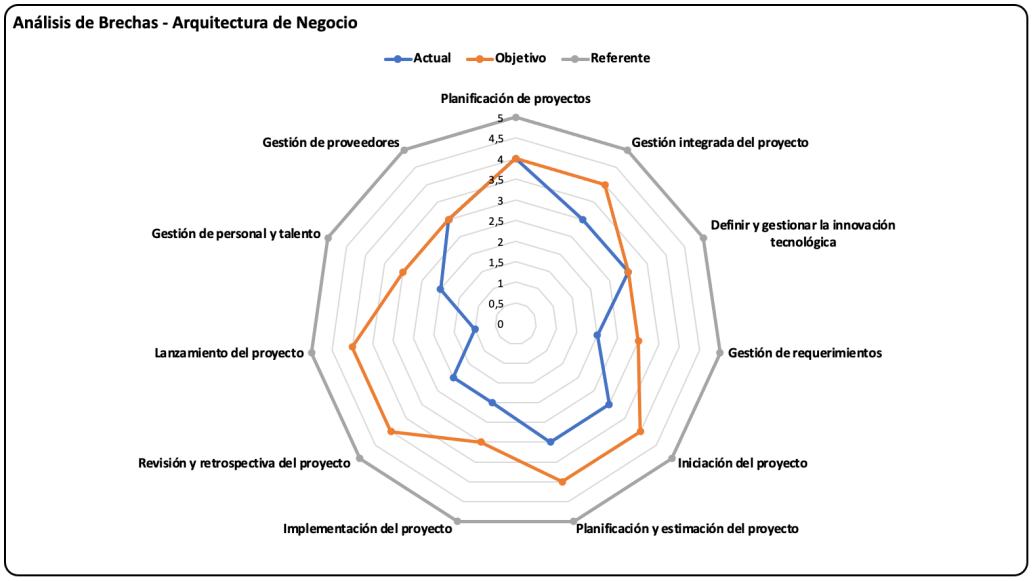


Figura 3.5 Gráfico radar de análisis de brechas

Tabla 3.13 y representados visualmente en la **Figura 3.5**, se concluye que se debe trabajar en un aspecto cultural para reforzar el uso de marcos ágiles de trabajo. Esto no debe ser únicamente a nivel del área de Tecnologías de la Información, si no a nivel general debido a que, en los nuevos roles propuestos, el involucramiento de las áreas funcionales es imprescindible para el éxito de las iniciativas, así como el apoyo de la parte interesada a nivel institucional.

En agilismo, los equipos de trabajo deben ser autocontrolados y multidisciplinarios evidenciando de esta manera la carencia de un rol de pruebas que ayude al aseguramiento de la calidad del software que se entrega a producción, junto con herramientas que faciliten los procesos de despliegue.

3.4.1 Iniciativas para cerrar brechas

En base a la arquitectura de negocio actual y objetivo se realizó un análisis del cual resultaron varias brechas que podrían ser cerradas con los siguientes proyectos:

- Programa de capacitación continua para el equipo del área en temas tecnológicos y metodológicos que apoyen a la gestión y trabajo diario basado en marcos de trabajo ágiles.
- Formalizar un proceso de aseguramiento de la calidad para gestionar actividades de pruebas a los productos desarrollados por el equipo. Se debe incorporar al equipo un rol de aseguramiento de la calidad de software o capacitar a uno o varios miembros del equipo para realizar estas tareas.

4 Arquitectura de Aplicaciones / Datos

El objetivo principal de este capítulo es abordar de manera integral aspectos arquitectónicos en los dominios de aplicaciones y datos, analizar la situación actual y desarrollar las arquitecturas objetivos en los dominios propuestos como parte de la solución al problema empresarial complejo manejado en el presente proyecto.

La Universidad de Las Américas tiene un número amplio de aplicaciones para manejar temas administrativos, educacionales y de manejo interno en el área de Sistemas de Información que se usan en las actividades diarias de desarrollo.

4.1 Arquitectura de aplicaciones objetivo

En este apartado, se detallará las funcionalidades que se buscan cubrir para cada grupo de aplicaciones detallada en la **Figura 2.12 Arquitectura de aplicaciones** ilustrada en el capítulo 2.

4.1.1 Catálogos y tipos de aplicación

4.1.1.1 Aplicaciones de planificación

Este tipo de aplicaciones cumplen un rol preponderante en el desarrollo de proyectos de todo tipo y más aún en proyectos tecnológicos en donde se busca tener un equilibrio entre el tiempo, costo y alcance. Para planificar los proyectos y tareas de la Fábrica de Software se requiere que las herramientas cumplan con las siguientes funcionalidades:

- Compatibilidad de marcos de trabajo Agile DevOps.
- Herramientas de planificación.
- Modelamiento de roles scrum.
- Gestión de *sprints*.
- Modelamiento de épicas, historias de usuarios, *backlogs.*
- Estimación de esfuerzos.
- Generación de listas de tareas.
- Ayuda para equipos ágiles

- Ejecución de pruebas
- Desarrollo de software
- Monitorización de errores
- Informes avanzados
- Paneles de control y paneles de pared personalizables
- Herramientas de colaboración
- Herramientas de planificación
- Opciones de seguridad y administración avanzadas
- Interfaz móvil
- Tableros Scrum, Kanban y Scrumban
- Definición y seguimiento de objetivos

4.1.1.2 Aplicaciones de Desarrollo

Estas aplicaciones son utilizadas en las actividades de desarrollo. Es en donde se codifica los requisitos funcionales y no funcionales que deben cumplir las aplicaciones software. Las aplicaciones de desarrollo deben cumplir las siguientes funcionalidades:

- Permitir la creación de aplicaciones en diferentes lenguajes de desarrollo.
- Integración con frameworks multiplataforma
- Capacidades de desarrollo de aplicaciones web, escritorio, servicios, móviles.
- Capacidades para depuración de código
- Herramientas de control de versiones
- Integración con herramientas colaborativas
- Conexión con fuente de datos SQL *Server* y *Oracle*.
- Manejo de extensiones que permitan personalizar el entorno de desarrollo.
- Integración con herramientas de automatización de pruebas.
- Integración con herramientas de integración continua.
- Integración con cualquier nube como *Azure, Amazon Web Services, Google Cloud Platform*.
- Capacidades para desarrollar aplicaciones del tipo *Low Code*

4.1.1.3 Aplicaciones de Lanzamiento

Tras haber concluido las tareas de desarrollo, pruebas y aceptación final del producto, se debe realizar el despliegue en los ambientes productivos. Si bien esta actividad puede ser netamente manual, implica una serie de riesgos que van desde la seguridad hasta la fiabilidad. En este sentido, se busca implementar herramientas que cumplan con las siguientes funcionalidades:

- Garantizar un código consistente y de calidad que esté fácilmente disponible para los usuarios.
- Proporcionar una forma rápida, fácil y segura de automatizar la creación de sus proyectos y ponerlos a disposición de los usuarios.
- Soportar cualquier lenguaje de desarrollo.
- Capacidades de compilación multiplataforma.
- Integración con herramientas de manejo de versiones de código.
- Automatización de procesos de compilación, pruebas y entregas.
- Flujos de trabajo con funcionalidad de contenedores nativa.

4.1.1.4 Aplicaciones de Operación

Una vez que la aplicación está en operación, es necesario realizar tareas de monitorización para garantizar el correcto funcionamiento y desempeño. Es necesario obtener visibilidad total sobre las aplicaciones, la infraestructura y la red para recibir alertas que permitan tomar medidas y generar conclusiones a partir de los registros y los datos de telemetría. Las funcionalidades deseadas son las siguientes:

- Almacenar y analizar la telemetría de las aplicaciones en operación
- Capacidades de Inteligencia Artificial y aprendizaje automático para encontrar patrones de comportamientos.
- Administración de problemas.
- Administración de servicios de TI.
- Administración de eventos e información de seguridad.
- Supervisión disponibilidad, rendimiento y uso de aplicaciones.
- Integración con herramientas y procesos *DevOps*.
- Seguimiento de las solicitudes, los flujos, y los tiempos de respuesta de métricas en vivo.

- Monitorización de la infraestructura.
- Visualización de resultados mediante *dashboards*.

4.1.2 Mapeo funcionalidades solución comercial

Dado que el estándar en la Fábrica de Software está orientado al uso de herramientas proporcionadas por Microsoft, se determina el uso de los servicios en la nube ofrecidos por *Azure DevOps Services* para cada elemento del grupo de aplicaciones especificado en la **Figura 2.12 Arquitectura de aplicaciones**. En la **Figura 4.1** se muestra el mapeo procesos con las aplicaciones.

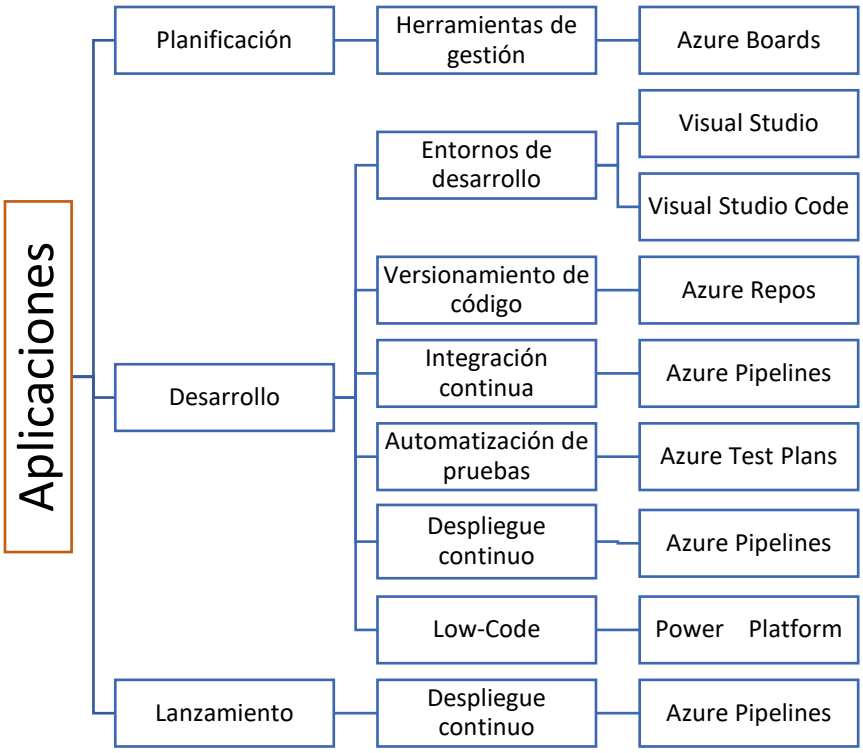


Figura 4.1 Mapeo procesos con aplicaciones

4.1.2.1 Descripción de las aplicaciones

- ***Azure Board***

  Servicio que ofrece herramientas de planeación ágiles para manejar sprints, backlogs interactivos, planificación para que los equipos de desarrollo puedan planear su trabajo, hacer un seguimiento utilizando tableros Agile.

- ***Visual Studio***

  Es un IDE (*Integrated Development Environment*) que provee amplias funcionalidades para que los desarrolladores de software puedan crear aplicaciones del tipo *Web, Desktop, Windows Services, API Rest Services,* aplicaciones móviles, entre otras opciones. Las aplicaciones creadas, están principalmente desarrolladas usando el lenguaje C#.

- ***Visual Code***

  Es un editor de código hecho por *Microsoft* para que funcione en los sistemas operativos *Windows, Linux, MacOS* que soporta características como resaltado de sintaxis, *debugging, ingelligent code* y demás funciones propias de un IDE como *Visual Studio*. Puede ser usado con una variedad amplia de lenguajes de programación tales como *Phyton, Java, Node.js*, entre otros. Esta herramienta fue liberada bajo la Licencia MIT que es una licencia permisiva de software libre.

- ***Azure Repos***

  Permite gestionar repositorios Git privados gratuitos, solicitudes de incorporación de cambios y búsqueda de código, revisiones y aprobaciones de código Git más efectivas con discusiones encadenadas e integración continua para cada cambio para mantener una alta calidad del código.

- ***Azure Pipelines***

  Servicio que ayuda a gestionar la Integración continua (CI) y entrega continua (CD). Funciona con cualquier lenguaje, plataforma y nube. Automatiza las compilaciones y las implementaciones para poder dedicar menos tiempo a estos procesos.

  La integración continua es la práctica que usan los equipos de desarrollo para automatizar la combinación y la prueba de código. Ayuda a detectar errores al principio del ciclo de desarrollo, lo que hace que sean menos costosos de corregir. Las pruebas automatizadas se ejecutan como parte de este proceso para garantizar la calidad. Los artefactos se generan a

partir de sistemas de integración y se alimentan de procesos de lanzamiento para impulsar implementaciones frecuentes. (¿Qué es Azure Pipelines?, 2022)

La entrega continua es un proceso mediante el cual el código se compila, prueba e implementa en uno o varios entornos de prueba y producción. Los sistemas de integración continua generan artefactos implementables, incluida la infraestructura y las aplicaciones. Los procesos de versión automatizados consumen estos artefactos para publicar nuevas versiones y correcciones en los sistemas existentes. Los sistemas de supervisión y alertas se ejecutan continuamente para impulsar la visibilidad de todo el proceso de entrega continua. (¿Qué es Azure Pipelines?, 2022)

- ***Azure Test Plans***

  Es una solución de gestión de pruebas basado en un navegador web que proporciona todas las capacidades para crear pruebas manuales planificadas, pruebas de aceptación del usuario, pruebas exploratorias y recopilación de comentarios de las partes interesadas. Permite la automatización de pruebas mediante la ejecución de pruebas dentro de Azure pipelines.

- ***Power Apps***

  Power Apps es un conjunto de aplicaciones, servicios y conectores, así como una plataforma de datos que proporciona un entorno de desarrollo de aplicaciones ágil para crear aplicaciones personalizadas para las necesidades de su empresa (Microsoft, ¿Qué es Power Apps?, 2022).

- ***Azure Monitor***

  Ayuda a maximizar la disponibilidad y el rendimiento de sus aplicaciones y servicios mediante una solución integral para recopilar, analizar y actuar sobre la telemetría desde su nube y entornos locales. Azure monitor ayuda en gran manera a lo siguiente:

  - Detectar y diagnosticar problemas entre aplicaciones y dependencias con Application Insights.

- Correlacionar problemas de infraestructura, información de máquinas virtuales y de contenedores.
- Recopilar datos de los recursos monitoreados Crear visualizaciones

4.1.3 Aplicaciones por proceso

En la **Figura 4.2**, se detalla la relación entre los procesos estratégicos y misionales de la Fábrica de Software interna de UDLA con las aplicaciones que son parte de la arquitectura objetivo-planteada y que se usarán en cada uno de los procesos.

Figura 4.2 Relación entre los procesos con las aplicaciones

4.2 Arquitectura de aplicaciones y datos objetivo

En esta sección se aborda la arquitectura de aplicaciones objetivo basado en las herramientas recomendadas por DevOps para cada uno de los procesos identificados en el capítulo anterior, que permitirán implementar agilidad en las fases de desarrollo software. Se plantea el uso de los servicios disponibles en *Azure DevOps Service*, lo que implicaría la migración a la nube de muchos de los procesos que actualmente se manejan de manera local.

Dentro de la arquitectura actual de la UDLA se tiene dos tipos de fuentes de datos: estructurados y no estructurados. En la **Figura 4.3** se muestra un diagrama de los tipos de datos.

Figura 4.3 Tipos de datos

4.2.1 Datos estructurados

Los datos estructurados tienen un formato estandarizado, una estructura bien definida, se ajustan a un modelo de datos y son fácilmente accesibles para humanos y software. Este tipo de datos suele almacenarse en una base de datos.

4.2.2 Datos no estructurados

Los datos no estructurados no encajan en ningún otro patrón y no tienen una estructura reconocible. No está organizado y no se puede almacenar lógicamente. Los datos no estructurados no encajan en ninguna estructura de base de datos, no tienen reglas ni formatos, y los programas no pueden utilizarlos fácilmente.

Como parte de los datos no estructurados se cuenta con un catálogo de reportes que permitan dar seguimiento a los proyectos, medir esfuerzos en el desarrollo.

4.2.2.1 Reportes

a) Reportes Gestión de requerimientos

Los reportes para la gestión de proyectos van a contemplar los siguientes parámetros.

- Porcentaje de requisitos satisfechos por la solución propuesta.
- Porcentaje de objetivos del caso de negocio satisfechos por la solución propuesta.
- Porcentaje de riesgos de los requisitos no cubiertos por una adecuada respuesta al riesgo.
- Nivel de satisfacción de las partes interesadas con los requisitos Número de excepciones de la solución observadas durante la etapa de las revisiones.
- Porcentaje de partes interesadas que no aprueban la solución en relación con el caso de negocio.
- Nivel de satisfacción de las partes interesadas con los requisitos.
- Porcentaje de partes interesadas que no aprueban la solución en relación con el caso de negocio.

b) Reportes gestión de la innovación

- Percepciones y retroalimentación de las partes interesadas de la empresa respecto a la innovación en I&T.
- Porcentaje de iniciativas implementadas con un claro vínculo a un objetivo empresarial.
- Porcentaje de oportunidades habilitadas por nuevas tecnologías identificadas.
- Frecuencia de la investigación y exploración del entorno realizadas para identificar ideas y tendencias innovadoras.
- Porcentaje de iniciativas implementadas que logran los beneficios previstos.
- Porcentaje de iniciativas de pruebas de concepto exitosas para poner a prueba tecnologías emergentes u otras ideas de innovación.
- Número de iniciativas de prueba de concepto evaluadas y aprobadas para su posterior implementación.
- Indicador de aumento de la cuota de mercado o competitividad debido a innovaciones.
- Número de lecciones aprendidas y oportunidades de mejora captadas para su uso futuro.

- Número de iniciativas de prueba de concepto que han sido apalancadas con la inversión real.

c) Reporte de gestión personal y talento

- Identificar habilidades y competencias clave que no se encuentren en la matriz de recursos.
- Número de brechas identificadas entre las habilidades requeridas y las disponibles.
- Número de programas de capacitación proporcionados.
- Porcentaje de contratistas que firman el marco de control empresarial.
- Frecuencia de las revisiones periódicas llevadas a cabo para garantizar la exactitud y el cumplimiento con la ley, del personal del contratista.
- Tiempo utilizado por cada empleado a tiempo completo (FTE) en trabajos y proyectos.
- Número y valor de las recompensas otorgadas a la persona.

d) Gestionar el conocimiento

- Porcentaje de información clasificada validada.
- Porcentaje de pertinencia de los tipos de contenido, artefactos e información estructurada y no estructurada
- Porcentaje de satisfacción de las partes interesadas con la organización y contextualización de la información en conocimiento
- Porcentaje de conocimiento disponible usado realmente
- Porcentaje de satisfacción del usuario con los conocimientos
- Frecuencia de actualización
- Nivel de satisfacción de los usuarios

e) Reportes Gestión de proyectos

- Porcentaje de proyectos exitosos conforme con la estrategia estándar definida.
- Número de actualizaciones de la estrategia de gestión de proyectos, buenas prácticas, herramientas y plantillas.

- Porcentaje de partes interesadas que aprueban la necesidad empresarial,
  alcance, resultado previsto y nivel de riesgo del proyecto.
- Porcentaje de proyectos en los que las partes interesadas reciben una clara declaración por escrito que define la naturaleza, alcance y beneficio del proyecto.
- Nivel de satisfacción de las partes interesadas con la participación.
- Porcentaje de partes interesadas efectivamente involucradas.
- Porcentaje de proyectos activos llevados a cabo sin mapas de valor del
  proyecto válidas y actualizadas.
- Porcentaje de hitos o tareas terminadas vs. el plan.

4.2.2.2 Repositorio de buenas prácticas

La capacidad de identificar y transferir las mejores prácticas dentro de una organización es fundamental para lograr una ventaja competitiva. Las mejores prácticas son aquellas técnicas o métodos que mejoran la satisfacción del cliente incorporándolas a nuestros procesos. Por esta razón se plantea contar con un repositorio actualizado con documentación sobre las mejores prácticas de desarrollo de software.

A nivel general la documentación debe contar con información sobre:

- Estandarizar las reglas del desarrollo
- Testeo de código
- Optimización
- Seguridad
- Documentación

4.2.3 Diagrama de aplicaciones y datos

En la **Figura 4.4** se indica la interoperabilidad entre la arquitectura de datos y la arquitectura de aplicaciones donde se describe que almacena cada aplicación.

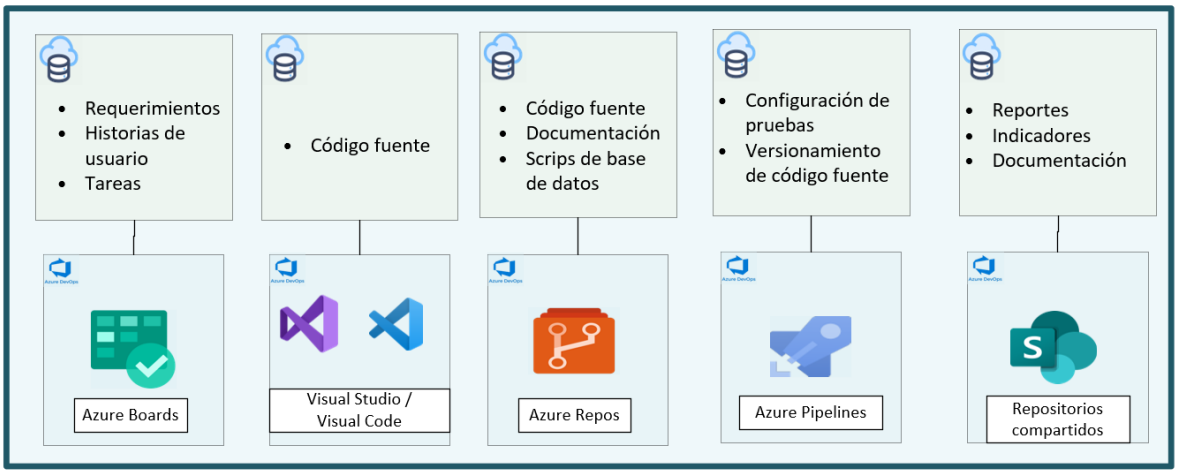


Figura 4.4 Interoperabilidad entre la arquitectura de datos y aplicaciones

4.3 Arquitectura actual e Iniciativas que cierran brechas

4.3.1 Arquitectura actual de aplicaciones

El objetivo principal de la Fábrica de Software de UDLA, es generar valor a la organización a través del desarrollo de iniciativas tecnológicas que soporten la operación de las diferentes áreas de la Universidad. Estas actividades se han apoyado en un reducido grupo de herramientas para cubrir el ciclo de vida de desarrollo de software y se han venido gestionando de con un enfoque bastante manual sin contar procesos de automatización que agiliten la entrega de valor. A continuación, en la **Figura 4.5**, se detallan las aplicaciones que actualmente se usan en las etapas que tradicionalmente se manejan en la fábrica de software:

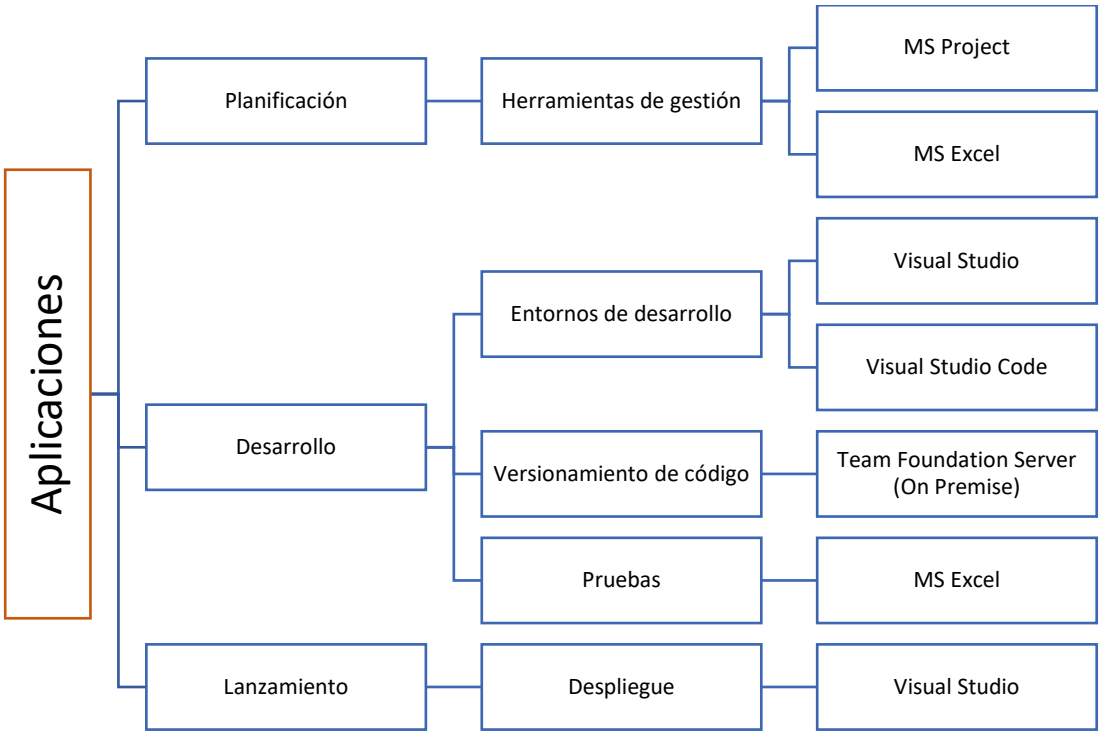


Figura 4.5 Arquitectura actual de aplicaciones

4.3.2 Análisis de Brechas

En la **Tabla 4.1**Figura 4.1, se realiza un análisis de brechas que es resultado de analizar la arquitectura actual y la objetivo que se plantea en el presente capítulo.

Tabla 4.1 Análisis de brechas entre aplicación actual y objetivo

Tipo	Aplicación	Actual	Objetivo	Referente	Brecha
Planificación	Azure Boards	1	4	5	• La planificación se lleva en cronogramas hechos en MS Project que a pesar de ser una execelente herramienta, se dificulta el seguimiento, el detalle de las tareas y trabajar con marcos ágiles. • Se dificulta el registro, seguimiento y manejo de históricos de los requerimientos levantados
Desarrollo	Entornos de desarrollo	4	4	5	• El estándar de UDLA es usar herramientas Microsoft y en este caso Visual Studio, por lo que no existe un brecha como tal.
	Versionamiento de código	3	4	5	• Se cuenta con Team Foundation Server para el versionamiento de código, pero es una herramienta On Premise y representa un alto riesgo debido a la posibilidad latente de un ataque informatico que de como resultado la pérdida del código fuente de las aplicaciones creadas.
	Integración continua	0	3	5	• No existe una aplicación que permita la automatización para la integración continua del código desarrollado en los ambientes requeridos.
	Automatización de pruebas	0	3	5	• No existe aplicaciones que ayuden a la automatización de pruebas sobre los desarrollos realizados.
	Despliegue continuo	0	3	5	• No existe una aplicación que permita la automatización para la integración continua del código desarrollado en los ambientes requeridos.
	Low-Code	2	3	5	• Falta capacitación en este tipo de herramientas. • Al ser un herramienta que democratiza la creación de aplicaciones mediante un esquema de Low-Code, hace falta estándares y políticas para normar su uso.
Lanzamiento	Despliegue continuo	0	3	5	• No existe una aplicación que permita la automatización del despliegue continuo de las aplicaciones en los ambientes requeridos.
	Promedio	**1,25**	**3,38**		

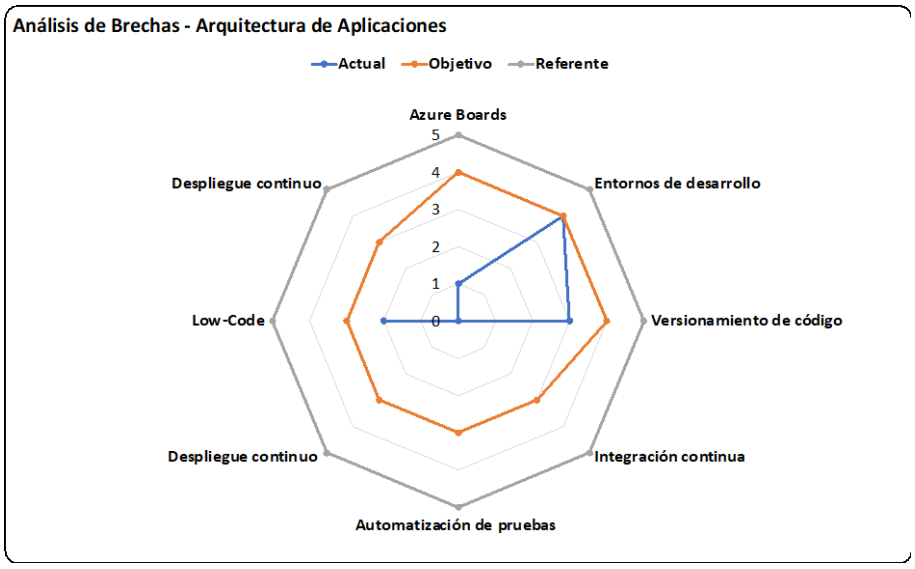


Figura 4.6 Radar de resultado de análisis de brechas

De los resultados obtenidos en el análisis de brechas **Figura 4.6** y **Tabla 4.1**, se evidencia la falta de varias herramientas que mejore, soporten y automaticen las actividades diarias que lleva a cabo los colaboradores de la Fábrica de Software interna de UDLA.

4.3.3 Iniciativas para cerrar brechas

En base al análisis de brechas realizado, se plantea que podrían ser cerradas con los siguientes proyectos:

- Implementar un proyecto de migración al esquema en la Nube ofrecido por Azure DevOps Services que nos permita usar las herramientas descritas en la arquitectura de aplicaciones objetivo.
- Programa de capacitación al personal designado en las herramientas para automatizar procesos de integración, despliegue continuo y pruebas.

5 Arquitectura de infraestructura base

Un papel primordial en el esquema del desarrollo de software lo cumple la infraestructura tecnológica que está formada por hardware, software base, redes de comunicación, esquemas de monitoreo, entre otros elementos que son necesarios para el despliegue de las aplicaciones creadas por las Fábricas de Software en general. En este capítulo se aborda el desarrollo de una arquitectura de infraestructura objetivo que soporte y habilite las arquitecturas objetivo de negocio, aplicaciones y datos planteadas en los capítulos anteriores permitiendo de esta manera la generación de valor a la organización.

5.1 Arquitectura de infraestructura objetivo

Se tomará como referencia los modelos de infraestructura en la nube que trabajen en concordancia con las prácticas y herramientas *DevOps* descritas en los anteriores capítulos. Los modelos en la nube se pueden dividir en los siguientes tipos:

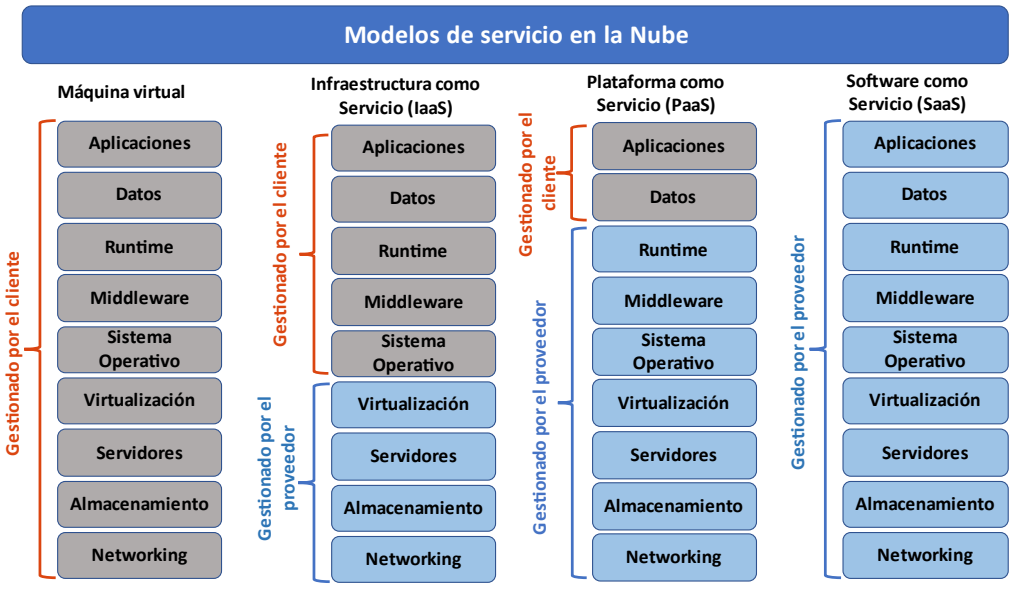


Figura 5.1 Modelo de servicios en la nube

5.1.1 Máquinas Virtuales

Este modelo es el que más se parece a tener un ambiente On-Premise debido a que el cliente administra de inicio a fin todos los componentes de infraestructura tecnológica. Al estar en la nube, tiene ciertas facilidades como escalabilidad, elasticidad, seguridad que brinda la nube en su conjunto.

5.1.2 Infraestructura como Servicio (IaaS)

Está formado por los bloques de creación fundamentales para la TI en la nube. Permite acceder a características de *networking*, a los equipos y al espacio de almacenamiento. Ofrece un alto nivel de flexibilidad y administración sobre los recursos de infraestructura base.

5.1.3 Plataforma como Servicio (PaaS)

En este modelo ya no se requiere administrar la infraestructura de TI como *hardware* o sistemas operativos. No hay necesidad de preocuparse por los recursos, capacidades, mantenimiento, parches, entre otros aspectos que conlleva la administración de servidores. De esta manera los clientes se centran en la implementación y administración de sus aplicaciones.

5.1.4 Software como Servicio (SaaS)

El proveedor gestiona de punta a punta todos los componentes de la infraestructura de TI. En la mayor parte de casos, el *software* como servicio se refiere a aplicaciones de usuario final, así el cliente solo debe preocuparse por cómo utilizar y sacarle el mayor provecho a la aplicación que usa.

5.1.5 Diagrama de ambientes

En base al modelo de infraestructura como servicio que ofrece Azure se plantea el diagrama de infraestructura objetivo de la Figura 5.2**.** Esta cuenta con dos subscripciones que son *DevTest y Production*.

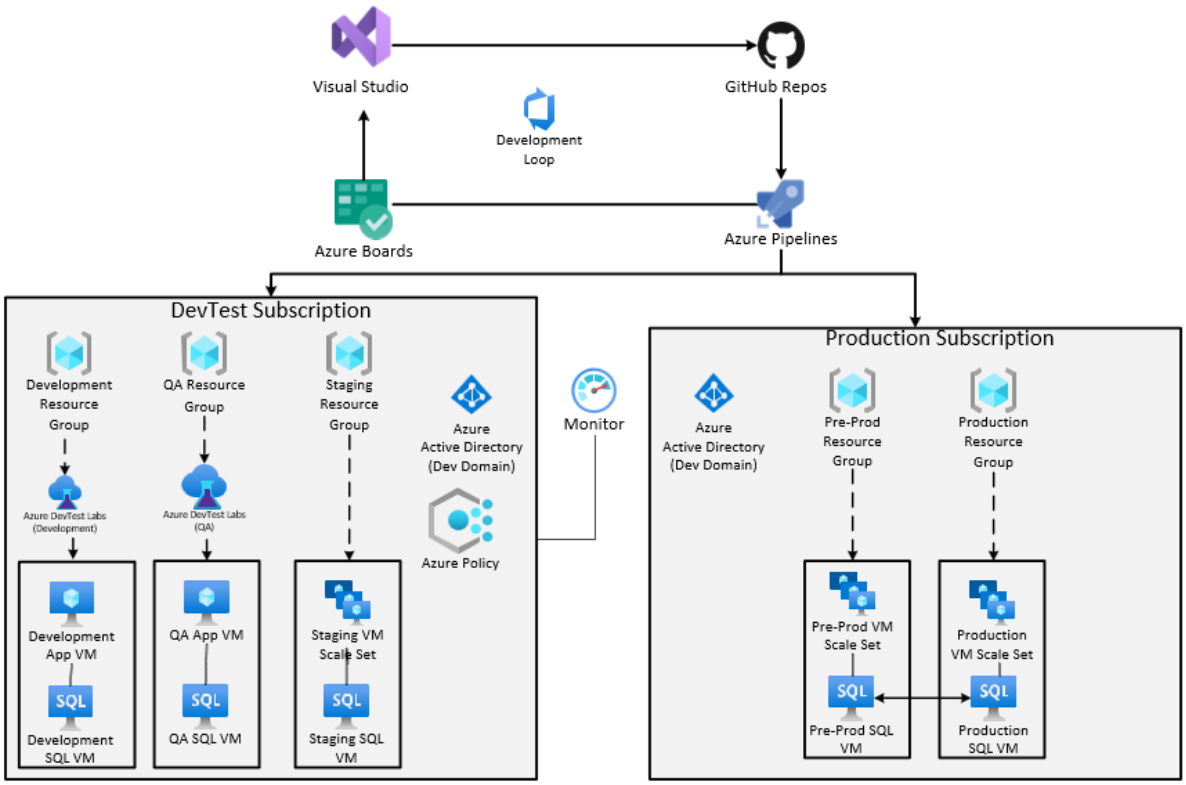


Figura 5.2 Diagrama de infraestructura objetivo

- ***DevTest***: Dentro de esta subscripción se cuenta con tres ambientes que ayudaran a realizar las validaciones necesarias sobre las aplicaciones para asegurar su correcto funcionamiento, estos son:
  - **a)** ***Development***: En este ambiente se llevan a cabo una serie de pruebas preliminares por parte de los desarrolladores antes de pasar a las siguientes etapas.
  - **b)** ***Quality Assurance (QA)***: El entorno de prueba permite a los ingenieros de control de calidad probar código nuevo y modificado, ya sea mediante técnicas automatizadas o de manera manual.
  - **c)** ***Staging***: Es una réplica casi exacta del ambiente de producción, para garantizar que el software funcione correctamente con todos sus componentes en conjunto.

Cada uno de estos ambientes cuenta con *Azure DevTest Labs* que es un servicio de Azure DevOps para crear, usar y administrar fácilmente máquinas virtuales de infraestructura como servicio *(IaaS)* y entornos de plataforma como servicio *(PaaS)* en un laboratorio que ofrece bases

preconfiguradas y artefactos para crear máquinas virtuales. (Microsoft Learn, 2022)

Los propietarios del laboratorio pueden crear máquinas virtuales preconfiguradas que tengan herramientas y que los usuarios necesiten. Las directivas de laboratorio y otros métodos controlan el uso y los costos. (Microsoft Learn, 2022)

- ***Production:*** En esta suscripción se cuenta con el ambiente de preproducción y producción. El ambiente de preproducción es una réplica exacta de producción.

5.1.6 Aproximación de especificaciones de las máquinas

En la

Tabla 5.1 se identifica las siguientes características óptimas a nivel computacional para los equipos que debe disponer la Fábrica de software:

Tabla 5.1 Especificación de máquinas

Tipo	Características	Descripción
Servidor bajo subscripción en Azure	**Plan:** *Azure DevTest* **Instancia:** Estándar A3 **Total de *Cores*:** 4 **Memoria:** 7 GB RAM **Disco:** 285 GB	Plan de Azure para DevOps en ambiente de desarrollo y pruebas. Los precios dependen de la región en la cual esté el data center.
Laptop	**Marca:** HP **Serie:** EliteBook 840 **Procesador:** Core I7 11va Gen **Memoria:** 16 GB RAM **Disco:** 512 GB SSD	Equipo personal para desarrollo
Desktop	**Marca:** HP **Serie:** Pro Desk 600 **Procesador:** Core I7 11va Gen **Memoria:** 16 GB RAM **Disco:** 512 GB SSD	Equipo personal para desarrollo

5.2 Arquitectura actual e iniciativas que cierran de brechas

5.2.1 Arquitectura actual de infraestructura

La arquitectura actual en donde la Fábrica de software de UDLA prueba sus desarrollos, está administrado por el área de infraestructura tecnológica que en el organigrama de TI es paralela al área de desarrollo. Ellos son los encargados de administrar los servidores On-Premise existentes en el centro de datos de la Universidad. A continuación, en la **Figura 5.3** se muestra un diagrama que representa la arquitectura de infraestructura base:

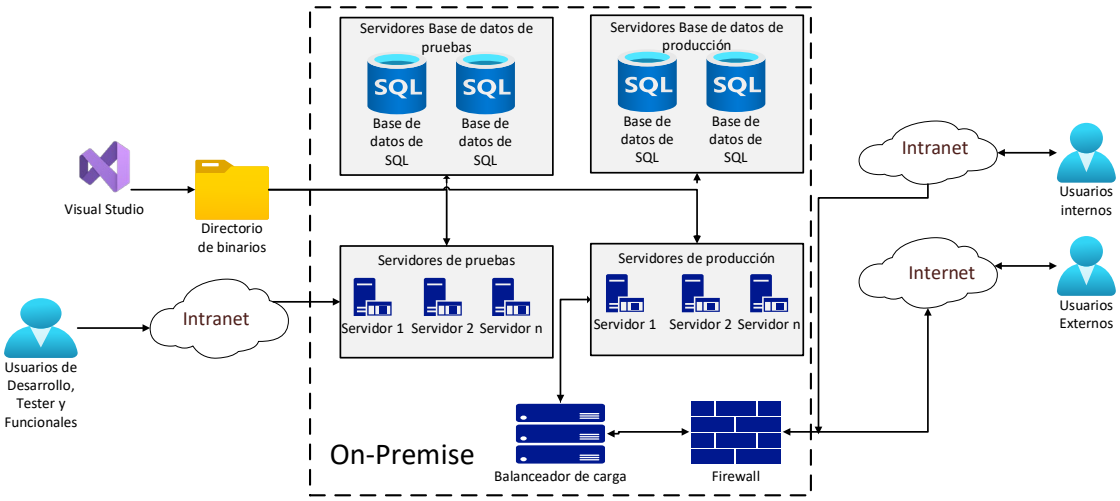


Figura 5.3 Infraestructura actual

En la **Figura 5.3** se evidencia que existen creados varios servidores tanto de pruebas como de producción en donde se van publicando las aplicaciones creadas por el área de desarrollo.

5.2.2 Análisis de Brechas

En la **Tabla 5.2**, se muestra el análisis de brechas realizado entre la arquitectura actual y objetivo para la arquitectura de infraestructura.

Tabla 5.2 Análisis de brechas de infraestructura base

Ambiente	Servidores	Actual	Objetivo	Referente	Brecha
Ambientes no productivos	Servidores de Desarrollo	0	4	5	• No existen ambientes de desarrollo para que los desarrolladores realicen pruebas en un ambiente no productivo y validar sus cambios fuera de sus equipos personales. • Existe una administración netamente manual para la gestión de los deploy de las aplicaciones.
	Servidores de Test	2	4	5	• Los ambientes de test son de uso compartido entre desarrolladores y funcionales que realizan las pruebas. Puede existir confusión de versiones al momento de los pasos a producción. • En la mayoría de casos, los servidores de pruebas no son fiel reflejo de producción dificultando de esa manera encontrar bugs reportados en producción. • Existe una administración netamente manual para la gestión de los deploy de las aplicaciones.
	Servidores de Staging	0	4	5	• No existe servidores de Staging que sea lo más parecido a los ambientes productivos para que se puedan realizar el correcto esquema de pruebas de las aplicaciones de todos los componentes en su conjunto.
	Servidores de Pre Producción	0	4	5	• No existe servidores de preproducción que sea una replica de los ambientes productivos para que se puedan realizar el correcto esquema de pruebas de las aplicaciones de todos los componentes en su conjunto.
	Promedio	**0,50**	**4,00**		

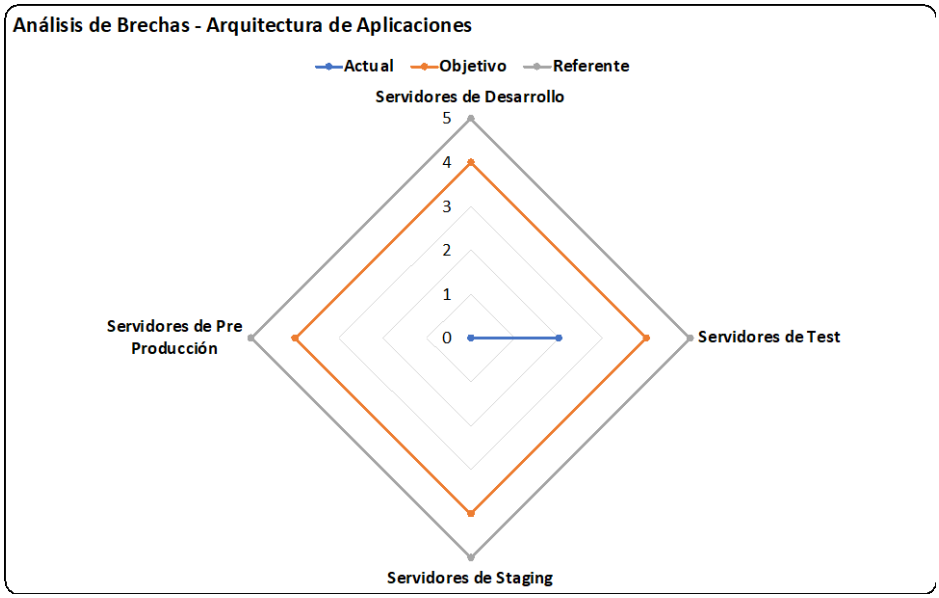


Figura 5.4 Radar de resultado de brechas de infraestructura base

De los resultados obtenidos en el análisis de brechas de la **Tabla 5.2** y la **Figura 5.4**, se evidencia una brecha grande en la disponibilidad de ambientes no productivos que apoyen a la gestión de desarrollo de software. Una tendencia con mucha aceptación en el mercado es migrar a la nube los componentes tecnológicos aprovechando de esta manera los beneficios que esta presta y en

el caso de los ambientes no productivos, se los podría gestionar de mejor manera.

5.2.3 Iniciativas para cerrar brechas

Tomando como referencia el análisis de brechas previamente realizado, se plantea los siguientes proyectos:

- Implementar un proyecto creación y migración de ambientes no productivos a la nube de Azure que permita apalancar un esquema de DevOps para las aplicaciones desarrolladas en la Fábrica de Software de UDLA.
- Implementar políticas y procedimientos para regular y normar el uso de los ambientes no productivos de tal manera que se aproveche de la mejora manera la capacidad en nube disponible.

6 Oportunidades y soluciones

En este capítulo se aborda las iniciativas o proyectos para cerrar las brechas identificadas en las arquitecturas de negocio, aplicaciones e infraestructura para la Fábrica de Software de UDLA. Para esto, en la **Tabla 6.1** se condensa las iniciativas divididas en cada una de las arquitecturas.

Tabla 6.1 Iniciativas de cierre de brechas

Arquitectura	Código	Iniciativa
Negocio	NE01	Programa de capacitaciones al equipo de desarrollo en temas tecnológicos y metodológicos.
	NE02	Establecimiento de un área de aseguramiento de la calidad de software.
Aplicaciones	AP01	Implementación de Azure DevOps utilizando como piloto una aplicación de producción.
Infraestructura	IN01	Uso de ambientes no productivos en la nube de Azure bajo el esquema de integración continua de DevOps.
	IN02	Implementar políticas y procedimientos para regular y normar el uso de los ambientes no productivos.

6.1 Arquitectura de negocio

6.1.1 Iniciativa: Programa de capacitaciones al equipo de desarrollo en temas tecnológicos y metodológicos

Esta iniciativa tiene como objetivo principal el contar con un equipo de desarrollo preparado para afrontar los nuevos retos de negocio que tiene UDLA y que necesariamente se apalancan en la tecnología, como facilitador de los procesos que conllevan.

Específicamente para el área de la Fábrica de Software de UDLA, se plantea de manera inicial, capacitación en los siguientes temas para afianzar habilidades duras y blandas dentro del equipo.

- Marcos de trabajo ágil tomando como referencia Scrum
- DevOps
- *Power Platform*

6.1.1.1 Conceptualización de la iniciativa

En la **Figura 6.1** se resume los aspectos para poner en marcha la iniciativa

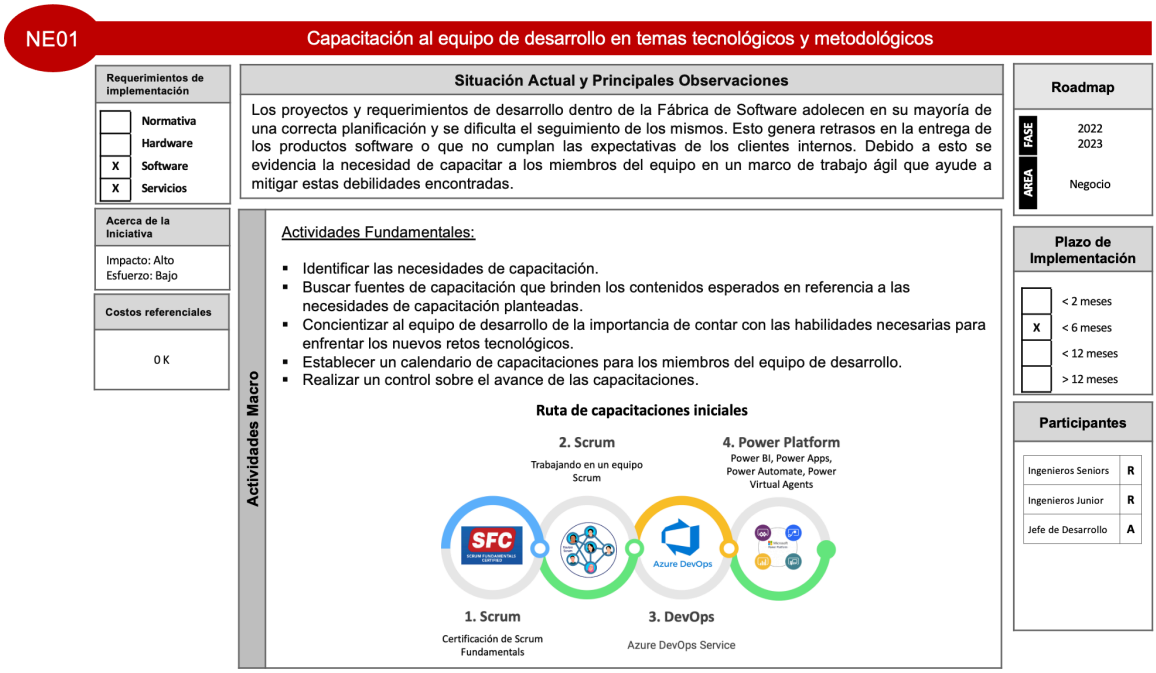


Figura 6.1 Capacitación al equipo de desarrollo en temas tecnológicos y metodológicos

6.1.1.2 Detalle de la implementación de la iniciativa

a) Scrum

Se plantea que la capacitación en el marco de trabajo Scrum sea tomada por el equipo de desarrollo en forma gratuita en la página web https://big-agile.teachable.com/. Se deben enfocar en dos cursos inicialmente:

En la

Tabla 6.2 se muestra el detalle de la capacitación en Scrum Fundamentals

Tabla 6.2 Detalle certificación de scrum fundamentals

Certificación de Scrum Fundamentals	
Objetivos del curso:	• Tener una base de conocimientos homogénea en relación con los conceptos fundamentales de Scrum. • Empezar gestión de cambio de *Mindset* de los miembros del equipo
Temas para tratar:	• Bases de Scrum • Ágil: Valores y Principios • Roles de Scrum • Reuniones de Scrum • Valores de Scrum • Definición de Terminado • Scrum: El marco de trabajo • La Guía Oficial de Scrum
Características del curso	**Duración aproximada de los videos:** 4 horas **Tipo de clases:** educación virtual mediante videos asincrónicos que explican de cada tema con preguntas al final de cada uno. **Idioma:** español **Precio:** gratuito

En la

Tabla 6.3 se muestra el detalle de la capacitación en Scrum Fundamentals

Tabla 6.3 Detalle curso trabajando en un equipo scrum

Trabajando en un equipo Scrum	
Objetivos del curso:	• Conocer a fondo los conceptos y definiciones en el marco de trabajo Scrum que permitan al desarrollador trabajar de manera fluida en equipos scrum. • Mejorar la satisfacción de los clientes internos, enfocándonos a entregar valor. • Mejorar la dinámica del trabajo del equipo, mejorando su ambiente para experimentar, fallar y crecer. Generando de esta manera equipos creativos y motivados.
Temas para tratar:	• Módulo 1: *Agile* • Módulo 2: *Scrum* • Módulo 3: *Sprinting* • Módulo 4: *Dev Team* • Módulo 5: *Product Owner* • Módulo 6: *ScrumMaster* • Módulo 7: *The Product Backlog* • Módulo 8: *Long-term Planning* • Módulo 9: *Sprint Planning* • Módulo 10: *Meetings* • Módulo 11: *Scaling*
Características del curso	• **Duración aproximada de los videos:** videos cortos de entre 7 y 12 minutos que suman un total de 8 horas aproximadamente. • **Tipo de clases:** Educación virtual mediante videos asincrónicos que explican de cada tema con preguntas al final de cada uno. Adicional se cuenta con talleres de trabajo para afianzar los conocimientos adquiridos. • **Idioma:** inglés • **Precio:** gratuito

b) DevOps

Es una cultura de mejores prácticas que integra el desarrollo y las tareas operativas, asegurando la entrega de nuevas funciones y la estabilidad del proyecto de forma continua y rápida, con un enfoque en la comunicación, la integración y la colaboración.

Para la capacitación en *DevOps* se plantea seguir la ruta de aprendizaje que ofrece *Microsoft Learn* en su plataforma gratuita*: https://learn.microsoft.com/es-es/training/paths/evolve-your-devops-practices/*. En la **Tabla 6.4** se muestra el detalle del curso de DevOps

Tabla 6.4 Detalle curso DevOps

DevOps	
Objetivos del curso:	• Comprender cómo los mapas de secuencias de valores pueden ayudar a evaluar los procesos y tecnologías actuales. • Aprender a planificar y hacer un seguimiento mediante Azure Boards. • Optimizar las cargas de trabajo entre equipos de Agile.

Temas para tratar:	• Parte 1: Introducción a Azure DevOps
	  ○ Evaluación del proceso de desarrollo de software existente
	  ○ Introducción a Azure DevOps
	  ○ Elegir un enfoque de Agile para el desarrollo de software
	  ○ Administración de planes de entrega de software ágil entre equipos
	• Parte 2: Creación de aplicaciones con Azure DevOps
	  ○ Creación de una canalización de compilación con Azure Pipelines

	○ Implementar un flujo de trabajo de código en la canalización de compilación mediante Git y GitHub ○ Ejecución de pruebas de calidad en la canalización de compilación mediante Azure Pipelines ○ Administrar las dependencias de compilación con Azure Artifacts ○ Hospedar un agente de compilación propio en Azure Pipelines • Parte 3: Implementación de aplicaciones con Azure DevOps ○ Crear una canalización de versión en Azure Pipelines ○ Creación de una canalización de varias fases con Azure Pipelines ○ Ejecución de pruebas funcionales en Azure Pipelines ○ Ejecución de pruebas no funcionales en Azure Pipelines ○ Administración de la cadencia de versiones en Azure Pipelines mediante patrones de implementación ○ Automatización de implementaciones de Azure Functions con Azure Pipelines ○ Automatización de implementaciones de contenedores de Docker con Azure Pipelines ○ Automatización de las implementaciones de Kubernetes de varios contenedores con Azure Pipelines
Características del curso	• **Duración aproximada del curso:** 7 horas • **Tipo de capacitación:** plataforma online gratuita que permite acceder a un conjunto de

	formaciones para la adquisición y perfeccionamiento de habilidades digitales. • **Idioma:** español • **Precio:** gratuito

c) Power Platform

Microsoft Power Platform es una plataforma que permite a los usuarios y organizaciones crear aplicaciones de forma rápida y sencilla con Power Apps, analizar datos con Power BI y automatizar procesos para aumentar la productividad con Microsoft Power Automate, independientemente de las habilidades técnicas.

Para Power Platform se puede seguir la ruta de aprendizaje de Microsoft con un curso introductorio posteriormente se debería seguir los siguientes cursos:

- Introducción a Microsoft Power Platform
- Introducción a Power Apps
- Introducción a Dataverse
- Introducción a Power BI
- Introducción a Power Automate
- Introducción a Power Virtual Agents

En la

Tabla 6.5 se muestra el detalle del curso de Introducción a Microsoft Power Platform

Tabla 6.5 Detalle curso Introducción a Microsoft Power Platform

Introducción a *Microsoft Power Platform*	
Objetivos del curso:	• Conocer los componentes y las características principales de *Microsoft Power Platform.* • Identificar cuándo usar cada aplicación que compone *Microsoft Power Platform* para crear soluciones empresariales. • Comprender los conectores. • Conocer el valor de usar *Microsoft Power Platform* para crear soluciones empresariales.

Temas para tratar:	• Introducción • ¿Qué es *Microsoft Power Platform*? • El valor empresarial de *Microsoft Power Platform* • Conectores de datos • Prevención de pérdida de datos, cumplimiento, privacidad y accesibilidad • Poner en práctica lo aprendido • Prueba de conocimientos • Resumen y recursos
Características del curso	• **Duración aproximada del curso:** 36 min • **Tipo de capacitación:** plataforma online gratuita que permite acceder a un conjunto de formaciones para la adquisición y perfeccionamiento de habilidades digitales. • **Idioma:** español • **Precio:** gratuito

6.1.2 Iniciativa: Establecimiento de un área de aseguramiento de la calidad de software

El aseguramiento de la calidad del software tiene como objetivo que el producto satisfaga las necesidades del usuario de una manera proactiva. Se trata de crear una cultura universal en esta área del mundo del software con tecnologías de alta calidad y en constante crecimiento que garanticen una mejor experiencia, ahorren costos y tiempo.

6.1.2.1 Conceptualización de la iniciativa

En la **Figura 6.2** se resume los aspectos para poner en marcha la iniciativa

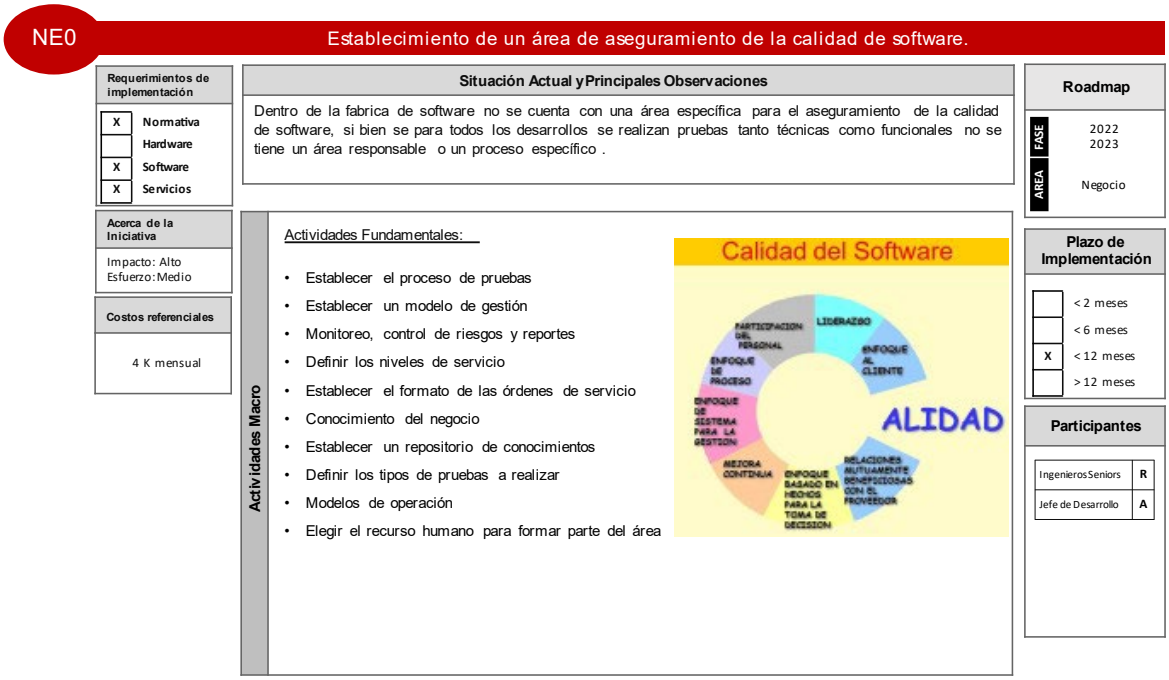


Figura 6.2 Establecimiento de un área de aseguramiento de calidad de software

6.1.2.2 Detalle de la implementación

Para establecer el área aseguramiento de calidad de software se debe realizar lo siguiente:

- Establecer el proceso de pruebas
- Establecer un modelo de gestión
- Monitoreo y control de riesgos y reportes
- Definir los niveles de servicio
- Establecer el formato de las órdenes de servicio
- Conocimiento del negocio
- Establecer un repositorio de conocimientos
- Definir los tipos de pruebas a realizar
- Establecer modelos de operación
- Elegir el recurso humano para formar parte del área

6.1.2.3 Características de la implementación

a) Los tipos de pruebas que principalmente se deben realizar para validar el proceso son:
   - Pruebas unitarias

- Pruebas de integración
- Pruebas de validación
- Pruebas del sistema
- Pruebas de carga y estrés

b) Incluir en los reportes de pruebas lo siguiente:
   - ¿Qué fue probado?
   - ¿Qué no fue probado y por qué?
   - Resultados de las pruebas
   - Conclusiones

c) Dentro de la selección del recurso humano para el área se debe tener en cuenta requisitos técnicos como:
   - Formación académica en Ingeniería en Sistemas, en *Software*, Industrial, Electrónica o afines.
   - Experiencia laboral de mínimo 4 años de experiencia en desarrollo y/o *testeo* de aplicaciones.

d) Los objetivos del cargo del equipo de pruebas serán:
   - Diseño y creación de herramientas de *Software* que permitan probar las funcionalidades y certificar la calidad de los productos generadas por el área de desarrollo.
   - Preparación de ambientes para verificación de los productos desarrollados.
   - Generación de procesos que permitan probar rápida y confiablemente las plataformas desarrolladas por la empresa (automatización).
   - Alta interacción con áreas de Desarrollo y Soporte de la empresa
   - Responsable de la certificación y puesta en producción de nuevos servicios y/o plataformas de software.

e) Las tareas del cargo de los integrantes del equipo de pruebas serán:
   - Ejecutar determinados scripts de pruebas manuales bajo supervisión.
   - Utilizar herramientas de prueba automatizadas y básicas.
   - Registrar resultados e informa sobre problemas.
   - Desarrollar un entendimiento de la función de las pruebas como herramienta para mejorar diseños y como proceso de validación.

- Diseñar casos de prueba, crea scripts y datos de ensayos, y automatiza las tareas repetibles trabajando según los requisitos o las especificaciones establecidas.
- Definir las condiciones de prueba para determinados requisitos.
- Ejecutar y registra pruebas manuales y automatizadas de acuerdo con los planes de prueba.
- Preparar análisis e informes sobre actividades, resultados, problemas y riesgos de las pruebas.
- Diseñar casos y scripts de prueba bajo su propia dirección, mapeados para identificar criterios predeterminados, registrando e informando los resultados de las pruebas.
- Participar en revisiones de requisitos, diseños y especificaciones, y utiliza esta información para diseñar planes de prueba y condiciones de ensayo.
- Aplicar estándares acordados para especificar y realizar pruebas manuales y automatizadas. Automatiza las tareas de prueba y crea cobertura de pruebas a través de una infraestructura nueva o existente.
- Preparar análisis e informes sobre actividades, resultados, problemas y riesgos de las pruebas.

6.2 Arquitectura de aplicaciones

6.2.1 Iniciativa: Implementación de Azure DevOps utilizando como piloto una aplicación de producción

Las prácticas DevOps, a más de ser una tendencia en el mundo del software, ayudan a agilizar y mejorar los procesos de desarrollo y entrega de software consiguiendo como resultado un beneficio económico que se refleja en la reducción de costos por las automatizaciones y la detección de posibles fallos en etapas tempranas del desarrollo de software. El éxito de la implementación de las prácticas DevOps puede tener un impacto significativo a través de una mayor colaboración organizacional, seguridad y eficiencia.

6.2.1.1 Conceptualización de la iniciativa

En la **Figura 6.3** se resume los aspectos para poner en marcha la iniciativa

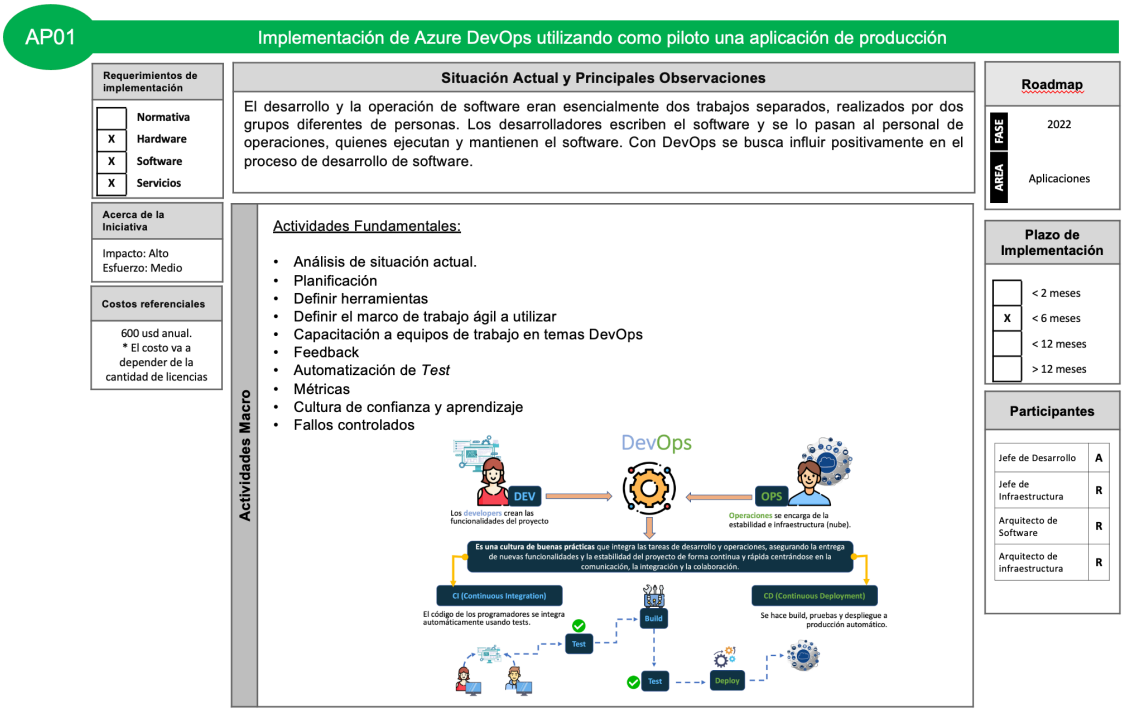


Figura 6.3 Implementación de Azure DevOps utilizando como piloto una aplicación de producción

6.2.1.2 Detalle de la implementación

A continuación, se detalla la estrategia a utilizar para la implementación de DevOps en la Fábrica de Software de UDLA.

a) Análisis de situación actual.
b) Planificación
c) Definir herramientas
d) Definir el marco de trabajo ágil a utilizar
e) Capacitación a equipos de trabajo en temas *DevOps*
f) Realizar *feedback* continuos
g) Automatización de *Tests*
h) Definir Métricas

6.2.1.3 Características de la implementación

Una vez definidos la estrategia para una implementación de *DevOps*, se aterriza en la herramienta *Azure DevOps Services* de *Microsoft*. Para lo cual se detalla lo siguiente:

a) Costos

- Gratis para equipos pequeños de hasta 5 usuarios.
- Desde $6 usd por usuario por mes

Los dos planes indicados incluyen *Azure Pipeline, Azure Boards, Azure Repos, Azure Artifacts*.

b) Pasos

- Determinar un proyecto software que se esté en producción, de preferencia que nos sea un sistema principal y con pocas funcionalidades.
- Ir a https://azure.microsoft.com/es-es/products/devops/ para iniciar sesión con la cuenta de UDLA.
- En *Azure DevOps* crear una organización.
- Crear un panel ágil para administrar el proyecto en *Azure Boards*.
- Crear un repositorio en *Azure Repos* con el código fuente del proyecto software seleccionado preferiblemente en estándar GIT.
- Definir una estrategia de *branching* para el código fuente
- Definir Pipelines para:
    - Verificar que se cumplan estándares de codificación.
    - Verificar que el código compile.
    - Ejecutar pruebas unitarias.
    - Generar los artefactos (archivos con los compilados)
    - Publicación de los compilados.
- Configurar la infraestructura para el ambiente productivo en donde desplegar el proyecto *software* mediante el uso de *Azure Functions*, *App Services, Virtual Machines*.
- Crear un *Pipeline* para los *Release*
- Realizar las pruebas funcionales.
- *Configurar Applications Insights* para el monitoreo de rendimiento y logueo de errores de las aplicaciones.

6.3 Arquitectura de Infraestructura base

6.3.1 Iniciativa: Uso de ambientes no productivos en la nube de Azure bajo el esquema de integración continua de DevOps

Una tendencia en las organizaciones a nivel de infraestructura tecnológica es dejar de tener servidores *On-Premises* debido a los costos de mantenimiento y preocupaciones que implica mantener estos recursos.

La infraestructura como servicio (IaaS) es una forma de informática en la nube que proporciona recursos informáticos virtualizados. Los ambientes no productivos (*DevTest*) son un enfoque de desarrollo de software que integra las pruebas en una etapa temprana de la fase de desarrollo reduciendo los costos y la sobrecarga de los entornos de desarrollo y pruebas, a la vez que facilita un desarrollo más rápido a través de la integración e implementación automatizada de máquinas virtuales e imágenes de estas. (Learn Microsoft, 2022).

6.3.1.1 Conceptualización de la iniciativa

En base a lo anterior indicado, en la **Figura 6.4** se presenta una conceptualización de la iniciativa

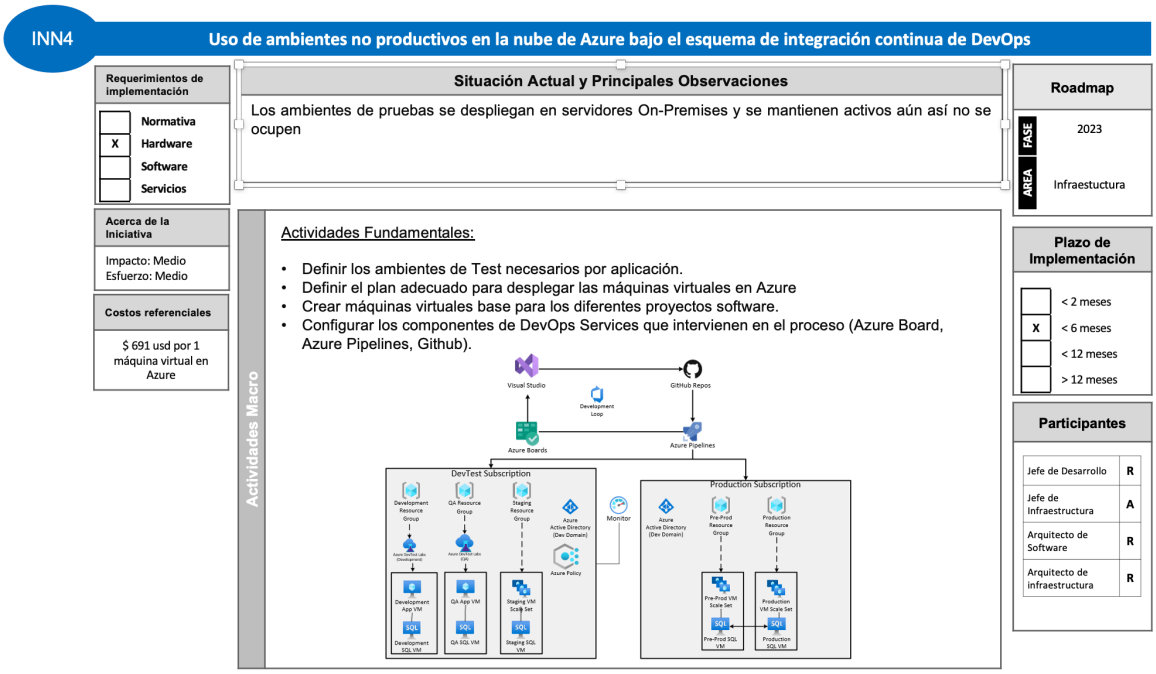


Figura 6.4 Uso de ambientes no productivos en la nube de Azure bajo el esquema de integración continua de DevOps

6.3.1.2 Detalle de la implementación

Para el Área de la Fábrica de Software se propone llevar los ambientes no productivos de las principales aplicaciones a la nube de Azure para maximizar las ventajas brindadas por *Azure DevOps*. En base a esto se deberán implementar los siguientes ambientes:

a) Entornos de desarrollo

Son las estaciones de trabajo de los miembros del equipo que cuentan con el software necesario para realizar actividades de desarrollo. Se pueden tener estaciones de trabajo de los siguientes tipos:

- Máquinas desktop o laptops
- *Microsoft Dev Box*
- *Visual Studio Code* para la *Web*

b) Entornos de *Test*

Este entorno está principalmente enfocado para los miembros del equipo que realizan labores de pruebas sobre las aplicaciones desarrolladas. La idea principal es que dispongan de imágenes de Máquinas Virtuales con las versiones de las aplicaciones listas para probar.

Se debe contar con dos sub ambientes, para garantizar que se cubra de manera integral todas las etapas de pruebas.

- *Quality Assurance (QA)*
- *Staging*

c) Componentes

Para los entornos anteriormente descritos, se necesita trabajar en conjunto con los siguientes componentes:

- ***Azure DevTest Labs*.** Laboratorios con las herramientas y software necesario para crear entornos.
- ***Azure VM Imagen Builder*.** Proporciona imágenes de máquinas virtuales en base a referencias dadas
- ***Shared Image Gallery*.** Repositorio de imágenes de máquinas Virtuales.
- ***GitHub*.** Plataforma de hospedaje, control y manejo de versiones de código de fuente.
- ***Azure Pipelines*.** Maneja la implementación de las imágenes de las máquinas virtuales.
- ***Azure Key Vault*.** Almacena de forma segura y controla el acceso a claves de APIs, contraseñas y certificados.
- ***Azure Boards*.** Administra el trabajo de los proyectos en ejecución mediante marcos de trabajo ágil.
- ***Azure Active Directory*.** Plataforma para gestionar la identidad empresarial y de los miembros de esta.
- ***Azure Monitor.*** Monitorea las Máquinas Virtuales en los entornos propuestos recopilando datos y mostrándolos en paneles informativos.

6.3.1.3 Características de la implementación

a) Microsoft Dev Box.

En caso de que se haga uso de esta herramienta, se deberá tomar en cuenta lo siguiente:

- Requerimientos
  - Windows 10 Enterprise o Windows 11 Enterprise
  - *Microsoft EndPoint Manager*

- Azure Active Directory P1
- Licenciamiento Microsoft 365 E3, E5, A3, A5, Education Student Use

- Costos
  - Computo

Característica	Precio
4 vCPU, 16 GiB Memory	$0,99 por hora
8 vCPU, 32 GiB Memory	$1,98 por hora

  - Almacenamiento

Característica	Precio
SSD 256 GiB	$0,053 por hora
SSD 512 GiB	$0,105 por hora
SSD 1024 GiB	$0,21 por hora

b) **Entornos de *Test***

La recomendación de Microsoft para este ambiente es tener máquinas virtuales de la Serie A, cuya configuración de memoria y rendimiento de CPU son adecuadas para este tipo de cargas de trabajo. Son económicas y son una opción de bajo costo para empezar en Azure.

- Características
  - **Plan:** *Azure DevTest*
  - **Instancia:** Estándar A3
  - **Total de *Cores*:** 4
  - **Memoria:** 7 GB RAM
  - **Disco:** 285 GB
- Precio
  - **Número de horas:** 160
  - **Precio por hora:** 0,360 usd
  - **Total:** $57,60 usd

6.3.2 Iniciativa: Implementar políticas y procedimientos para regular y normal el uso de los ambientes no productivos

Dentro de la fábrica de software, es común que los desarrolladores necesiten máquinas virtuales (VM) y entornos de desarrollo, a medida que iteran en las aplicaciones. Así mismo, en el área de calidad de software se necesitarán entornos idénticos a producción para realizar las pruebas respectivas.

Para evitar conflictos es necesario establecer políticas y normas que regulen el uso de los entornos y establecer lineamientos entre el rendimiento que necesitan y los costos.

6.3.2.1 Conceptualización de la iniciativa

En base a lo anterior indicado, en la **Figura 6.5** se presenta una conceptualización de la iniciativa.

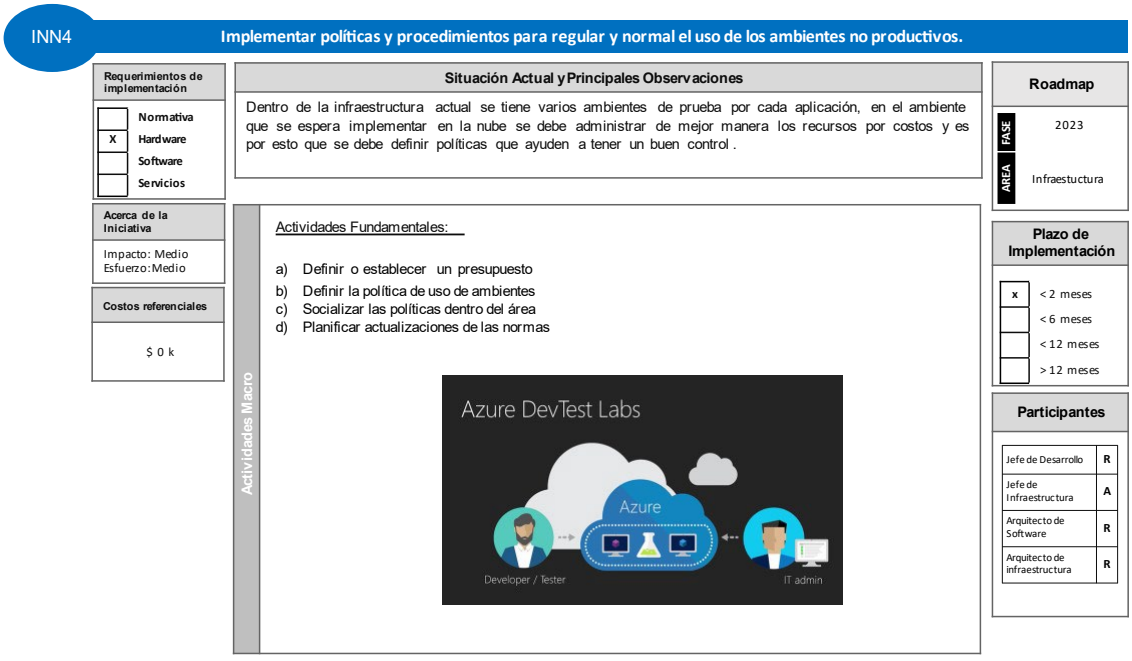


Figura 6.5 Implementar políticas y procedimientos para regular y normal el uso de los ambientes no productivos

6.3.2.2 Detalle de la implementación

Para realizar la implementación de las políticas y normas, se debería seguir los siguientes lineamientos en base a costos de los servicios.

a) Definir o establecer un presupuesto

Se debería definir o tener claro el presupuesto destinado para las suscripciones ya que acorde a esto se puede supervisar y controlar los costos.

Dentro de la suscripción se tienen ciertas acciones que se las puede realizar de forma gratuita en la

Tabla 6.6 se muestra el detalle:

Tabla 6.6 Detalle precios Azure DevTest Labs

Gratis	Facturado
• Crear un laboratorio de desarrollo y pruebas. • Crear plantillas en el laboratorio. • Usar plantillas como base para crear máquinas virtuales (VM). • Agregar un repositorio de artefactos. Los orígenes de artefactos son repositorios de Git (VSO-GIT, GitHub, etc.) en los que se insertan los artefactos (archivos JSON basados en ARM). • Instalar artefactos después de aprovisionar una máquina virtual. Los artefactos pueden ser: • Herramientas que quiere instalar en la máquina virtual, tales como agentes, Fiddler, Visual Studio, etc. • Acciones que quiere ejecutar en la máquina virtual, tales como clonar repositorios o configurar algo. • Aplicaciones que desea probar.	• Storage. El laboratorio crea automáticamente cuentas de almacenamiento para almacenar lo siguiente: • Archivos VHD utilizados para crear plantillas de laboratorio personalizadas • Discos de SO de las máquinas virtuales creadas en el Laboratorio • Más información sobre los precios de Azure Storage • Virtual Network. El laboratorio crea automáticamente una red virtual y todas las máquinas virtuales se agregan a ella. Más información sobre Virtual Network. • Máquinas virtuales creadas en el Laboratorio. Más información sobre los precios de Linux Virtual Machines. • Azure Key Vault. El Laboratorio necesita credenciales de usuario para comunicarse con el repositorio de Git privado de un usuario. El laboratorio almacena estas credenciales en Azure Key Vault. Más información sobre los precios de Key Vault.

b) Definir la política de uso de ambientes

En base al presupuesto establecido se puede establecer una política y control de uso de ambientes en base a lo siguiente:

- Limitar el número de máquinas virtuales que cada usuario puede crear o reclamar.
- Permitir solo determinados tamaños de máquina virtual en el laboratorio.
- Configurar directivas de apagado automático e inicio automático para detener y reiniciar todas las máquinas virtuales en determinadas horas del día. El apagado automático de las máquinas virtuales no se aplica a los recursos de PaaS en entornos.
- Administración de destinos de costos y notificaciones.

Para definir esta política se debería establecer reuniones entre los involucrados y basarse en las aplicaciones, requerimientos actuales y presupuesto para establecer límites y normas.

c) Socializar las políticas dentro del área

Una vez definidas las normas y políticas se debe realizar una socialización dentro del área de la fábrica de software para informar el uso de los ambientes y cuáles son las limitantes que se tiene de esta forma cada uno podrá planificar mejor su trabajo y desarrollos.

d) Planificar actualizaciones de las normas

Debido a que estos servicios pueden tener cambios y pueden existir cambios dentro de las suscripciones como mejoras o nuevos servicios se debe mantener en constante revisión y actualizar de ser el caso. Para esto se debería definir un responsable que este en constante revisión y de ser el caso actualización de las políticas.

7 Plan de Migración

En este capítulo se prioriza las iniciativas identificadas en las arquitecturas de Negocio, aplicaciones e infraestructura basado en el impacto producido y el esfuerzo necesario para implementarlas. Se propondrá una hoja de ruta que de una visión para la ejecución de los estas.

7.1 Priorización

7.1.1 Análisis de impacto

El análisis de impactos se representa mediante una matriz de correlación de los objetivos de la fábrica de software con las iniciativas. Se evaluarán mediante la rúbrica de la

Tabla 7.1.

Tabla 7.1 Escala de impacto de las iniciativas

Escala de Impacto		
Bajo:	Entre 0 - 0,7	Bajo
Medio:	Entre 0,7 y 1,4	Medio
Alto:	Entre 1,4 y 2	Alto

Tabla 7.2 Impacto de las iniciativas para cierre de brechas

				Objetivos								
				20%	15%	15%	20%	15%	15%	1		
No	Dominio	Id	Iniciativa	Sostenibilidad de los servicios de TI	Contribuir a la experiencia del cliente interno	Crear e implementar software de manera ágil que opere en forma segura y resiliente	Prestación de servicios I&T conforme a los requisitos del negocio	Fortalecer los controles de aseguramiento de la calidad al software desarrollado	Desarrollar innovación tecnológica para generar productos disruptivos	Valoración cualitativa	Impacto
1	Negocio	NE01	Programa de capacitaciones al equipo de desarrollo en temas tecnológicos y metodológicos							2,00	Alto
2	Negocio	NE02	Establecimiento de un área de aseguramiento de la calidad de software							1,50	Alto
3	Aplicaciones	AP01	Implementación de Azure DevOps utilizando como piloto una aplicación de producción							1,50	Alto
4	Infraestructura	IN01	Uso de ambientes no productivos en la nube de Azure bajo el esquema de integración continua de DevOps							0,75	Medio
5	Infraestructura	IN02	Implementar políticas y procedimientos para regular y normal el uso de los ambientes no productivos							0,80	Medio

Del análisis de la **Tabla 7.2**, se considera que lo pertinente sobre las cinco iniciativas planteadas, es dar mayor prioridad a las tres de valor más alto ya que generarán mayor impacto y valor en el proceso de fortalecimiento planteado en la Fábrica de Software.

7.1.2 Análisis de Esfuerzo

En la **Tabla 7.5** se evalúan las iniciativas planteadas tomando en cuenta los esfuerzos económicos, la complejidad en implementarlos y la capacidad de TI para asimilar estas nuevas operaciones tecnológicas. La evaluación se realiza mediante la rúbrica de las tablas Tabla 7.3 y Tabla 7.4.

Tabla 7.3 Escala de esfuerzos para las iniciativas

Escala de Esfuerzo		
Bajo:	Entre 1 - 1,7	Bajo
Medio:	Entre 1,7 y 2,4	Medio
Alto:	Entre 2,4 y 3	Alto

Tabla 7.4 Escala de esfuerzos de recursos económicos para las iniciativas

Valor	Recursos Económicos	Complejidad	Capacidad IT	
1	Menor a $10.000	Baja	Listo	Bajo
2	Entre $10.000 y $60.000	Media	En Proceso	Medio
3	Mayor a $60.000	Alta	A Futuro	Alto

Tabla 7.5 Iniciativas en base al esfuerzo económico

				Criterios Esfuerzo				
				40%	30%	30%		
No	Área	Id	Iniciativa	Recursos Económicos	Complejidad	Capacidad TI	Suma ponderada	Esfuerzo
1	Negocio	NE01	Programa de capacitaciones al equipo de desarrollo en temas tecnológicos y metodológicos				1,30	Bajo
2	Negocio	NE02	Establecimiento de un área de aseguramiento de la calidad de softw are				2,30	Medio
3	Aplicaciones	AP01	Implementación de Azure DevOps utilizando como piloto una aplicación de producción				2,20	Medio
4	Infraestructura	IN01	Uso de ambientes no productivos en la nube de Azure bajo el esquema de integración continua de DevOps				1,90	Medio
5	Infraestructura	IN02	Implementar políticas y procedimientos para regular y normal el uso de los ambientes no productivos				1,90	Medio

7.1.3 Fases

En la

Tabla 7.6, se combina los resultados obtenidos del análisis de impacto y esfuerzo priorizándolos en tres fases para su ejecución posterior mediante la planificación plasmada en un cronograma de actividades con sus respectivas estimaciones.

Tabla 7.6 Iniciativas ordenas por fases de implementación

No	Dominio	Id	Iniciativa	Impacto	Esfuerzo	Prioridad	Fase
1	Negocio	NE01	Programa de capacitaciones al equipo de desarrollo en temas tecnológicos y metodológicos	Alto	Bajo	Habilitante	1
5	Aplicaciones	AP01	Implementación de Azure DevOps utilizando como piloto una aplicación de producción	Alto	Medio	Alta	2
4	Negocio	NE02	Establecimiento de un área de aseguramiento de la calidad de softw are	Alto	Medio	Alta	2
3	Infraestructura	IN01	Uso de ambientes no productivos en la nube de Azure bajo el esquema de integración continua de DevOps	Medio	Medio	Alta	3
2	Infraestructura	IN02	Implementar políticas y procedimientos para regular y normal el uso de los ambientes no productivos	Medio	Medio	Media	3

7.2 Análisis de dependencias

En la **Tabla 7.7** se muestra un análisis de dependencias entre las iniciativas planteadas para determinar en base a esto el orden en que se podrían atender.

Tabla 7.7 Análisis de dependencias

Id.	Dominio	Iniciativa	Dependencia
NE01	Negocio	Programa de capacitaciones al equipo de desarrollo en temas tecnológicos y metodológicos	No depende de otro proyecto es el punto de partida para iniciar con las iniciativas de DevOps
NE02	Negocio	Establecimiento de un área de aseguramiento de la calidad de software	No tiene dependencias
AP01	Aplicaciones	Implementación de Azure DevOps utilizando como piloto una aplicación de producción	Depende de NE01, se debe tener conocimientos sobre DevOps
IN01	Infraestructura	Uso de ambientes no productivos en la nube de Azure bajo el esquema de integración continua de DevOps	Depende de IN02, para iniciar con implementaciones sea en pruebas y en producción se debe tener claro el procedimiento y normativas
IN02	Infraestructura	Implementar políticas y procedimientos para regular y normal el uso de los ambientes no productivos	Depende de NE01 se debe tener un conocimiento al menos de las bases de DevOps para iniciar reuniones de establecimiento de políticas y normas para el uso de ambientes

7.3 Plan de migración detallado

A continuación, se muestra una planificación de actividades y fechas a alto nivel que muestra la estimación de tiempos de las iniciativas planteadas en las diferentes fases.

En la Tabla 7.8 Plan de migración con tiempos de duración**Tabla 7.8** se muestra cada una de las iniciativas ordenas por fase y prioridad con las fechas de ejecución de cada una.

Tabla 7.8 Plan de migración con tiempos de duración

Fase	Dominio	Id	Iniciativas	Fecha Inicio	Fecha Fin	Días de duración
1	Negocio	NE01	Programa de capacitaciones al equipo de desarrollo en temas tecnológicos y metodológicos	15/12/2022	31/1/2023	34
2	Aplicaciones	AP01	Implementación de Azure DevOps utilizando como piloto una aplicación de producción	1/2/2023	15/3/2023	31
2	Negocio	NE02	Establecimiento de un área de aseguramiento de la calidad de software	10/1/2023	15/4/2023	69
3	Infraestructura	IN02	Implementar políticas y procedimientos para regular y normal el uso de los ambientes no productivos	1/5/2023	31/5/2023	23
3	Infraestructura	IN01	Uso de ambientes no productivos en la nube de Azure bajo el esquema de integración continua de DevOps	1/6/2023	15/7/2023	32

7.4 Roadmap

En la **Figura 7.1** se muestra en una línea de tiempo el orden en el que las iniciativas se irán implementando de acuerdo con el plan de migración.

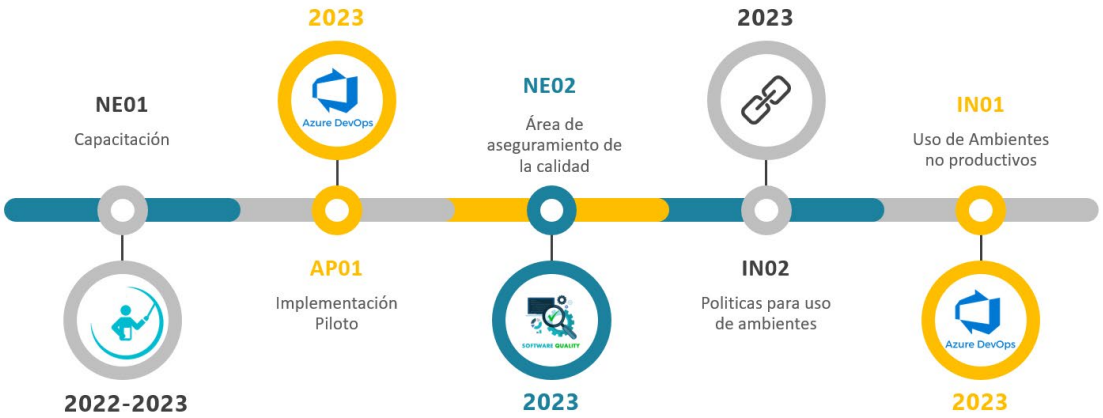


Figura 7.1 Roadmap de implementación de iniciativas

Conclusiones

- Los cambios en las organizaciones son mucho más rápidos en comparación a años anteriores, lo que implica que las áreas de desarrollo deben estar preparadas para poder responder a las nuevas necesidades de una manera ágil y dinámica, en donde se reduzca el *time to market* de los productos generados y contribuir de esa manera a los objetivos estratégicos empresariales. Esto genera escenarios de incertidumbre que se deben abordar con enfoques ágiles como Scrum para generar productos funcionales en iteraciones cortas de tiempo o a su vez plantear la elaboración de productos mínimos viables que generen valor.
- Es importante hacer énfasis en un cambio de mentalidad o *mindset* no solamente de los miembros de la Fábrica de Software de UDLA, también de los altos directivos y todas las personas que son parte del área de TI y de la organización en general para que de esa manera se puedan abordar los requerimientos institucionales de una manera integral. Es importante que se adopte orgánicamente la implementación de un marco de trabajo

ágil que ayude a generar valor en todas las instancias de UDLA de manera rápida, eficaz y confiable.

- Este trabajo evidencia la importancia de apoyarse en herramientas tecnológicas para facilitar el ciclo de vida de desarrollo software que va desde identificar el problema hasta ponerlo en producción y monitorear su funcionamiento; así como, tener siempre a la mano los estándares y buenas prácticas de la industria en referencia al desarrollo de software para apuntalar los procesos que permitan un proceso de mejora continua y fortaleciendo esta fábrica de software.
- El factor humano es fundamental e imprescindible en todas las organizaciones, tomando como base esto, se debe procurar el mantener un equipo de trabajo empoderado, motivado, comprometido y capacitado para la ejecución de todos los proyectos software que se emprendan como parte de la estrategia organizacional.
- El aprovechar las bondades de las plataformas del tipo Low Code contribuye a democratizar el desarrollo, poniendo al alcance de todos los miembros de la organización la oportunidad de crear sus propias soluciones a problemas cotidianos dentro de sus ámbitos laborales y generar ganancias rápidas y palpables.
- El presente trabajo deja una enseñanza importante que impacta en la manera de cómo abordar los problemas empresariales que se van presentando a diario en las organizaciones. Forja en nosotros como profesionales el pensamiento crítico en donde predomine la estrategia por sobre los temas operativos, permitiendo de esta manera ser eficaces y asertivos en la generación de valor basados en la toma de decisiones.

Recomendaciones

- Implementar como política institucional el requisito que los miembros de la fábrica de software anualmente obtengan al menos dos nuevas certificaciones en temas que estén acorde a los procesos o tecnologías que se manejan en la Fábrica de Software. Esto debería ir acompañado de un programa de incentivos, de tal modo que la formación sea tomada como un aspecto motivacional y de crecimiento.
- Si bien el implementar DevOps acarreará beneficios a toda el área y por consiguiente a la organización, se recomienda que la siguiente iniciativa a trabajar sobre la base de la implementación de DevOps, sea la adopción de prácticas de seguridad en cada fase del ciclo de desarrollo *software* conocidas como DevSecOps.
- Se recomienda realizar una segunda iteración en la cual se consideren buenas prácticas de seguridad durante el ciclo de vida del software. Se deberían tomar en cuenta técnicas de desarrollo basados en las buenas prácticas recomendadas por OWASP y establecer una política de desarrollo seguro donde se definan todos los lineamientos tales como: responsables, documentación técnica, protección de claves, seguridad en los ambientes, herramientas para la gestión de código y versionamiento recomendado por la práctica.

Bibliografía

¿Qué es Azure Pipelines? (01 de 08 de 2022). Obtenido de Microsoft.com: https://docs.microsoft.com/es-es/azure/devops/pipelines/get-started/what-is-azure-pipelines?view=azure-devops

A. Perez. (07 de 09 de 2021). *¿Conoces la metodología del Project Management Institute (PMI)?* Obtenido de OBS Business School: https://www.scribbr.es/detector-de-plagio/generador-apa/new/webpage/

Alvarado, K. L. (2021). *6. Fases y Procesos Scrum ::*. Obtenido de 6. Fases y Procesos Scrum ::: https://www.katherineluyoalvarado.com/6-fases-y-procesos-scrum/

Cmmi Institute. (2022). *Desarrollo CMMI*. Obtenido de cmmiinstitute: https://cmmiinstitute.com/cmmi/dev/

FlowForma. (Enero de 2021). *No Code Vs Low Code Process Automation*. Obtenido de FlowForma: https://www.flowforma.com/no-code-vs-low-code-process-automation

Greaves , K., & Laing, S. (2013). *The Manifesto for Agile Software Testing* . Obtenido de Growingagile: https://www.growingagile.co.za/2015/04/the-testing-manifesto/

Herrera Uribe, E., & Valencia Ayala, L. (Mayo de 2007). Del manifiesto ágil sus valores y principios. *Scientia Et Technica, 13*(34), 381-386.

ISACA. (2021). Construyendo un modelo de madurez para COBIT 2019. *ISACA JOURNAL*, 2.

Isolution. (10 de 12 de 2020). *CMMI - Áreas de proceso clave*. Obtenido de https://isolution.pro/es/t/cmmi/cmmi-process-areas/cmmi-areas-de-proceso-clave: https://isolution.pro/es/t/cmmi/cmmi-process-areas/cmmi-areas-de-proceso-clave

Learn Microsoft. (2022). *DevTest y DevOps para soluciones IaaS*. Obtenido de https://learn.microsoft.com/es-es/azure/architecture/solution-ideas/articles/dev-test-iaas

Manifiesto por el Desarrollo Ágil de Software. (2001). Obtenido de Manifiesto por el Desarrollo Ágil de Software: http://agilemanifesto.org/iso/es/manifesto.html

McFeeley, B. (1996). *IDEALSM: A User's Guide for Software Process Improvement.*

Microsoft. (2020). *Customer Stories*. Obtenido de Powerapps: https://powerapps.microsoft.com/es-es/customer-stories/

Microsoft. (2021). *Gartner® Magic Quadrant™ for Enterprise Low-Code Application Platforms*. Obtenido de Microsoft: https://info.microsoft.com/ww-landing-gartner-MQ-for-LCAP-landing-and-confirmation.html?lcid=en-us

Microsoft. (22 de Marzo de 2022). *¿Qué es Power Apps?* Obtenido de Documentación de Power Apps: https://docs.microsoft.com/es-es/power-apps/powerapps-overview

Microsoft. (22 de Marzo de 2022). *¿Qué es Power Apps?* Obtenido de Documentación de Power Apps: https://docs.microsoft.com/es-es/power-apps/powerapps-overview

Microsoft Azure. (2021). *DevOps frente al método ágil*. Obtenido de Microsoft Azure: https://azure.microsoft.com/es-es/overview/devops-vs-agile/

Microsoft Learn. (27 de septiembre de 2022). Obtenido de Microsoft Learn: https://learn.microsoft.com/es-es/azure/devtest-labs/devtest-lab-overview

Netapp. (2019). *¿Qué es DevOps?* Obtenido de Netapp: https://www.netapp.com/es/devops-solutions/what-is-devops/

NetApp. (2019). *¿Qué es DevOps? - Explicación de prácticas y beneficios*. Obtenido de NetApp: https://www.netapp.com/es/devops-solutions/what-is-devops/

Owasp. (2022). *Who is the OWASP® Foundation?* Obtenido de Owasp: https://owasp.org/

Salazar, A. (16 de 10 de 2016). *prozessgroup*. Obtenido de Procesos de SCRUM: http://www.prozessgroup.com/procesos-de-scrum/#:~:text=Los%20procesos%20de%20Scrum%20corresponden,dentro%20de%20un%

Sayers. (19 de 06 de 2021). *DevOps at scale: How to build your software factory*. Obtenido de https://techbeacon.com/devops/devops-scale-how-build-your-software-factory.

Sfia. (2022). *Acerca de SFIA*. Obtenido de sfia-online: https://sfia-online.org/es/about-sfia/about-sfia

SFIA 8. (28 de Septiembre de 2021). *Acerca de SFIA*. Obtenido de SFIA online: https://www.sfia-online.org/es/about-sfia

Sicma21. (02 de 12 de 2021). *Soluciones Integrales para la Industria 4.0*. Obtenido de https://www.sicma21.com/que-es-software-factory/#Por_que_necesitas_una_software_factory: https://www.sicma21.com/que-es-software-factory/#Por_que_necesitas_una_software_factory

Universidad de Las Américas. (2020). *MISIÓN Y VISIÓN*. Obtenido de Portal Udla: https://www.udla.edu.ec/misionvision/