



MAESTRÍA EN GERENCIA DE SISTEMAS Y TECNOLOGÍA EMPRESARIAL

Fortalecimiento de la fábrica de software in house en una entidad financiera

AUTOR

Mayra Alejandra Moreno Tamayo

AÑO

2021



UNIVERSIDAD DE LAS AMÉRICAS

FACULTAD DE POSGRADOS

TEMA:

FORTALECIMIENTO DE LA FÁBRICA DE SOFTWARE IN

HOUSE EN UNA ENTIDAD FINANCIERA

TRABAJO DE TITULACIÓN PRESENTADO EN CONFORMIDAD CON LOS  
REQUISITOS ESTABLECIDOS PARA OPTAR POR EL TÍTULO DE  
MAGISTER EN GERENCIA EN SISTEMAS Y TECNOLOGÍA EMPRESARIAL

TUTOR

MsC. Germán Pancho

AUTOR:

MAYRA ALEJANDRA MORENO TAMAYO

AÑO:

2021

## RESUMEN

El presente trabajo muestra un análisis de la situación actual de la fábrica de software de una prestigiosa entidad bancaria constituida en Ecuador. La entidad financiera es considerada una de las más grandes del país, pero no la mejor, ya que posee varias deficiencias tecnológicas que han causado pérdida de prestigio y clientes.

Se evidencia que el proceso de desarrollo de software y la metodología actual tienen varias deficiencias que afectan a la seguridad, a la eficiencia de los equipos, satisfacción del cliente, disponibilidad y estabilidad de las aplicaciones. Además de que existe mucha burocracia en sus procesos que ralentizan la entrega de los proyectos.

Luego del análisis realizado se han identificado varias oportunidades de mejora para solventar estas deficiencias. El uso del marco de trabajo ágil Scrum será el punto de inicio de la transformación digital recomendada para la organización, además de poner al cliente en el centro de los proyectos, escuchándolos y fabricando software que se adapte a sus necesidades.

Existen varios procesos manuales que pueden ser fácilmente reemplazados por procesos automáticos, para ello la implementación incremental de una cultura DevOps ayudará a generar innovación rápidamente y mermar la burocracia existente.

## **ABSTRACT**

This work shows an analysis of the software factory's current situation of a prestigious bank institution in Ecuador. The financial institution is considered one of the largest in the country, but not the best because it has several technological deficiencies.

It is evident that the software development process and the current methodology have several security deficiencies, equipment efficiency, customer satisfaction, availability and applications stability. In addition, there is a lot of bureaucracy in its processes that slow down the delivery of projects.

After the analysis, several improvement opportunities have been identified to solve these deficiencies. The starting point recommended is to use the agile Scrum methodology for the digital transformation. In addition, to putting the client at the center of the projects, listening to them and manufacturing software that adapts to their needs.

There are several manual processes that can be easily replaced by automatic processes. Implements a DevOps culture incremental adoption will help drive innovation quickly and reduce existing bureaucracy.

## Tabla de contenido

1. Fase preliminar .....	1
1.1 Contexto .....	1
1.2 Matriz FODA .....	4
1.3 BMM .....	5
1.4 Organización impactada.....	5
1.5 Stakeholders y expectativas de valor .....	7
1.6 Marcos de referencia .....	7
1.7 Equipo de arquitectura .....	10
1.7.1 Roles y responsabilidades.....	11
1.8 Catálogo de principios .....	12
2. Visionamiento arquitectónico.....	13
2.1 Requerimientos de alto nivel .....	14
2.2 Visionamiento y escenarios de la solución.....	15
2.2.1 Empresas referentes.....	15
2.2.2 Escenario de fábrica de software ideal .....	18
2.2.3 Escenario de fábrica de software objetivo .....	20
2.2.4 Resultado de visionamiento.....	21

2.3	Análisis de brechas.....	21
2.4	Definición de arquitectura objetivo.....	23
2.4.1	Target de arquitectura de negocio.....	23
2.4.2	Target de arquitectura de datos .....	24
2.4.3	Target de arquitectura de aplicaciones .....	25
2.4.4	Target de arquitectura de infraestructura base.....	26
3.	Arquitectura de Negocios.....	28
3.1	Arquitectura base de negocio.....	28
3.1.1	Fases.....	28
3.1.2	Roles .....	32
3.2	Arquitectura objetivo de negocio.....	33
3.2.1	Fases.....	34
3.2.2	Eventos utilizados durante cada sprint.....	37
3.2.3	Artefactos utilizados durante cada sprint .....	37
3.2.4	Roles y responsabilidades involucrados en el marco de trabajo: ...	38
3.2.5	Análisis de brechas.....	41
4.	Arquitectura de aplicaciones/datos.....	43
4.1	Arquitectura base de aplicaciones .....	43

4.2	Arquitectura base de datos/información .....	47
4.2.1	Datos estructurados.....	47
4.2.2	Datos no estructurados.....	47
4.3	Arquitectura objetivo de datos y aplicaciones.....	48
4.4	Análisis de brechas.....	53
5.	Arquitectura de infraestructura base.....	57
5.1	Arquitectura base de Infraestructura.....	57
5.1.1	Ambiente de desarrollo.....	60
5.1.2	Ambiente productivo o de producción.....	61
5.2	Arquitectura objetivo de infraestructura.....	63
5.2.1	Ambientes para desarrollo.....	64
5.2.2	Implementación de Google Cloud Platform .....	65
5.3	Análisis de brechas.....	67
6.	Oportunidades y soluciones .....	68
6.1	Consolidación de iniciativas.....	68
6.2	Conceptualización de iniciativas.....	69
6.2.1	Proyecto PN01: Implementación marco de trabajo ágil Scrum.....	69
6.2.2	Proyecto PN02: Talleres de capacitación de nuevos roles y responsabilidades.....	73

6.2.3 Proyecto PN03: Definir una metodología ágil de pruebas (objetivo automatización).....	75
6.2.4 Proyecto PN04: Adopción de una cultura DevOps en la organización.....	78
6.2.5 Proyecto PA01: Usar herramientas de apoyo para DevOps.....	81
6.2.6 Proyecto PA02: Crear una célula dedicada a la implementación de un CRM institucional.....	85
6.2.7 Proyecto PA03: Adquirir licencias de Jira para todas las personas involucradas en el nuevo marco de trabajo. ....	87
6.2.8 Proyecto PI01: Usar los servicios de infraestructura de Google Cloud Platform.....	88
6.2.9 Proyecto PI02: Usar ApiGee para la gestión de APIs .....	90
<b>7. Plan de Migración.....</b>	<b>91</b>
7.1 Análisis de impacto.....	91
7.2 Análisis de esfuerzo .....	92
7.3 Fases.....	93
7.4 Análisis de dependencias .....	94
7.5 Roadmap.....	95
7.6 Cronograma de actividades .....	95
7.7 Conclusiones .....	98



7.8	Recomendaciones.....	98
-----	----------------------	----

## Índice de figuras

Figura 1 Matriz FODA .....	4
Figura 2 BMM (Modelo de motivación de negocio).....	5
Figura 3 Equipo de arquitectura.....	11
Figura 4 Diseño conceptual de requerimientos de alto nivel .....	15
Figura 5 Cultura Ágil en Spotify (Deloitte, 2020) .....	17
Figura 6 Marco de acción Segacy .....	18
Figura 7 Flujo de trabajo Scrum (Roles, artefactos y eventos) (Pablo Marichal, 2016).....	20
Figura 8 Escenario de fábrica objetivo .....	21
Figura 9 Gráfico resultante del análisis de brechas .....	23
Figura 10 Arquitectura de negocio adaptado de: (Agesic, s. f.).....	24
Figura 11 Arquitectura de datos objetivo.....	24
Figura 12 Herramientas utilizadas en DevOps adaptado de: (Singh, 2019) .....	25
Figura 13 Arquitectura de infraestructura base .....	27
Figura 14 Arquitectura base de negocio.....	33
Figura 15 Arquitectura objetivo de negocio .....	41
Figura 16 Arquitectura base de aplicaciones .....	47
Figura 17 Arquitectura objetivo de aplicaciones.....	53
Figura 18 Gráfico resultante del análisis de brechas .....	55

Figura 19 Arquitectura base de infraestructura .....	60
Figura 20 Ambientes/Entornos actuales, adaptado de (Autentia, 2020).....	61
Figura 21 Arquitectura objetivo de infraestructura .....	63
Figura 22 Ambientes/Entornos objetivo, adaptado de (Autentia, 2020).....	65
Figura 23 Dashboard de Compute Engine de Google .....	66
Figura 24 Dashboard de App Engine de Google .....	67
Figura 25 Herramienta User story mapping (How to Use a Story Map in Jira, 2019).....	72
Figura 26 Modelo SFIA .....	74
Figura 27 Niveles de madurez de pruebas en TMMi (TMMi Foundation, 2021)76	
Figura 28 Flujo DevOps (RRHHDigital, 2019).....	82
Figura 29 Roadmap de iniciativas.....	95

## Índice de tablas

Tabla 1 .....	6
Organización impactada.....	6
Tabla 2 .....	7
Stackeholders y expectativas de valor .....	7
Tabla 3 .....	8
Marco de referencia Scrum.....	8
Tabla 4 .....	9
Marco de referencia TMMI .....	9
Tabla 5 .....	10
Marco de referencia SFIA .....	10
Tabla 6 .....	11
Matriz RACI .....	11
Tabla 7 .....	12
Principio: Soluciones orientadas a generar valor al negocio .....	12
Tabla 8 .....	12
Principio: Fallar rápido, fallar barato .....	12
Tabla 9 .....	13
Principio: Calidad del software.....	13

Tabla 10.....	13
Principio: Arquitectura escalable basada en microservicios.....	13
Tabla 11.....	22
Análisis de brechas de los referentes.....	22
Tabla 12.....	35
Roles con sus respectivas actividades durante la Fase de Ideación y diseño .	35
Tabla 13.....	36
Roles con sus respectivas actividades durante la Fase de Construcción.....	36
Tabla 14.....	42
Análisis de brechas en cada Fase de Desarrollo.....	42
Tabla 15.....	54
Niveles de madurez de las aplicaciones actuales.....	54
Tabla 16.....	56
Análisis de brechas de aplicaciones y soluciones propuestas.....	56
Tabla 17.....	57
Detalle de equipos de infraestructura utilizados actualmente.....	57
Tabla 18.....	68
Detalle de equipos de infraestructura utilizados actualmente.....	68
Tabla 19.....	69

Consolidación de iniciativas en base a cada arquitectura .....	69
Tabla 20 .....	69
Proyecto PN01: Implementación marco de trabajo ágil Scrum. ....	69
Tabla 21 .....	71
Proyecto PN01: Implementación marco de trabajo ágil Scrum. ....	71
Tabla 22 .....	73
Proyecto PN02: Talleres de capacitación de nuevos roles y responsabilidades. .....	73
Tabla 23 .....	75
Proyecto PN02: Definir una metodología ágil de pruebas (objetivo automatización). ....	75
Tabla 24 .....	78
Proyecto P0N4: Adopción de una cultura DevOps en la organización. ....	78
Tabla 25 .....	81
Proyecto PA01: Implementación de herramientas de apoyo a DevOps. ....	81
Tabla 26 .....	85
Proyecto PA02: Crear una célula dedicada a la implementación de un CRM institucional. ....	85
Tabla 27 .....	87
Proyecto PA03: Adquirir licencias de Jira para todas las personas involucradas en el nuevo marco de trabajo.....	87

Tabla 28.....	88
Proyecto PI01: Adopción de una infraestructura Cloud Computing. ....	88
Tabla 29.....	90
Proyecto PI02: Uso de API Gateway con distintos ambientes de desarrollo. ...	90
Tabla 30.....	92
Análisis de impacto de iniciativas .....	92
Tabla 31 .....	93
Análisis de esfuerzo de iniciativas .....	93
Tabla 32.....	94
Fases de implementación .....	94
Tabla 33.....	94
Análisis de dependencias entre iniciativas .....	94
Tabla 34.....	95
Cronograma para la implementación marco de trabajo ágil Scrum .....	95
Tabla 35.....	96
Cronograma para la adopción de una cultura DevOps en la organización.....	96
Tabla 36.....	96
Cronograma para los talleres de capacitación de nuevos roles y responsabilidades.....	96
Tabla 37 .....	96

Cronograma para la implementación de herramientas de apoyo a DevOps ....	96
Tabla 38 .....	96
Cronograma para la adopción de una infraestructura Cloud Computing. ....	96
Tabla 39 .....	97
Cronograma para definir una metodología ágil de pruebas.....	97
Tabla 40 .....	97
Cronograma para crear una célula dedicada a la creación de un CRM institucional. ....	97
Tabla 41 .....	97
Cronograma para adquirir licencias de Jira para todas las personas involucradas en el nuevo marco de trabajo y capacitación. ....	97
Tabla 42 .....	97
Cronograma para usar API Gateway con distintos ambientes de desarrollo....	97



## **DECLARACIÓN DE AUDITORÍA**

“Declaro que este trabajo es original, de mi 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.”

A handwritten signature in blue ink, appearing to read 'Mayra Moreno Tamayo', is centered on the page.

Mayra Alejandra Moreno Tamayo

1718445040

## **DEDICATORIA**

Este proyecto va dedicado a las tres personas que más amo en mi vida, son mi esposo y mis hijitos que gracias a su apoyo y amor he logrado culminar uno más de mis objetivos.

## **1. Fase preliminar**

### **1.1 Contexto**

La entidad bancaria referente que se ha utilizado para construir el proyecto Capstone, tiene varios años brindando a los ecuatorianos servicios y sistemas de carácter financiero, como concesión de préstamos, gestión de ahorro, entre otros, además cuenta con una posición de liderazgo en el mercado ecuatoriano siendo una de las tres instituciones más grandes del país.

Actualmente las entidades del sector público y privado del sistema financiero se encuentran bajo la supervisión y normativa de la Superintendencia de Bancos con el objetivo de preservar su seguridad, estabilidad y transparencia y así proteger los ahorros del público en general, por tal motivo cualquier proyecto, mejora o cambio en sus procesos internos o productos, deben ser notificados y normados por dicha institución.

La solvencia y la capacidad de las entidades financieras para gestionar los riesgos y cumplir con sus obligaciones con el público, se incluyen dentro de la calificación de riesgo realizada por la Superintendencia de Bancos. En el 2020 la firma calificadora PCR le otorgó la calificación AAA, denotando así que tiene una fuerte estabilidad y una sobresaliente trayectoria.

En la actualidad, debido a la alta demanda de servicios y aplicaciones digitales, surge la necesidad de adoptar una transformación digital, colocando al cliente en el centro de este gran reto que implica el reinventarse.

El Gerente del área de Desarrollo de Aplicaciones y Mantenimiento del Banco, sostiene que el desarrollo de soluciones tecnológicas, que buscan dar respuesta a los problemas complejos existentes en el resto de las áreas del Banco, son sometidos a constantes modificaciones debido a cambios en el entorno, en normativas, políticas externas o internas, actualización en el ámbito tecnológico, definiciones o reglas de negocio, entre otras.

Cuando estas modificaciones o cambios son introducidos a la mitad o en el peor de los casos al final del proyecto, impacta al área de desarrollo generando:

- Mayor esfuerzo para el equipo de desarrollo ya que, si la fecha de entrega del proyecto no se extiende, deberá realizar las modificaciones solicitadas extendiendo su horario laboral. Produciendo así desmotivación del equipo.
- Inicio de un proceso de control de cambios.
- Que se requiera modificar el cronograma de trabajo, en ciertos casos se extiende varios días o incluso semanas, retrasando el despliegue en producción.
- Eventualmente suelen solicitarse cambios que no son considerados cambios como tal, más bien son nuevos requerimientos, mismos que no fueron levantados al inicio del proyecto.
- Al introducir cambios o nuevos requerimientos casi a la culminación del proyecto, produce que el equipo de desarrollo tenga que rehacer el código y la lógica de negocio o incluso hacer de cero ciertas funcionalidades.
- Que se deba hacer un nuevo análisis y si el cambio requiere de una capacidad o conocimiento que el equipo de desarrollo no tiene, retrasa más aún la entrega del proyecto.

El tener cronogramas de trabajo extensos y el basarse únicamente en metodologías tradicionales de desarrollo de software, ha provocado que las entidades financieras competidoras entreguen soluciones tecnológicas que el Banco tardó mucho en subir al ambiente de producción.

Previo al cierre de un proyecto se realizan pruebas de aceptación de usuario (UAT), el objetivo de este tipo de pruebas es que el usuario de su aprobación para que el sistema pueda ser desplegado en producción; sin embargo en ocasiones, durante estas pruebas se detecta que existen funcionalidades que no realizan lo que se esperaba, que no se contempló ciertos requerimientos, que la aplicación no resulta del todo intuitiva, entre otras consideraciones que llevarán a prolongar la entrega del producto o servicio.

En el área de Desarrollo de aplicaciones y mantenimiento, no existe un equipo (team) de Control de calidad de software, las funcionalidades implementadas son testeadas entre los desarrolladores. La ausencia de un equipo especializado de pruebas se ha visto reflejado en el reporte de errores en la fase de pruebas de aceptación de usuario (UAT), como en el ambiente de producción, lo que genera un reproceso que en ciertos casos puede ocasionar grandes pérdidas económicas, así como daños en la reputación de la entidad. Con el apoyo de un equipo de testers, se podrá automatizar pruebas logrando así un ciclo de pruebas más ágil.

Algunas áreas del Banco utilizan aplicaciones y herramientas que se encuentran obsoletas o sistemas a los que ya no se les brinda soporte, lo que conlleva a que no puedan interoperar entre ellas.

Existe un incumplimiento de buenas prácticas de desarrollo de software, a lo que se llama deuda técnica, a largo plazo esto producirá costos extras para la empresa.

Es así que esta entidad financiera tiene un problema en la eficiencia de las soluciones tecnológicas que ofrece, ya que los proyectos son entregados lejos de la fecha planificada y sin contar con la calidad que deberían, lo cual repercute en pérdidas económicas de millones de dólares. Estas deficiencias impactan directamente en las aplicaciones y servicios que el Banco ofrece a la ciudadanía, por lo que se puede perder credibilidad y la confianza de sus clientes.

El Banco necesita actualizar sus políticas de desarrollo de software, tanto generales y específicas, estableciendo lineamientos para asegurar que las áreas que realizan adquisición, implementación y mantenimiento del software, mismo que vaya a ser utilizado en el Banco, se ejecute considerando las mejores prácticas y estándares internacionales, además de cumplir con los requerimientos que indica el ente regulatorio.

Se requiere mejorar los tiempos de entrega de las soluciones tecnológicas con el fin de apoyar a la productividad de todas las áreas del Banco. El área de TI,

además de trabajar dando mantenimiento a sistemas e implementando nuevas iniciativas que apalancan tecnológicamente a las distintas áreas para cumplir con sus metas y generar valor al negocio, debe buscar implementar proyectos de innovación que permitan al Banco estar a un nivel tecnológico mucho más alto, respecto a las entidades financieras competidoras.

## 1.2 Matriz FODA

Con el objetivo de conocer la situación actual del área de la fábrica de desarrollo de software, a continuación, se describe un análisis tanto de las fortalezas para poder consolidarlas, como de las oportunidades para poder aprovechar sus ventajas, de las debilidades para planificar cómo minimizarlas y de las amenazas para preparar a la organización a trabajar en ellas y poder convertirlas en oportunidades.



Figura 1 Matriz FODA

### 1.3 BMM

Con el objetivo de entregar a los stakeholders información relevante que fundamente las decisiones estratégicas que se van a tomar, se planteará el Modelo de Motivación de Negocio.

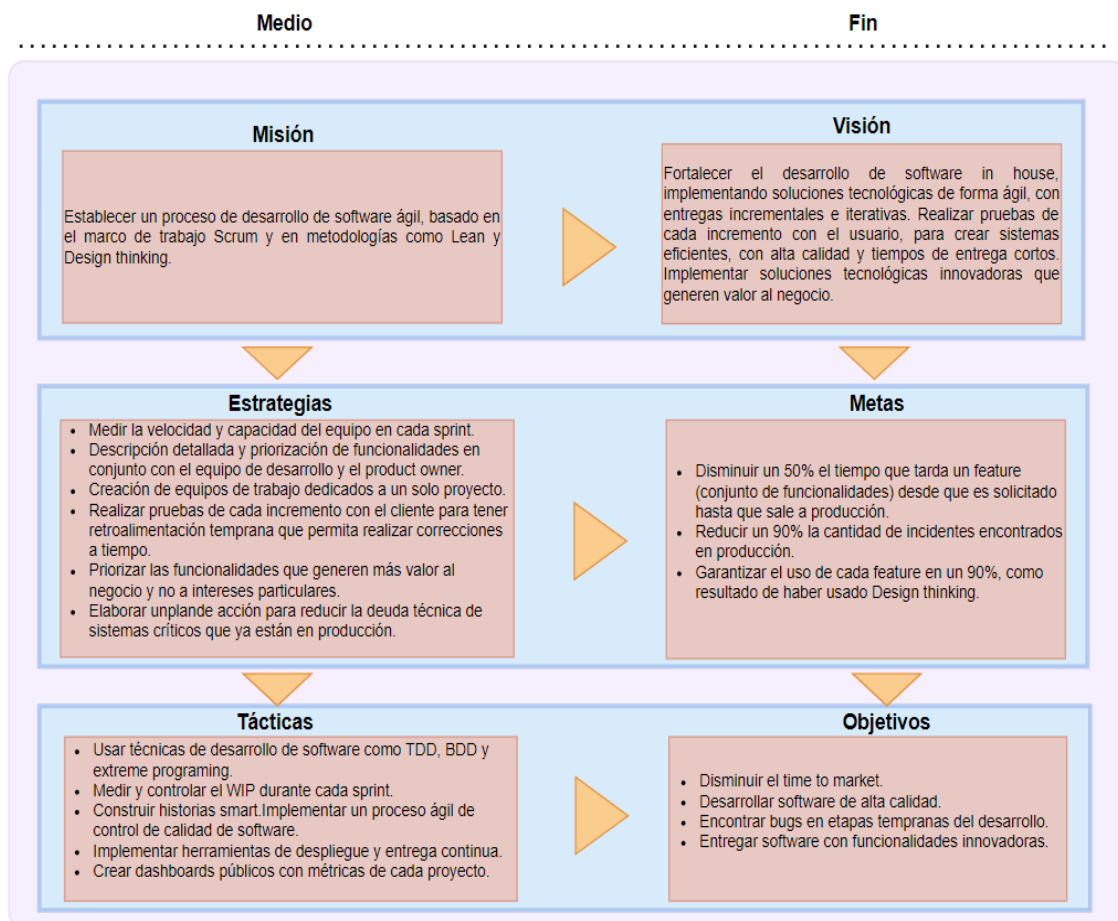


Figura 2 BMM (Modelo de motivación de negocio)

### 1.4 Organización impactada

Las áreas de la organización son clasificadas por el nivel de impacto, ya sea alto, medio y bajo. En la siguiente tabla se describe de qué forma se ve impactada cada una de las áreas.

Tabla 1  
Organización impactada

Área	Nivel de impacto	Descripción del impacto
Vicepresidencia Ejecutiva de Tecnología y Operaciones	Alto	<ul style="list-style-type: none"> <li>• Cambio en metodología del ciclo de desarrollo de software.</li> <li>• Cambio en herramientas y métricas para la calidad del software.</li> <li>• Entrega, despliegue e integración continua con los sistemas y arquitectura del Banco.</li> </ul>
Gerencia de seguridad de la información	Medio	Redefinir los estándares de seguridad para el desarrollo y aceptación de software externo.
Centro digital	Alto	<ul style="list-style-type: none"> <li>• Usar nuevas metodologías de desarrollo definidas.</li> <li>• Generar planes de acción para reducir y remediar la deuda técnica.</li> <li>• Asignación de equipos necesarios para la implementación.</li> </ul>
Área de abastecimiento del Banco	Bajo	Coordinar el cumplimiento de la política de adquisición o contratación del software.
Oficina de Transformación	Bajo	Coordinar y priorizar actividades para el cumplimiento de los principios normados.
Áreas del Banco que realicen desarrollo de software	Alto	Cambio en las directrices de la política y procedimientos derivados para el desarrollo de software.
Auditoría interna	Bajo	Validar el cumplimiento de las políticas.
Proveedores de servicios TI	Bajo	Cumplir con las políticas y procedimientos derivados de adquisición de software.
Cliente interno	Alto	<ul style="list-style-type: none"> <li>• Desarrollo y pruebas en conjunto con el usuario final.</li> <li>• Implementación basada en Design thinking</li> </ul>



## 1.5 Stakeholders y expectativas de valor

En la siguiente tabla se enlistan los distintos interesados dentro de la organización y sus respectivas expectativas de valor acerca de esta iniciativa.

Tabla 2  
Stackeholders y expectativas de valor

UNIDAD	CARGO	Comprensión actual	Se requiere la comprensión	Compromiso actual	Compromiso requerido	Expectativas de valor
Vicepresidencia	Vicepresidente de tecnología y operaciones	2	5	3	5	Contar con un área de TI que tenga procesos eficientes apalancando a todas las áreas del banco y así pueda generar valor al negocio.
Gerencia de desarrollo de aplicaciones y mantenimiento	Gerente de desarrollo de aplicaciones y mantenimiento	3	5	3	5	Contar con un área de TI que provea un servicio de calidad tanto a clientes externos como internos. Mejorar el nivel de satisfacción de los servicios proporcionados por TI.
Gerencia Arquitectura de operaciones	Gerente Arquitectura de operaciones	3	5	3	5	Contar con un área de TI que implemente soluciones tecnológicas innovadoras.
Gerencia Infraestructura de TI	Gerente Infraestructura de TI	2	5	3	5	Contar con sistemas de información robustos y confiables.
Servicios generales	Asistente de servicios generales	2	3	2	3	Garantizar la correcta toma de decisiones por medio de procesos de TI estandarizados y confiables.
Departamento financiero	Gerente administrativo financiero	2	4	2	4	Cumplir con presupuestos establecidos. Implementar sistemas que incrementen la productividad y permitan trabajar con grandes volúmenes de datos.

## 1.6 Marcos de referencia

Para el desarrollo de software, se utilizará el marco de trabajo Scrum, mientras que, para la implementación de los procesos de pruebas de la nueva área de Control de calidad de software, se utilizará como marco de referencia TMMI.

Tabla 3  
Marco de referencia Scrum

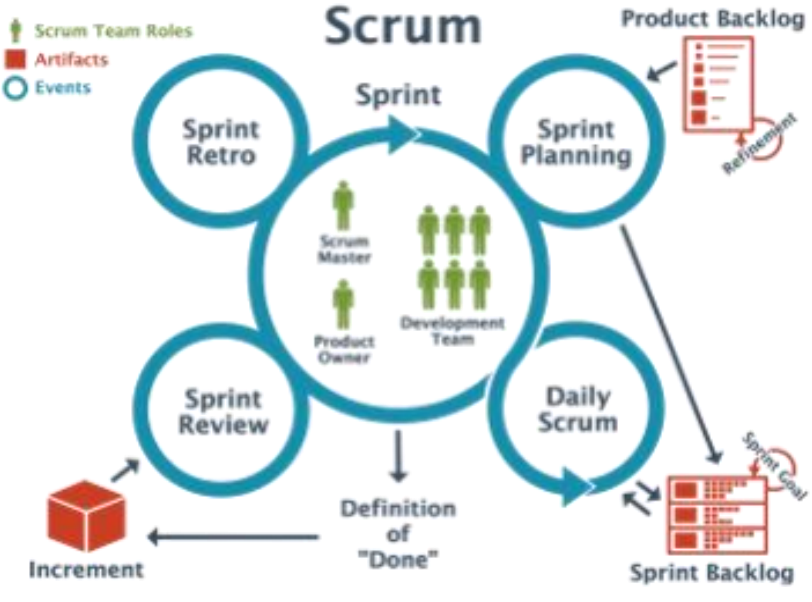
<b>ÁREA</b>	Desarrollo de aplicaciones y mantenimiento
<b>DESCRIPCIÓN</b>	<p><b>Scrum.</b> Es un marco de trabajo utilizado para desarrollar productos complejos. La base de Scrum es el pensamiento Lean y el control empírico de procesos, es decir, la transparencia, inspección y adaptación. Scrum es un conjunto de buenas prácticas y reglas que proveen la estructura necesaria para ayudar al equipo a reducir la complejidad técnica, comercial e interpersonal del desarrollo del software en su conjunto y que, también capacita a equipos auto organizados para alcanzar sus objetivos. El enfoque de Scrum para el desarrollo de productos es iterativo e incremental con múltiples bucles de retroalimentación.</p>
<b>REFERENTE</b>	 <p>El diagrama ilustra el ciclo de Scrum. En el centro se encuentran los roles: Scrum Master, Product Owner y el Development Team. El ciclo comienza con el Sprint Planning, donde se seleccionan ítems del Product Backlog para el Sprint. Durante el Sprint, se realizan Daily Scrums. Al finalizar el Sprint, se realiza el Sprint Review y se crea un Incremento. El ciclo se cierra con el Sprint Retrospective, que alimenta de nuevo al Sprint Planning. Los artefactos clave son el Product Backlog, el Sprint Backlog y el Incremento. Los eventos son el Sprint, el Daily Scrum y el Sprint Retrospective.</p>

Tabla 4  
Marco de referencia TMMI

<b>ÁREA</b>	Área de Control de calidad de Desarrollo de aplicaciones y mantenimiento
<b>DESCRIPCIÓN</b>	TMMi es un marco de referencia para definir o mejorar procesos de prueba de manera continua. Cubre todas las actividades del ciclo de vida de desarrollo de software, desde la planificación, construcción y evaluación de soluciones tecnológicas. (iSQI, 2021)
<b>REFERENTE</b>	<p><b>Nivel 5: Optimización</b></p> <ul style="list-style-type: none"> <li>• Optimización del proceso de pruebas</li> <li>• Control de calidad</li> <li>• Prevención de defectos</li> </ul> <p><b>Nivel 4: Gestión y Medición</b></p> <ul style="list-style-type: none"> <li>• Programa para la medición de pruebas</li> <li>• Evaluación de la calidad del software</li> <li>• Vista preliminar avanzada grupal</li> </ul> <p><b>Nivel 3: Definido</b></p> <ul style="list-style-type: none"> <li>• Organización de pruebas</li> <li>• Programa de entrenamiento de pruebas</li> <li>• Ciclo de vida e integración de pruebas</li> <li>• Pruebas no-funcionales</li> <li>• Revisiones grupales</li> <li>• Control y monitor</li> <li>• Ciclo de vida</li> </ul> <p><b>Nivel 2: Gestionado</b></p> <ul style="list-style-type: none"> <li>• Técnicas y métodos de pruebas</li> <li>• Planificación de pruebas</li> <li>• Metas y pólizas de pruebas</li> <li>• Ambiente de Pruebas</li> <li>• Póliza y planificación</li> <li>• Monitor, control, diseño, ejecución</li> </ul> <p><b>Nivel 1: Inicio</b></p> <ul style="list-style-type: none"> <li>• Proceso caótico</li> <li>• Sin entendimiento de coste o calidad</li> </ul> <p><b>TMMi FOUNDATION</b></p>

Tabla 5  
Marco de referencia SFIA

<b>ÁREA</b>	Desarrollo de aplicaciones y mantenimiento
<b>DESCRIPCIÓN</b>	<p><b>SFIA</b> es un marco o modelo de referencia que describe las habilidades y competencias que los profesionales de una organización necesitan tener en roles relacionados a tecnologías de información, transformación digital e ingeniería de software. (SFIA, 2021)</p>
<b>REFERENTE</b>	

### 1.7 Equipo de arquitectura

Para este trabajo se debe tener el patrocinio del Gerente de Desarrollo de Aplicaciones y Mantenimiento, a quien se informará acerca del avance y deberá aprobar o no las decisiones que se requieran.

El equipo de arquitectura está compuesto por el líder de arquitectura, arquitecto de aplicaciones, arquitecto de datos, arquitecto de negocios y arquitecto de tecnología.

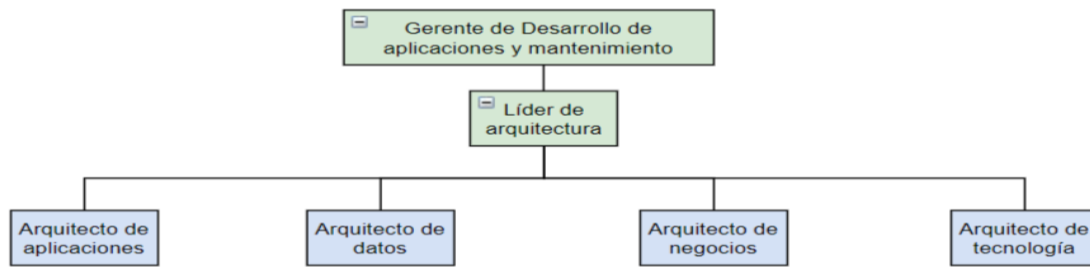


Figura 3 Equipo de arquitectura

### 1.7.1 Roles y responsabilidades

Tabla 6  
Matriz RACI

MATRIZ RACI	Gerente de ADM	Arquitecto líder	Arquitecto de aplicaciones	Arquitecto de datos	Arquitecto de negocios	Arquitecto de tecnología
Reestructurar la visión de arquitectura empresarial	RA	R	R	R	R	R
Reforzar el sistema de gestión del área TI del Banco	RA	A	C	C	C	C
Gestionar la difusión de objetivos, metas, tácticas y decisiones tomadas	R	R	C	C	C	C
Gestionar la mejora y automatización de procesos	R	R	C	C	RA	C
Establecer nuevas y fortalecer las estructuras organizativas que existen	R	A	C	C	C	C
Definir roles y responsabilidades	R	A	R	R	R	R
Definir habilidades y competencia profesional	R	R	R	R	R	R
Gestionar la comunicación de políticas internas y externas	R	RA	C	C	R	C
Reestructurar la infraestructura, servicios y aplicaciones	R	R	C	C	C	RA
Promover y gestionar la mejora continua del área de desarrollo de aplicaciones y mantenimiento.	R	R	R	R	R	R

## 1.8 Catálogo de principios

En esta sección se presentan los principios arquitectónicos que se van a seguir durante todo el proyecto para llevar a cabo la iniciativa:

Tabla 7  
Principio: Soluciones orientadas a generar valor al negocio

Código:	MM01
Principio	Soluciones orientadas a generar valor al negocio
Dominio	Negocio
Definición	Todas las funcionalidades realizadas deben estar alineados a los Objetivos and Key Results (OKRs) y necesidades del negocio.
Motivación	<ul style="list-style-type: none"> <li>• Crear soluciones que generen valor desde el primer entregable.</li> <li>• Generar el mayor valor posible en cada iteración.</li> </ul>
Implicaciones	<ul style="list-style-type: none"> <li>• Modificación de prioridades de desarrollo.</li> <li>• Generación de roadmap estratégico.</li> </ul>

Tabla 8  
Principio: Fallar rápido, fallar barato

Código:	MM02
Principio	Fallar rápido, fallar barato
Dominio	Negocio
Definición	Sacar entregables que generen valor en cada iteración lo más pronto posible con el objetivo de recolectar la mayor retroalimentación del usuario.
Motivación	<ul style="list-style-type: none"> <li>• La retroalimentación generada crea mejores funcionalidades.</li> <li>• Se escucha al usuario y se atiende a sus necesidades.</li> </ul>
Implicaciones	<ul style="list-style-type: none"> <li>• Se genera soluciones en conjunto con el usuario.</li> <li>• Las funcionalidades creadas solucionan necesidades a todo un arquetipo.</li> </ul>

Tabla 9  
Principio: Calidad del software

Código:	MM03
Principio	Calidad del software
Dominio	Tecnología
Definición	Entregar soluciones y servicios de alta calidad a clientes internos y externos de la organización
Motivación	El valor percibido por los usuarios del producto o servicio depende de la calidad de estos.
Implicaciones	<ul style="list-style-type: none"> <li>• Establecer procesos automáticos que garanticen la calidad de software.</li> <li>• Crear políticas y restricciones de despliegue a producción.</li> </ul>

Tabla 10  
Principio: Arquitectura escalable basada en microservicios

Código:	MM04
Principio	Arquitectura escalable basada en microservicios
Dominio	Aplicación
Definición	Arquitectura creada pensando en un alto crecimiento y fácil mantenibilidad.
Motivación	Crear un plan estratégico más eficiente que no haga perder tiempo en escalabilidad automática.
Implicaciones	<ul style="list-style-type: none"> <li>• Aprobación de RRHH para contratación de expertos.</li> <li>• Contar con un presupuesto anual para tecnología.</li> <li>• Implementar cloud computing para concentrarse en el producto.</li> </ul>

## 2. Visionamiento arquitectónico

En este capítulo se analizarán 3 referentes, que han sido elegidos debido a la semejanza que tienen con el concern, en base a dicho análisis se obtendrá el resultado del visionamiento, análisis de brechas y se revisarán ciertos requerimientos que se deben seguir para determinar la arquitectura objetivo en alto nivel.

## 2.1 Requerimientos de alto nivel

En base al análisis realizado en el Modelo de Motivación de Negocio, se identifica que se requiere las siguientes especificaciones:

- Reestructurar los equipos de desarrollo de aplicaciones, estableciendo los roles mencionados en el marco de trabajo Scrum: Product owner, Scrum master y equipo de desarrollo (programadores, diseñadores web, testers, arquitectos, entre otros).
- Realizar capacitaciones a las personas de acuerdo a sus perfiles para mejorar el conocimiento tanto técnico como de gestión de proyectos de desarrollo.
- Utilizar herramientas que contribuyan a la transparencia del trabajo del equipo y la gestión del proyecto.
- Capacitar a los equipos de desarrollo sobre el marco de trabajo Scrum, metodologías ágiles y herramientas que se utilizarán para el desarrollo de software.
- Crear un área de Control de calidad en la fábrica de software, que implemente un proceso ágil de pruebas, estándares de calidad y procesos de pruebas automatizadas.
- Elaborar capacitaciones sobre buenas prácticas de programación para dar mantenimiento a las aplicaciones actuales que tienen deuda técnica y para que en los nuevos desarrollos el código sea limpio, fácil de mantener, sin errores y con pruebas unitarias, evitando así la deuda técnica (alto porcentaje de código con errores, ausencia de buenas prácticas).
- Implementar procesos que, apoyados en herramientas, permitan realizar versionamientos concurrentes y sin errores.
- Utilizar la metodología Design Thinking para mejorar la experiencia del usuario, al utilizar las aplicaciones que se están desarrollando.



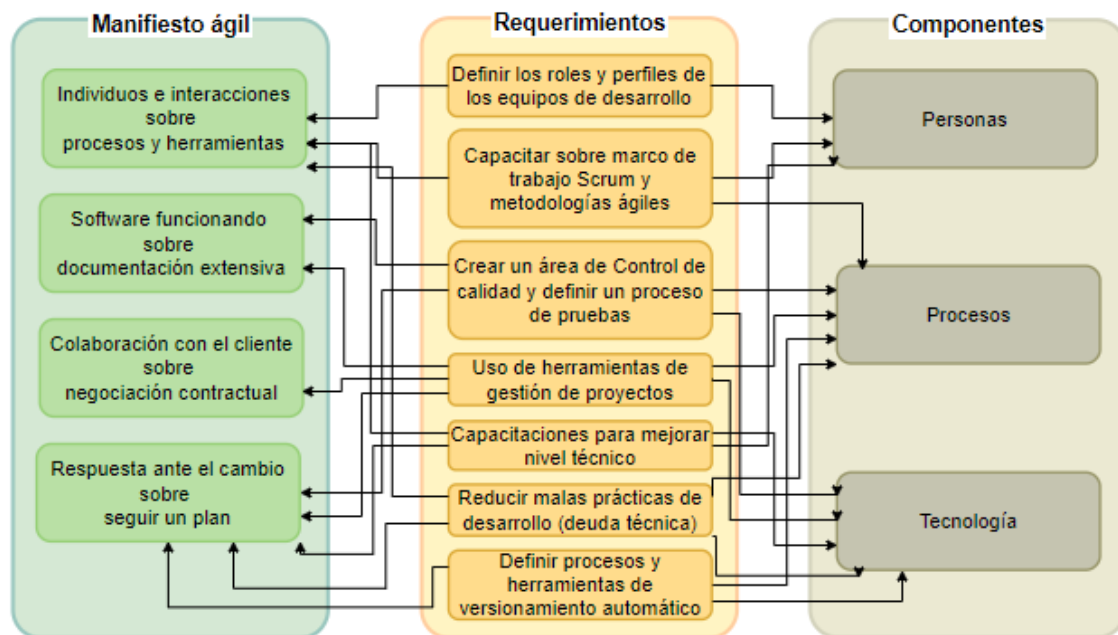


Figura 4 Diseño conceptual de requerimientos de alto nivel

## 2.2 Visionamiento y escenarios de la solución

Las fábricas de software giran alrededor de entornos que cambian constantemente durante todo el ciclo de desarrollo, muchas de ellas han tenido que implementar procesos más ágiles para poder satisfacer a las cambiantes demandas comerciales (Rising & Janoff, 2000).

Aquellas empresas que han adoptado una cultura ágil, es decir, que impulsan a las personas y a los procesos de desarrollo, a adaptarse fácil y rápidamente a los cambios, evidencian una mejora importante en la forma de trabajar. Se requiere de un alto grado de compromiso de cada equipo de trabajo para alcanzar excelentes resultados en el desarrollo de sus soluciones tecnológicas.

### 2.2.1 Empresas referentes

Para el desarrollo del visionamiento, se analizan 3 casos de éxito de empresas que usan como marcos referentes a Scrum, TMMI y SFIA.

### 2.2.1.1 Referente de Scrum

Spotify es una plataforma para escuchar música vía streaming (tecnología que permite oír contenidos desde internet sin tener que descargar previamente los datos al dispositivo, el software utilizado es construido y gestionado utilizando metodologías ágiles.

Entre las prácticas ágiles que se llevan a cabo y que con ellas han logrado tener muchos beneficios, son:

- La empresa cuenta con 250 personas ubicadas en 3 países, los han dividido en 30 pequeños equipos llamados escuadrones, cada uno de ellos funcionando como un startup independiente, centrándose así en una función específica.
- Trabajan por iteraciones, desarrollando productos mínimos viables (MVP), es decir, entregando con frecuencia pequeñas versiones que aporten valor al negocio.
- Cada escuadrón cuenta con su propio espacio de trabajo y un organigrama que, a la hora de gestionar proyectos, se considera como una estructura plana.
- En cada equipo se encuentra un Product owner, que además de proporcionar al equipo Scrum los criterios de aceptación de cada funcionalidad, se hace cargo de las relaciones con los otros equipos.
- Los integrantes de los escuadrones son lo suficientemente autónomos como para contactarse directamente con los *stakeholders*.
- Existen las llamadas tribus, que son conformaciones de escuadrones, en las que no debe haber más de 100 personas. Son equipos que tienen algo en común o tienen cierta dependencia. Realizan reuniones semanales con el fin de compartir conocimiento o eliminar obstáculos.
- Para poder compartir experiencias y buenas prácticas, se tienen los “Capítulos” y “Gremios”, conformados por más de 100 personas que se reúnen mensualmente para tener mayor colaboración y comunicación.

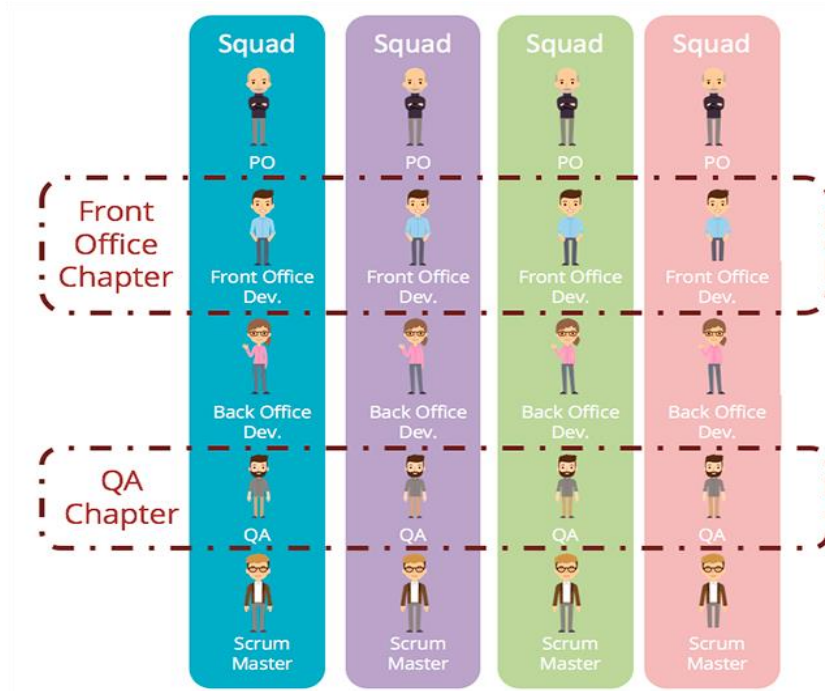


Figura 5 Cultura Ágil en Spotify (Deloitte, 2020)

### 2.2.1.2 Referente de TMMI

Indra una compañía internacional de consultoría y tecnología, ha implantado TMMi (Test Maturity Model integration) en sus centros de producción, lo que le beneficia de la siguiente manera:

- Mejora el proceso y procedimientos de pruebas de software para asegurar la calidad de sus aplicaciones.
- Logran una mayor satisfacción en el cliente gracias a las soluciones tecnológicas entregadas con un alto estándar de calidad.
- Permite compartir entre todos los colaboradores la misma cultura organizacional, metodologías de desarrollo, procesos y herramientas.

Gracias a esta certificación Indra demuestra tener un óptimo proceso de pruebas, logrando una mejor estabilidad y confianza en sus productos; destacándola frente a otras empresas por tener esta ventaja competitiva.

Además de ser reconocida como una empresa que implementa desarrollos y mantenimientos de software con los procedimientos y estándares más exigentes

a nivel internacional, se posiciona como la pionera en certificarse en el modelo TMMi (Indra, 2017)

### 2.2.1.3 Referente de SFIA

Segacy es una consultora chilena que asesora a las empresas en la creación de perfiles de competencias y habilidades laborales necesarias para operar en la industria de TI. Ha implementado el marco referente SFIA (Skills Framework for the Information Age) en su propia organización logrando mejorar el rendimiento de TI generando más valor para la empresa.

Tiene conocimientos sólidos y una gran experiencia en aplicar el modelo SFIA en cuanto a definir las habilidades en conjunto con la mejora de procesos y define un marco de acción en el que habla de identificar 3 roles: gobierno: que es quien establece estándares, universidad: es quien prepara a las personas y empresa: es en donde la gente se desarrolla profesionalmente. (SFIA | Chile, s. f.)



Figura 6 Marco de acción Segacy

### 2.2.2 Escenario de fábrica de software ideal

Los roles, procesos y artefactos necesarios en una fábrica de software ideal, deben estar estructurados de la siguiente forma:

### Roles:

- Cada equipo de trabajo debería tener los siguientes roles: Product owner, Scrum master, equipo Scrum (programadores, testers, diseñadores, entre otros, todo depende del proyecto).
- Cada persona debe contar con una certificación en Fundamentos de Scrum.

### Procesos que se llevarían a cabo durante todo el ciclo de desarrollo:

- Inception (también conocido como sprint cero)
- Planificación del sprint (Sprint planning – ceremonia de Scrum).
- Análisis y revisión diaria (Daily – ceremonia de Scrum)
- Refinamiento de historias de usuario.
- Revisión del incremento (Sprint review – ceremonia de Scrum).
- Reunión de retrospectiva (Sprint retrospective – ceremonia de Scrum).

### Artefactos utilizados en Scrum:

- Product backlog: listado de todas las historias de usuario del producto.
- Sprint backlog: listado de historias de usuario que se trabajarán durante el sprint en curso.
- Incremento: es una nueva versión del producto que ofrece valor al negocio.

## The Agile: Scrum Framework

Información de los ejecutivos,  
el equipo, los implicados,  
los clientes, los usuarios, etc.

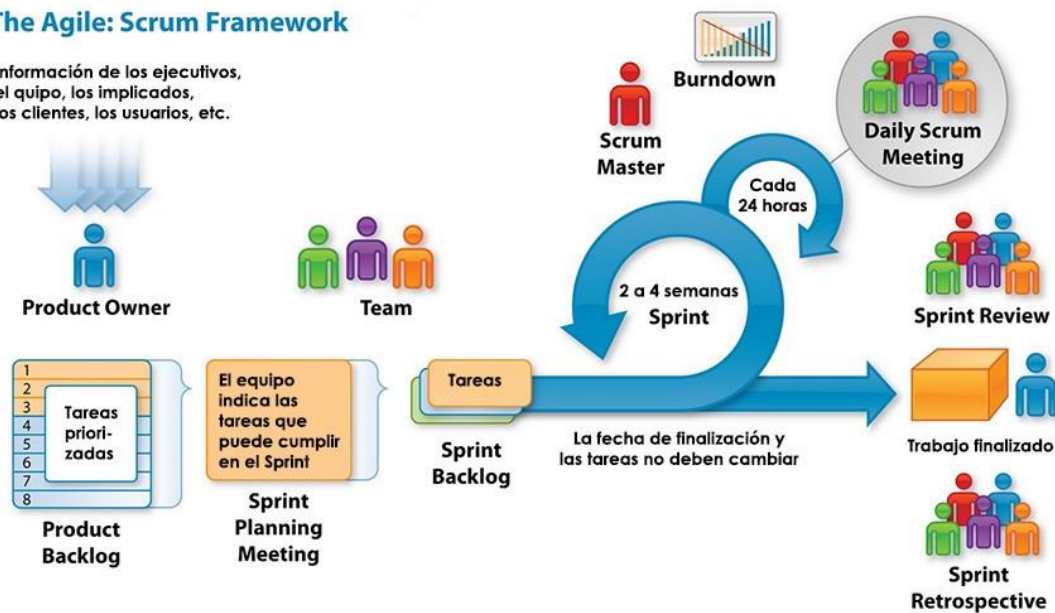


Figura 7 Flujo de trabajo Scrum (Roles, artefactos y eventos) (Pablo Marichal, 2016)

### 2.2.3 Escenario de fábrica de software objetivo

A continuación, se describe el escenario de la fábrica de software objetivo:

- Debe contener todo lo descrito en el escenario de fábrica de software ideal.
- Los integrantes del equipo deben tener conocimiento sobre TMMI y Scrum.
- Los testers del equipo de Control de calidad debe estar especializado en TMMI y tener la certificación emitida por el ISTQB (Certified Tester Foundation Level).

TMMI se enfoca en el proceso de control de calidad de software y cuenta con 5 niveles de madurez. Al utilizarlo en los proyectos de software, produce un impacto positivo en la calidad de las aplicaciones y en el esfuerzo de las pruebas.

La integración de todo el modelo de prueba TMMI implica tener mucha documentación, pero al combinarlo con Scrum se reduce dicha documentación y mejora todo el proceso de prueba. Al final se obtendrá el porcentaje de

cobertura de TMMI y se determinarán las áreas de mejora. (Unudulmaz & Kalıpsız, 2020)

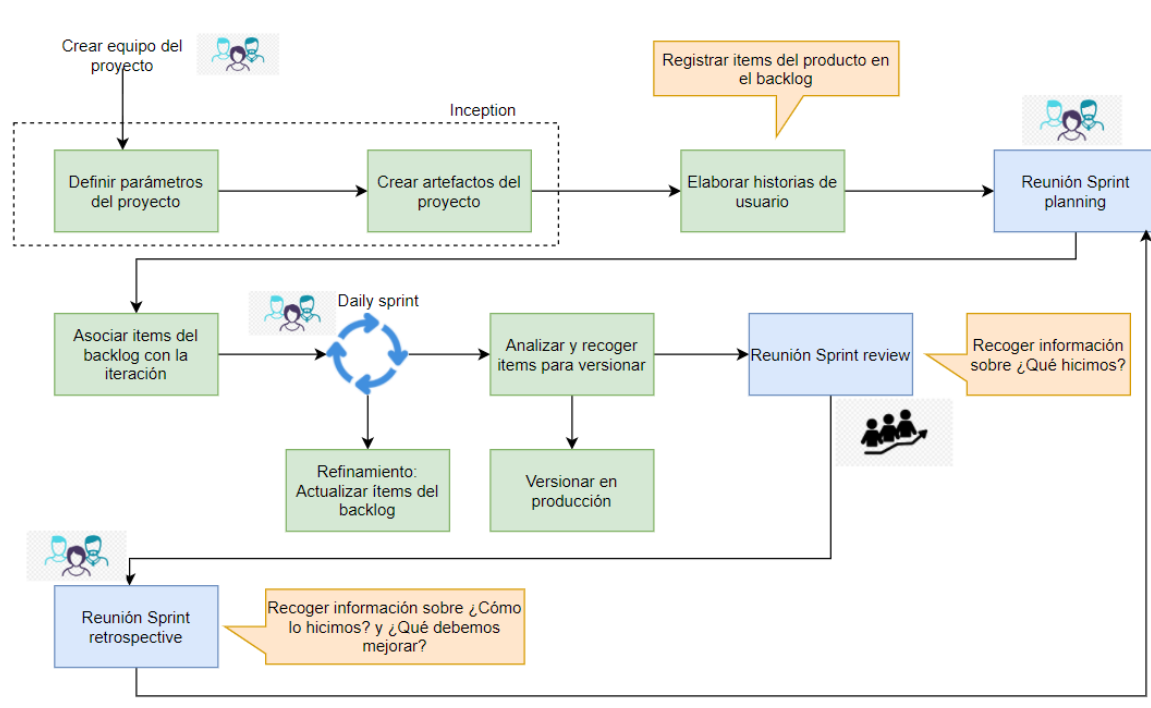


Figura 8 Escenario de fábrica objetivo

## 2.2.4 Resultado de visionamiento

Debido a que las soluciones tecnológicas implementadas por el área de desarrollo tienen un gran impacto en las áreas de negocio del Banco, se determina que se debe tener la visión de cumplir con la fábrica de software objetivo, para proporcionar versiones recurrentes de las aplicaciones en tiempos cortos y con estándares altos de calidad.

## 2.3 Análisis de brechas

Una vez identificados los componentes a fortalecer, se realizará un análisis de brechas para determinar cómo se encuentra la arquitectura actual en relación a la arquitectura objetivo.

Tabla 11  
Análisis de brechas de los referentes

	BRECHAS		BASE	META	REFERENTE
Personas SFIA	Planificar	Diseñar roles y estructura. Planificar la fuerza de trabajo.	2	4	5
	Adquirir	Reclutar habilidades correctas	3	5	5
	Desplegar	Asignar trabajo por capacidad	1	4	5
	Evaluar	Evaluar competencias y desempeño	2	4	5
	Analizar	Identificar brechas y oportunidades	1	4	5
	Desarrollar	Proporcionar caminos de carreras. Construir capacidad y rendimiento.	1	4	5
	Recompensar	Compensar y premiar	1	3	5
Procesos TMIMI / Scrum (Metodologías ágiles)	Política de pruebas	PP	0	3	5
	Estrategia de pruebas	EP	0	3	5
	Indicadores de desempeño de pruebas	IDP	0	3	5
	Alcance de pruebas	AP	0	3	5
	Estimación de pruebas	EP	0	3	5
	Gestión de proyectos ágiles	GPA	0	4	5
	Proceso ágil de desarrollo de software	AGDS	2	5	5
	Proceso automático de versionamiento	PAV	1	4	5
Tecnología Scrum	Metodologías ágiles	Scrum	0	4	5

Nivel 0: No existe implementación.

Nivel 1: El nivel de implementación es muy incipiente o incompleto.

Nivel 2: La documentación es básica, no se dispone de documentación formal.

Nivel 3: Ya lleva implementado un tiempo superior a 6 meses, se dispone de documentación formal.

Nivel 4: La implementación es guiada por buenas prácticas y adopta tecnologías referentes. Su desempeño es bueno.

Nivel 5: La implementación ha madurado sustancialmente, es un referente. Se evidencia un proceso gradual de innovaciones.



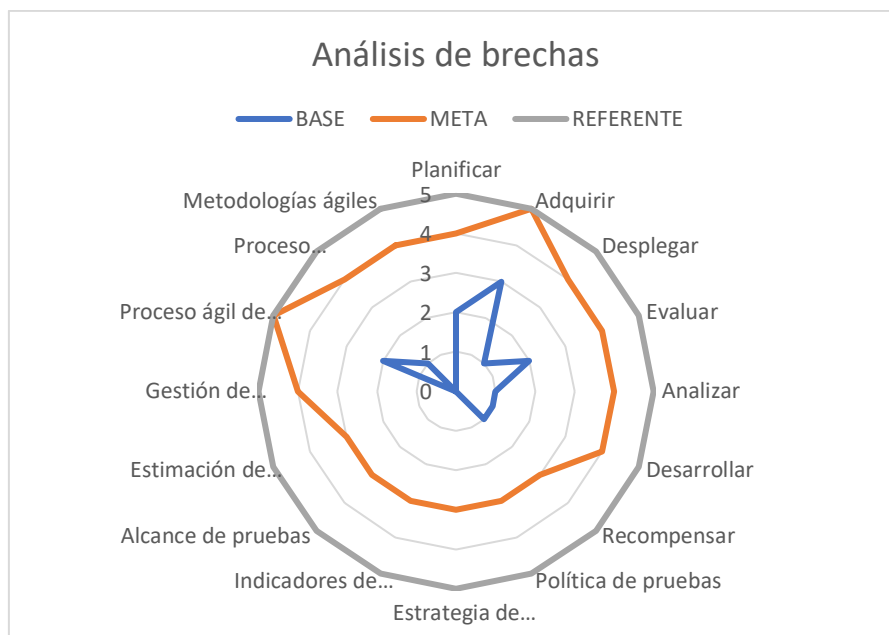


Figura 9 Gráfico resultante del análisis de brechas

En base a este análisis se evidencia que la arquitectura actual es incipiente, ya que no se utilizan procesos y prácticas ágiles en el desarrollo de software, adicional no se tiene una estructura definida de los roles y competencias de las personas.

Esencial en el desarrollo de soluciones tecnológicas es además contar con procesos ágiles de pruebas, para entregar software de alta calidad.

## 2.4 Definición de arquitectura objetivo

La solución del concern se basará en 3 marcos ágiles que son: Scrum, TMMI y SFIA, con el propósito de fortalecer la fábrica de software in house del Banco y llegar al nivel de madurez deseado.

### 2.4.1 Target de arquitectura de negocio

Para cumplir con la arquitectura de negocio objetivo, debe haber un apalancamiento tecnológico que permita cumplir con la visión de la entidad financiera: “Ser el Banco más grande y el mejor”, y pueda volverse más competitiva y tener oportunidad de innovar.

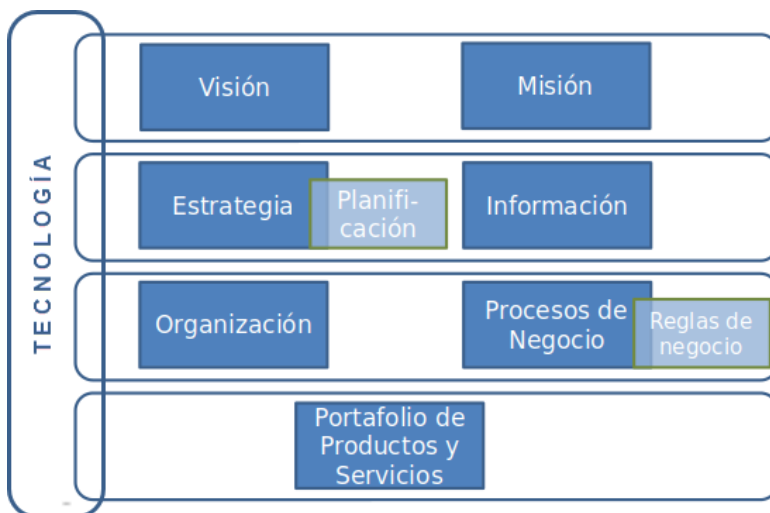


Figura 10 Arquitectura de negocio adaptado de: (Agesic, s. f.)

### 2.4.2 Target de arquitectura de datos

Para llegar a la arquitectura de datos objetivo, se deben definir procesos estandarizados de gestión de información para que el Banco pueda adquirir, clasificar, distribuir y evaluar la información que proveen o reciben los procesos.

Deben determinarse indicadores que permitan evidenciar si se han alcanzado los objetivos esperados por la fábrica de desarrollo, los mismos que estarán alineadas a las metas del Banco.

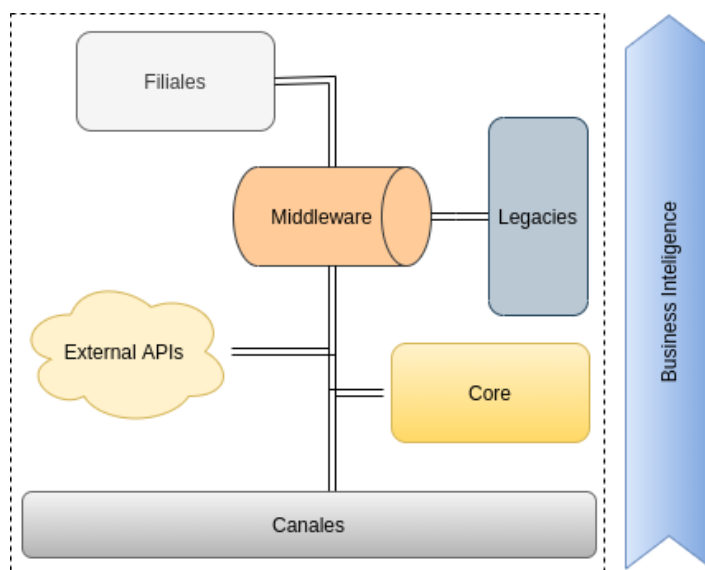


Figura 11 Arquitectura de datos objetivo

### 2.4.3 Target de arquitectura de aplicaciones

Con el objetivo de ser la mejor institución financiera del país es necesario brindar a los clientes aplicaciones estables, rápidas y con la mayor disponibilidad posible, además de entregar innovación constantemente al usuario cumpliendo los mismos parámetros de calidad.

Para llegar a la arquitectura de aplicaciones objetivo, se debe lograr disponer de aplicaciones totalmente integradas entre ellas con el objetivo de lograr una cohesión de los procesos de desarrollo de software además de planificar el desarrollo de nuevas aplicaciones centrándose en las necesidades del usuario y del negocio.

Para cada fase del ciclo de software existen varias aplicaciones que ayudan a mejorar la eficiencia del proceso y a su vez se integran entre sí de una forma simple, con esto se busca implementar los insumos necesarios para que el tiempo de entrega de las aplicaciones sea lo menor posible y que el usuario pueda las aplicaciones rápidamente.

Se debe entender el ambiente en el cual se desplegarán las aplicaciones e identificar los usuarios que las utilizarán y cómo lo harán.

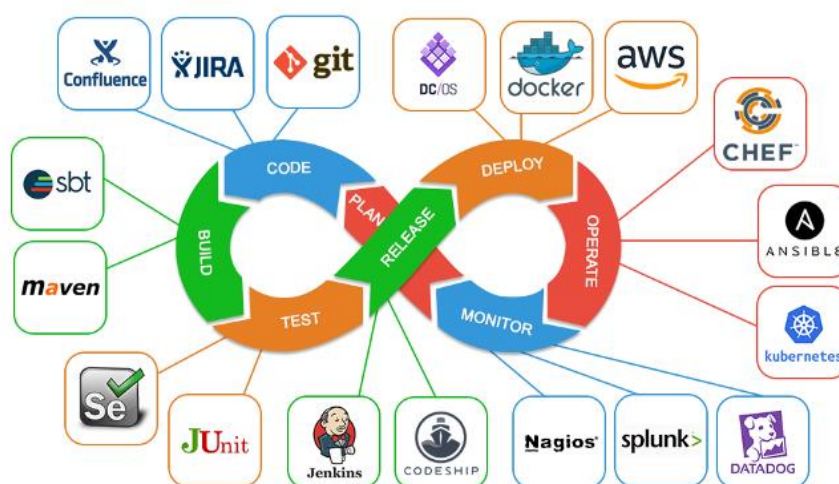


Figura 12 Herramientas utilizadas en DevOps adaptado de: (Singh, 2019)

#### 2.4.4 Target de arquitectura de infraestructura base

La institución financiera cuenta con varios niveles de alimentación de información, los cuales deben estar centralizados y se los debe organizar correctamente.

A continuación, se describen los principales medios de obtención de datos dentro de la institución:

**Canales:** agencias, canales digitales, BPM, entre otros que consumen y proporcionan datos al sistema bancario.

**External APIs:** redes interconectadas que permiten la conexión con agentes externos de regulación, control y gestión, por ejemplo:

- La Superintendencia de Bancos.
- Control del código swift para identificar al Banco receptor cuando se realiza una transferencia internacional.
- Banred para transferencias interbancarias locales.
- Bancos con convenios.
- Alianzas estratégicas con empresas locales y extranjeras.

**Legacies:** programas considerados antiguos y con poco o nulo mantenimiento, entre ellos se encuentran:

- Comercio exterior
- Tesorería
- Garantías
- Canje
- RRHH Adam system
- Activos fijos, cuentas por cobrar y pagar
- Contabilidad

**Business intelligence:** consumo transversal de datos con el objetivo de obtener información valiosa para el negocio, además se necesita otorgar una alta experiencia de usuario, reduciendo el tiempo en agencias.

Los principales modelos de datos existentes son:

- Data Warehouse de transacciones y clientes.
- Modelos de análisis en conjunto con datos proporcionados por SIB, SRI, BCE.
- CRM para otorgar la mayor información necesaria para el análisis crediticio de un cliente.
- SAS para emprendedores.

**Filiales:** nacionales y extranjeras que complementan la información interna de la institución.

**Middleware:** encargado de la orquestación del origen y destino de los datos.

**Core:** centraliza los datos, además realiza un análisis de comportamiento de débito y crédito en cada transacción.

El objetivo es tener datos de calidad, mejorar la integración, velocidad y escalabilidad de los sistemas nuevos y ya existentes.

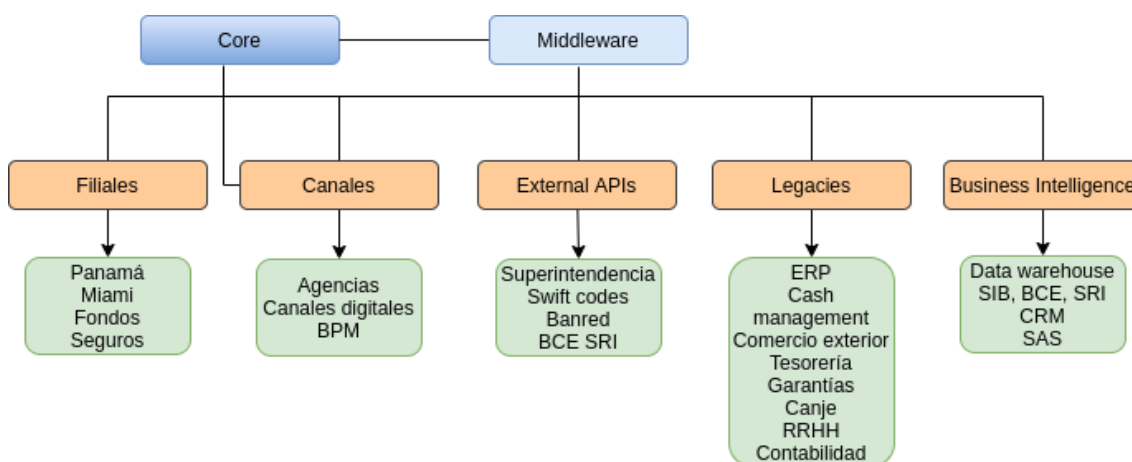


Figura 13 Arquitectura de infraestructura base

### **3. Arquitectura de Negocios**

En esta sección se describe y analiza el funcionamiento de los procesos actuales, a las personas y a la organización con el fin de encontrar las brechas existentes y proponer iniciativas para mejorar dicho funcionamiento.

#### **3.1 Arquitectura base de negocio**

Actualmente en la fábrica de software del Banco se utiliza una metodología cascada, es decir, los proyectos pasan por todas las fases de desarrollo de una manera lineal, debiendo finalizar una fase para continuar con la siguiente, las fases son: Inception, Conceptualización, Ejecución, Pruebas y Certificación.

##### **3.1.1 Fases**

Las fases que intervienen actualmente en el proceso de desarrollo de software son las siguientes:

###### **3.1.1.1 Inception**

- Se coordina un Kick Off de la iniciativa.
- Se coordinan los talleres de diseño de la solución donde participan el Business Analyst, expertos funcionales requeridos y Arquitectos de software.
- El experto de negocio del área que corresponda, junto con el Equipo de proyecto se encargan del levantamiento y refinamiento de los requerimientos funcionales y no funcionales.
- El Business analyst obtiene las estimaciones y formalización de las necesidades del cliente.
- El Arquitecto de la solución levanta el diseño arquitectónico de alto nivel estudiando los requerimientos no funcionales del sistema (usabilidad, interfaz de usuario, rendimiento, confiabilidad, almacenamiento, prototipos) y asegurando su entendimiento y acordando con el cliente objetivos/restricciones de diseño, además, en el diseño de alto nivel se

entenderán las características y capacidades de los ambientes de software y hardware requeridos.

### **3.1.1.2 Conceptualización**

- Se coordinan los talleres de diseño basados en el estudio del problema actual, procesos de negocio, sistemas existentes y discusiones con los usuarios.
- En caso de realizarse modificaciones sobre soluciones existentes deben participar los proveedores relacionados.
- Refinar las necesidades funcionales.
- El Equipo de desarrollo levantará el diseño arquitectónico de bajo nivel estudiando los requerimientos no funcionales del sistema (usabilidad, interfaz de usuario, rendimiento, confiabilidad, almacenamiento, prototipos) y asegurando su entendimiento y acordando con el cliente objetivos y restricciones de diseño.
- El Líder de proyecto se asegurará de:
- Velar por el cumplimiento de que la planificación acordada para esta fase se cumpla dentro de los plazos establecidos.
- Gestionar la asignación del Equipo de trabajo necesario.
- Asegurar que los objetivos esperados de la solución estén claramente identificados.
- En caso de que no se cumpla lo indicado deberá informar a los interesados sobre el problema para tomar las acciones correctivas en conjunto con el Equipo de Proyecto.
- Cuidar de que el alcance de la solución definido se enmarque dentro de lo definido en el Project charter.
- Cuando el proyecto corresponde a una solución que requiere de contratación de un tercero, ejecuta en paralelo el proceso de contratación externa.

### 3.1.1.3 Ejecución (Construcción)

El Equipo de Desarrollo será el encargado de:

- Gestionar la creación de ambientes y entornos a utilizar.
- Preparar los datos de prueba necesarios.
- Basarse en el Project charter definido para la implementación de la solución.
- Generar los repositorios de código y ramas donde se almacenará el código desarrollado y se mantendrá su respectivo versionamiento.
- Definir los estándares para el desarrollo de la codificación y basarse en ellos al momento de programar.
- Revisión de código por pares, a la cual llaman pruebas back to back.
- Implementación de requerimientos funcionales y no funcionales.
- Generación de pruebas unitarias y de integración.
- Validaciones de seguridad como usuarios, permisos y reglas de entrada y salida de firewall.

El Líder del Proyecto será el encargado de:

- Vela por que se cumpla el cronograma establecido, tanto el tiempo como el alcance definido.
- Gestiona la asignación de los integrantes del equipo requeridos para esta fase.
- Reporta los avances de la implementación a los interesados.
- Gestiona los impedimentos y necesidades del equipo.
- Elabora las estimaciones en tiempo y costo de los posibles controles de cambios.
- Asigna las tareas al Equipo de desarrollo.
- Genera órdenes de compra tanto de hardware como software.
- Responsable por el cumplimiento del presupuesto aprobado del proyecto.
- Gestiona los riesgos del proyecto.
- Gestiona las comunicaciones del proyecto con los interesados.



- Realiza el seguimiento del avance de la construcción.
- Solventa dudas funcionales que pueda tener el equipo de desarrollo.
- Evalúa el impacto (costo–beneficio) de los cambios que pueda presentar el proyecto.

#### **3.1.1.4 Pruebas**

- El equipo de desarrollo realizará:
  - Despliegue unificado del código.
  - Pruebas de integración.
  - Pruebas de conectividad.
  - Pruebas funcionales y no funcionales.

#### **3.1.1.5 Certificación**

- Los expertos funcionales realizarán:
  - Ejecución de pruebas de los requerimientos solicitados.
  - Pruebas de aceptación de usuario (si aplican).
  - Acompañamiento y soporte técnico a usuario.
  - Pruebas de Rendimiento (si aplican).
  - Pruebas de Seguridad.
  - Al aceptar que todo es correcto, elabora Plan de despliegue.
- El Líder del Proyecto se encargará de:
  - Obtener autorizaciones para que el equipo de desarrollo finalmente despliegue en producción.

Del análisis realizado en cada etapa de desarrollo de software, se evidencia que existen ciertas desventajas ante la implementación de proyectos cambiantes, ya que apenas en la fase final se puede probar el producto o servicio y resulta difícil introducir cambios cuando el proyecto ya se encuentra avanzado.

Con respecto a las personas y organización, la fábrica de software se conforma por un equipo jerárquico que trabaja en cascada con roles bien definidos. El Gerente de Desarrollo de Aplicaciones y Mantenimiento está a la cabeza del

área, bajo su rol se encuentran el Líder de proyecto, Arquitectos de software, Business analyst y Desarrolladores en ese orden jerárquico, además de un Experto de negocio externo al área de desarrollo, que da la pauta inicial para la implementación de las soluciones.

### 3.1.2 Roles

A continuación, se encuentra el detalle de los roles mencionados:

- **Gerente de Desarrollo de aplicaciones y Mantenimiento** (1 persona)

Es aquel responsable de apoyar constantemente a la organización a través de la toma de decisiones clave respecto a proyectos de tecnología enfocados en los objetivos de la institución.

- **Líder de proyecto** (1 persona)

Experto en materia de gestión de proyectos que controla todas las fases del mismo. Da seguimiento a las fechas comprometidas, verifica si el proyecto cumple con la planificación inicial. Su rol es importante y crítico en el tema de liderazgo hacia el equipo teniendo el propósito de cumplir con los objetivos del proyecto.

- **Business Analyst** (1 persona)

Realiza el análisis de factibilidad de los proyectos de software y realiza la propuesta de las soluciones al Gerente del área para su respectiva aprobación.

- **Arquitecto de Software** (2 personas)

Encargado de buscar soluciones tecnológicas consistentes, escalables y de alto desempeño. Debe adaptar las soluciones a los objetivos de la empresa con componentes reutilizables que satisfagan las necesidades comerciales cambiantes. Da seguimiento al desarrollo para comprobar la correcta implementación de la arquitectura propuesta.

- **Desarrolladores** (4 personas)

Desarrolla la arquitectura propuesta cumpliendo altos estándares de calidad y los tiempos establecidos. Ejecuta el plan estratégico y se ajusta a la visión técnica. Participa en todas las fases de la implementación de la solución. Al no existir un área de control de calidad dedicada, el equipo de desarrollo realiza pruebas back to back para garantizar un correcto desarrollo de funcionalidades.



Figura 14 Arquitectura base de negocio

De lo que se puede notar, no existe un rol de Tester para que pueda certificar el correcto funcionamiento del producto, en este caso son los desarrolladores quienes hacen de juez y parte, corriendo así el riesgo de tener errores en producción, además se tiene una jerarquía que muchas veces no da paso a la creatividad del equipo y más bien genera un entorno burocrático.

### 3.2 Arquitectura objetivo de negocio

En vista de que en la fábrica de software también se llevan a cabo proyectos en los que hay mucha incertidumbre, son complejos e impredecibles, es recomendable guiarse en un marco de trabajo ágil. Scrum se adapta a estas necesidades debido a que permite realizar entregas parciales del producto de forma iterativa e incremental y los cambios son bienvenidos en cualquier momento del desarrollo (Sutherland, 2020).

### **3.2.1 Fases**

El proceso objetivo consiste en las siguientes fases:

#### **3.2.1.1 Onboarding**

Es un proceso de incorporación de los integrantes de la célula, en el que el Gerente de Desarrollo de aplicaciones y Mantenimiento da ciertos lineamientos del proyecto y luego se imparten talleres de capacitaciones técnicas y metodologías a las nuevas células, entre ellas se tienen las siguientes:

- Técnicas: En qué consiste la programación en pares, qué es DevOps y herramientas a utilizar, proceso ágil de control de calidad, Creación de pruebas unitarias y de integración.
- Metodológicas:
  - Scrum: eventos y artefactos, roles y responsabilidades.
  - Kanban: creación de tablero de trabajo, tipos de incidencias, estado de incidencias, métricas.

#### **3.2.1.2 Ideación y diseño**

Este proceso da como resultado una pila de tareas y funcionalidades que serán parte del producto.

En esta etapa se impartirán los siguientes talleres:

- Identificación de arquetipos: consiste en definir perfiles de personas en los que se representan comportamientos, intereses, motivaciones, entre otros patrones, con el fin de identificar clientes.
- Creación de Customer journey map: se utiliza este artefacto con el propósito de identificar los dolores de los clientes al pasar por ciertos procesos y así determinar las oportunidades de mejora.
- Identificación del problema: para esto se realizan entrevistas tanto a proveedores, clientes, interesados, entre otros dependiendo del proyecto,

para poder determinar los problemas que tienen las distintas partes involucradas.

- Visión y propósito del proyecto: en base a la información obtenida se crea tanto la visión como el propósito del proyecto para alinear los objetivos del equipo.
- Creación de User story mapping: es una co creación entre todo el equipo de las tareas y funcionalidades del producto (backlog), basándose en las fechas de cumplimiento de los objetivos, se obtiene el roadmap del proyecto que se encontrará siempre sujeto a cambios.
- Prototipado: Se diseñan y proponen los prototipos del producto.

A continuación, se muestra las actividades que realiza cada rol durante esta fase:

Tabla 12  
Roles con sus respectivas actividades durante la Fase de Ideación y diseño

Rol	Actividades
UX y Scrum master	<ul style="list-style-type: none"> <li>•Facilitación de talleres.</li> <li>•Impulsan a la co creación de los distintos artefactos.</li> </ul>
Expertos de negocio	<ul style="list-style-type: none"> <li>•Entregan información útil para el desarrollo de los artefactos (journey del cliente, user story mapping, OKRs, entre otros) y la creación de la pila de funcionalidades.</li> <li>•Dan feedback al final de cada sprint a las células en las cuales colaboran.</li> </ul>
UI / UX	<ul style="list-style-type: none"> <li>•Diseño de prototipos iniciales.</li> <li>•Elaboración y ejecución de entrevistas a clientes o futuros usuarios.</li> </ul>
Gerente de Desarrollo de Aplicaciones y Mantenimiento	<ul style="list-style-type: none"> <li>•Analiza y complementa la información necesaria para dar inicio al proyecto.</li> <li>•Seguimiento mensual de OKRs de cada célula.</li> </ul>

### 3.2.1.3 Construcción

De aquí en adelante inicia la Construcción basándose en Scrum, el trabajo se realiza en iteraciones entregando un incremento al final de cada iteración.

Dentro de un sprint el trabajo que se realizará es el siguiente:

Tabla 13  
Roles con sus respectivas actividades durante la Fase de Construcción

Rol	Actividades
UI / UX	<ul style="list-style-type: none"> <li>• Diseño de prototipos para futuros sprints.</li> <li>• Constante investigación, por ejemplo: acerca de la satisfacción de los clientes con las nuevas funcionalidades.</li> </ul>
Arquitecto de software	<ul style="list-style-type: none"> <li>• Define lineamientos de arquitectura del proyecto.</li> <li>• Lidera la parte técnica.</li> </ul>
Equipo de desarrolladores	<ul style="list-style-type: none"> <li>• Desarrollo de funcionalidades.</li> <li>• Pruebas unitarias y de integración.</li> </ul>
Equipo de control de calidad	<ul style="list-style-type: none"> <li>• Diseño de casos de prueba, documentación de guías de usuario, ejecución de pruebas: funcionales, no funcionales, de rendimiento, automatización, entre otras.</li> <li>• Una vez certificado, es decir, no existen errores, da paso al trabajo del ingeniero DevOps.</li> </ul>
Ingeniero DevOps	<ul style="list-style-type: none"> <li>• Creación de flujos de prueba en plataformas de integración continua.</li> <li>• Despliegue del nuevo desarrollo en producción.</li> </ul>

Cuando se ejecute el sprint final se desplegará el último incremento, teniendo así en producción todo el producto terminado.

### 3.2.1.4 Mantenimiento

La fase final y posterior al último despliegue del producto se llama Mantenimiento, en la que se brindará soporte cuando se deba realizar pequeños

cambios, permitiendo que el producto cumpla totalmente con la satisfacción del cliente.

### 3.2.2 Eventos utilizados durante cada sprint

Los eventos dentro de este marco de trabajo son:

- **Sprint:** es la parte principal del marco de trabajo, el equipo define su duración fija de un mes o menos, al finalizar el sprint se obtiene un entregable potencialmente usable. Un sprint comienza inmediatamente después de finalizar el anterior.
- **Scrum planning:** se plantea el objetivo del sprint y se planifica junto con todo el equipo las tareas que se trabajarán durante esta iteración y estiman cada una de ellas.
- **Daily scrum:** es una sincronización diaria en la que el equipo inspecciona cuán cerca de cumplir el objetivo están y mencionan si tienen impedimentos para determinar quién podría apoyar.
- **Scrum retrospective:** es una reunión en la que el equipo identifica qué es lo que se ha hecho bien durante el sprint y en qué se debería mejorar, se obtienen accionables y los respetivos responsables, en las próximas retros se da seguimiento.
- **Scrum review:** a esta reunión asisten todos los interesados ya que el equipo presenta el incremento obtenido y recibe feedback para poder mejorar en próximos sprints.
- **Refinamiento:** no es como tal un evento de Scrum, sin embargo, es aconsejable hacerlo para poder afinar las tareas de los próximos sprint, de esta forma en la planning las tareas están lo suficientemente entendibles con todos los criterios de aceptación necesarios.

### 3.2.3 Artefactos utilizados durante cada sprint

Los artefactos utilizados son:

- **Lista de pendientes del producto** (Product backlog): es un listado de los pendientes del producto definidos hasta el momento, el cual se encuentra ordenado y priorizado. El listado no tiene fin, va evolucionando constantemente, inicialmente se dispone de aquellos requisitos que han sido bien entendidos
- **Lista de pendientes del sprint** (Sprint backlog): Es un conjunto de tareas que el equipo ha seleccionado del Product backlog porque consideran que son necesarias para cumplir con el objetivo del sprint.
- **Incremento:** es la suma de todos los elementos trabajados durante el sprint, más los incrementos obtenidos en anteriores sprints. Este incremento debe ser siempre utilizable. (Sutherland, 2020)

La nueva fábrica de desarrollo del Banco propone contar con varias células ágiles o equipos de trabajo multidisciplinarios, cuyos integrantes están alineados a un objetivo común.

Los equipos entregan productos de forma iterativa e incremental, evaluando las oportunidades de mejora a través de eventos de retroalimentación de los incrementos presentados. Las entregas incrementales aseguran que siempre estará disponible una versión potencialmente útil y funcional del producto.

Las células estarán compuestas por los siguientes roles un Product Owner, un Scrum Master y el Equipo de desarrollo compuesto por nueve personas (dos desarrolladores backend, dos desarrolladores frontend, un tester, un diseñador UX y otro UI), a todo este equipo se le llama Scrum team. Dependiendo del proyecto se podrán agregar otros roles al Equipo de desarrollo.

#### **3.2.4 Roles y responsabilidades involucrados en el marco de trabajo:**

- **Product owner** (1 persona)

Es el responsable de la iniciativa y de maximizar el valor del producto o servicio que está implementando el equipo. Es el único responsable de gestionar las



necesidades de los interesados y consolidarlas en una pila del producto, priorizando aquellas que generan más valor al negocio y al producto.

- **Scrum master** (1 persona)

Es el responsable de promover y velar porque se cumpla el marco de trabajo. Es un líder que está al servicio del equipo y colabora con la gestión de impedimentos, además de facilitar el apoyo de las partes interesadas.

- **Equipo de desarrollo** (9 personas)

Conformado por profesionales de tecnología y expertos de negocio, trabaja en conjunto para cumplir con el incremento del producto en cada sprint. Cada célula dependiendo de la necesidad del producto y negocio, estará compuesta por dos backends, dos frontends, una o dos personas de control de calidad, una persona de user experience (UX), una persona para user interface (UI), un ingeniero devops y un arquitecto de software.

- **Backends Developers** (2 personas)

Equipo de ingenieros especializados en el back de la aplicación, enfocándose en patrones de diseño, performance y en cumplir una arquitectura robusta.

- **Frontends Developers** (2 persona)

Desarrolladores especializados en parte visual de la solución, su objetivo principal es cumplir con el diseño propuesto por el UI, creando un proyecto amigable y de fácil uso.

- **Control de Calidad (QA)** (1 persona)

Se especializan en realizar meticulosas pruebas funcionales y de usuario con el objetivo de ser el último filtro de calidad de cara al cliente. Colaboran también en la automatización de pruebas y adaptabilidad al diseño.

- **User Experience (UX)** (1 persona)

Profesional encargado de la experiencia de usuario completa de la solución. Es la persona que propone las soluciones tecnológicas que se adapten a las necesidades de los usuarios, enfocándose en encontrar y resolver sus puntos de dolor. A través del uso de varias técnicas de investigación proporciona valiosa información al equipo y Product owner de cara a mejorar constantemente la solución y en ciertos casos replantear desde el inicio.

- **User Interface (UI)** (1 persona)

En conjunto con el UX, es la persona encargada de crear las propuestas de los diseños de la solución, dando mayor importancia a la facilidad de uso y solventando los puntos de dolor descubiertos en la investigación del UX. Encargado de realizar y probar los prototipos de la solución o soluciones antes de pasar al equipo de desarrollo.

- **Arquitecto** (1 persona)

Es el líder técnico de la célula, analiza las necesidades tecnológicas de la solución y plantea una arquitectura robusta, escalable y de alto desempeño que pueda ser realizada por el equipo de desarrollo de software y devops.

- **Ingeniero DevOps** (1 persona)

El ingeniero DevOps es la persona encargada de la infraestructura de las soluciones, su misión es implantar de manera óptima y con los recursos disponibles la arquitectura propuesta. Además, mantiene automatizados todos los procesos posibles de tal manera que facilite y reduzca el tiempo del ciclo de vida de desarrollo, con esto ayuda a disminuir el time to market de las funcionalidades de la solución. Se encarga de algunos procesos operativos y de la disponibilidad de la solución.

Sobre todas las células se encuentra el Gerente de Desarrollo de Aplicaciones y Mantenimiento que es la persona que realiza un mentoring, más no toma las decisiones ya que la célula es independiente y es guiada por el Product owner, es el principal stakeholder que monitorea los avances y descubrimientos de la célula en cada sprint review. Encargado de presentar resúmenes ejecutivos de cara a los miembros de la mesa directiva y otros gerentes, toma decisiones de creación, reasignación o replanteamiento de las células. Todo el equipo Scrum emplea un estilo de liderazgo de servicio hacia sus clientes (Maximini, 2015).

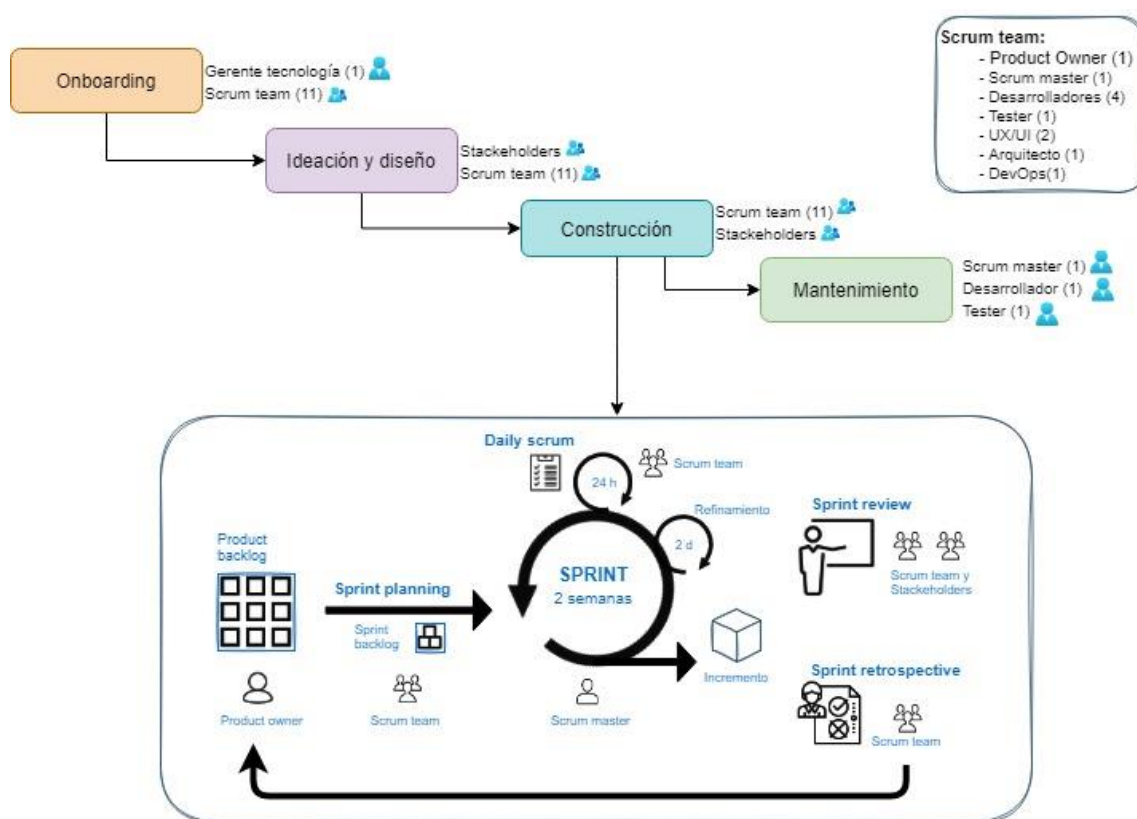


Figura 15 Arquitectura objetivo de negocio

### 3.2.5 Análisis de brechas

A continuación, se describirán las brechas encontradas en el modelo actual de procesos, personas y organización y se mostrará de qué forma se cerrarán dichas brechas.

Tabla 14  
Análisis de brechas en cada Fase de Desarrollo

Fase de desarrollo	Brechas actuales	Soluciones con el nuevo modelo
Inception	Al inicio se levantan todos los requerimientos que contendrá el producto, definiendo un cronograma; así cuando existen cambios, se complica realizar modificaciones en la planificación.	El realizar entregas continuas y recibir feedback temprano, permitirá no implementar funcionalidades que no generarán valor al usuario o al negocio. No se tiene un cronograma de todo el proyecto, sino que los requerimientos van cambiando a medida que cambia el entorno.
Conceptualización	Se define un alcance del proyecto en base al cronograma establecido.	En base a prácticas ágiles se elabora un roadmap que da una visión en alto nivel de qué incrementos se irán entregando en cada sprint.
Ejecución	No existe un tester que realice pruebas constantes de los desarrollos, se espera a finalizar el desarrollo de todo el producto para mostrarlo al usuario.	El realizar entregas continuas y recibir feedback temprano, permitirá no implementar funcionalidades que no generarán valor al usuario o al negocio, es decir, el cliente siempre se encuentra revisando cada incremento. Para que una funcionalidad pueda ser finalizada debe tener la aprobación del tester.
Pruebas	Los desarrolladores realizan pruebas al final de toda la implementación, no son pruebas exhaustivas. En caso de encontrar errores, resulta muy costoso resolverlos. No existe documentación que confirme el estado de las pruebas. No existe una manera formal de registrar los errores encontrados.	Existirá un tester en cada célula, que diseñe y ejecute pruebas funcionales, no funcionales, entre otras y no permitirá continuar el resto de desarrollos hasta que los errores se encuentren solucionados. Los errores son reportados en una herramienta colaborativa.
Certificación	El producto final es mostrado al cliente (pruebas UAT), se dan	Bajo un marco de trabajo ágil, el cliente siempre es parte de cada

	cuenta de que no se entendieron bien ciertos requerimientos, de que olvidaron mencionar funcionalidades, no se encuentran satisfechos.	entregable y ofrece feedback constante.
--	--	---

En base al análisis realizado se determina que la aplicación de Scrum y de prácticas ágiles permitirán implementar productos y servicios que generen gran valor tanto al negocio como al cliente, permitiéndole tener una experiencia satisfactoria a los usuarios.

#### 4. Arquitectura de aplicaciones/datos

En este capítulo se identifica la utilidad de las aplicaciones actuales, se definen planes de instalación e integración para aquellas aplicaciones que se implementarán, sus interacciones y relación con los procesos centrales de la organización (Josey, 2011).

##### 4.1 Arquitectura base de aplicaciones

En la fábrica de software actualmente se utilizan aplicaciones que conllevan mucho trabajo manual. A continuación, se analizarán algunas de ellas:

- **Servidores de aplicaciones**

Son programas de ordenador distribuidos que ofrecen un entorno con una arquitectura de alto rendimiento para la ejecución de un programa de aplicación. Usualmente se trata de un dispositivo de software que ofrece servicios de la aplicación a los clientes, pueden ser computadoras, dispositivos móviles, televisores, tablets, entre otros.

Características de los servidores de aplicaciones:

- Soportan una gran cantidad de estándares como HTML, XML, SSL, etc.
- Permiten conexión a varias fuentes de datos, sistemas y dispositivos

- Tienen una capa de middleware que permite comunicarse con otros servicios.

En la actualidad los servidores son físicos y no soportan una escalabilidad automática, es decir, es necesario de que un ingeniero se encargue de las configuraciones, réplicas y soporte constante para ofrecer una escalabilidad tanto horizontal como vertical. En el Banco los servidores de aplicaciones utilizados son WebSphere, Internet Information Server y Jboss, al no existir escalabilidad automática produce un dolor operativo para el área de infraestructura.

Los servidores de aplicación soportan cualquier arquitectura de soluciones, por el momento se encuentran implementadas aplicaciones en las cuales la interfaz de usuario y la capa de acceso a datos se encuentran en un mismo programa ubicado sobre los servidores locales, lo cual implica un innecesario desperdicio de recursos y disminuye la disponibilidad en caso de fallos ya que es necesario sustituir todo el programa en caso de alguna corrección.

- **IDE's de desarrollo** (Integrated Development Environment)

Un IDE es un entorno de desarrollo que combina herramientas de diseño e implementación de aplicaciones en una sola interfaz gráfica (GUI). Es un entorno digital empleado para la implementación de software en cualquier lenguaje de programación con el objetivo de agilizar este proceso con un ambiente integral de trabajo. (Piao, 2021)

Dependiendo del lenguaje de programación a desarrollar existen IDE's dedicados y otros adaptables mediante un plugin. Existen en el mercado varias ofertas de IDE's tanto en versiones *community* sin costo, como versiones *professionals* pagadas e inclusive versiones *ultimate* que son pagadas y cuentan con todas las bondades del IDE.

Los IDE's que se utilizan actualmente en la entidad financiera son:

- **IntelliJ IDEA**

Es un entorno de desarrollo integrado de programas informáticos creado por JetBrains y que proporciona las versiones *community* que es gratuita y tiene funcionalidades básicas y la versión *professional* que ofrece la suite completa. Incluye soporte para Java 8 aunque se le puede instalar cualquier versión, además mediante plugins se puede desarrollar en varios lenguajes de programación. Las dos ediciones tienen soporte para sistema de control de versiones y servidores embebidos.

Para la arquitectura objetivo se ha elegido Java como lenguaje de programación principal, en conjunto con IntelliJ ofrecen una potente combinación.

Java es uno de los lenguajes más conocidos por su versatilidad, potencia y escalabilidad, además con mucha oferta de profesionales en el mercado brinda una suite de desarrollo muy completa para la creación de una robusta arquitectura de aplicaciones tanto web como móvil.

Características de Java:

- Simple, basado en las mejores prácticas de C y C++.
- Independiente de la plataforma, se puede ejecutar en cualquier tipo de hardware.
- Garbage collector, es una funcionalidad muy útil que va limpiando bloques de memoria que no están siendo utilizados.
- Distribuido, proporciona una biblioteca estándar y herramientas para que pueda hacerlo distribuido.
- Orientado a objetos, es el estilo de programación más popular.
- Interpretado, el intérprete de java ejecuta directamente el código del objeto.

- **Postman**

Permite crear y enviar peticiones http a servicios REST a través de una interfaz gráfica de usuario. Es una potente herramienta que se utiliza principalmente para el *testing* de API mediante un protocolo http.

Características:

- Permite crear y compartir colecciones de peticiones, las colecciones son un conjunto de solicitudes.
- Permite realizar *testing* dinámico usando distintas variables de entorno dependiendo del ambiente: test, producción, desarrollo, etc.

- **Otras aplicaciones o herramientas**

- **Jira Software**

Es un software que ayuda con el seguimiento y gestión operativa de tareas, errores e incidencias. Forma parte de una suite de productos diseñados para ayudar a los equipos a gestionar el trabajo. Jira está enfocado en la gestión de proyectos de desarrollo de software basados en la metodología ágil.

Características:

- Posee plantillas de proyectos profesionales con workflows de trabajo predefinidos.
- Realiza un acompañamiento en base a historias de usuario o casos de uso durante todo el ciclo de vida del software.
- Proporciona varios tableros basados en principios Kanban o Scrum listos para usar.
- Creación de informes, recopila datos y los pone a disposición del equipo.
- Permite integración con varias herramientas.



A continuación, se muestra la Arquitectura base de aplicaciones:

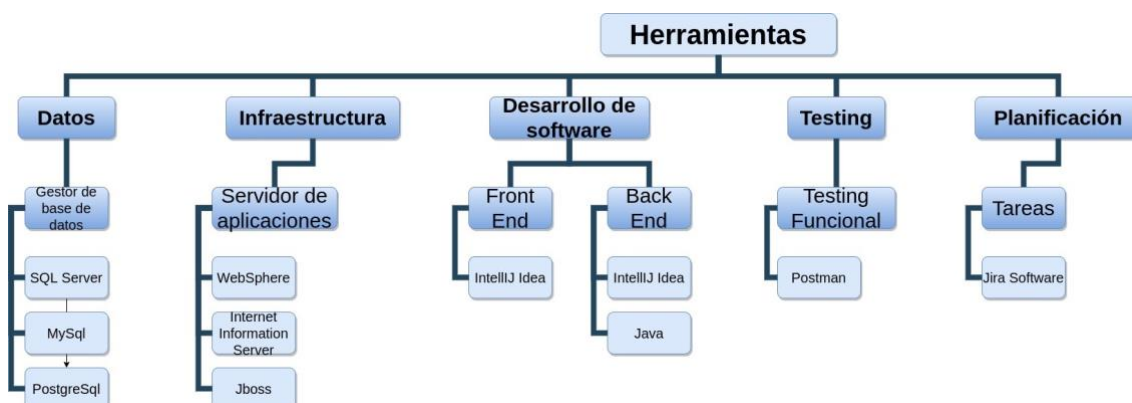


Figura 16 Arquitectura base de aplicaciones

## 4.2 Arquitectura base de datos/información

Existen varias fuentes de datos que alimentan el core de datos de la institución y son de dos tipos: datos estructurados y datos no estructurados.

### 4.2.1 Datos estructurados

Son aquellos que se encuentran en bases de datos relacionales, en la institución existen varios tipos de motores de bases de datos como MySQL, PostgreSQL y Sql Server. Mediante un middleware que orquesta los datos de la institución se tiene un acceso lento, desordenado y poco accesible a estos datos estructurados. El middleware se alimenta de los datos de filiales, *legacies*, external APIs, de canales bancarios y del core.

La cantidad de datos en tráfico son de aproximadamente 3 millones de clientes y de 100 millones de transacciones al mes, por este motivo el acceso no es eficiente y no permite determinar un perfil bancario del cliente con rapidez.

### 4.2.2 Datos no estructurados

Son los datos que no se encuentran en una base de datos relacional, es decir, no tienen una estructura interna identificable. Actualmente estos datos se

encuentran de manera no muy ordenada en un servidor ftp compartido, al cual no todos los colaboradores tienen acceso.

Los tipos de datos no estructurados almacenados son:

- Documentos Word, Excel, PDF.
- Correos electrónicos.
- Videos y audios.

Dentro de estos documentos se encuentra información valiosa sobre clientes, proveedores y flujos de trabajo, que ahora mismo resulta difícil encontrarla.

Además, al no tener la data consolidada se otorga una mala experiencia al usuario en el momento de acceder a agencia ya que el asesor está obligado a navegar por varias plataformas independientes para realizar un análisis del perfil bancario de dicho cliente lo cual toma mucho tiempo.

### **4.3 Arquitectura objetivo de datos y aplicaciones**

Se utilizarán las herramientas actuales desglosadas en la arquitectura base y se añadirán herramientas claves para la transformación digital y la fábrica de software ideal como son: Jenkins, Visual Studio Code con el lenguaje híbrido React native, Sonar cloud, tecnologías de Cloud computing y Signal fx las cuales se detallan a continuación.

- **Jenkins**

Es un servidor de integración continua que permite automatizar cualquier tipo de proceso, gratuito y de código abierto es uno de los más utilizados. Basado en tareas, orquesta un conjunto de forma secuencial y ordenada con el objetivo de mejorar el tiempo de desarrollo y librar de estas responsabilidades al equipo de desarrollo. Por sus grandes capacidades se puede utilizar para varios propósitos más.

Dado el enfoque actual de transformación digital y considerando que en el

corazón de dicha transformación se encuentra el cambio hacia una cultura DevOps, Jenkins es considerado como un aliado clave en este proceso, ya que ayuda a automatizar varios procesos anteriormente hechos a mano permitiendo a los recursos enfocarse en generar el mayor valor posible al incremento sin tener que preocuparse por estos procesos operativos y en consecuencia reduciendo el time to market. (Borja Hernández, 2017)

### Características

- Orquestador y automatizador de procesos.
  - Utilizado para ejecución de tareas manuales a horas determinadas.
  - Ejecución distribuida en base a eventos.
  - Dashboard con visualización de alertas o estado de procesos que lo hace fácil de usar.
- 
- **Visual Studio Code**

Es un IDE de desarrollo que soporta varios lenguajes de programación, diseñado y desarrollado por Microsoft, es una interfaz gráfica de usuario cuyo código fuente fue entregado a la comunidad, por ende, es una plataforma gratuita. Ideal para el desarrollo de aplicaciones móviles, aunque es compatible con lenguajes de programación enfocados en el backend. En la fábrica de software con el uso de Visual Studio Code se va a implementar el lenguaje híbrido de programación React native.

El objetivo de las células es entregar el mayor valor posible dentro del sprint, por lo tanto, lo ideal es usar lenguajes híbridos de programación debido a que se codifica una única vez y se puede compilar el mismo código para diferentes clientes como Android y IOs. En la actualidad React native se ha convertido en uno de los lenguajes híbridos de programación más robustos y completos, favorito para el desarrollo de aplicaciones móviles de manera ágil, fue fabricado e implementado por Facebook.

Características de React native:

- Tiene compatibilidad de compilación para iOS y Android.
- Usa el paradigma de programación reactiva que se basa en el flujo de datos asíncronos.
- Basado en componentes, permite crear componentes reutilizables lo cual ayuda considerablemente a la escalabilidad de las aplicaciones.
- Comunidad activa y actualizaciones constantes.
- Versatilidad, se puede encontrar versiones tanto para el desarrollo web como para el desarrollo híbrido móvil.

- **SonarCloud**

Es una plataforma que permite analizar la calidad del código realizando un análisis estático, esto corresponde a evaluar un software sin ejecutarlo. El objetivo es advertir sobre problemas o mejoras en una temprana etapa del desarrollo, además facilita un conjunto de métricas para la optimización del código.

Características:

- Soporta más de 20 lenguajes de programación.
- Existen dos versiones SonarQube el cual requiere instalación previa y SonarCloud que es una versión en la nube.
- Usa diversas herramientas de análisis estático de código como checkstyle, PMD, findBugs, entre otros.
- Permite configurar reglas dinámicas

- **Cloud computing**

El concepto de cloud computing se usa para describir un nuevo modelo de computación basado en la nube. Las plataformas de servicio de Cloud computing proporcionan infraestructura, almacenamiento de archivos, uso de aplicaciones o conexión de dispositivos sin ocupar recursos de hardware propios y aprovechando la infraestructura ya existente del proveedor.

- **Google Cloud Platform**

Es la suite que proporciona google para el manejo de la infraestructura y servicios en la nube. Permite crear, administrar y escalar aplicaciones, sitios web o servicios usando la disponibilidad y arquitectura de google con un costo de renta por lo utilizado.

Características:

- Alta agilidad y escalabilidad instantánea.
- Diversos modelos de cobranza.
- Infraestructura global lo que otorga una alta disponibilidad.
- Tecnología madura con clientes grandes.
- Encriptan los datos y canales de comunicación por defecto.

- **Selenium UI**

Proporciona un entorno de pruebas que se utiliza para comprobar que el software cumpla con los criterios de aceptación iniciales. Esta herramienta permite grabar y depurar los casos de prueba y automatizar su ejecución con esto ayuda a reducir el tiempo de testing.

Características:

- Ayuda con ejecución de test complejos y reduce las horas de trabajo
- Facilidad de creación y ejecución de test
- Las acciones se ejecutan paso a paso

- **Jmeter**

Es una aplicación con software libre que permite hacer pruebas masivas de carga y con esto mide el rendimiento del desarrollo. Dispone de una interfaz gráfica de usuario que permite crear los pasos a seguir para cumplir con los casos de prueba. (Zapata, 2021)

### Características:

- Se usa para pruebas de base de datos, pruebas funcionales y aplicaciones web.
  - Simula una carga pesada a los servidores o grupo de servidores.
  - Licencia open source y una interfaz de fácil uso.
  - Independiente de la plataforma ya que es un programa 100% java.
- **Signal FX**

Es un software SaaS que permite visualizar, automatizar, monitorear y analizar las métricas del comportamiento de la infraestructura, aplicaciones, microservicios, contenedores y funciones. Permite observar en tiempo real dichas métricas con el objetivo de encontrar rápidamente los errores y causas para de esta manera resolverlos.

### Características:

- Permite crear alertas por varios medios como sms y email.
- Permite muchos tipos de análisis como de rendimiento, tendencia, de causa raíz y en tiempo real.
- Permite la creación de diagramas y dashboards personalizados.
- Permite un tiempo de prueba gratuito, pero no existe una versión community.
- Arquitectura de datos objetivo.
- Debido a la cantidad de datos de clientes, proveedores y funcionamiento que maneja la institución existe un gran potencial de realizar una arquitectura de datos basada en agilidad.

- **Confluence**

Es una herramienta que servirá como único repositorio de datos no estructurados, permite crear y organizar todo el trabajo de forma colaborativa. Además, tiene integración con Jira software, permitiendo así tener una trazabilidad entre las tareas del proyecto y su respectiva documentación.

Características:

- Forma parte de la suite de aplicaciones de Atlassian.
- Posee un conjunto de plantillas html prediseñadas para fácil documentación.
- Existe una versión gratuita y te permiten un tiempo de prueba para la versión paga.

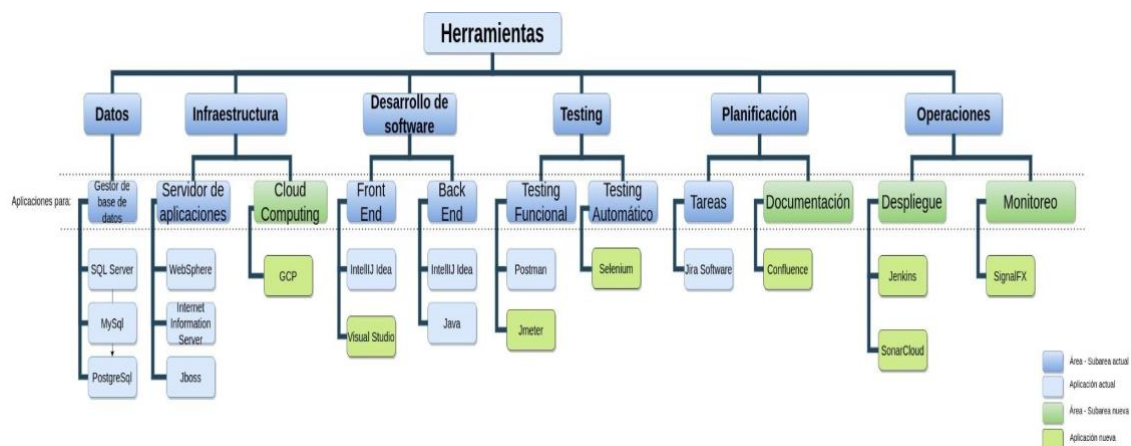


Figura 17 Arquitectura objetivo de aplicaciones

#### 4.4 Análisis de brechas

En la actualidad la institución se encuentra en un proceso de transformación digital lo cual es el paso inicial para el cambio hacia una nueva fábrica de software que se rige a metodologías y un mindset ágil. Con esto se busca reducir el tiempo de desarrollo y los problemas actuales del proceso.

Tabla 15  
Niveles de madurez de las aplicaciones actuales

Valoración Madurez/Capacidad					
		Aplicación	Estado Actual	Mediano Plazo	Referente
Datos	Gestores de base de datos	Sql Server	5	5	5
		MySql	5	5	5
		PostgreSql	2	4	5
Infraestructura	Servidores de aplicación	WebSphere	5	5	5
		IIS7	5	5	5
		JBoss	4	5	5
	Cloud Computing	Google Cloud Platform	1	3	5
Desarrollo de software	Frontend	IntellJ Idea	3	4	5
		Visua Studio Code	1	3	5
	Backend	IntellJ Idea	3	4	5
Testing	Funcional	Postman	4	5	5
		Jmeter	1	3	5
	Automatizado	Selenium	1	3	5
Planificación	Tareas	Jira Software	3	4	5
	Documentación	Confluence	2	3	5
Operaciones	Despliegue	Jenkins	1	3	5
		SonarCloud	2	3	5
	Mantenimiento	Signal FX	1	2	5

Niveles de madurez:

**Nivel 1:** No se tiene conocimiento de la herramienta, es necesaria una investigación ardua para su implementación.

**Nivel 2:** Se tiene un conocimiento básico de la herramienta, es necesaria una prueba de concepto antes de la implementación.

**Nivel 3:** Existen recursos con experiencia para la implementación y administración de la herramienta.

**Nivel 4:** La herramienta está implementada y hay recursos que lo administran, falta explotar todo su potencial.

**Nivel 5:** Herramienta implementada en su totalidad y otorga un retorno de la inversión constante.



## Estado Actual, Mediano Plazo y Referente

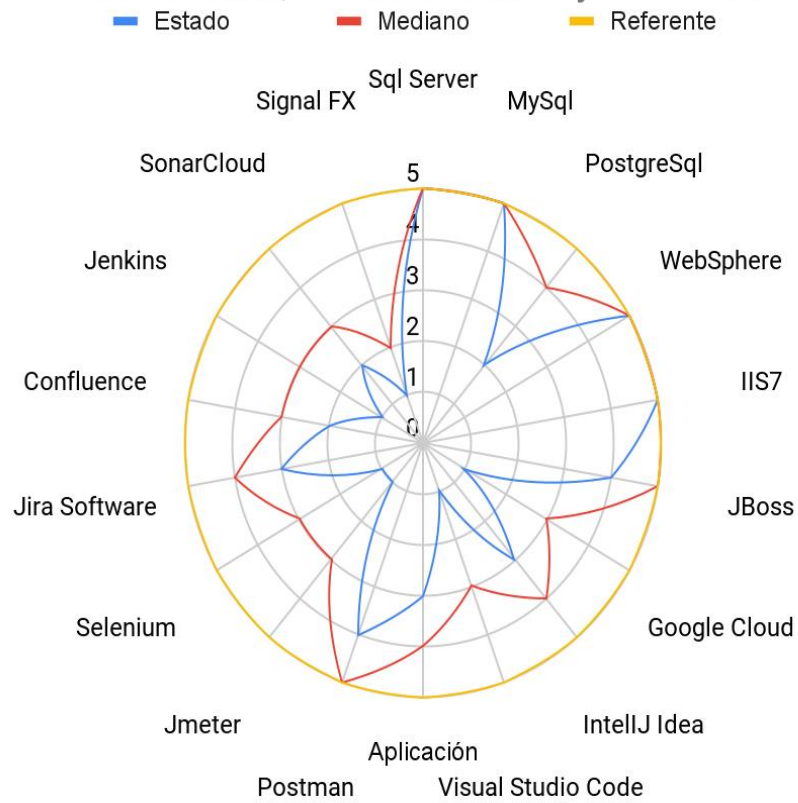


Figura 18 Gráfico resultante del análisis de brechas

En la siguiente tabla se listan las propuestas para cerrar las brechas existentes:

Tabla 16  
Análisis de brechas de aplicaciones y soluciones propuestas

Brechas actuales	Soluciones con el nuevo modelo
Existen muchos trabajos manuales, como el despliegue de las aplicaciones en producción y el manejo y soporte de servidores físicos.	Implementar servidores de aplicaciones en la nube reducirá las horas de trabajo de los ingenieros de infraestructura considerablemente, lo mismo sucederá al implementar Jenkins para el despliegue e integración continuos, ya que permitirá al ingeniero de software concentrarse solo en generar valor y no en tareas operativas.
Todas las pruebas realizadas son manuales lo cual requiere de tiempo de los desarrolladores, además al ser manuales suelen escaparse varios errores que son detectados por el usuario en producción.	Al automatizar las pruebas funcionales y de carga, el objetivo es similar, se garantiza una aplicación con alta calidad en poco tiempo. Además, en conjunto con las herramientas de monitoreo, brindan una detección rápida y causas de posibles problemas, cubriendo todo el ciclo de vida de la fábrica de software.
Se encuentran totalmente separadas el área de Desarrollo y Operaciones, existen varios inconvenientes al tratar de culpase entre ellos por errores presentados.	Al unir todas las herramientas se logra un potente ambiente de trabajo basado en DevOps, todos unidos con una alta cohesión de equipo que busca la reducción del time to market y con el cliente en el corazón de las soluciones.
La data no está centralizada, se encuentra desordenada en distintas fuentes.	Se debe crear un CRM institucional con el propósito de consolidar en una misma interfaz la data de las distintas fuentes y así permita poder disponer de un perfil lo más exacto posible del cliente.  El CRM debe ser una solución interna del Banco y lo realizaría una célula de la fábrica de software.

Como se visualiza en la tabla anterior, con el nuevo modelo se establecen varias soluciones a las brechas existente, que una vez conceptualizadas y

especificadas se consolidarán como proyectos de la arquitectura de aplicaciones y datos.

## 5. Arquitectura de infraestructura base

La entidad financiera mantiene una infraestructura base robusta que constantemente se modifica para adaptarse a las necesidades y cantidad de clientes, aun así, no ha logrado ofrecer un nivel de disponibilidad acorde al prestigio de la institución.

### 5.1 Arquitectura base de Infraestructura

Se detalla a continuación las características de los principales equipos utilizados por la institución:

Tabla 17  
Detalle de equipos de infraestructura utilizados actualmente

Equipo	Tipo	Características	Detalles
Servidores	Servidor Dell PowerEdge R440	<ul style="list-style-type: none"> <li>• Capacidad de disco duro: 1000 GB SSD</li> <li>• Modelo del procesador: 4110</li> <li>• Tipo de memoria interna: DDR4-SDRAM</li> <li>• Velocidad de rotación de disco duro: 7200 RPM</li> <li>• Montaje en rack: Si</li> </ul>	Servidor dedicado para el entorno de desarrollo y pruebas.
	Servidor Dell PowerEdge R910	<ul style="list-style-type: none"> <li>• Chasis: servidor Dell R910 Gen II de 4 bahías con CPU de la serie E7</li> <li>• Procesadores: 4 x Intel Xeon E7-4870 2,40 GHz 10 Core.</li> <li>• Configuración de memoria: 256 GB – 64 x 4GB PC3-10600R DDR3 registrado.</li> <li>• Tarjeta RAID: controlador RAID PERC H700 con caché de 512 MB.</li> <li>• Fuentes de alimentación: 4 x 1100 W PSU para R910.</li> <li>• Acceso remoto: iDRAC6 Enterprise</li> <li>• Discos duros: 4 x Evo 500 GB SSD para la serie R de Dell</li> </ul>	Servidor dedicado para montar en producción las aplicaciones y páginas web desarrolladas.

Computadores	Laptop Lenovo Ideapad 330S	<ul style="list-style-type: none"> <li>• Procesador: Intel® Quad Core i7-8550U de 8.ª generación</li> <li>• Sistema operativo: Windows 10 Home</li> <li>• Pantalla: FHD IPS de 39,62 cm (15,6") y resolución 1920 x 1080.</li> <li>• Memoria: DDR4 integrada de 4 GB + SODIMM de 8 GB.</li> <li>• Almacenamiento: Unidad SSD PCIe de 256 GB.</li> <li>• Batería: Hasta 7 horas; Rapid Charge.</li> <li>• Dimensiones: 35,84 cm x 24,41 cm x 1,94 cm.</li> </ul>	Laptop específica para el desarrollo de las aplicaciones.
	Apple MacBook A1181	<ul style="list-style-type: none"> <li>• Negro MacBook 13" pulgadas.</li> <li>• 2,16 GHz procesador Intel Core 2 Duo.</li> <li>• 2 GB de RAM (ampliable a 4GB Max).</li> <li>• Disco duro de 160 GB.</li> <li>• Soporta hasta OSX 10.7.5 Lion.</li> </ul>	Dedicada para el desarrollo y mantenimiento de aplicaciones para IOs.
	PCs de escritorio	<ul style="list-style-type: none"> <li>• Sistema operativo: Windows 7.</li> <li>• UPC: Intel Core i7-9700 de 8 núcleos a 3GHz.</li> <li>• Tipo de memoria: DDR4 de 2666 MHz.</li> <li>• Disco duro: 1 TB.</li> </ul>	Dedicada para algunos desarrolladores menos expertos y para las pruebas de control de calidad.
	Laptop Dell Inspiron 15 serie 3000	<ul style="list-style-type: none"> <li>▪ Sistema operativo: Windows® 8.1 64 bit.</li> <li>▪ Tarjeta de video: Intel HD Graphics.</li> <li>▪ Pantalla: 15.6-inch HD (1366 x 768) Truelife LED-Backlit Display.</li> <li>▪ Memoria: 4GB Single Channel DDR3L.</li> <li>▪ Disco duro: 500 GB 5400 RPM (standard) with options up to 1 TB 5400 RPM SATA Hard Drive.</li> </ul>	Laptop específica para el desarrollo de las aplicaciones.

Dispositivos móviles	Dispositivo Android	<ul style="list-style-type: none"> <li>▪ Tamaño: 76.3 mm x 159.2 mm x 8.7 mm.</li> <li>▪ Peso: 183 g.</li> <li>▪ Pantalla: 6.4 pulgadas.</li> <li>▪ Procesador: HUAWEI Kirin 810, CPU: Octa-Core.</li> <li>▪ S.O. EMUI10.0.1 (Basado en Android10.0)</li> <li>▪ Memoria: 6 GB de RAM + 128 GB de ROM</li> <li>▪ Batería: 4,200 mAh</li> <li>▪ NFC: JNY-L22A</li> </ul>	Destinado para el desarrollo y pruebas de las aplicaciones móviles en Android.
	Dispositivo IOs	<ul style="list-style-type: none"> <li>▪ Pantalla: 4.7", 1334 x 750 pixels</li> <li>▪ Procesador: Apple A11 Bionic</li> <li>▪ RAM: 2GB</li> <li>▪ Almacenamiento: 64GB/256GB</li> <li>▪ Batería: 1850 mAh</li> <li>▪ OS: iOS 11</li> <li>▪ Peso: 148 g</li> </ul>	Destinado para el desarrollo y pruebas de las aplicaciones móviles en IOs.
Conexión al Core Bancario	Bus Omnicanal		Orquesta el tráfico pertinente hacia el core bancario.

El diagrama de infraestructura base de la fábrica de software se detalla a continuación:

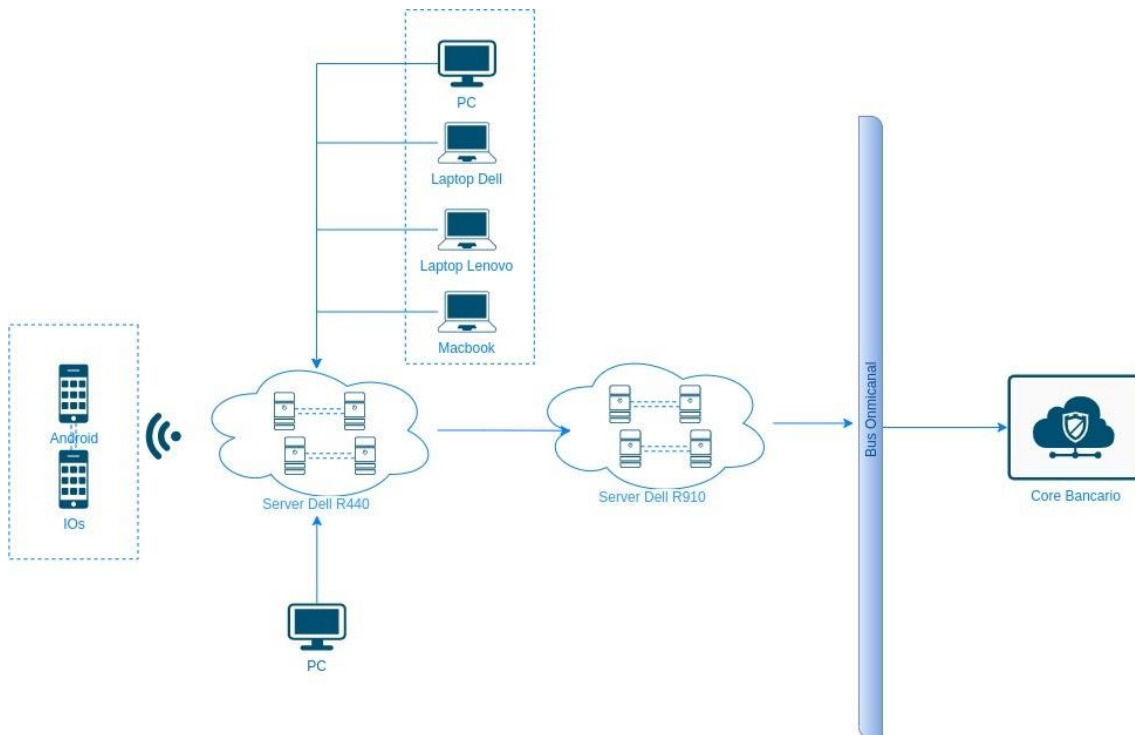


Figura 19 Arquitectura base de infraestructura

En la fábrica de software se dispone de 2 ambientes: desarrollo y producción, además cada desarrollador levanta un ambiente local en el cual implementa su código. Ambiente o entorno de desarrollo de software es un conjunto de procedimientos y herramientas necesarias para que el desarrollador realice su trabajo diario, por ejemplo: codificación, pruebas, ejecución de aplicaciones, entre otras.

En medida de lo posible se trata de mantener los ambientes lo más parecidos a producción, con el objetivo de que las pruebas sean lo más reales posibles y así detectar errores en etapas tempranas del desarrollo.

### 5.1.1 Ambiente de desarrollo

Ambiente en el cual los desarrolladores realizan la construcción del software, este ambiente tiene bajos recursos y se trata de tener la data lo más parecida al

entorno de producción. Es un entorno creado para equivocarse, hacer pruebas de concepto, crear funcionalidades y hacer pruebas funcionales de forma colaborativa ya que los desarrollos convergen en este ambiente.

### 5.1.2 Ambiente productivo o de producción

Maneja las transacciones reales de los usuarios, este entorno tiene características de seguridad y disponibilidad fuertes y robustas de cara al cliente. Normalmente no es permitido realizar configuraciones o cambios durante horario normal sino en horas de la noche o madrugada.

Si se detecta un error en este ambiente el equipo de desarrollo debe lograr replicarlo en ambiente de desarrollo y corregirlo de inmediato.

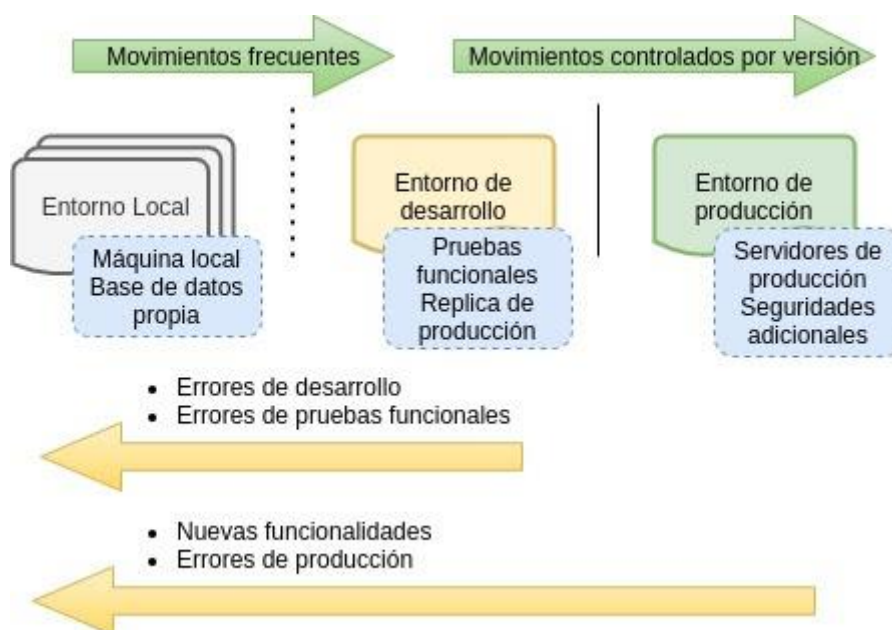


Figura 20 Ambientes/Entornos actuales, adaptado de (Autentia, 2020)

Actualmente los entornos de desarrollo interactúan según el diagrama de la figura anterior, los desarrolladores crean su código en su ambiente local y lo pasan al ambiente de desarrollo en donde realizan pruebas back to back y de integración. Luego en el mismo ambiente de desarrollo el equipo realiza pruebas funcionales, una vez certificado que no existen errores se realiza el paso a

producción poco controlado, ya que no se efectúan revisiones del código a desplegar.

A continuación, se detallan las características de los servidores de aplicaciones actuales:

- **WebSphere**

Es una plataforma que proporciona toda la infraestructura necesaria para alojar una aplicación, fue desarrollado por IBM y está construido con especificaciones de java y otros estándares de dominio público para desarrollar aplicaciones.

Características:

- Ejecuta las aplicaciones web con un perfil ultraligero.
- Posee varios gestores de bases de datos.
- Proporciona datos y creación de informes para análisis en tiempo real.
- Posee documentación robusta y una comunidad grande que aporta continuamente con soluciones a problemas recurrentes.

- **Internet Information Server**

Es un conjunto de servicios que proporcionan un paquete de software para Windows server, se usa para alojar aplicaciones y contenidos en la web.

Características:

- Tiene una interfaz gráfica de usuario que permite gestionar sitios web y usuarios asociados.
- Permite adicionar módulos o extensiones que agregan funcionalidad al servidor.
- Disponible para internet e intranet, siendo uno de los más usados en la intranet.



- **Jboss**

Es un servidor de aplicaciones desarrollado en 100% en java y de código abierto, por este motivo se lo puede usar en cualquier sistema operativo.

Características:

- Ofrece un gran rendimiento para aplicaciones empresariales.
- Puede ser descargado, manipulado y distribuido sin restricciones de licenciamiento.
- Basado en el estándar Java 2EE.

## 5.2 Arquitectura objetivo de infraestructura

Se ilustra a continuación el detalle de la arquitectura objetivo, en la que se puede evidenciar que se utilizará la misma infraestructura actual y para mejorarla se agregarán componentes y ambientes específicos que potenciarán dicha infraestructura para un mejor entorno de desarrollo.

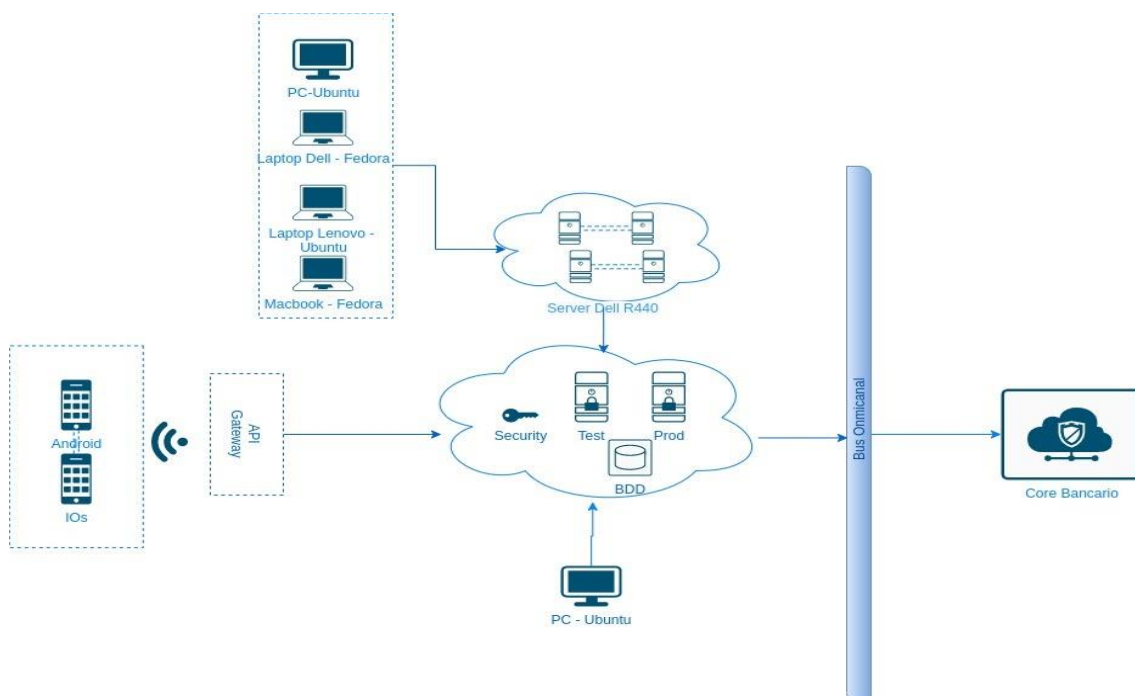


Figura 21 Arquitectura objetivo de infraestructura

Es necesario implementar un ambiente de desarrollo colaborativo en el cual los desarrolladores unan sus funcionalidades y puedan realizar pruebas integrales. Además, se debe crear un ambiente de pruebas, así al no compartir el mismo ambiente entre desarrollo y control de calidad, hay más certeza de que los datos no estén corruptos.

Uno de los objetivos a cumplir con el uso de metodologías ágiles, es la detección temprana de errores, la implementación de este nuevo ambiente apalancará a alcanzar este propósito.

## **5.2.1 Ambientes para desarrollo**

En base al análisis de la sección previa, se determina que los ambientes o entornos para desarrollo con los que deben contar la fábrica de software son:

### **5.2.1.1 Ambiente local**

Cada desarrollador debe tener localmente en su computador una réplica lo más exacta posible al ambiente real, tanto en back como front end.

### **5.2.1.2 Ambiente de desarrollo**

Es el ambiente destinado a integrar las funcionalidades de los desarrolladores y además realizar pruebas técnicas, de esta manera se puede detectar errores en una etapa temprana.

### **5.2.1.3 Ambiente de Test o Pruebas**

Se trata de un entorno de trabajo más limpio y robusto, bastante cercano a los datos de producción. En este ambiente se realizan réplicas constantemente desde el entorno de producción, se tiene especial cuidado con datos sensibles como cuentas o contraseñas. Se usa comúnmente para realizar las pruebas finales antes de lanzar el feature/funcionalidad a producción.

Al ser un entorno aislado al de desarrollo, los datos son lo más reales posible y no hay data corrupta generada por las pruebas de los desarrolladores.

#### 5.2.1.4 Ambiente de producción

Entorno robusto que ofrece una infraestructura con alta disponibilidad para ser usada por el cliente.

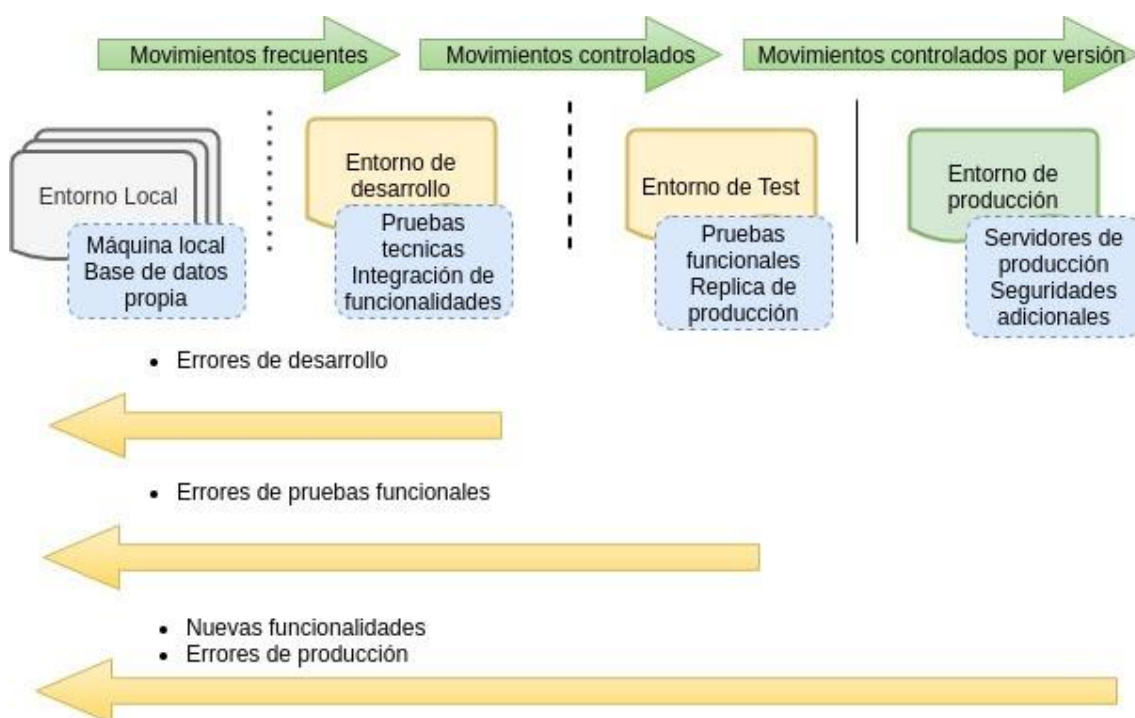


Figura 22 Ambientes/Entornos objetivo, adaptado de (Autentia, 2020)

Con esta implementación de ambientes de desarrollo se puede llevar un mejor control del código que está siendo subido a producción.

#### 5.2.2 Implementación de Google Cloud Platform

Uno de los retos que se tiene que afrontar al momento de crear una aplicación empresarial es la escalabilidad y la disponibilidad de la aplicación.

Actualmente este proceso es muy costoso en horas hombre y dinero para la fábrica de software y por ende para la entidad financiera. Una de las soluciones planteadas por las grandes empresas de tecnología como Google, es compartir

su infraestructura a pequeñas, medianas y grandes empresas cobrando una renta por lo que consumen y otorgando un acuerdo de nivel de servicio muy alto.

Partiendo de esta gran oportunidad de mejora se ha planteado la inserción paulatina de la nube a la infraestructura actual, con un costo inicial elevado debido a la curva de aprendizaje que conlleva la implementación.

Google cloud platform ofrece varios servicios en la nube como IaaS y PaaS que se adaptan a la necesidad de la aplicación a desarrollar. La diferencia entre estos servicios es el nivel de personalización de la solución, el nivel de administración y la responsabilidad.

### 5.2.2.1 IaaS

Permite elegir la infraestructura desde el nivel más bajo incluyendo sistema operativo, servidor de aplicaciones, además se puede escalar según las necesidades, para esto Google ofrece el servicio de Compute Engine.

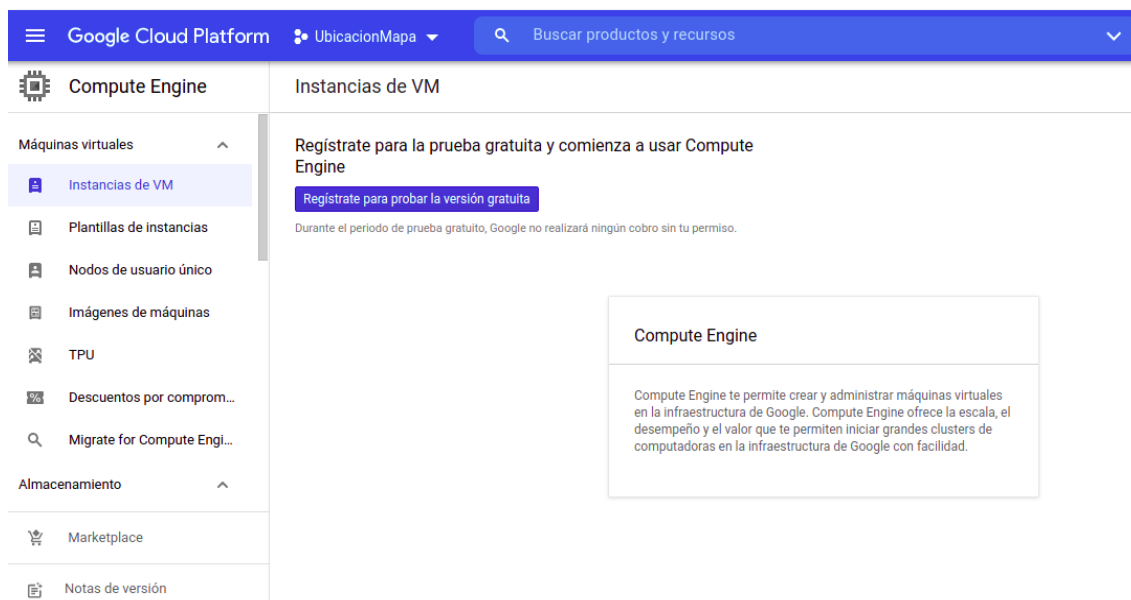


Figura 23 Dashboard de Compute Engine de Google

### 5.2.2.2 PaaS

Este servicio permite a los desarrolladores enfocarse 100% en el código y ofrece una infraestructura completa sin necesidad de una administración. Provee todos los recursos necesarios al código subido por los desarrolladores a la nube.

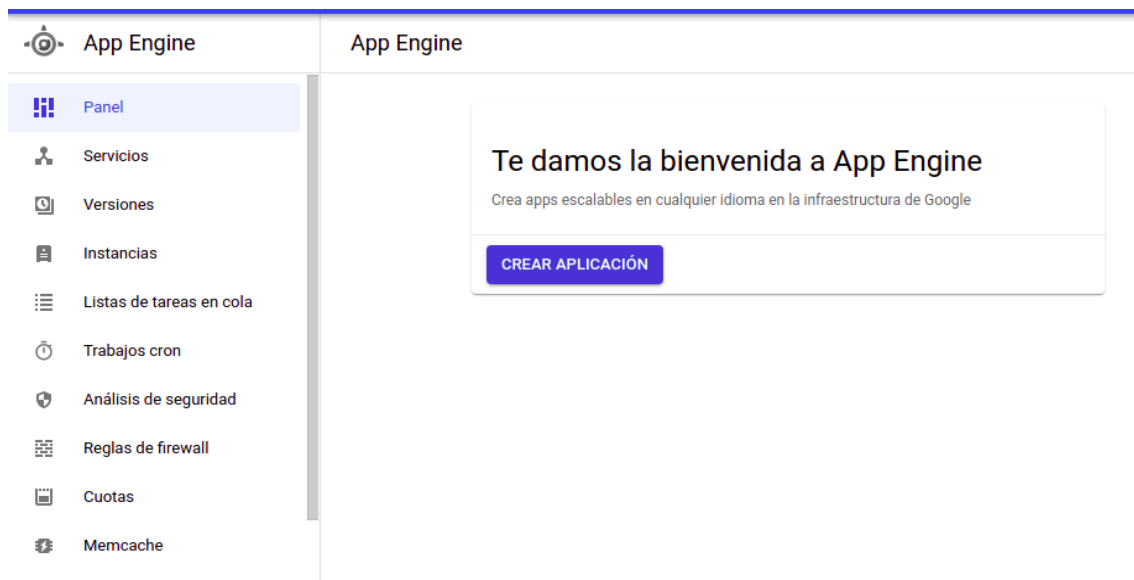


Figura 24 Dashboard de App Engine de Google

## 5.3 Análisis de brechas

A continuación, se describen las brechas existentes en la arquitectura actual y las soluciones que conllevan la arquitectura objetivo.

Tabla 18  
Detalle de equipos de infraestructura utilizados actualmente

Brechas actuales	Soluciones con el nuevo modelo
Al utilizar el mismo ambiente tanto para desarrollo como para pruebas, genera data corrupta lo cual retrasa el proceso de pruebas o a su vez no permite detectar errores.	Se agregará un ambiente de test o pruebas, utilizado solo por el área de control de calidad.
Existe escalabilidad limitada y la disponibilidad depende del trabajo humano, lo cual genera lentitud en las aplicaciones y cargas de trabajo.	Se plantea la inserción de un modelo de Cloud computing que, en conjunto con los servidores físicos, permitirá tener una infraestructura de desarrollo híbrida y así poder mejorar la disponibilidad de las aplicaciones.  A largo plazo: se tendrá la mayoría de la infraestructura en la nube (es el ideal).
Windows levanta muchos servicios que no son necesarios en el momento del desarrollo, lo cual ralentiza la implementación de la solución.	Se plantea la instalación de un sistema operativo basado en Unix para mejorar los tiempos de desarrollo a corto plazo.
No existe un control para que el servidor atienda solo aquellas peticiones que son reales y seguras.	Para la orquestación de peticiones realizadas por los clientes (móviles, web, etc) se implementará un API Gateway que filtre solo las peticiones que sean reales y seguras.

## 6. Oportunidades y soluciones

Luego de realizar un análisis de las brechas existentes en cada arquitectura surgen oportunidades de mejora que se detallan en este capítulo.

### 6.1 Consolidación de iniciativas

A continuación, se consolida en varias iniciativas las propuestas que se presentaron en el transcurso del proyecto y que permitirán cumplir con las respectivas arquitecturas objetivo.

Tabla 19  
Consolidación de iniciativas en base a cada arquitectura

Arquitectura	Id	Iniciativa
Negocio	PN01	Implementación marco de trabajo Scrum.
	PN02	Crear área de QA y definir una metodología ágil de pruebas.
	PN03	Adopción de una cultura DevOps en la organización.
Aplicaciones/Datos	PA01	Usar herramientas de apoyo para DevOps.
	PA02	Crear una célula dedicada a la implementación de un CRM institucional.
Infraestructura	PI01	Usar los servicios de infraestructura de Google Cloud Platform.
	PI02	Usar ApiGee para la gestión de APIs

## 6.2 Conceptualización de iniciativas

En esta sección cada una de las iniciativas propuestas serán conceptualizadas y especificadas.

### 6.2.1 Proyecto PN01: Implementación marco de trabajo ágil Scrum.

Tabla 20  
Proyecto PN01: Implementación marco de trabajo ágil Scrum.

Proyecto PN01: Implementación marco de trabajo ágil Scrum.	
Situación Actual y Principales Observaciones	<p>No se tiene foco en el cliente, por lo que no existe una buena relación ya que no se le entregan productos que le generen valor.</p> <p>No existe un marco de trabajo que se adapte a las necesidades de aquellos proyectos que deben ir cambiando a medida que va cambiando el entorno.</p>
Actividades fundamentales	<ul style="list-style-type: none"> <li>• Contratación Agile coach.</li> <li>• Talleres de capacitación a la organización sobre el nuevo marco de trabajo.</li> <li>• Adquirir licencias de Jira.</li> </ul>

Plazo de implementación	Menor a 6 meses.
Costos referenciales	<ul style="list-style-type: none"> <li>• Contratación Agile coach: \$3400 mensuales.</li> <li>• Talleres de capacitación a cargo del Agile coach.</li> </ul>

El nuevo marco de trabajo a implementar en la fábrica de software propone definir correctamente desde el inicio los problemas y necesidades y disponibilizar entregables en tiempos cortos y con calidad, todo esto teniendo como base un equipo estable multidisciplinario que trabaja de la mano con las áreas que intervienen en el proceso.

- **Contratación de un Agile coach**

Inicialmente es indispensable contratar un Agile coach con experiencia en apoyo a células ágiles cuyo desarrollo sea enfocado en productos para entidades financieras.

En esta iniciativa, los talleres a todas las áreas de la organización consisten en capacitaciones de alto nivel, los detalles serán afinados cuando dicha área deba estar incluida en algún proyecto de la fábrica de software.

El conocimiento impartido será acerca de cómo será la nueva reestructuración y de cómo se trabajará en cada uno de los proyectos, es importante recibir dichas capacitaciones debido a que pasarán a ser *stakeholders* y deberán asistir al *Sprint review*, que es un evento en el cual la célula muestra a los interesados el incremento resultante del sprint finalizado, al concluir el evento recibirán retroalimentación valiosa y el equipo lo tomará en cuenta en los próximos sprints.

- **Priorización del backlog del producto**

En ocasiones debido a las retroalimentaciones recibidas suelen cambiar las priorizaciones del *backlog*, cambiando así el *roadmap* previsto, sin embargo, gracias a la entrega continua que se obtiene al trabajar con Scrum, estos



cambios serán siempre bienvenidos por la célula ya que así el producto o servicio se acercará cada vez más al resultado que realmente espera el cliente.

Para la priorización de los elementos del *backlog* se utilizará la técnica de MoSCoW, es una de las formas más fáciles y útiles de poner un orden en el listado de pendientes por hacer. El equipo y las partes interesadas utilizan esta técnica para colaborar y priorizar los requisitos con el objetivo de ofrecer valor al cliente desde el inicio.

Los resultados de MoScoW se clasifican como escala nominal, se agrupan todos los requisitos que tengan prioridad similar. (Hudaib, 2018)

Tabla 21  
Proyecto PN01: Implementación marco de trabajo ágil Scrum.

Mo	<i>Must have</i> (Debe tener)	Estas características son críticas para el proyecto, deben ser tomadas en cuenta obligatoriamente.
S	<i>Should have</i> (Debería tener)	Estas características son críticas, pero no indispensables.
Co	<i>Could have</i> (Podría tener)	Estas características no son críticas, pero estaría muy bien tenerlas.
W	<i>Won't have</i> (No se van a hacer)	Estas características no aportan valor.

Adaptado de (Mulder, 2017)

- ***User story mapping***

Una vez priorizado el *backlog*, se deberá actualizar el *roadmap*, para ello se utiliza el artefacto llamado *User story mapping*, el mapeo de historias de usuario es una herramienta valiosa para el desarrollo de software, ayuda al equipo a mantenerse enfocado en los usuarios y sus necesidades y a la vez tener una visión de las características individuales del producto. (Patton & Economy, 2014)

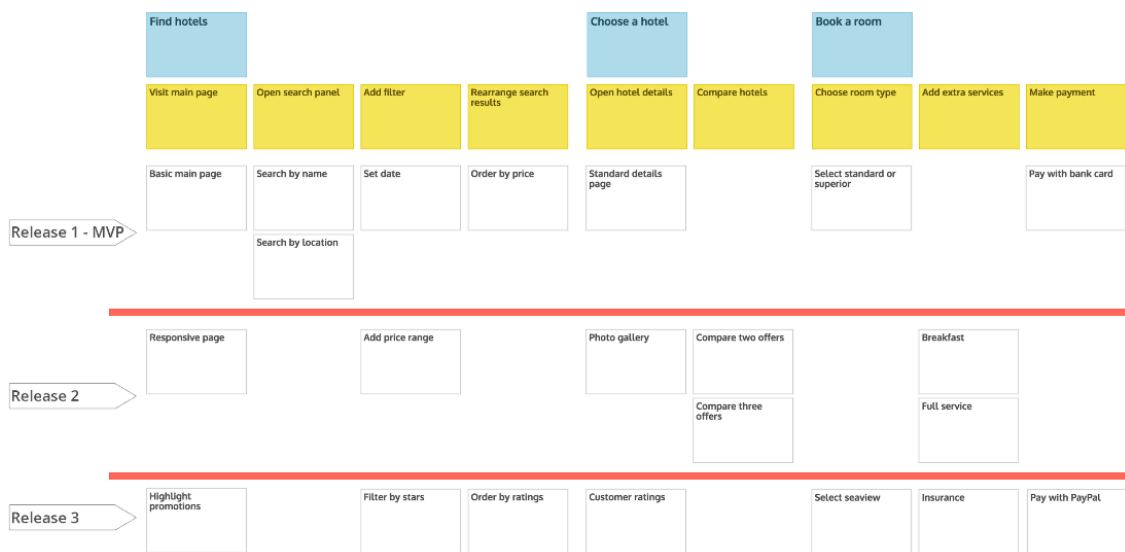


Figura 25 Herramienta User story mapping (How to Use a Story Map in Jira, 2019)

La construcción de esta herramienta se la puede hacer en alguna herramienta colaborativa como Mural o Miro, y se debe tomar en cuenta lo siguiente:

1. Se deben colocar los objetivos, en el caso de la fábrica de software, se trazan objetivos cada 3 meses y se da seguimiento a ellos en cada *Sprint review*. De igual forma al inicio se colocan las funcionalidades o épicas del producto (una épica es una historia de usuario muy grande que estará compuesta por varias historias de usuario).
2. Una vez que ya se tiene la columna vertebral del producto, se empiezan a agregar las funciones escribiéndolas como historias de usuario, utilizando el formato: Quién utilizará, Qué necesita y Para qué lo necesita.
3. Se priorizan las historias de usuario descritas.
4. Es opcional mapear los versionamientos.
5. Se colocan etiquetas a las historias de usuario del número de sprint en el que será trabajado, de esta forma se tendrá una visión estimada del avance de las funcionalidades en cada sprint.

- **Metodología XP** (Programación extrema)

Parte de las habilidades que debe tener el equipo de desarrollo, puntualmente los ingenieros de programación, es que deben tener conocimiento sobre las

prácticas que propone la metodología XP (Programación extrema), entre todo el equipo de desarrollo de la fábrica de software deberán definir cuáles aplicar a nivel de todas las células.

Son técnicas que mejoran la eficiencia de los desarrolladores, permitiéndoles tener una programación más organizada y ágil, logrando fortalecer conocimientos entre ellos por ejemplo a través de la técnica *pair programming* que consiste en trabajar en pares para poder equiparar conocimientos.

Finalmente, el resultado de aplicar esta metodología es que los resultados tengan mayor calidad, es decir, reduce la tasa de errores notablemente.

### 6.2.2 Proyecto PN02: Talleres de capacitación de nuevos roles y responsabilidades.

Tabla 22

Proyecto PN02: Talleres de capacitación de nuevos roles y responsabilidades.

Proyecto PN02: Talleres de capacitación de nuevos roles y responsabilidades.	
Situación Actual y Principales Observaciones	Los equipos de trabajo disponen de un Líder de proyecto que gestiona todo tipo de proyectos bajo un marco de trabajo cascada, además no cuentan con asignación de personas a tiempo completo y no disponen de todos los roles necesarios para que el ciclo de desarrollo sea más ágil dando como resultado productos de mayor calidad.
Actividades fundamentales	<ul style="list-style-type: none"> <li>• Talleres de capacitación a los colaboradores de la fábrica de software sobre los nuevos roles y responsabilidades definidas.</li> <li>• Talleres de capacitación al resto de áreas de la organización.</li> </ul>
Plazo de implementación	Menor a 3 meses.
Costos referenciales	Talleres de capacitación a cargo del Agile coach.

La implementación de Scrum conlleva a plantear nuevos roles que, a pesar de tener ciertas habilidades definidas, cuando la situación amerita comparten responsabilidades, logrando así cumplir con el objetivo del sprint.

El equipo de trabajo es llamado célula y está compuesto por pocas personas con el objetivo de tener una mejor comunicación y su trabajo sea más productivo. Existen 3 roles principales dentro del *Scrum team* y son *Product owner*, *Scrum master* y *Developers*, dentro de este último se pueden tener varios profesionales con distintas habilidades como desarrolladores, testers, arquitectos, entre otros.

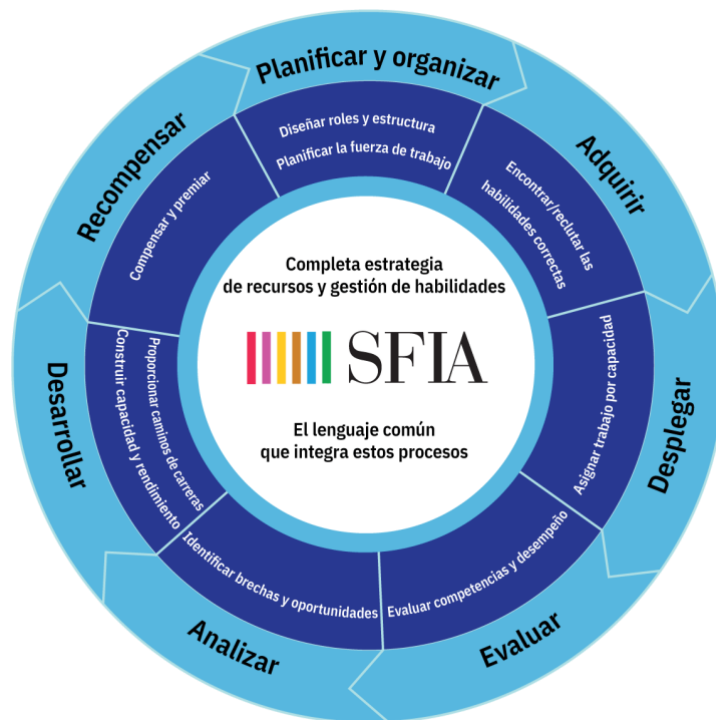


Figura 26 Modelo SFIA

Con el propósito de que todos cumplan con las habilidades necesarias se implementará SFIA, este es un modelo de referencia que muestra aquellas habilidades indispensables y necesarias de desarrollar para poder operar sistemas de información.

### 6.2.3 Proyecto PN03: Definir una metodología ágil de pruebas (objetivo automatización)

Tabla 23

Proyecto PN02: Definir una metodología ágil de pruebas (objetivo automatización).

Proyecto PN03: Definir una metodología ágil de pruebas (objetivo automatización).	
Situación Actual y Principales Observaciones	Los equipos de trabajo no cuentan con un tester, los programadores realizan pruebas funcionales cuando el producto se encuentra terminado, lo que ha venido generando insatisfacción en los clientes al encontrarse con varios errores en producción.
Actividades fundamentales	<ul style="list-style-type: none"> <li>• Contratar un Líder de Control de calidad con experiencia en pruebas ágiles y automatización de pruebas.</li> <li>• Contratar personal con perfil de tester, experiencia en pruebas manuales y automatizadas.</li> </ul>
Plazo de implementación	Menor a 3 meses.
Costos referenciales	<ul style="list-style-type: none"> <li>• Contratación líder de QA: \$2.000</li> <li>• Contratación tester: \$1.500</li> </ul>

Para dar inicio a esta iniciativa se debe primero contratar un Líder de Control de calidad que cuente con experiencia en implementación de procesos ágiles de pruebas, herramientas de reporte/seguimiento de incidencias y en procesos de automatización con distintas herramientas como: Selenium, Jmeter y SoapUI, estas herramientas son gratuitas es así que no requiere de licencias.

El Líder de Control de calidad se encargará de:

- Verificar disponibilidad de recursos y asignar los testers a las células correspondientes.
- Planificar reuniones periódicas para homologar conocimientos de todos los testers.
- Es responsable de definir tanto el foco de las pruebas, como las metas y objetivos que contemplará el Plan maestro de prueba, así como la estrategia a utilizar.

- Definir formato de los entregables y ubicación de repositorio.

El proceso inicial propuesto para el control de calidad dentro de cada célula es el siguiente:

Los Testers se encargarán de:

- Actualizar y ejecutar el Plan de pruebas y estrategia de pruebas definidas.
- Detectar, reportar y dar seguimiento a los defectos encontrados durante los distintos tipos de pruebas.
- Generar y mantener actualizados los artefactos utilizados en la gestión de pruebas.
- Reportar los riesgos detectados.
- Apoyar al área de negocio en la ejecución de pruebas (UAT).
- Asegurar la calidad en todas las etapas del ciclo de desarrollo incluso en aquellas tempranas.

Para el proceso de pruebas en general se utilizará como marco referencial el de TMMi (*Test Maturity Model Integration*) que propone una evaluación y mejora continua de las prácticas de pruebas de una organización. TMMi define 5 niveles de madurez de pruebas los mismos que consisten en:



Figura 27 Niveles de madurez de pruebas en TMMi (TMMi Foundation, 2021)

La rúbrica correspondiente es:

### **Nivel 1 – Inicial**

Pruebas desordenadas.

### **Nivel 2 – Gestionado**

Existe un enfoque claro sobre los métodos de prueba a realizar, estos varían de un proyecto a otro dentro de la organización. Las pruebas están basadas en estrategias y políticas, las prácticas de pruebas en este nivel son:

- Plan, monitoreo y control sobre las actividades de pruebas.
- Técnicas de diseño de pruebas definidas
- Control sobre la ejecución de pruebas.
- Ambiente de pruebas bien definido.

Esto proporciona una dirección para la planificación de la supervisión y el control, la tecnología de diseño de pruebas y cada proyecto puede controlar la ejecución de la prueba. En el nivel 2, también hay un entorno de prueba para el proyecto.

### **Nivel 3 – Definido**

Todos los proyectos siguen los mismos estándares y procedimientos en toda la organización. El nivel 2 aún está en progreso, el equipo ahora se está organizando, el plan de capacitación piloto está en progreso, las pruebas se han integrado en el ciclo de vida del desarrollo y se han integrado en todos los proyectos desde las primeras etapas de desarrollo. Se planifica y ejecutan pruebas no funcionales en todos los proyectos y se hacen revisiones continuamente.

### **Nivel 4 – Medido**

La medición de actividades y resultados se aplican cuidadosamente al comienzo de todos los proyectos para asegurarse de que el proyecto se encuentre lo más libre de defectos en cada etapa de desarrollo. Según la práctica de revisión presentada en el nivel 3, todos los proyectos utilizan la revisión avanzada.

### **Nivel 5 – Optimización**

En este nivel, se evalúan todas las actividades y resultados y se realizan actividades de optimización para asegurar una mejora continua para prevenir defectos y optimizar la calidad.

## 6.2.4 Proyecto PN04: Adopción de una cultura DevOps en la organización.

Tabla 24

Proyecto P0N4: Adopción de una cultura DevOps en la organización.

Proyecto PN04: Adopción de una cultura DevOps en la organización.	
Situación Actual y Principales Observaciones	Se evidencia un exceso de burocracia en los procesos, además falta de colaboración y cohesión entre los equipos y departamentos. Los colaboradores y proyectos siguen con el uso de metodologías tradicionales.
Actividades fundamentales	<ul style="list-style-type: none"> <li>• Contratar una empresa consultora externa.</li> <li>• Crear documentación sobre la capacitación de DevOps y socializarla al resto de áreas.</li> <li>• Iniciar con la adopción incremental.</li> <li>• Realizar retroalimentación usando el modelo de madurez.</li> </ul>
Plazo de implementación	2 meses – constantemente
Costos referenciales	Capacitaciones iniciales \$25.000

Esta iniciativa tiene como objetivo reducir la burocracia evidenciada en los procesos internos, mejorar la colaboración y la integración entre los departamentos de desarrollo y operaciones y cerrar las brechas existentes. DevOps como cultura, fortalecerá estas debilidades y será aplicada de manera incremental.

- **Contratar una empresa consultora externa**

Para poder realizar la adopción DevOps de una manera sencilla se va a contratar los servicios de una empresa consultora que es experta en el tema.

Con un gran portafolio de clientes internacionales y varias historias de éxito Mckinsey se posiciona como el aliado ideal para realizar el acompañamiento y guía de una adopción incremental exitosa.



Para iniciar la adopción de la cultura DevOps se dictará a un grupo de colaboradores una capacitación con las mejores prácticas realizadas por parte de la consultora.

- **Crear documentación para la capacitación de DevOps y socializarla al resto de áreas**

Con el objetivo de depender cada vez menos del acompañamiento y guía de la consultora y disminuir los gastos que eso implica se realizará la documentación de las charlas recibidas para que el resto de la organización reciba esa información dictada por el personal interno del Banco.

Al iniciar con estas charlas internas será necesaria aún la corrección de la consultora hasta que el equipo interno pueda afianzar y apropiarse los conceptos. La idea principal es que en cada charla dictada sea menos indispensable la guía de la consultora y el personal interno sea capaz de realizar estas charlas con gran eficiencia a las distintas áreas y personal nuevo de la institución

- **Iniciar con la Adopción Incremental**

Un enfoque incremental en lugar de un enfoque de adopción agresivo permite a la organización minimizar el riesgo y el costo de una adopción de DevOps y otorga el tiempo de aprendizaje para construir las habilidades y el impulso necesarios para lograr una implementación exitosa.

Es importante plantearse objetivos pequeños que permitan a la organización obtener la mayor cantidad de retroalimentación posible de modo que las lecciones aprendidas permitan plantearse objetivos cada vez más grandes hasta que se pueda obtener un proceso y ejecución del mismo equilibrados.

Pasos para la adopción incremental

- El pensamiento en sistemas se refiere a crear un proceso de desarrollo de software integrado usando herramientas especializadas que se

detallaran en la iniciativa de implementación de herramientas DevOps. Se automatizarán aquellos procesos manuales que puedan ser reemplazados por un proceso automático como son el despliegue del software en cada ambiente, la ejecución de las pruebas unitarias y pruebas de integración, el chequeo de la calidad del software y las pruebas automatizadas en cada proyecto o aplicación.

- Amplificar bucles de retroalimentación, cada versión del software será entregado al cliente lo más pronto posible para así poder obtener comentarios constructivos y mejorar el proceso de desarrollo y la codificación.
- Cultura de la experimentación y aprendizaje continuos, ya que existe una apertura a investigar nuevas soluciones y a mejorar los procesos continuamente.

## 6.2.5 Proyecto PA01: Usar herramientas de apoyo para DevOps

Tabla 25

Proyecto PA01: Implementación de herramientas de apoyo a DevOps.

Proyecto PA01: Implementación de herramientas de apoyo a DevOps.	
Situación Actual y Principales Observaciones	Existe un ciclo de desarrollo de software lento e ineficiente, además de varios procesos manuales que podrían ser automatizados. Se verifica la existencia código sucio, con vulnerabilidades y sin cobertura.
Actividades fundamentales	<ul style="list-style-type: none"> <li>• Usar Git versión Enterprise (50GB of Packages storage)</li> <li>• Usar Docker (Dockerfile estandar para todos los proyectos)</li> <li>• Crear los nuevos proyectos móviles con React-Native (Lenguaje Híbrido desarrollado por Facebook)</li> <li>• Instalar y configurar Jenkins en la nube (Jenkins on Ubuntu 14.04 LTS)</li> <li>• Adquirir y configurar SonarCloud versión Private(pago por líneas de código analizadas)</li> </ul>
Plazo de implementación	2 meses - mejoras constante
Costos referenciales	<ul style="list-style-type: none"> <li>• Licencia anual GitHub \$500</li> <li>• GCP pago por uso y por cantidad de ambientes aproximadamente \$80.000 anual</li> <li>• SonarQueue \$150 - \$2.000 al año depende de la cantidad de líneas analizadas</li> </ul>



Figura 28 Flujo DevOps (RRHHDigital, 2019)

Se instalarán los insumos necesarios para que el ciclo de desarrollo de software sea eficiente en cada uno de los pasos y para que el entregable salga lo más pronto posible a ambientes productivos.

Al implementar las herramientas adecuadas la fusión entre los roles y deberes nos ayudan a crear un ambiente ideal para temas de escalabilidad, confiabilidad y administración de la carga.

#### **Herramientas que se implementarán:**

- **Git**

Los pasos para la implementación de esta herramienta en la organización son:

- Configurar una cuenta tipo empresa en github.com.
- Ingresar el método de pago y elegir el plan que se adapte a la cantidad de repositorios a usar.
- Crear los repositorios correspondientes a los proyectos que existen actualmente en la institución.
- Crear una conexión SSH entre las computadoras locales de los desarrolladores y el repositorio remoto.
- Subir a cada repositorio de GitHub el código de su respectivo proyecto.

Cada vez que se cree un proyecto o servicio nuevo se deberá crear su respectivo repositorio en GitHub.

- **Google Cloud Platform**

Los pasos para implementar GCP en la organización son:

- Al ingresar a la consola de Google Cloud Platform se debe crear una cuenta para iniciar las configuraciones.
- Configurar el método de pago y configurar los presupuestos
- Crear una organización de GCP utilizando el hosting del dominio de la institución. Para esto GCP nos proporciona un archivo plano que deberá ser agregado a la configuración del hosting.
- Crear los roles con sus respectivos permisos en GCP
- Crear los usuarios de la organización y asignarles un rol dentro de la plataforma.
- Crear los proyectos existentes en el banco dentro de Google Cloud Platform. Cada proyecto manejará su presupuesto individualmente, por este motivo hay que configurar dicho presupuesto cada vez que se cree un proyecto.

- **Docker**

Para la implementación es necesario crear un archivo de configuración de Docker en cada servicio o aplicación creada. En este archivo se configura la imagen a usar y las dependencias del ambiente necesario para que funcione el servicio.

Para usar las configuraciones de Docker, Google Cloud Platform nos brinda un servicio de administración y escalabilidad de contenedores llamado Cloud Run.

Habilitar el servicio de Cloud Run en cada proyecto que use Docker, luego configurar Cloud Run para que identifique el DockerFile de los proyectos.

- **Inserción de React Native**

Para usar React-Native para el desarrollo de aplicaciones móviles es necesario que los desarrolladores front usen computadores especializados y con mayor potencia. De preferencia se recomienda el uso de computadores Mac ya que se integra de manera sencilla con el desarrollo en React-Native.

Si el presupuesto no permite la adquisición de portátiles Mac, una segunda opción sería la instalación de sistemas operativos tipo Unix en los computadores de los desarrolladores front end.

Los nuevos proyectos de innovación o proyectos que se van a migrar tanto en móvil como en web serán desarrollados en React y React-Native, con esto aseguramos mayor rapidez en la entrega de versiones.

- **Jenkins**

Jenkins es una herramienta gratuita de código abierto que se adapta a múltiples sistemas operativos, nosotros la instalaremos en un servidor local de la institución que tiene Fedora.

Para comenzar la configuración es necesario descargar la versión LTS de Jenkins que es la versión estable e instalarla en el servidor.

Luego de la instalación se debe configurar las integraciones con GitHub y SonarCloud para disparar el proceso de construcción al en el momento que el desarrollador integre su código a GitHub y medir la calidad y cobertura del código. (Hernández, 2017)

- **SonarCloud**

Para implementar SonarCloud se seguirán los siguientes pasos:

- Crear una cuenta en SonarCloud.io y configurar el método de pago.

- Se seleccionará la facturación tipo privada para poder analizar 2 millones de líneas, esto debido a la gran cantidad de proyectos del banco.
- Integrar SonarCloud con GitHub para analizar el código generado en los repositorios.
- Modificar las reglas existentes en SonarCloud para que se adapten a los estándares de calidad del Banco.

### 6.2.6 Proyecto PA02: Crear una célula dedicada a la implementación de un CRM institucional

Tabla 26

Proyecto PA02: Crear una célula dedicada a la implementación de un CRM institucional.

Proyecto PA02: Crear una célula dedicada a la implementación de un CRM institucional.	
Situación Actual y Principales Observaciones	Se verifica que los datos y la información se encuentran en varios repositorios, bases de datos y archivos planos de manera distribuida por toda la institución
Actividades fundamentales	<ul style="list-style-type: none"> <li>• Identificar los roles necesarios para la célula.</li> <li>• Realizar la inducción de la célula.</li> <li>• Presentación de la célula.</li> </ul>
Plazo de implementación	6 meses – 1 año
Costos referenciales	Sueldos \$300.000 anuales

- Identificar los roles necesarios para la célula

Luego de un primer análisis de necesidades de la célula se identifica la necesidad de los siguientes roles.

- *Product Owner*: será reclutado desde el área de negocio.
- *User Experience/User Interface*: existen perfiles similares en otras áreas, por lo tanto, se procede con entrevistas internas para reclutamiento.
- *Desarrolladores Front End y Back End*: perfiles técnicos que no existen en la organización por tal motivo se procede a contratarlos.

- Tester: el perfil no existe en la organización por tal motivo se procede a contratarlo.
- Arquitecto de soluciones: este perfil existe en el área de tecnología, de modo que se procede a entrevistas internas y de no aprobar se procede a contrato.
- Ingeniero DevOps: perfil que no existe en la organización por tal motivo se procede a contratarlo.
- Scrum master: se realiza reclutamiento interno y de no encontrarlo se contrata uno.

- **Realizar la inducción de la célula**

Una vez reclutados los perfiles necesarios para la célula, se procede a realizar las capacitaciones respectivas con respecto al marco de trabajo Scrum y la cultura DevOps.

Inducción acerca de los objetivos y propósito de la célula. Presentación de la situación actual del problema e identificación de posibles soluciones a nivel de negocio y su implementación tecnológica.

- **Presentación de la célula**

La célula deberá ser presentada públicamente a la organización y en especial a los gerentes. Para esta presentación la célula deberá defender su razón de ser, sus objetivos como célula a mediano y largo plazo, deberá tener identificado plenamente el problema y situación actual además de sus posibles soluciones.

Deben mostrar un road map y un backlog inicial, también deberán tener claros los MVP's es decir, el mínimo producto que pueda salir al usuario para comenzar a recibir los comentarios con retroalimentación y sus fechas de lanzamientos.



## 6.2.7 Proyecto PA03: Adquirir licencias de Jira para todas las personas involucradas en el nuevo marco de trabajo.

Tabla 27

Proyecto PA03: Adquirir licencias de Jira para todas las personas involucradas en el nuevo marco de trabajo.

Proyecto PA03: Adquirir licencias de Jira para todas las personas involucradas en el nuevo marco de trabajo.	
Situación Actual y Principales Observaciones	Existe un ciclo de desarrollo de software lento e ineficiente, además de varios procesos manuales que podrían ser automatizados. Los colaboradores y proyectos siguen con el uso de metodologías tradicionales.
Actividades fundamentales	<ul style="list-style-type: none"> <li>• Comprar 50 licencias de Jira Software para los colaboradores de la fábrica (incluye uso del repositorio Confluence).</li> <li>• Configuración de tableros en Jira y creación de proyectos y asignación de personas.</li> <li>• Creación de espacios de repositorio en Confluence.</li> <li>• Capacitación del uso de la herramienta Jira.</li> </ul>
Plazo de implementación	Menor a 3 meses
Costos referenciales	\$8.700 anuales

Esta iniciativa se unifica con el proyecto PN01: Implementación marco de trabajo Scrum; debido a que Jira es indispensable para trabajar con Scrum ya que une a los equipos y les permite trabajar en torno a objetivos comunes, cuenta con opciones que promueven realizar entregas iterativas e incrementales. (Atlassian, 2021)

A través de esta herramienta colaborativa, el equipo logra poner en práctica algunos de los valores de Scrum, por ejemplo: la transparencia, proporciona tableros ágiles que logran mantener sincronizado al equipo y transparentar el trabajo a todos sobre qué está pendiente por hacer, en progreso de desarrollo, en progreso de pruebas y aquel que se encuentra ya finalizado.

## 6.2.8 Proyecto PI01: Usar los servicios de infraestructura de Google Cloud Platform.

Tabla 28  
Proyecto PI01: Adopción de una infraestructura Cloud Computing.

Proyecto PI01: Adopción de una infraestructura Cloud Computing.	
Situación Actual y Principales Observaciones	Se verifica deficiencias en la escalabilidad de los sistemas ya que es necesaria la intervención humana en el proceso.
Actividades fundamentales	<ul style="list-style-type: none"> <li>• Configurar la cuenta que administrará la plataforma.</li> <li>• Crear los proyectos existentes.</li> <li>• Configurar método del pago y presupuestos por proyecto.</li> <li>• Creación de una organización atada al hosting del Banco.</li> </ul>
Plazo de implementación	2 meses – mejora constante
Costos referenciales	\$80.000 por año - pago por uso

Los pasos para implementar Google Cloud Platform en la organización son:

- Al ingresar a la consola de GCP se debe crear una cuenta para iniciar las configuraciones.
- Configurar el método de pago y configurar los presupuestos
- Crear una organización de GCP utilizando el hosting del dominio de la institución. Para esto GCP nos proporciona un archivo plano que deberá ser agregado a la configuración del hosting.
- Crear los roles con sus respectivos permisos en GCP
- Crear los usuarios de la organización y asignarles un rol dentro de la plataforma.
- Crear los proyectos existentes en el banco dentro de Google Cloud Platform. Cada proyecto manejará su presupuesto individualmente, por este motivo hay que configurar dicho presupuesto cada vez que se cree un proyecto.

- Cada célula de trabajo tiene sus propias necesidades informáticas primarias por lo que GCP ofrece varias soluciones disponibles como IaaS con compute engine para ejecutar máquinas virtuales sin administrar la infraestructura, PaaS con app engine el cual proporciona un ambiente completo de construcción y escalabilidad permitiendo a los desarrolladores concentrarse 100% en su código.

Para poder usar los servicios de GCP primero se debe crear una cuenta en la consola y luego agregar un método de pago.

Una vez autenticado se necesita crear una organización en la consola para poder crear los proyectos dentro de ella. Para crear la organización necesita integrar la consola de GCP con su proveedor de hosting mediante un archivo de configuración proporcionado por GCP, esto proporciona un dominio en el cual se puede utilizar el email y publicar las APIs.

Existe la opción de crear presupuestos para controlar el nivel de gasto de cada proyecto y que la facturación no sea muy elevada. Con este sencillo proceso queda activa toda la suite de servicios de google que se puede utilizar de manera independiente en cada proyecto.

#### Características de la iniciativa

- Identificarse en la consola de GCP.
- Organización crearla usando el hosting que use el Banco para integrar el dominio.
- Proyectos cada célula de trabajo es considerada un proyecto nuevo dentro de la consola de GCP y manejará su facturación y presupuesto de manera independiente.

## 6.2.9 Proyecto PI02: Usar ApiGee para la gestión de APIs

Tabla 29

Proyecto PI02: Uso de API Gateway con distintos ambientes de desarrollo.

Proyecto PI02: Uso de API Gateway con distintos ambientes de desarrollo.	
Situación Actual y Principales Observaciones	Se verifica que la adaptabilidad de los servicios no es buena ya que para un cambio pequeño es necesario volver a construir la solución lo cual menora la disponibilidad de las aplicaciones.
Actividades fundamentales	<ul style="list-style-type: none"> <li>• Crear nuevo ambiente de desarrollo</li> <li>• Implementar un API Gateway para las nuevas soluciones.</li> <li>• Integrar ApiGee con Google Cloud Platform</li> </ul>
Plazo de implementación	2 meses
Costos referenciales	<ul style="list-style-type: none"> <li>• Licencia ApiGee \$6.000 anual</li> <li>• Nuevo ambiente \$15.000 anual</li> </ul>

- **Crear nuevo ambiente de desarrollo**

En el proceso de adopción de DevOps es requerido que el código pase por diferentes ambientes de pruebas, construcción y control de errores. Existe una gran importancia con respecto a la retroalimentación en cada uno de los ambientes de desarrollo, por lo cual es necesario crear un nuevo ambiente dirigido a obtener muchos comentarios constructivos por parte del equipo de calidad y otros desarrolladores con el objetivo de mejorar el código y encontrar errores en etapas tempranas.

Debido a que cada ambiente de desarrollo tiene costo en la nube por el uso de los servicios disponibles es recomendable crear únicamente un ambiente adicional ya que proporciona la retroalimentación y los filtros suficientes para tener un ciclo de desarrollo estable y escalable.

- **Implementación del API Gateway**

El objetivo de implementar un API Gateway es incrementar la disponibilidad de los servicios y las aplicaciones ya que no es necesario reconstruirlas con cada cambio que no implique lógica en el código. Los cambios pueden ser datos de autenticación, transformaciones, administración de cuotas, caches entre otras.

Apigee es una plataforma que nos permite gestionar el ciclo de vida de las APIs usando un conjunto de políticas que nos permite proteger, desplegar, escalar y supervisar estas. Además, proporciona un portal de desarrolladores que permite monetizar las APIs con disponibilidad para medir el rendimiento y uso.

## **7. Plan de Migración**

El plan de migración presenta la priorización de las iniciativas propuestas en torno al impacto y esfuerzo de cada una de ellas, finalmente se determina el orden de implementación separado en 4 fases.

### **7.1 Análisis de impacto**

En la siguiente tabla se analiza el impacto de cada iniciativa, relacionándolas con los objetivos de la fábrica de software.

La rúbrica utilizada para medir el impacto de cada iniciativa es: Bajo (0 – 0.7), Medio (0.8 – 1.4), Alto (1.5 - 2).

Tabla 30  
Análisis de impacto de iniciativas

No	Arquitectura	Id	Iniciativa	Habilitante	Objetivos						Valoración Cuantitativa	Impacto	
					19%	18%	14%	14%	13%	10%			10%
					Sostenibilidad de Servicios TI	Mejorar la experiencia de usuario y tiempo en agencias	Eficiencia de procesos, integración y TI, gestión de conocimiento y efectividad laboral	Desarrollos estables y sin errores	Servicios de TI ágiles y adaptables al cambio	Versionamiento de código y repositorios centralizados	Automatización de procesos y mejora en tiempos de entrega		
1	Negocio	PN01	Implementación marco de trabajo Scrum.	SI								1.6	Alto
2	Negocio	PN02	Talleres de capacitación de nuevos roles y responsabilidades.	SI								1.3	Medio
3	Negocio	PN03	Definir una metodología ágil de pruebas (objetivo automatización).									1.2	Medio
4	Negocio	PN04	Adopción de una cultura DevOps en la organización.	SI								1.5	Alto
5	Aplicaciones / Datos	PA01	Implementación de herramientas de apoyo a DevOps.									1.3	Medio
6	Aplicaciones / Datos	PA02	Crear una célula dedicada a la creación de un CRM institucional.									0.7	Bajo
7	Infraestructura	PI01	Adopción de una infraestructura Cloud Computing.									1.0	Medio
8	Infraestructura	PI02	Uso de Api Gateway con distintos ambientes de desarrollo.									0.6	Bajo

Escala de Impacto	
Bajo: entre 0-0.7	
Medio: entre 0.7-1.4	
Alto: entre 1.4-2	

## 7.2 Análisis de esfuerzo

Para el análisis de esfuerzo se consideraron tres criterios de esfuerzo: recursos económicos, complejidad y capacidad de TI. Con el resultado obtenido se lo categorizó en tres niveles de esfuerzo basados en la siguiente rúbrica: Bajo (1 – 1.7), Medio (1.7 – 2.4), Alto (2.4 – 3).

Tabla 31  
Análisis de esfuerzo de iniciativas

No	Área	Id	Iniciativa	Criterios Esfuerzo			Suma Ponderada	Esfuerzo
				Recursos Económicos	Complejidad	Capacidad TI		
1	Negocio	PN01	Implementación marco de trabajo Scrum.	2	2	3	2.33	Medio
2	Negocio	PN02	Talleres de capacitación de nuevos roles y responsabilidades.	1	1	3	1.67	Bajo
3	Negocio	PN03	Definir una metodología ágil de pruebas (objetivo automatización).	1	1	2	1.33	Bajo
4	Negocio	PN04	Adopción de una cultura DevOps en la organización.	1	3	3	2.33	Medio
5	Aplicaciones / Datos	PA01	Implementación de herramientas de apoyo a DevOps.	3	3	2	2.67	Alto
6	Aplicaciones / Datos	PA02	Crear una célula dedicada a la creación de un CRM institucional.	2	2	3	2.33	Medio
7	Infraestructura	PI01	infraestructura Cloud Computing.	3	3	2	2.67	Alto
8	Infraestructura	PI02	Uso de Api Gateway con distintos ambientes de desarrollo.	1	2	2	1.67	Bajo

Escala de Esfuerzo
Bajo: entre 1-1.7
Medio: entre 1.7-2.4
Alto: entre 2.4-3

Costos
Bajo: entre \$150-\$30.000
Medio: entre \$30.001-\$70.000
Alto: \$70.001 en adelante

### 7.3 Fases

Se combinaron los resultados obtenidos en el análisis de impacto y el análisis de esfuerzo y se dividieron las iniciativas en fases. Estas fases ayudarán más adelante a crear el cronograma de actividades para la implementación.

Tabla 32  
Fases de implementación

No	Área	Id	Iniciativa	Habilitante	Impacto	Esfuerzo	Prioridad	Fase
1	Negocio	PN01	Implementación marco de trabajo Scrum.	SI	Alto	Medio	SI	1
2	Negocio	PN02	Talleres de capacitación de nuevos roles y responsabilidades.	SI	Medio	Bajo		2
3	Negocio	PN03	Definir una metodología ágil de pruebas (objetivo automatización).		Medio	Bajo		3
4	Negocio	PN04	Adopción de una cultura DevOps en la organización.	SI	Alto	Medio	SI	1
5	Aplicaciones / Datos	PA01	Implementación de herramientas de apoyo a DevOps.		Medio	Alto	SI	2
6	Aplicaciones / Datos	PA02	Crear una célula dedicada a la creación de un CRM institucional.		Bajo	Medio		4
7	Infraestructura	PI01	Adopción de una infraestructura Cloud Computing.		Medio	Alto	SI	2
8	Infraestructura	PI02	Uso de Api Gateway con distintos ambientes de desarrollo.		Bajo	Bajo		4

## 7.4 Análisis de dependencias

Existen dependencias entre las iniciativas, como muestra la siguiente tabla:

Tabla 33  
Análisis de dependencias entre iniciativas

No	Iniciativas	Implementación marco de trabajo Scrum.	Talleres de capacitación de nuevos roles y responsabilidades.	Definir una metodología ágil de pruebas (objetivo automatización).	Adopción de una cultura DevOps en la organización.	Implementación de herramientas de apoyo a DevOps.	Crear una célula dedicada a la creación de un CRM institucional.	Adopción de una infraestructura Cloud Computing.	Uso de Api Gateway con distintos ambientes de desarrollo.
1	Implementación marco de trabajo Scrum.		X						
2	Talleres de capacitación de nuevos roles y responsabilidades.	X							
3	Definir una metodología ágil de pruebas (objetivo automatización).				X	X			
4	Adopción de una cultura DevOps en la organización.	X	X						
5	Implementación de herramientas de apoyo a DevOps.				X			X	
6	Crear una célula dedicada a la creación de un CRM institucional.	X	X						
7	Adopción de una infraestructura Cloud Computing.				X				
8	Uso de Api Gateway con distintos ambientes de desarrollo.					X		X	



## 7.5 Roadmap

En base al orden de implementación analizado en el capítulo anterior, se establece un *roadmap* en el que se tiene una visión general de la secuencia del futuro desarrollo de cada iniciativa y más adelante permitirá informar a los *stakeholders* acerca del progreso.

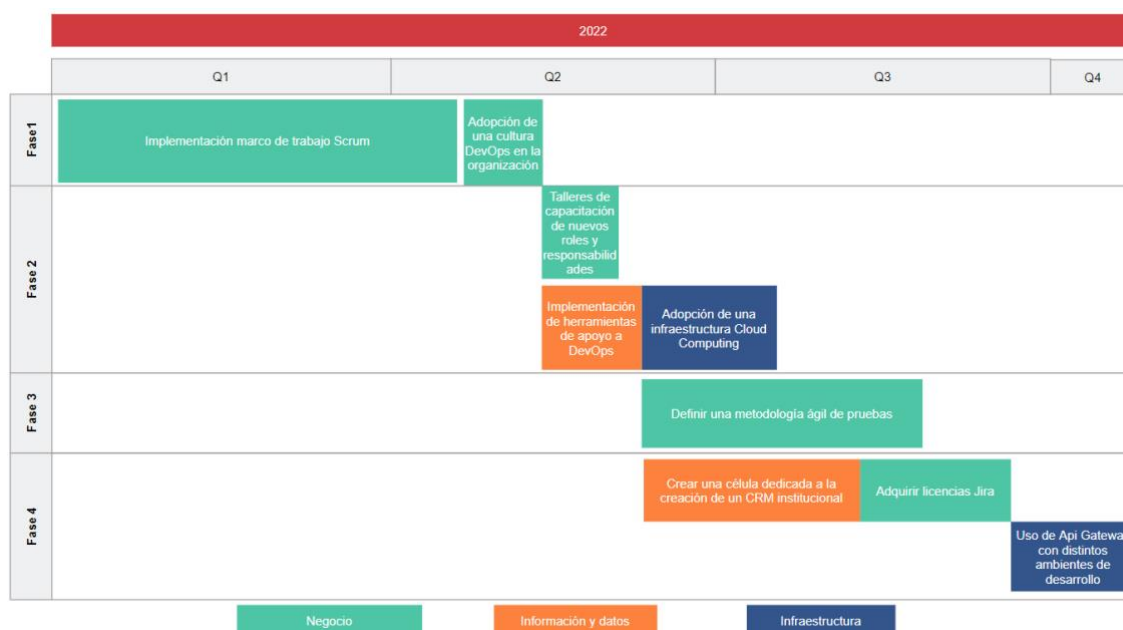


Figura 29 Roadmap de iniciativas

## 7.6 Cronograma de actividades

Tabla 34  
Cronograma para la implementación marco de trabajo ágil Scrum

Id	Nombre	Duración	Comienzo	Fin
1	Contratación de Agile coach	6 días	02/01/2022	11/01/2022
2	Taller sobre Scrum al área de tecnología	10 días	12/01/2022	26/01/2022
3	Taller sobre Scrum al resto de la organización	15 días	27/01/2022	17/02/2022
4	Taller definición de OKRs de la fábrica de software	5 días	18/02/2022	25/02/2022
5	Taller definición de OKRs en cada célula ágil, alineados a los OKRs de la fábrica	5 días	28/02/2022	07/03/2022
6	Taller de User story mapping en cada célula	10 días	08/03/2022	22/03/2022
7	Taller de priorización del backlog de cada célula	20 días	23/03/2022	20/04/2022
8	Taller de Jira (creación de tableros y dinámica ágil de trabajo)	8 días	21/04/2022	02/05/2022

Tabla 35  
Cronograma para la adopción de una cultura DevOps en la organización.

Id	Nombre	Duración	Comienzo	Fin
9	Capacitación DevOps al área de tecnología	5 días	03/05/2022	10/05/2022
10	Inducción a nuevos colaboradores	15 días	11/05/2022	25/05/2022
11	Creación de célula(s) piloto	2 días	26/05/2022	27/05/2022
12	Presentación de células a gerencias	1 día	30/05/2022	30/05/2022
13	Ejecución del Modelo de madurez	4 días	31/05/2022	06/06/2022

Tabla 36  
Cronograma para los talleres de capacitación de nuevos roles y responsabilidades.

Id	Nombre	Duración	Comienzo	Fin
14	Taller de nuevo roles al área de tecnología - Capacitación - Definición de roles a cada colaborador	5 días	07/06/2022	13/06/2022
15	Taller de nuevo roles al resto de la organización	3 días	14/06/2022	16/06/2022

Tabla 37  
Cronograma para la implementación de herramientas de apoyo a DevOps

Id	Nombre	Duración	Comienzo	Fin
16	Configuración de cuenta en Github	1 día	07/06/2022	07/06/2022
17	Subir repositorios actuales a Github	1 día	08/06/2022	08/06/2022
18	Crear un docker file genérico para los nuevos servicios	2 días	09/06/2022	10/06/2022
19	Instalar Jenkins	1 día	13/06/2022	13/06/2022
20	Integrar Jenkins con Github	2 días	14/06/2022	15/06/2022
21	Integrar Jenkins con SonarCloud	2 días	16/06/2022	17/06/2022
22	Configuración de despliegue en la nube	3 días	20/06/2022	22/06/2022
23	Configuración de cuenta en SonarCloud	1 día	23/06/2022	23/06/2022
24	Configuración de reglas de SonarCloud	1 día	24/06/2022	24/06/2022
25	Integración de SonarCloud con Github	2 días	27/06/2022	29/06/2022

Tabla 38  
Cronograma para la adopción de una infraestructura Cloud Computing.

Id	Nombre	Duración	Comienzo	Fin
26	Configuración de la cuenta en GCP	2 días	30/06/2022	01/07/2022
27	Creación de organización en GCP con hosting de la institución	3 días	04/07/2022	06/07/2022
28	Creación de los proyectos actuales	5 días	07/07/2022	14/07/2022

Tabla 39  
Cronograma para definir una metodología ágil de pruebas.

Id	Nombre	Duración	Comienzo	Fin
29	Contratación de líder de control de calidad	6 días	30/06/2022	08/07/2022
30	Contratar perfiles de tester (pruebas manuales y automáticas)	10 días	11/07/2022	25/07/2022
31	Definición y socialización de proceso ágil de pruebas. Publicación documentación en Confluence.	10 días	26/07/2022	09/08/2022
32	Asignar los tester a las células	3 días	10/08/2022	14/08/2022
33	Definición formato de plan de pruebas, entregables y ubicación en repositorio	8 días	15/08/2022	26/08/2022

Tabla 40  
Cronograma para crear una célula dedicada a la creación de un CRM institucional.

Id	Nombre	Duración	Comienzo	Fin
34	Creación de la célula CRM	1 día	30/06/2022	30/06/2022
35	Elección del Product Owner	2 días	01/07/2022	04/07/2022
36	Inducción del equipo	5 días	05/07/2022	12/07/2022
37	Identificar fuentes de datos	15 días	13/07/2022	03/08/2022
38	Presentación de la célula y MVP	1 día	04/08/2022	04/08/2022

Tabla 41  
Cronograma para adquirir licencias de Jira para todas las personas involucradas en el nuevo marco de trabajo y capacitación.

Id	Nombre	Duración	Comienzo	Fin
39	Negociar y adquirir 50 licencias de Jira	8 días	05/08/2022	16/08/2022
40	Configuración de tableros en Jira	15 días	17/08/2022	07/09/2022
41	Capacitación del uso de la herramienta	3 días	08/09/2022	12/09/2022
42	Creación de espacios de repositorios en Confluence	5 días	13/09/2022	20/09/2022

Tabla 42  
Cronograma para usar API Gateway con distintos ambientes de desarrollo

Id	Nombre	Duración	Comienzo	Fin
43	Configuración de nuevo ambiente de desarrollo	5 días	23/09/2022	29/09/2022
44	Configuración de Api Gee	5 días	30/09/2022	07/10/2022
45	Integración de Api Gee con GCP	3 días	10/10/2022	12/10/2022

## 7.7 Conclusiones

1. El no involucrar al cliente durante todo el proceso de desarrollo de software, causa que se implementen funcionalidades que no son utilizadas por el usuario.
2. La reputación de la institución financiera depende fuertemente de sus servicios digitales, al ser estos de baja calidad la popularidad del Banco se ve afectada y existen pérdidas económicas y de clientes.
3. La cultura tradicional de la institución se encuentra fuertemente arraigada en sus procesos y colaboradores, razón por la cual los proyectos no se terminan o tardan demasiado en llegar a los clientes.

## 7.8 Recomendaciones

1. Realizar un cambio cultural de manera incremental, comenzando con los departamentos que generen una mayor satisfacción del cliente como es el de tecnología.
2. Motivar y ayudar al personal interno a capacitarse en el uso e implementación de las nuevas herramientas y metodologías que ayudaran a generar procesos más ágiles, de esta manera se podrá cubrir la demanda de roles que generará la transformación digital.
3. Al existir células de trabajo independientes, el conocimiento y prácticas no son socializados, por lo que se recomienda la creación de reuniones por rol, llamadas *chapters*.

## Referencias

- Acerca de SFIA. (s. f.). [Página]. SFIA. Recuperado 26 de abril de 2021, de <https://sfia-online.org/es/about-sfia>
- Deloitte, E. (2020). *Introducción al modelo «agile» de Spotify*. <https://www2.deloitte.com/es/es/pages/technology/articles/introduccion-modelo-agile-spotify.html>
- Indra, C. (2017). *Indra asegura la mayor fiabilidad de su software con la certificación TMMi nivel 3 de siete centros de producción en cinco países* | *indra*. <https://www.indracompany.com/es/noticia/indra-asegura-fiabilidad-software-certificacion-tmmi-nivel-3-centros-produccion-paises>
- Pablo Marichal. (2016). *Apuntes para entender esta metodología ágil*. <https://medium.com/@dotpablo/serie-scrum-apuntes-para-entender-esta-metodolog%C3%ADa-%C3%A1gil-1-c054f4f23b2f>
- Patton, J., & Economy, P. (2014). *User Story Mapping: Discover the Whole Story, Build the Right Product*. O'Reilly Media, Inc.
- Rising, L., & Janoff, N. S. (2000). The Scrum software development process for small teams. *IEEE Software*, 17(4), 26-32. <https://doi.org/10.1109/52.854065>
- Singh, J. (2019, julio 16). *DevOps without DevOps tools*. Medium. <https://faun.pub/devops-without-devops-tools-3f1deb451b1c>

Atlassian. (2021). *Atlassian*. Obtenido de <https://www.atlassian.com/es/software/jira/features/scrum-boards>

Autentia. (2020). *DEVOPS Cultura de entrega de valor*. Madrid.

Foundation, T. (2021). *TMMi Foundation*. Obtenido de <https://www.tmmi.org/>

Hernández, P. B. (03 de 10 de 2017). Automatización de despliegues mediante VMware, Jenkins y Robot Framework. Universitat Politècnica de València.

Hudaib, A. (2018). Requirements Prioritization Techniques Comparison. *Modern Applied Science*.

iSQI. (2021). *iSQI*. Obtenido de <https://isqi.org/es/36-tmmi-professional.html>

Josey, A. (2011). *TOGAF Version 9.1 – A Pocket Guide*.

Maximini, D. (2015). *The Scrum Culture*. Suiza.

Mulder. (2017). *Toolshero*. Obtenido de <https://www.toolshero.es/administracion/metodo-moscow/>

Piao, Y. C. (2021). Distributed Architecture for an Integrated Development Environment, Large Trace Analysis, and Visualization. *Sensors*.

RRHHDigital. (28 de 05 de 2019). *RRHHDigital*. Obtenido de [http://www.rrhhdigital.com/secciones/tecnologia-e-innovacion/136859/Que-es-DevOps-y-que-debes-saber-para-convertirte-en-ello?target=\\_self](http://www.rrhhdigital.com/secciones/tecnologia-e-innovacion/136859/Que-es-DevOps-y-que-debes-saber-para-convertirte-en-ello?target=_self)

SFIA. (2021). *The global skills and competency framework for the digital world.*

Obtenido de <https://sfia-online.org/>

Sutherland, K. S. (2020). *La Guía Scrum. La Guía Definitiva de Scrum: Las Reglas del Juego.*

Zapata, C. M. (2021). Comparación de las características de algunas herramientas de software para pruebas de carga. *Revista Avances en Sistemas e Informática.*

