

**MANUAL PARA EL DESARROLLO E IMPLEMENTACIÓN DE SOLUCIONES INFORMÁTICAS**

La información contenida en este documento

es confidencial y de propiedad de la Cooperativa de

“Ahorro y Crédito Vicentina Manuel Esteban Godoy Ortega” Ltda.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ELABORADO** | **REVISADO** | | | | |
|  |  |  | |  |  |
| Julio Guarderas Maldonado  **JEFE DE DESARROLLO DE SISTEMAS** | Luis Montaño Montaño  **DIRECTOR DE TECNOLOGÍA DE LA INFORMACIÓN** | Lenin Naranjo Piedra  DIRECTOR **DE GESTIÓN Y DESARROLLO** | | Arelis Cabrera Jiménez  **JEFA DE LA UNIDAD DE RIESGOS** | Juan Bustamante Cueva  **PRESIDENTE DEL COMITÉ DE TECNOLOGÍA DE LA INFORMACIÓN** |
| **APROBADO** | | | | | |
| El presente documento fue aprobado por el Consejo de Administración en sesiones del \_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ según consta en actas números \_\_, \_\_ y \_\_ respectivamente | | | | | |
|  | | |  | | |
| Galo Galindo André  **PRESIDENTE** | | | Ketty León Veintimilla  **SECRETARIA** | | |

**VERSIÓN DEL DOCUMENTO Y CONTROL DE CAMBIOS**

**VERSIÓN 01-2024**

**Fecha de la última actualización:** XX de enero de 2024

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **DETALLE DE CAMBIOS** | | | | |
| **DETALLE DEL CAMBIO REALIZADO** | **APROBADO POR** | **DOCUMENTO SOPORTE** | **FECHA DE APROBACIÓN** | **VERSIÓN** |
| Se actualizó todo el documento, siendo los cambios generales los siguientes:   * Actualización de las responsabilidades; * Actualización de las políticas; * Actualización de los cargos de acuerdo con la estructura organizacional de la Cooperativa; * Actualización de flujogramas y procedimientos de acuerdo con lo establecido en el Manual de Procesos; * Modificación del formato de presentación del manual de acuerdo con la nueva plantilla detallada en el Manual de Procesos; * Actualización del glosario de términos; * Actualización de los Anexos: * Incorporación de Anexos; y * Actualización de los logos institucionales en todos los Anexos. | Consejo de **Administración** |  |  | VERSIÓN 01 - 2020 |
| Se actualizó todo el documento, siendo los cambios generales los siguientes:   * Actualización de las responsabilidades; * Actualización de políticas; * Actualización de los cargos de acuerdo con la estructura organizacional de la Cooperativa; * Actualización de flujogramas y procedimientos de acuerdo con lo establecido en el Manual de Procesos; * Modificación del formato de presentación del manual de acuerdo con la nueva plantilla detallada en el Manual de Procesos; * Actualización del glosario de términos; * Actualización de los Anexos: * Incorporación de Anexos; * Actualización de los logos institucionales en todos los Anexos; y, * Se incluyeron aspectos de agilismo, DEVOPS y de seguridad de la información. | Consejo de **Administración** |  |  | VERSIÓN 01 - 2024 |

[1. GLOSARIO DE TÉRMINOS 4](#_Toc427001414)

[2. ALCANCE 7](#_Toc318020301)

[3. RESPONSABILIDADES 8](#_Toc1853718410)

[4. POLÍTICAS 13](#_Toc1232986088)

[4.1 GENERALES 13](#_Toc1527439786)

[4.2 DE LA GESTIÓN DE PROYECTOS DE DESARROLLO DE SOLUCIONES INFORMÁTICAS 13](#_Toc1322908426)

[4.3 ETAPAS DEL DESARROLLO DE LAS SOLUCIONES INFORMÁTICAS 14](#_Toc701662983)

[4.3.1 ANÁLISIS 14](#_Toc1055469472)

[4.3.2 DISEÑO 15](#_Toc143643035)

[4.3.3 DESARROLLO 15](#_Toc649694646)

[4.3.4 PRUEBAS 16](#_Toc1499734669)

[4.3.5 IMPLEMENTACIÓN 17](#_Toc374671702)

[4.4 DOCUMENTACIÓN 18](#_Toc1049731518)

[4.5 DE LA CONTRATACIÓN DE SERVICIO DE DESARROLLO EXTERNO 18](#_Toc1546719345)

[5. DISPOSICIONES GENERALES 18](#_Toc312348789)

[6. DISPOSICIONES TRANSITORIAS 19](#_Toc1246402625)

[7. NORMATIVA INTERNA RELACIONADA 19](#_Toc1724187060)

[8. DIAGRAMAS DE PROCESO 19](#_Toc667312103)

[8.1 Análisis de requerimientos y factibilidades de soluciones informáticas 20](#_Toc1579002484)

[8.2 Planificación técnica inicial 21](#_Toc1000598626)

[8.3 Planificación de Sprint 22](#_Toc1005502216)

[8.4 Diseño de soluciones informáticas 23](#_Toc2145544895)

[8.5 Desarrollo de soluciones informáticas 26](#_Toc946086912)

[8.6 Pruebas Técnicas 26](#_Toc79658505)

[8.7 Pruebas de Usuario 27](#_Toc1086571615)

[8.8 Puesta en Producción 29](#_Toc1156392008)

[8.9 Plan de entrenamiento a los usuarios 29](#_Toc1599885160)

[9. PROCEDIMIENTO 30](#_Toc1719594734)

[9.1 Análisis de requerimientos y factibilidades de soluciones informáticas 31](#_Toc1979322150)

[9.2 Planificación técnica inicial 33](#_Toc1050392188)

[9.3 Planificación de Sprints 34](#_Toc1366856283)

[9.4 Diseño de soluciones informáticas 35](#_Toc244050112)

[9.5 Desarrollo de soluciones informáticas. 37](#_Toc143237689)

[9.6 Pruebas Técnicas 38](#_Toc1397163111)

[9.7 Pruebas de Usuario 41](#_Toc198333017)

[9.8 Puesta en Producción de las Soluciones Informáticas. 42](#_Toc1545835621)

[9.9 Plan de entrenamiento a los usuarios 44](#_Toc1945840250)

[10. ANEXOS 45](#_Toc237820127)

[ANEXO #1: Herramientas para el análisis 45](#_Toc1427705932)

[ANEXO #2: Herramientas de diseño 46](#_Toc1425526562)

[ANEXO #3: Herramientas de desarrollo 49](#_Toc1238752951)

[ANEXO #4: Herramienta para pruebas 56](#_Toc168130786)

[ANEXO #5: Herramientas de documentación 60](#_Toc1786163581)

[ANEXO #6: Arquitectura de desarrollo de software 63](#_Toc1644321107)

[ANEXO #7: Estándares de programación y base de datos 66](#_Toc1832173490)

[ANEXO #8: Informe de entrenamiento 75](#_Toc295737795)

[ANEXO #9: Desarrollo de software seguro 75](#_Toc1779077576)

[ANEXO #10: Requerimiento a la Dirección de Tecnología de la Información 78](#_Toc1063576364)

[ANEXO #11: Cronograma de trabajo 79](#_Toc340180713)

[ANEXO #12: Prototipado con Penpot 80](#_Toc580664737)

[ANEXO #13: Lineamientos del Proyecto 82](#_Toc495293116)

[ANEXO #14: Plan de Pruebas 84](#_Toc801766568)

[ANEXO #15: Bitácora de errores 86](#_Toc839784775)

[ANEXO #16: Solicitud de ejecución de pruebas técnicas 87](#_Toc520963005)

[ANEXO #17: Incidentes del usuario 89](#_Toc172928382)

[ANEXO #18: Registro de puesta en producción 90](#_Toc1543750256)

[ANEXO #19: Especificación de requerimientos 91](#_Toc1104988010)

[ANEXO #20: Estilo de programación en pares 95](#_Toc857328132)

[ANEXO #21: Planificación del sprint 96](#_Toc633161577)

[ANEXO #22: Kanban 98](#_Toc1756235780)

[ANEXO #23: Seguridad en el desarrollo de Software. 102](#_Toc1781165191)

[ANEXO #24: Registro de Logs 107](#_Toc781915666)

1. GLOSARIO DE TÉRMINOS

***Ambiente de desarrollo****:* son todos los servidores que tienen características similares a los servidores de producción y que sirven para hacer el desarrollo de la aplicación, su uso es directamente por el Desarrollador de Sistemas.

***Ambiente de producción****:* son todos los servidores que sirven para publicar la solución informática en funcionamiento real. Es el ambiente donde se registran las transacciones reales por los usuarios.

***Ambiente de pruebas****:* son todos los servidores con características similares a los servidores de producción y que sirven para hacer el control de calidad de la solución informática. Se la usa por lo general por el analista de sistemas informáticos y el analista de control de cambios.

***Analista de sistemas informáticos:*** persona encargada de realizar el levantamiento y depuración de los requerimientos, creación de historias de usuario y prototipos, ejecución de pruebas técnicas.

**Célula de trabajo:** es un grupo de personas que trabajan en conjunto de manera colaborativa y autónoma, para lograr un objetivo específico dentro de una organización.

***Control de calidad:*** es el conjunto de los mecanismos, acciones y herramientas utilizadas para detectar la presencia de errores. ​ Su función principal es asegurar que la solución informática cumpla con los requisitos mínimos de calidad.

***Desarrollo compartido:*** indica que la solución informática va a ser desarrollada por el departamento de desarrollo de sistemas de la Cooperativa en conjunto con una empresa externa de desarrollo de software.

***Desarrollo de soluciones informáticas:*** Se refiere al proceso de codificación de las soluciones informáticas.

***Desarrollo externo:*** indica que la solución informática va a ser desarrollada por una empresa de desarrollo externa al departamento de desarrollo de sistemas de la Cooperativa.

***Desarrollo interno:*** indica que la solución informática va a ser desarrollada por el departamento de desarrollo de sistemas de la Cooperativa.

***Dirección IP:*** Cadena numérica que identifica a una máquina en una red IP.

***Dueño del producto:*** Es el responsable de la toma de decisiones sobre los cambios de la solución informática, es quien aprueba las historias de usuario con su firma, es el dueño del proceso o su delegado quien conoce detalladamente el proceso y tiene poder de decisión. Si en las historias de usuario existen temas que no son de su conocimiento, podrá consultar a la(s) persona(s) que considere necesario.

***Encriptar:*** preparar un archivo o mensaje para que solo pueda interpretarse si se dispone de su contraseña o clave.

***Escenario de prueba:*** es el conjunto de condiciones o variables que definen un estado que debe validarse durante las pruebas para comprobar que la solución informática funciona conforme a lo esperado.

***Expediente técnico:*** es el conjunto de documentos técnicos generados durante todo el proceso del desarrollo de la solución informática.

***Framework:*** Es una estructura conceptual con módulos de software concretos que pueden servir de base para la organización y desarrollo de software. Normalmente incluye soporte de programas, bibliotecas, un lenguaje interpretado, entre otras herramientas satisfaciendo las necesidades más comunes del Desarrollador de Sistemas.

***Librería:*** Conjunto de módulos de programación o elementos que se utilizan para desarrollar y diseñar aplicaciones.

***Logs:*** Registros que guardan detalles importantes sobre eventos, acciones o datos relevantes durante la ejecución de sistemas informáticos. Se utilizan para monitorear, diagnosticar y auditar el funcionamiento de sistemas y aplicaciones.

***Pipeline:*** Es un conjunto automatizado de pasos interconectados que permiten el flujo continuo de trabajo en el desarrollo de software. Se utiliza para compilar, probar y desplegar aplicaciones de manera eficiente y repetible, lo que ayuda a garantizar la calidad del software y acelerar su entrega al mercado.

***Planificación de desarrollo:*** es la planeación de toda la etapa de desarrollo desde el análisis hasta la puesta en producción de la solución informática.

***Pruebas de regresión:*** son pruebas que realiza el analista de sistemas informáticos cuando se desarrolla un soporte para una solución informática. Su objetivo es validar que se haya corregido el error, o que la solución informática funcione correctamente con la nueva funcionalidad.

***Pruebas de unitarias de usuario:*** son las pruebas iniciales que realiza el analista de control de cambios previo a llamar a los usuarios a las pruebas. Su objetivo es validar que la solución informática funcione correctamente en base a los requerimientos solicitados.

***Pruebas integrales:*** son pruebas que realiza el analista de sistemas informáticos y tienen como objetivo validar que toda la solución informática funcione correctamente.

***Pruebas técnicas:*** son pruebas que realiza el analista de control de calidad y tienen como objetivo validar que la solución informática funcione según lo requerido por el dueño del producto.

***Requerimientos funcionales:*** Los requerimientos funcionales de la solución informática describen la funcionalidad o los servicios que se espera que éste provea. Estos dependen del tipo de software y de la solución informática que se esté desarrollando y de los posibles usuarios del software. Describen con detalle la función de sus entradas y salidas, excepciones, etc.

***Requerimientos no funcionales:*** Son restricciones de los servicios o funciones ofrecidos por la solución informática. Incluyen restricciones de tiempo, sobre el proceso de desarrollo, estándares, etc. Son aquellos requerimientos que no se refieren directamente a las funciones específicas que entrega la solución informática, sino a las propiedades emergentes de éste como la fiabilidad, la respuesta en el tiempo y la capacidad de almacenamiento.

***Scripts:*** conjunto de instrucciones que se ejecutan como una macro.

***Servidor:*** equipo que controla el acceso de los usuarios a una red y les da servicio e información.

***Soluciones Informáticas:*** Corresponde al sistema informático o conjunto de sistemas informáticos desarrollado en la Cooperativa, con personal interno, externo o desarrollo compartido.

***SP (Stored Procedure):*** Es un procedimiento el cual es almacenado físicamente en una base de datos, su implementación varía de un gestor de base de datos a otro, al ser ejecutado en respuesta a una petición de usuario es ejecutado directamente en el motor de base de datos, el cual usualmente corre en un servidor separado. Por lo cual posee acceso directo a los datos que necesita manipular y solo necesita enviar sus resultados de regreso al usuario, deshaciéndose de la sobrecarga resultante de comunicar grandes cantidades de datos salientes y entrantes.

***Sprint*:** En metodologías ágiles como Scrum, un sprint es un corto período, generalmente de una a cuatro semanas, en el que un equipo se enfoca en completar un conjunto definido de tareas. El objetivo es desarrollar, probar y entregar un incremento funcional del producto. Al final del sprint, se revisa el progreso y se ajustan estrategias para mejorar en futuros sprints.

***Stakeholders*:** se refiere a todas las partes interesadas que tienen algún interés en un proyecto, proceso o sistema en particular. Estas partes interesadas pueden incluir individuos, grupos, organizaciones u otras entidades que puedan verse afectadas por el resultado del proyecto o que puedan influir en su desarrollo

***Transacciones controladas:*** son las pruebas que se realizan en un ambiente cerrado de producción con un grupo limitado de usuarios. Su objetivo es validar que la solución informática funcione correctamente en producción en base a los requerimientos solicitados.

***Transferencia electrónica:*** es una transacción de fondos, realizadas por cualquier usuario habilitado para este fin, haciendo uso de los diferentes canales electrónicos de la Cooperativa.  
**QA:** es una abreviatura de "Aseguramiento de la Calidad" (en inglés, Quality Assurance). Se refiere al proceso de garantizar la calidad de un producto o servicio. En el contexto del desarrollo de software, QA se centra en asegurar que el software cumpla con los estándares de calidad definidos y funcione según lo previsto antes de su lanzamiento.

***Usuario final:*** Persona que va a manipular de manera directa un producto de software.

***Usuarios de pruebas:*** son las credenciales de los usuarios en el ambiente de pruebas, para poder acceder a las soluciones informáticas a ser validadas.

***Usuarios de transacciones controladas:*** son los usuarios preseleccionados, que cumplen con las características necesarias para ejecutar las transacciones controladas.

***Validación de Requerimientos:*** La validación de requerimientos se asegura de que todos los requerimientos de la solución informática han sido establecidos sin ambigüedad, sin inconsistencias, sin omisiones, que los errores detectados hayan sido corregidos, y que el resultado del trabajo se ajusta a los estándares establecidos para el proceso, el proyecto y el producto.

1. ALCANCE

Comprende desde la recepción del requerimiento de una solución informática, análisis, diseño, codificación, pruebas, implementación, documentación y mantenimiento de la solución informática.

1. RESPONSABILIDADES

Director de Tecnología de la Información (propietario del proceso):

* Gestionar la asignación de los recursos humanos, materiales y tecnológicos necesarios para llevar a cabo el desarrollo de soluciones informáticas (ciclo de vida del software) y buen desenvolvimiento de los proyectos;
* Validar el alcance de las soluciones informáticas;
* Definir la prioridad de los proyectos de tecnología que se originan para apoyar al Plan Operativa de la Cooperativa;
* Gestionar la solución de impedimentos que se presenten durante el desarrollo de soluciones informáticas; y,
* Analizar, definir y proponer la forma de desarrollo de las soluciones informáticas, pudiendo ser estas: desarrollo interno, desarrollo externo o un desarrollo compartido.

Jefe de Operaciones de Tecnología de la Información:

* Implementar en ambiente de producción las soluciones informáticas que hayan sido previamente certificadas;
* Configurar la infraestructura tecnológica requerida para cumplir adecuadamente con el ciclo de vida del software, en base a la disponibilidad de recursos; y,
* Analizar y sugerir mejoras a la infraestructura de hardware y de software propuesta por el área de desarrollo, para que se adapte a los requerimientos de las soluciones informáticas y proporcionen el mejor rendimiento posible.

Analista de Control de Cambios:

* Desarrollar y supervisar la implementación de estrategias de aseguramiento de calidad que se alineen con los objetivos del proyecto;
* Conocer el detalle de la solución informática que se está desarrollando;
* Diseñar casos y escenarios de prueba para evaluar la funcionalidad y calidad del software;
* Ejecutar pruebas manuales y/o automatizadas para verificar que la solución informática funcione correctamente;
* Registrar y documentar los resultados de las pruebas y cualquier defecto o error encontrado;
* Validar que la solución informática cumpla con los requisitos del cliente y los estándares de calidad;
* Comunicar el estado de las pruebas y la calidad del software a la célula de trabajo;
* Colaborar estrechamente con el equipo de desarrollo para mejorar la calidad del software;
* Contribuir a la mejora continua de los procesos de desarrollo y pruebas;
* Validar el expediente técnico de las soluciones informáticas;
* Custodiar el código fuente y ejecutable de las soluciones informáticas que se encuentran en el ambiente de producción;
* Coordinar la ejecución de pruebas de usuario y documentar las mismas en los formatos que corresponda;
* Coordinar la puesta en producción de las soluciones informáticas;
* Verificar que la solución informática disponga del manual de usuario;
* Custodiar el expediente técnico de las soluciones informáticas que se encuentran en ambiente de producción;
* Asegurar que la solución informática cumpla con las normativas y regulaciones aplicables si es necesario;
* Investigar y recomendar nuevas herramientas y enfoques para la mejora de las pruebas de software;
* Identificar y proponer soluciones para problemas y riesgos en el software; y,
* Gestionar la administración del sistema de control de versiones.

Jefe de Desarrollo:

* Liderar el desarrollo de soluciones informáticas;
* Investigar y proponer nuevos marcos de trabajo y tecnologías de desarrollo de soluciones informáticas;
* Coordinar la ejecución de actividades del personal a su cargo;
* Controlar el avance del desarrollo de las soluciones informáticas;
* Reportar al Director de Tecnología de la Información el avance del desarrollo de las soluciones informáticas;
* Gestionar la resolución de problemas que ocurren durante el desarrollo de las soluciones informáticas;
* Coordinar que el desarrollo de las soluciones informáticas cumpla con los estándares de seguridad, normativa emitida por los organismos de control, la normativa interna vigente y requerimientos encontrados en la fase de análisis;
* Coordinar el mantenimiento y soporte técnico de las soluciones informáticas, de conformidad al Plan Operativo de Tecnología de la Información;
* Colaborar en el análisis y factibilidad de las nuevas soluciones informáticas;
* Coordinar el entrenamiento de soluciones informáticas para el dueño del producto;
* Gestionar la corrección de los errores detectados en las soluciones informáticas;
* Gestionar el cumplimiento de los estándares de programación tanto para back-end como front-end; y,
* Gestionar la realización de capacitaciones al personal de desarrollo de sistemas en temas específicos sobre controles, seguridades informáticas y codificación segura para el desarrollo de soluciones informáticas.

Desarrollador de Sistemas:

* Analizar y comprender las necesidades, requerimientos y expectativas del propietario del proceso, mismas que se detallan mediante el formato ([Anexo #10](#_ANEXO__#11:), Requerimiento a la Dirección de Tecnología de Información);
* Elaborar el correo de Lineamientos del Proyecto en donde se definen las especificaciones del proyecto para la solución informática propuesta “[Anexo #13](#_ANEXO_#14:_Estatuto_1)”;
* Realizar la depuración de la información recopilada, organizar y proponer prioridades a los requerimientos establecidos;
* Diseñar la solución a los requerimientos del propietario del proceso, mediante prototipos no funcionales y diagramas de flujo;
* Verificar que el diseño, desarrollo y pruebas técnicas de la solución informática cumpla con los estándares de seguridad [Anexo #23](#_ANEXO__#11:), normativa emitida por los organismos de control, la normativa interna vigente y requerimientos encontrados en la fase de análisis;
* Elaborar y/o actualizar el expediente técnico y el manual de usuario de las soluciones informáticas;
* Elaborar el material a utilizar en el entrenamiento a los usuarios;
* Custodiar el expediente técnico de las soluciones informáticas en ambiente de desarrollo;
* Diseñar, codificar y ejecutar las pruebas unitarias de la solución informática de manera que cumpla con los estándares de seguridad, normativa emitida por los organismos de control, la normativa interna vigente y requerimientos encontrados en la fase de análisis;
* Validar que las soluciones informáticas cumplan con las necesidades del dueño del producto, rigiéndose en el análisis levantado;
* Trasformar el diseño de la solución informática, en estructuras de datos necesarios para el desarrollo;
* Participar en el proceso de análisis de la solución informática;
* Participar en la estimación de tiempo de los proyectos;
* Desarrollar y actualizar el código fuente de la solución informática;
* Elaborar el instructivo de instalación de la solución informática;
* Documentar el diseño de la base de datos y generar el diccionario de datos;
* Generar la documentación del código fuente;
* Documentar los procedimientos almacenados;
* Levantar el plan de pruebas ([Anexo #14](#_Plan_de_pruebas) – Plan de pruebas) que se usarán para la ejecución de las pruebas unitarias de las soluciones informáticas;
* Realizar pruebas de regresión para garantizar que las actualizaciones no afecten las funcionalidades existentes;
* Evaluar el rendimiento y la seguridad del software haciendo uso de las herramientas para pruebas [Anexo # 4](#_ANEXO_#5:_Herramienta_1);
* Cargar en el sistema de control de versiones el código fuente de las soluciones informáticas y la documentación técnica desarrollada;
* Participar en el proceso de pruebas de usuario cuando se le solicite;
* Corregir los errores reportados de las pruebas técnicas y de usuario;
* Otorgar el entrenamiento de soluciones informáticas al dueño del producto, y formalizar el mismo; e,
* Investigar y proponer nuevos marcos de trabajo y tecnologías de codificación de soluciones informáticas.

Arquitecto de Sistemas:

Contará con todas las responsabilidades del Desarrollador de Sistemas; además:

* Colaborar con los desarrolladores en el análisis de sus proyectos mediante la revisión y análisis de las propuestas de solución. Este apoyo permite garantizar que el alcance y las especificaciones funcionales estén alineados con los requisitos solicitados;
* Coordinar estrechamente con los equipos de desarrollo para comprender los requisitos y cambios en el software, garantizando una comunicación efectiva;
* Contribuir activamente a la mejora continua de los procesos de desarrollo y pruebas, identificando áreas de oportunidad y proponiendo soluciones;
* Gestionar eficientemente a los desarrolladores, asignando pruebas técnicas según la disponibilidad, las habilidades y capacidades individuales;
* Identificar proactivamente problemas y riesgos en el software durante las fases de prueba, proponiendo soluciones y medidas preventivas;
* Sugerir a los desarrolladores posibles casos de prueba y escenarios;
* Recomendar y asesorar a los desarrolladores sobre prácticas y técnicas de seguridad que deben implementar en sus proyectos; garantizando la seguridad del software desde las primeras etapas de desarrollo, reduciendo así el riesgo de exposición a posibles ataques; y,
* Diseñar la arquitectura y la estructura de la solución informática para garantizar que satisfaga las necesidades de los usuarios.

**Oficinista de Tecnología:**

* Recopilar información referente a las soluciones informáticas desarrolladas (nuevas) o actualizadas, que se encuentre en el ambiente de producción, con la finalidad de documentar los cambios que afecten el flujo de normal de los procedimientos;
* Elaborar instructivos que expliquen cómo utilizar el sistema de manera efectiva, incluyendo instrucciones paso a paso del proceso mediante la utilización de capturas de pantalla;
* Elaborar y/o actualizar regularmente los manuales de usuario y documentación para reflejar los cambios en el sistema, como nuevas funcionalidades, cambios del proceso o actualizaciones de versión;
* Trabajar en estrecha colaboración con la célula de trabajo para asegurar que la documentación refleje con precisión el funcionamiento del sistema;
* Evaluar la documentación y realizarle las mejoras según sea necesario, para garantizar que sea fácil de entender y utilizar por los usuarios finales; y,
* Asegurarse que la documentación cumpla con los estándares y normativas definidos por la Cooperativa.

Jefe de Seguridad de la Información:

* Revisar y ejecutar la configuración sugerida de las soluciones informáticas para garantizar que se cumpla las políticas de seguridad establecidas y las mejores prácticas ([Anexo #23](#_ANEXO_#27:_Seguridad_1)); y,
* Proponer alternativas de solución para cumplir con los requerimientos de seguridad de las soluciones informáticas, para ajustarse a lo requerido por la Cooperativa.

Director de procesos y control de gestión:

* Diseñar el proceso de los requerimientos.

Propietarios de los procesos:

* Designar los integrantes de la célula de la solución informática; y,
* Solicitar cambios a las soluciones informáticas.

Dueño del producto:

* Firma la solicitud de puesta en producción, confirmando que se ejecutaron las pruebas de usuario; y,
* Definir estrategias para la puesta en producción de las soluciones informáticas.

Célula de trabajo:

* Validar la Estructura de Desglose de trabajo de la solución informática, participando proactivamente en su elaboración, coordinando con las áreas involucradas, de ser necesario;
* Conocer el entorno de negocio, las necesidades y el objetivo que se persigue con la solución informática que se está actualizando o se va a desarrollar;
* Aprobar los cambios necesarios durante el desarrollo de la solución informática;
* Designar los usuarios de pruebas, quienes tendrán la responsabilidad de participar en las pruebas de usuario;
* Designar los usuarios de transacciones controladas, quienes tendrán la responsabilidad de participar en las transacciones controladas de la solución informática;
* Validar el manual de usuario de las soluciones informáticas puestas en producción;
* Gestionar (con las áreas que corresponda) la parametrización y configuración requerida por la solución informática; y,
* Definir estrategias para la puesta en producción de las soluciones informáticas.

1. POLÍTICAS
   1. GENERALES
2. Los trabajadores del área de desarrollo observarán, lo definido en el Manual para la Gestión de Seguridad de la Información, para acceder a la información confidencial de la Cooperativa;
3. El ambiente de desarrollo donde se alojan las soluciones informáticas estará aislado del ambiente de producción;
4. Las librerías, controladores y aplicativos necesarios para que las soluciones informáticas funcionen, se respaldarán en el repositorio asignado al área de desarrollo;
5. El Jefe de Desarrollo de Sistemas, al menos una vez al año, presentará al (la) responsable de Recursos Humanos las necesidades de capacitación del personal a su cargo, con el fin de garantizar que el ciclo de vida de desarrollo de software cumpla con estándares de seguridad;
6. El (la) responsable de Recursos Humanos gestionará la propuesta planteada por el Jefe de Desarrollo de Sistemas de manera independiente al plan anual de capacitación;
7. Los desarrolladores de sistemas son los responsables del cumplimiento de los estándares de programación definidos en la presente metodología, además de garantizar que la solución informática cumpla con los requisitos del cliente;
8. Mantener respaldos de toda la información relacionada con el desarrollo e implementación de las soluciones informáticas, entre las que constan documentos, historias de usuario, correos y prototipos, con el fin de garantizar la integridad y disponibilidad de estos datos; y,
9. Brindar la capacitación necesaria al personal afectado por la implementación de las soluciones informáticas, con el objetivo de familiarizarse con su funcionamiento y uso correcto.
   1. DE LA GESTIÓN DE PROYECTOS DE DESARROLLO DE SOLUCIONES INFORMÁTICAS
10. El Jefe de Desarrollo de Sistemas mantendrá una planificación de los proyectos de desarrollo para todas las soluciones informáticas que se requieran y se originen en: Plan operativo institucional, plan operativo de tecnología de la información y disposiciones de los organismos de control externo;
11. La planificación de los proyectos de desarrollo de las soluciones informáticas que mantiene el Jefe de Desarrollo de Sistemas se alineará al alcance y cronograma establecido en: Plan operativo institucional, plan operativo de tecnología de la información y disposiciones de los organismos de control externo;
12. El propietario del proceso podrá solicitar nuevos requerimientos no contemplados en la planificación de proyectos de desarrollo, coordinando con el director de procesos las mejoras requeridas, solicitud que la planteará usando el formato [Anexo #10](#_ANEXO__#11:);
13. El director de tecnología de la información gestionará con las áreas que correspondan la inclusión (o postergación justificada) del requerimiento planteado por organismos de control, áreas de control interno y/o por el propietario del proceso en la planificación de los proyectos de desarrollo;
14. El director de tecnología de la información gestionará la provisión de recursos necesarios para incluir nuevos requerimientos en la planificación de desarrollo;
15. La planificación de desarrollo contendrá al menos:

* Detalle de las actividades que se relacionen con el ciclo de vida de desarrollo de software (análisis, diseño, desarrollo, pruebas técnicas y control de calidad);
* Responsable de la actividad;
* Fecha de inicio;
* Fecha de fin;
* Porcentaje de avance;
* Notas; y,

1. El Jefe de Desarrollo de Sistemas propondrá nuevas formas de gestión de proyectos de desarrollo, a fin de procurar la innovación, agilidad y calidad del desarrollo de soluciones informáticas.
   1. ETAPAS DEL DESARROLLO DE LAS SOLUCIONES INFORMÁTICAS
      1. ANÁLISIS
2. El Desarrollador de sistemas solicitará al propietario del proceso la definición de la célula de trabajo responsable, con quien iniciará la fase de análisis;
3. El levantamiento de requerimientos de las soluciones informáticas se realizará mediante técnicas como: entrevista personal, observación de campo, lluvia de ideas, etc.;
4. El Desarrollador de sistemas documentará el análisis y las medidas de seguridad necesarias para el desarrollo ([Anexo #23](#_ANEXO_#27:_Seguridad_1)); describiendo de forma clara los requerimientos y los posibles problemas de seguridad, evitando el consumo innecesario de recursos o la aparición de requerimientos contradictorios;
5. El análisis de las soluciones informáticas será formalizado en el [Anexo #13](#_ANEXO_#14:_Estatuto_1);
6. Todo cambio no contemplado en el análisis debe ser resuelto por la célula de trabajo;
7. El Jefe de Desarrollo de Sistemas definirá cuando la solución informática requerirá la construcción de un prototipo no funcional. [Anexo #12](#_ANEXO_#12:_Prototipo)**;**
8. El Jefe de Desarrollo de Sistemas validará la arquitectura sobre la cual desarrollará la solución informática, [Anexo #5](#_ANEXO_#7:_Arquitectura); y,
9. El Jefe de Desarrollo de Sistemas propondrá al director de tecnología de la información nuevas arquitecturas, dependiendo de la solución tecnológica a ser desarrollada, las cuales serán incorporadas en el presente manual.
   * 1. DISEÑO
10. Todo requerimiento relacionado con una solución informática que implique cambios al modelo de base de datos deberá tener un diseño lógico y físico;
11. En el diseño de soluciones informáticas se utilizarán herramientas de modelado de base de datos como instrumento para la creación de la base de datos. [Anexo #2](#_ANEXO_#3:_Herramientas);
12. Los nombres de los objetos en la base de datos se codificarán con los estándares especificados en el [Anexo #7](#_￼ANEXO_#8:_Estándares);
13. El diseño lógico cumplirá con:

* Diseño de componentes de software para que sean reutilizados en otras soluciones informáticas;
* Especificaciones que consideren los requisitos de seguridad, rendimiento y depuración de datos;
* Diccionario de datos, siempre y cuando la solución tecnológica procese datos; y,
* Autenticación segura para el acceso de usuarios a las soluciones informáticas;

1. En base al análisis realizado, el Desarrollador de Sistemas definirá los diseños para la seguridad del sistema [Anexo #23](#_ANEXO_#27:_Seguridad_1) y la arquitectura [Anexo #6](#_Arquitectura_de_Desarrollo) sobre la cual se realizará el desarrollo; y,
2. La Célula de Trabajo y el jefe de operaciones de TI aceptan los diseños del desarrollador para que pueda empezar con el proceso de desarrollo.
   * 1. DESARROLLO
3. El desarrollo de soluciones informáticas se basará en el uso de herramientas de desarrollo, [Anexo #3](#_ANEXO_#4:_Herramientas);
4. El desarrollo de soluciones informáticas deberá producir software con calidad de producción, seguro ([Anexo #23](#_ANEXO_#27:_Seguridad)), probado y documentado;
5. Los estándares para la programación y base de datos son los establecidos en el [Anexo #7](#_￼ANEXO_#8:_Estándares);
6. Las reglas para el desarrollo seguro de las soluciones informáticas que incorporen “transferencias electrónicas” serán las establecidas en el [Anexo #9](#_ANEXO_#10:_Desarrollo); y,
7. Se deberá registrar logs de trazabilidad y de errores tanto en el código fuente como en la base de datos, conforme al [Anexo #24](#_ANEXO_#24:_Registro).
   * 1. PRUEBAS
8. El ambiente de pruebas mantendrá una configuración similar al ambiente de producción;
9. Todo error encontrado debe estar documentado y formalizado en la bitácora de errores [Anexo #15](#_ANEXO_#16:_Bitácora), en redacción clara y especifica de cómo reproducirlo para determinar el tiempo que puede tomar corregirlo;
10. Para toda solución informática se realizarán los siguientes tipos de pruebas:

* **Pruebas técnicas:** Pruebas ejecutadas por los Analistas de Control de Cambios. Su objetivo es validar la funcionalidad de cada uno de los módulos en forma individual y en forma conjunta, garantizando el correcto funcionamiento de la solución informática;
* **Pruebas de usuario:** Pruebas ejecutadas por el Analista de Control de Cambios en conjunto con el dueño del producto, y desarrollador se reunirán con el objetivo de validar todos los requerimientos establecidos para la solución informática, los errores se harán conocer en el formulario detallado en el [Anexo #17](#_ANEXO_#18:_Incidentes)**;** y,

1. Toda solución tecnológica será puesta en producción siempre y cuando haya superado con éxito las pruebas técnicas y pruebas de usuario.
   * + 1. PRUEBAS TÉCNICAS
2. El Analista de Control de Cambios validará que la solución informática cumpla con las políticas de desarrollo y seguridad ([Anexo #23](#_ANEXO_#27:_Seguridad)) establecidas en el presente manual;
3. Para las pruebas de las soluciones informáticas que involucren transferencias electrónicas se usará herramientas que permitan validar la seguridad de las mismas, [Anexo #4](#_ANEXO_#5:_Herramienta); y,
4. El Analista de Control de Cambios, en coordinación con el equipo de desarrollo asignado, realizará las pruebas técnicas que sean necesarias, usando el plan de pruebas [Anexo #14](#_ANEXO_#15:_Plan), para garantizar una adecuada calidad de la solución informática;
   * + 1. PRUEBAS DE USUARIO
5. El funcionamiento de la solución informática será validado íntegramente por el dueño del producto;
6. El Jefe de Seguridad de la Información participará en las pruebas de usuario;
7. De ser necesario el dueño del producto requerirá la participación de otros usuarios en la ejecución de las pruebas de usuario; y,
8. El desarrollador de la solución informática también participará en la validación de las pruebas de usuario si se lo requiere.
   * 1. IMPLEMENTACIÓN
9. El dueño del producto en coordinación con el Jefe de Desarrollo de Sistemas, definirán la estrategia del paso a producción de las soluciones informáticas desarrolladas;
10. Para la puesta en producción de las soluciones informáticas se aplicarán lo establecido en el Manual de Tecnología;
11. El Analista de Control de Cambios remitirá al Oficial de Seguridad de la Información las parametrizaciones necesarias para la puesta en producción de la solución informática;
12. El Jefe de Operaciones de Tecnología de la Información definirá el servidor donde se instalarán las soluciones informáticas ([Anexo #23](#_ANEXO_#27:_Seguridad_1));
13. El Jefe de Operaciones de Tecnología de la Información será el único autorizado para instalar las aplicaciones en los servidores de producción, y en su ausencia lo podrá hacer el Administrador de datos; y,
14. El Jefe de Desarrollo de Sistemas gestionará la entrega de las versiones finales del código fuente y ejecutable de todas las soluciones informáticas desarrolladas o actualizadas al Analista de Control de Cambios.
    * + 1. ENTRENAMIENTO
15. Toda solución informática (nueva) desarrollada por la Cooperativa que se ponga en ambiente de producción, incluirá un plan de entrenamiento para los usuarios finales;
16. Cuando se actualicen las soluciones informáticas que se encuentren en ambiente de producción, y estos cambios afecten el flujo normal de los procedimientos definidos, también se ejecutará un plan de entrenamiento;
17. El Jefe de Desarrollo de Sistemas entregará al dueño del producto el plan de entrenamiento, el mismo que se podrá ejecutar de forma presencial o telemática. El informe de entrenamiento puede referirse al [Anexo # 8](#_ANEXO_#9:_Informe); y,
18. El plan de entrenamiento contendrá un instructivo, con el detalle de los cambios realizados.
    1. DOCUMENTACIÓN

Toda solución informática que se desarrolle debe contener un expediente técnico, [Anexo #5](#_ANEXO_#6:_Herramientas).

* 1. DE LA CONTRATACIÓN DE SERVICIO DE DESARROLLO EXTERNO

1. El Director de Tecnología de la Información validará que los contratos de servicio de desarrollo de software (soluciones informáticas) contemplen lo siguiente:

* Procedimientos para proteger los bienes de la Cooperativa;
* Cumplimiento de las políticas relacionadas con la gestión de la seguridad de la información y seguridad física;
* Definiciones de derechos de propiedad intelectual;
* Formas de provisión del servicio ante situaciones de emergencia; y,

1. Los proveedores de servicio de desarrollo de software externo se regirán en las políticas y procedimientos estipulados en el Manual para la Gestión de Seguridad de la Información, para acceder a la información confidencial de la Cooperativa.

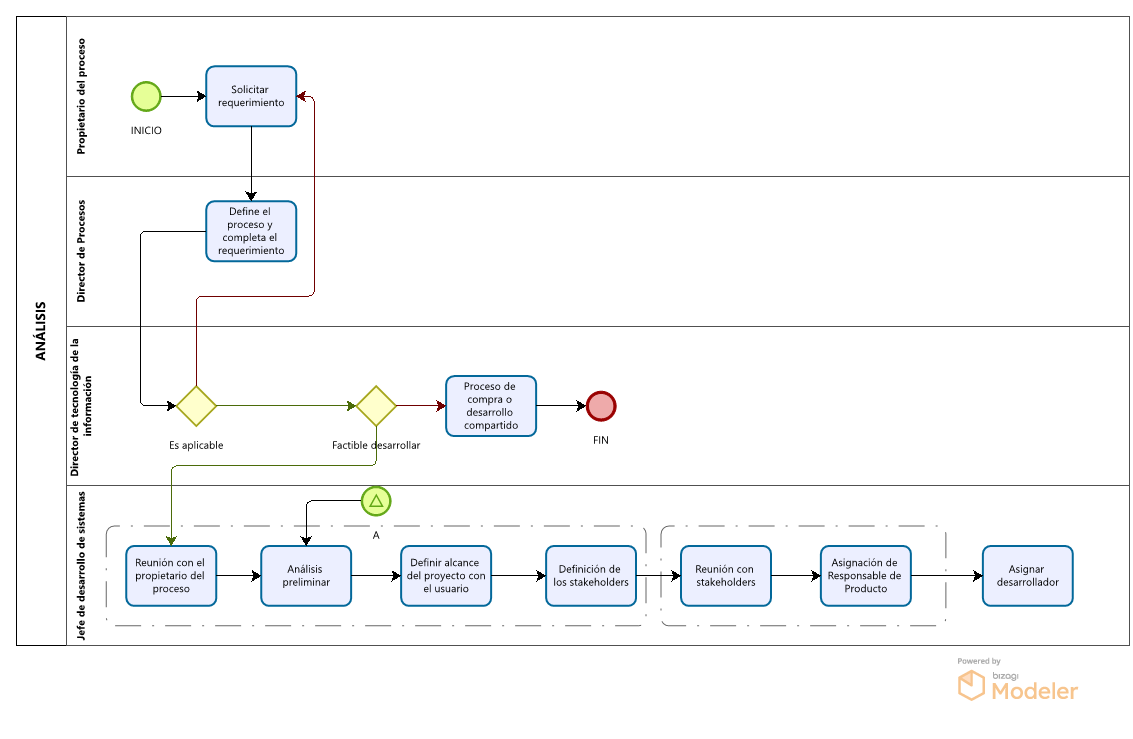
1. DISPOSICIONES GENERALES
2. Todos los trabajadores involucrados en el presente proceso, son responsables del cumplimiento obligatorio de lo establecido en este documento;
3. La inobservancia o incumplimiento de las responsabilidades, políticas y procedimientos derivará en la aplicación de sanciones por parte de la Administración de la Cooperativa, conforme a lo previsto en el Reglamento Interno de Trabajo;
4. Los trabajadores involucrados en el presente proceso, son responsables civil, penal y administrativamente, por las pérdidas que se produjeren ante su inobservancia, errores y/u omisiones a lo determinado en este documento;
5. Previo a imponer la sanción a los responsables de los errores u omisiones, el Auditor Interno emitirá un informe que contenga:

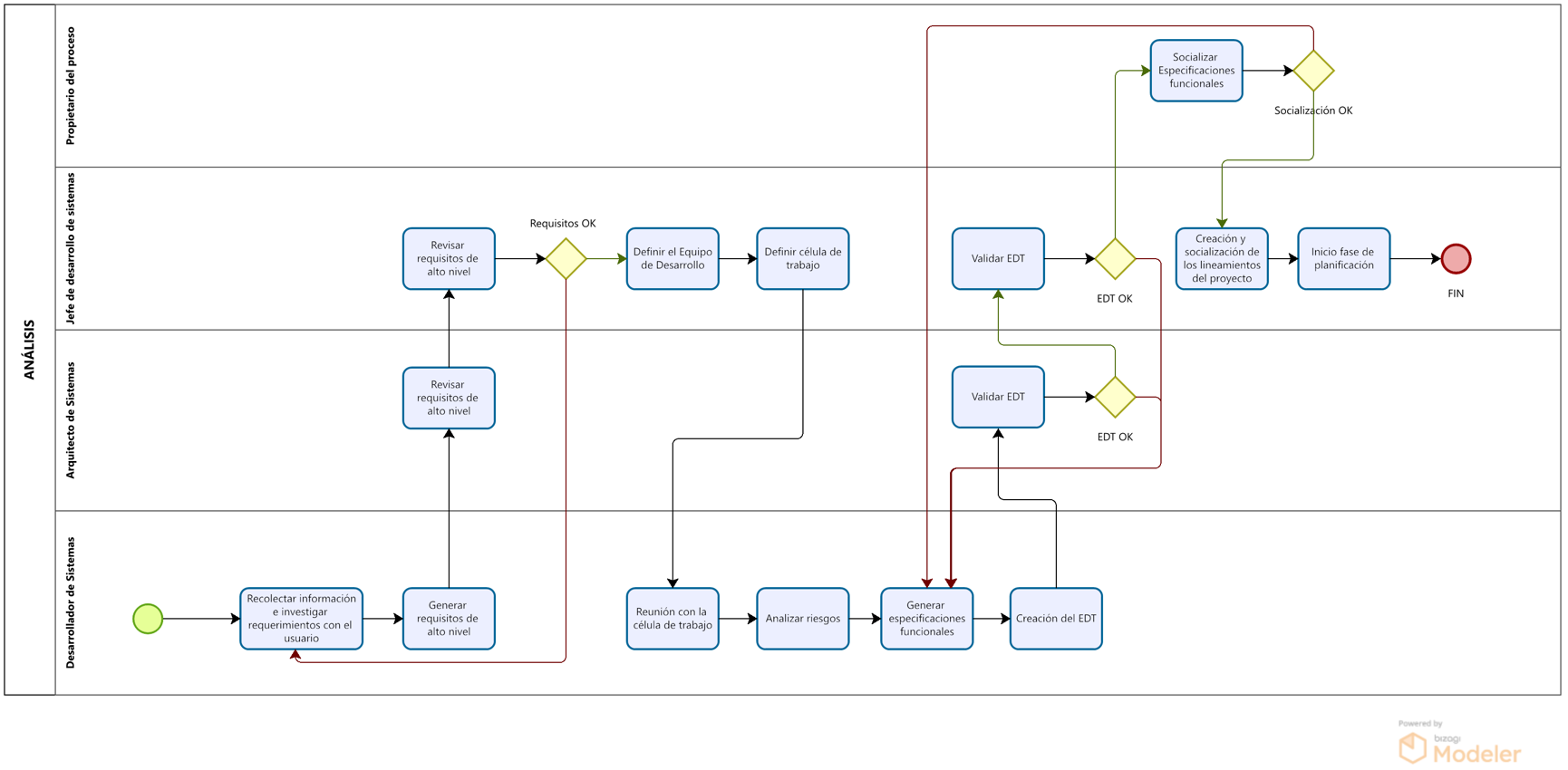
* Antecedentes;
* Disposiciones legales y reglamentarias inobservadas;
* Causas que originaron el incumplimiento del proceso establecido en el presente manual;
* Consecuencias y perjuicios ocasionados;
* Responsable(s) del error u omisión; y,
* Conclusiones y Recomendaciones.

1. El Gerente y/o Consejo de Administración, en base al informe señalado en el literal anterior, adoptarán las acciones que correspondan;
2. La difusión y actualización del presente documento se realizará conforme a lo estipulado en el “Manual para la administración de procesos”;
3. El presente documento es de lectura obligatoria para todos los trabajadores de la Cooperativa;
4. El Auditor Interno es el responsable de aplicar evaluaciones permanentes sobre el grado de suficiencia de conocimientos, a los trabajadores involucrados en el presente proceso;
5. Las evaluaciones se aplicarán al personal de la oficina matriz y de agencias, de acuerdo a lo previsto en el plan anual de trabajo de Auditoría Interna. Los resultados de las evaluaciones se comunicarán al Gerente;
6. Las evaluaciones se realizarán in situ con la finalidad de verificar el nivel de conocimientos de cada trabajador. Los cuestionarios de evaluación contendrán como mínimo 10 preguntas entre objetivas y de razonamiento, cada una tendrá un valor de 1 punto, el mismo que se lo transformará a porcentaje;
7. Cuando un trabajador obtuviese un puntaje inferior al 85%, deberá recibir nuevamente capacitación y rendir una nueva evaluación. Si nuevamente no alcanza el puntaje mínimo, se comunicará al Gerente y éste recomendará las acciones aplicables a cada caso;
8. El Gerente puede asumir todas las atribuciones otorgadas a los trabajadores de la Cooperativa, previstas en el presente documento; y,
9. El Gerente es el único autorizado para excepcionar las políticas y procedimientos establecidos en el presente documento.
10. DISPOSICIONES TRANSITORIAS
11. El presente documento entrará en vigencia a partir de la aprobación por parte del Consejo de Administración quedando sin efecto la versión anterior;
12. Derogar la versión anterior “Manual para el Desarrollo e Implementación de Soluciones Informáticas V\_01\_2021”; y,
13. Se deroga toda disposición interna que se contraponga a lo previsto en este documento;
14. NORMATIVA INTERNA RELACIONADA

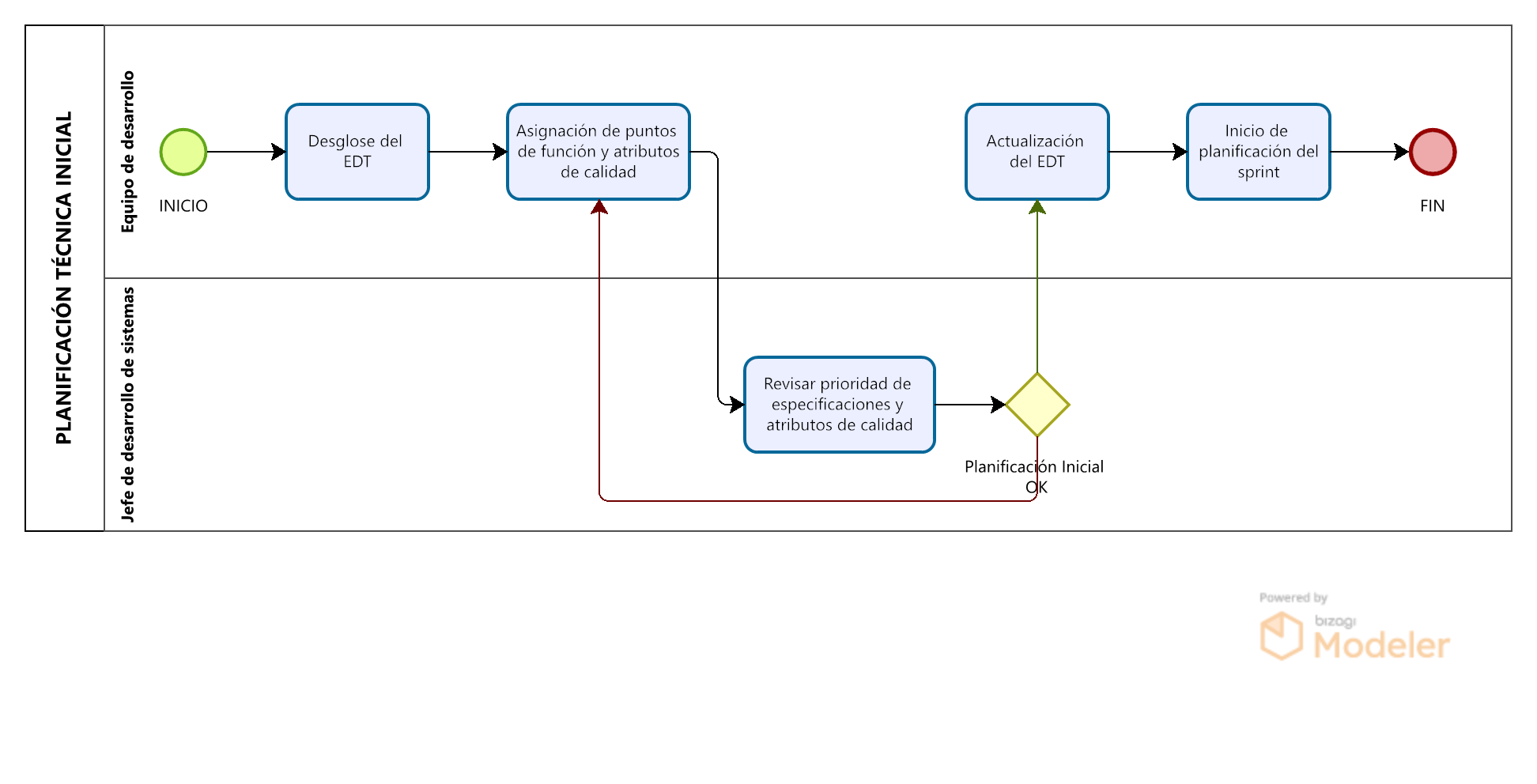
* Manual para la gestión de la seguridad de la información;
* Manual de Tecnología; y,
* Reglamento del Comité de Tecnología.

# 

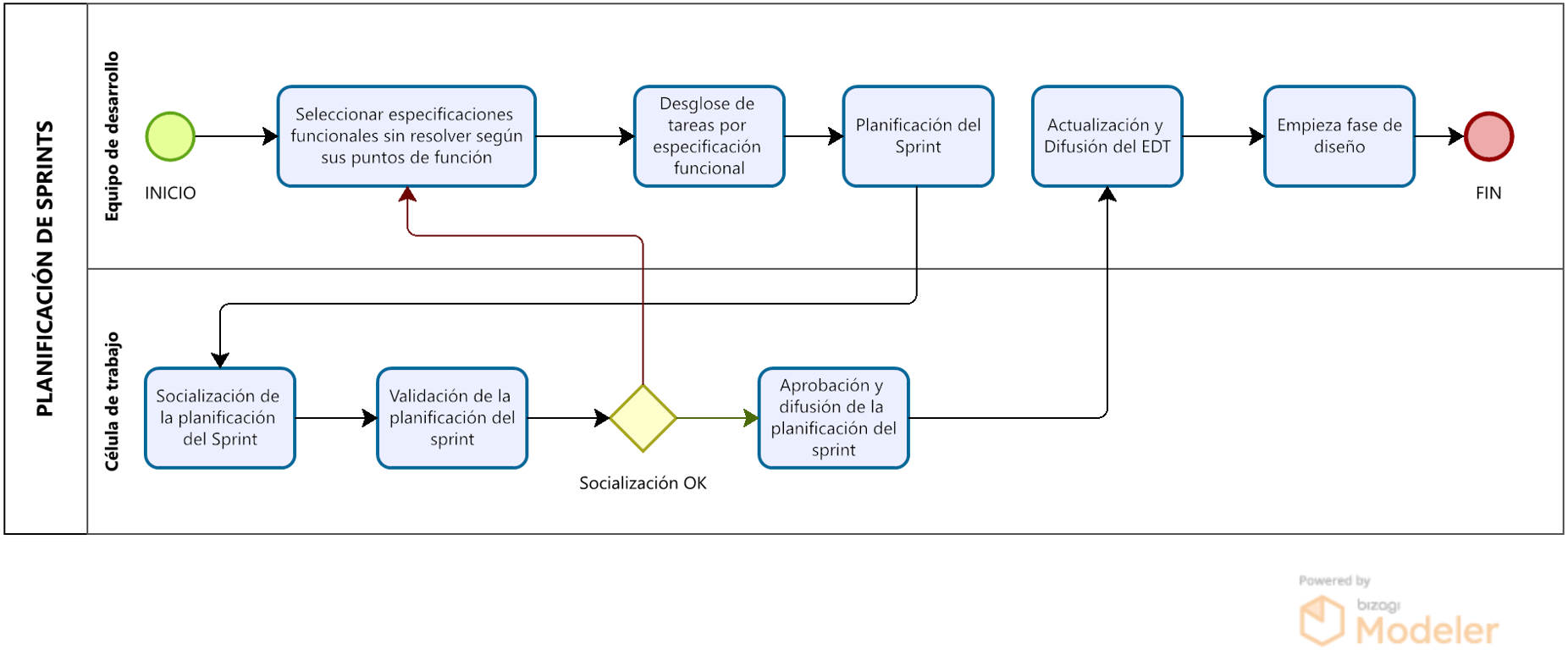
1. DIAGRAMAS DE PROCESO
   1. Análisis de requerimientos y factibilidades de soluciones informáticas



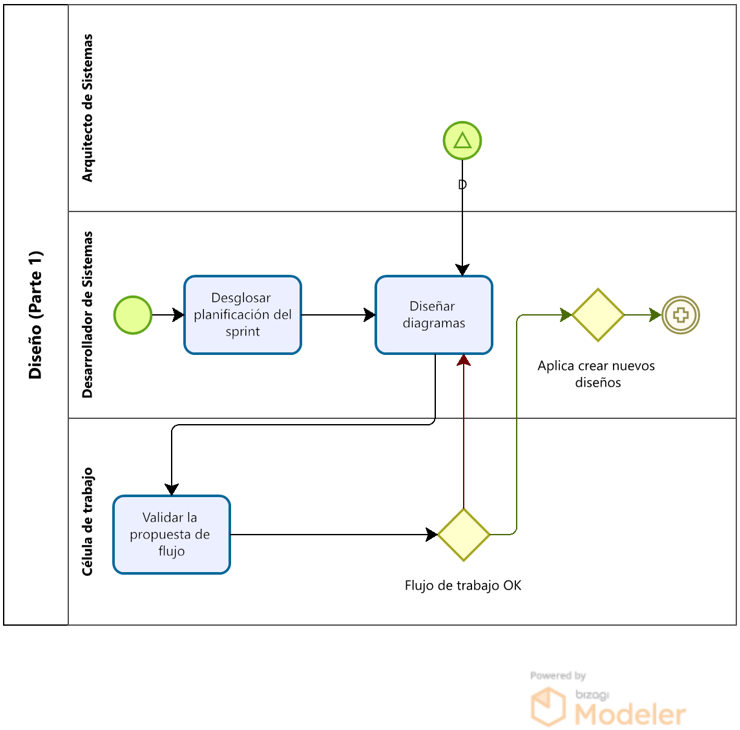
* 1. Planificación técnica inicial



* 1. Planificación de Sprint

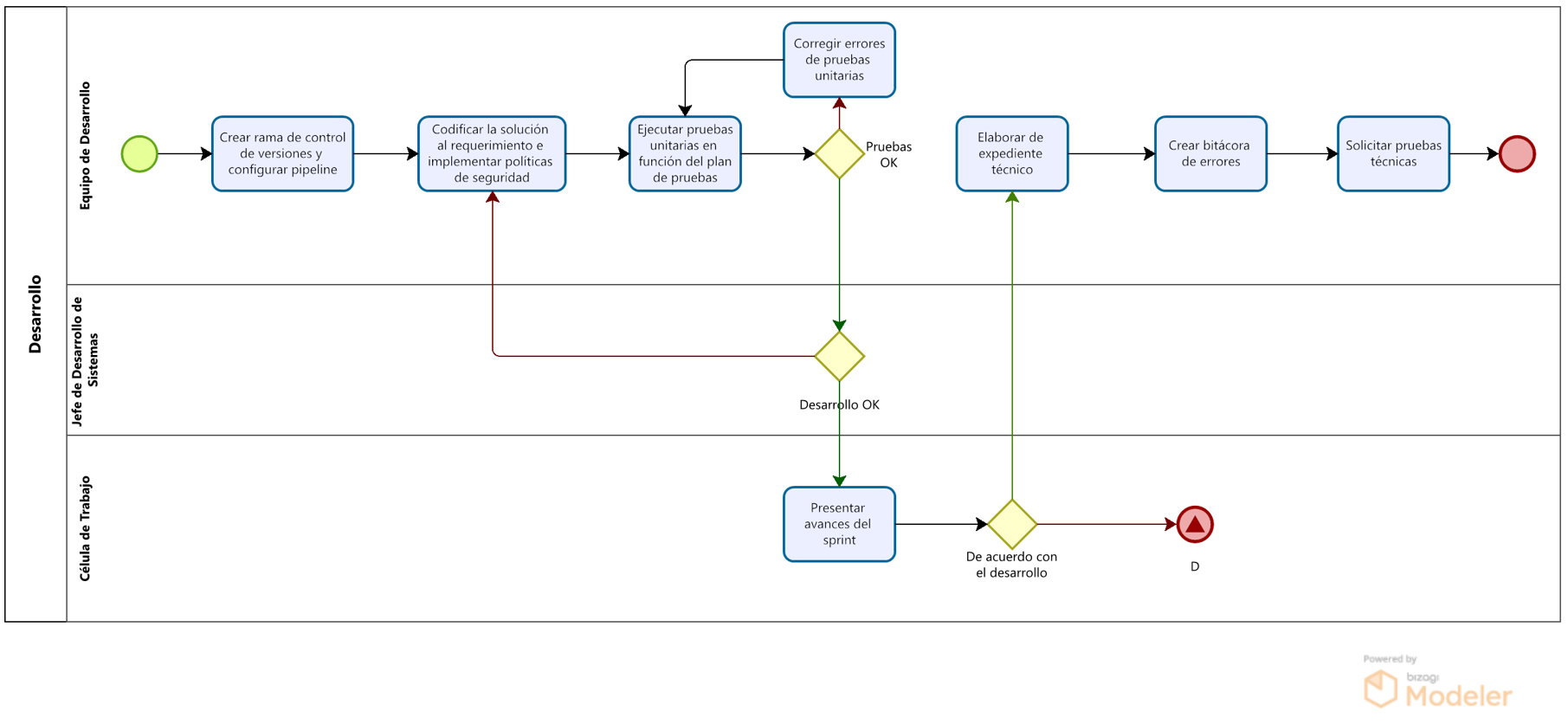


* 1. Diseño de soluciones informáticas

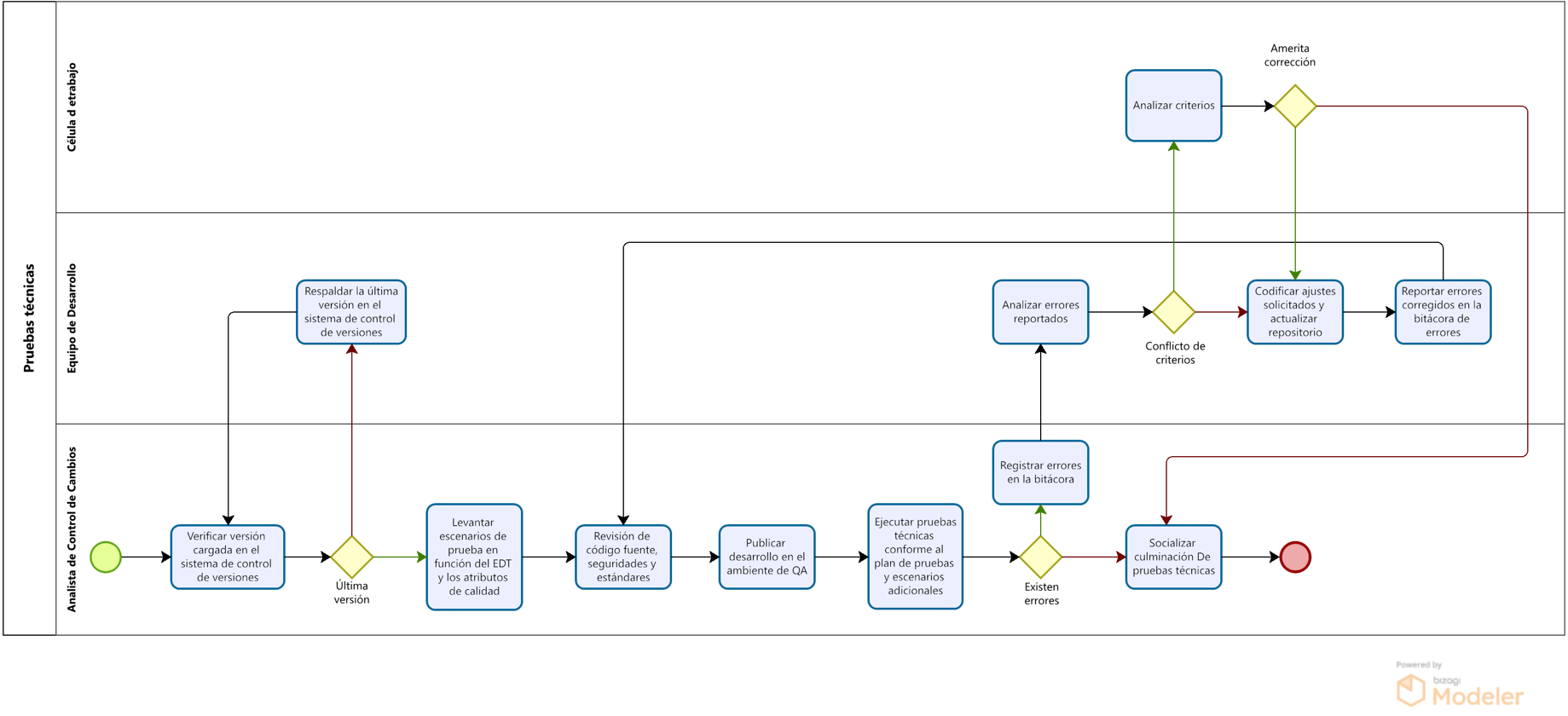




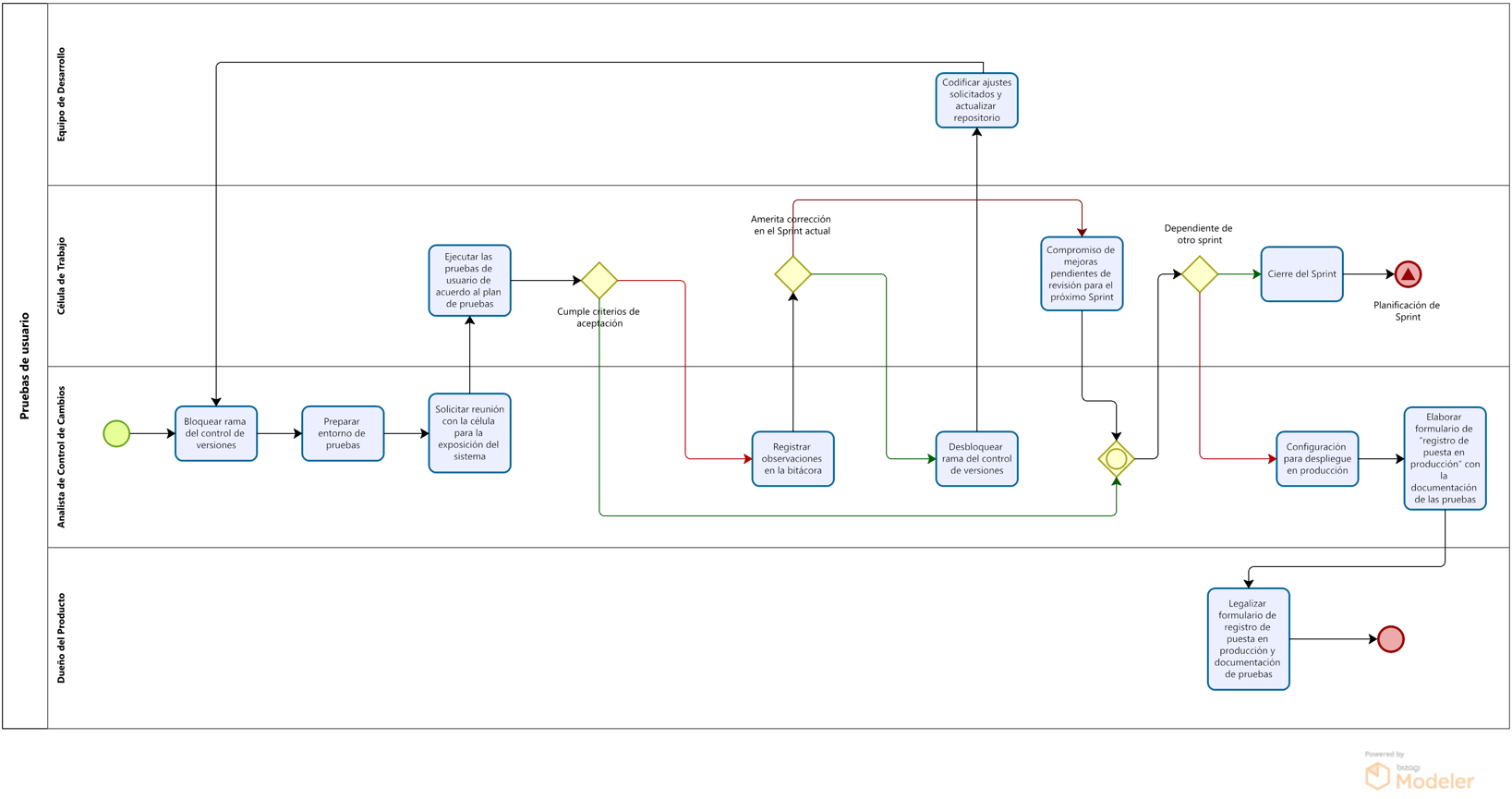
* 1. Desarrollo de soluciones informáticas



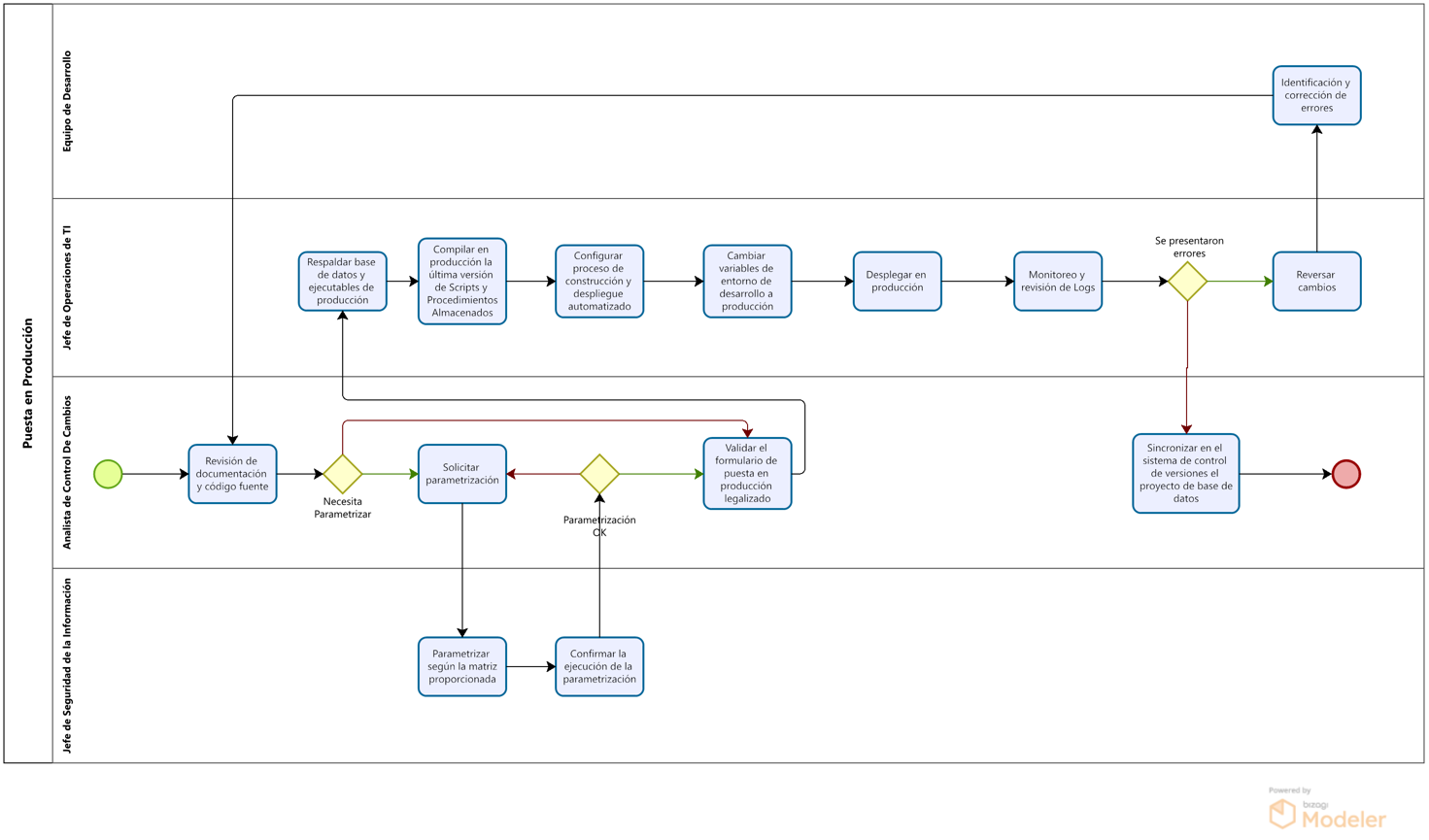
* 1. Pruebas Técnicas



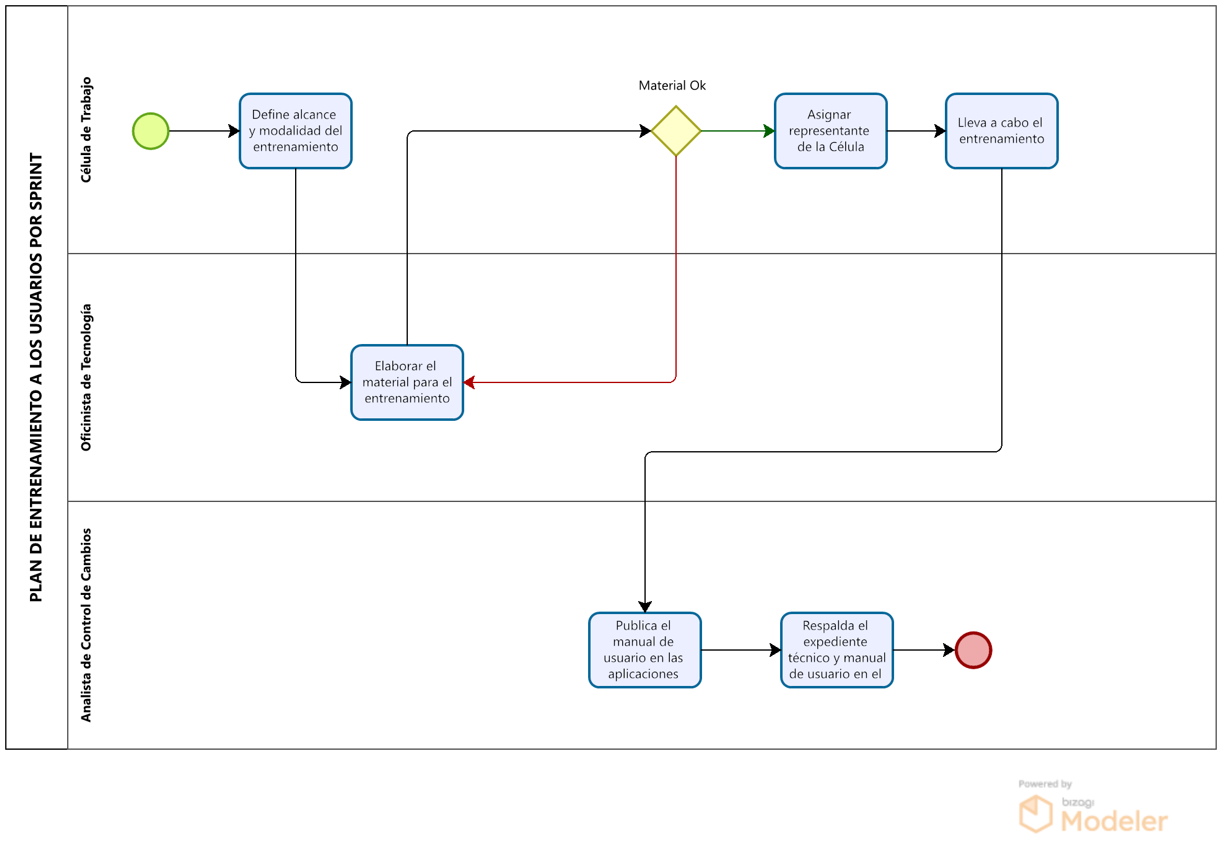
* 1. Pruebas de Usuario



* 1. Puesta en Producción



* 1. Plan de entrenamiento a los usuarios



1. PROCEDIMIENTO
   1. Análisis de requerimientos y factibilidades de soluciones informáticas

Propietario del proceso:

1. Solicitar requerimiento. – esta actividad da inicio al desarrollo de una solución informática. Los usuarios “Propietarios de los procesos” como (directores, jefes departamentales, organismos de control internos, etc.), solicitarán cualquier requerimiento al Director de Tecnología de la Información mediante el documento denominado “Requerimiento a la Dirección de Tecnología de Información” ([Anexo #10](#_ANEXO_#11:_Requerimiento)). No será necesario enviar este documento, para los requerimientos que constan en la planificación operativa o estratégica de la Cooperativa o del departamento de Tecnología de la Información, tampoco será necesario enviarlo en requerimientos producto de normativas externas emitidas por los entes de control externo; para el resto de casos, cuando por algún motivo justificado no sea posible enviar este documento, se deberá enviar el requerimiento detallado mediante un correo electrónico o cualquier otro medio del cual haya constancia física o electrónica.

Director de procesos:

1. Define el proceso que se llevara a cabo para cumplir el requerimiento, analiza la solución informática como una necesidad específica, si esta es aplicable, y resuelve el problema identificado, describiendo de manera precisa qué es lo que se necesita, cuáles son los objetivos que se buscan alcanzar, cuáles son los criterios de satisfacción y cuál es el impacto que tendrá en el negocio.

Director de Tecnología de la Información:

1. Recepta la solicitud del requerimiento y verifica la aplicabilidad del mismo, analizándolo si se ajusta a los objetivos estratégicos de la cooperativa o si es un proceso de mejora continua.
2. Analiza la prioridad del requerimiento para anteponer su ejecución a otros requerimientos planteados.
3. Revisa las mejores opciones en el mercado, si es factible desarrollar, pasará el requerimiento al Jefe de Desarrollo de Sistemas, caso contario se irá a un proceso de compra o se definirá si se realiza un desarrollo compartido con un proveedor externo.

Jefe de Desarrollo de Sistemas:

1. Agenda y realiza una reunión con el Propietario del proceso, en conjunto realizan un análisis preliminar del requerimiento solicitado, se obtiene información básica referente a la solución informática a desarrollar y se establece el alcance del proyecto con el Propietario del Producto. Se define en conjunto a las partes interesadas y afectadas por el proyecto, a partir de ahora referidos como “Stakeholders”.
2. Agenda y realiza una reunión con los Stakeholders, se socializa el proyecto y se realizan ajustes al alcance del proyecto en conjunto. Se designa una persona encargada de realizar el seguimiento del proyecto, a partir de ahora referido como “Responsable de Producto”.
3. Asigna a un Desarrollador de Sistemas el proyecto y le entrega documentación recopilada del análisis preliminar.

Desarrollador de Sistemas:

1. Realiza estudio del análisis preliminar del proyecto asignado. Recolecta la información necesaria para definir los requerimientos de software, haciendo uso de herramientas como: investigación, entrevistas, encuestas, observación, etc. [Anexo #1](#_ANEXO_#2:_Herramientas)**.**
2. Depura toda la información que contiene el requerimiento solicitado: elimina requerimientos repetidos, redundancias, etc., con la finalidad de generar requisitos de alto nivel claros y precisos de lo que se debe desarrollar. El [Anexo #19](#_ANEXO_#20:_Especificación) detalla las distintas fases y consideraciones para el levantamiento de requerimientos.
3. Se agenda y realiza una reunión con la Célula de Trabajo, que fue definida por el Jefe de Desarrollo en su momento, en la cual se responderán inquietudes respecto a los requisitos de alto nivel propuestos, preguntas referentes al producto, a los procesos y a los fundamentos o justificaciones para realizar el proyecto. Este proceso permitirá determinar y registrar hallazgos de problemas o inconvenientes en los procesos de negocio y determinar oportunidades de mejora.
4. En la reunión con la Célula de Trabajo se debe analizar e identificar los posibles riesgos en base a los requisitos de alto nivel, el análisis de riesgos podría abordar aspectos como: Vulnerabilidades de seguridad, posibles obstáculos técnicos que podrían surgir durante el desarrollo, riesgos relacionados con la disponibilidad de recursos necesarios, factores externos que podrían afectar el proyecto, impacto potencial en el negocio en caso de falla o retraso en el proyecto. Se debe detallar lo analizado en el [Anexo #13](#_ANEXO_#14:_Estatuto).
5. Se genera especificaciones funcionales para cada requisito de alto nivel. Se debe procurar que en cada especificación se pueda identificar: el usuario, la especificación funcional dentro del requisito de alto nivel y el propósito de dicha especificación, como se detalla en el [Anexo #19](#_ANEXO_#20:_Especificación) .
6. Se levantará la Estructura de Desglose de Trabajo con los requisitos de alto nivel y las especificaciones funcionales y el objetivo para el que nació el requerimiento. Este EDT deberá ser modificado gradualmente en cada iteración del proyecto, usando el formato detallado en el [Anexo #19](#_ANEXO_#20:_Especificación).

Arquitecto de Sistemas

1. Revisa el análisis preliminar, si los requisitos de alto nivel no están claros, o en desacuerdo al marco de trabajo levantado inicialmente, se regresará al Desarrollador de Sistemas con todas las dudas y sugerencias para que se aplique nuevamente la recolección de información.
2. Valida la Estructura de Desglose de Trabajo

Jefe de Desarrollo de Sistemas:

1. Revisa el análisis depurado después de la revisión con el Arquitecto de sistemas, si el análisis aún se encuentra con inconsistencias o falencias se regresará al Desarrollador de Sistemas para que analice nuevamente en base a las observaciones realizadas.
2. Asigna los desarrolladores que conformarán al Equipo de Desarrollo, procurando que cada equipo tenga al menos dos personas, el desarrollador que realizó el análisis puede o no ser parte del equipo de desarrollo. El número de desarrolladores asignados dependerá del alcance del proyecto y a discreción del Jefe de Desarrollo Sistemas. La información pertinente respecto a la forma de trabajo para proyectos se encuentra en el [Anexo #20](#_ANEXO_#21:_Pair).
3. En conjunto con el propietario del proceso, se define a los miembros que serán parte de la Célula de Trabajo, la cual deberá constar como mínimo de: Propietario de Proceso, Jefe de Desarrollo de Sistemas, Responsable del Producto, Equipo de Desarrollo, Seguridad de la Información, Dirección de proyectos, Dirección de riesgos, Dirección de Procesos y otros miembros que se considere oportunos.
4. Revisa la EDT, verificando que se ajuste a los requerimientos solicitados por el Propietario del Proceso para determinar si se puede cumplir técnicamente con el requerimiento. Si existen ajustes, los coordina con el Arquitecto de Sistemas, hasta llegar a un consenso para proceder a actualizar la EDT. Procede a agendar una reunión para socializar las especificaciones con el Propietario del Proceso y la Célula de Trabajo. Agenda una reunión con la Célula de Trabajo y el Propietario del Proceso.
5. Si el Propietario del Proceso aprueba la EDT, se elabora el correo denominado Lineamientos del Proyecto, el cual se puede enviar en físico para ser firmado por los miembros de la célula de trabajo o se puede enviar un correo dirigido a todos los involucrados según el formato establecido en el [Anexo #13,](#_ANEXO_#14:_Estatuto)  y comienza la Fase de Planificación Técnica Inicial.

Propietario del Proceso

1. Revisa y aprueba en conjunto con la Célula de Trabajo la EDT del proyecto; si es necesario, se reporta cambios a las especificaciones funcionales al Desarrollador de Sistemas encargado del análisis para reanalizarlos y corregirlos.
   1. Planificación técnica inicial

Equipo de Desarrollo

1. Desglosa la EDT, analiza las especificaciones funcionales de forma individual, asignándoles puntos de función y atributos de calidad. Los puntos de función determinarán la prioridad de una especificación, considerando: la importancia para el usuario, la complejidad técnica, el valor del negocio y la urgencia. Los atributos de calidad por parte se los asignará considerando: Viabilidad técnica, usabilidad, accesibilidad, escalabilidad, etc. Se pueden asignar aspectos de seguridad como un atributo de calidad clave y asignar puntos de función adicionales para implementar controles de seguridad específicos, [Anexo #19](#_ANEXO_#20:_Especificación). Para este análisis se debe considerar el plan de acción para los riesgos encontrados y registrados en el correo cuyo formato consta en el [Anexo #13](#_ANEXO_#13:_Lineamientos) con el fin de realizar las actividades necesarias según haya sido asignada su prioridad.

Jefe de Desarrollo de Sistemas:

1. Revisa y analiza la prioridad de las especificaciones funcionales en conjunto con el Equipo de Desarrollo, considerando los puntos de función y los atributos de calidad, si es necesario se despejan dudas y se vuelve a asignar puntos de función a las especificaciones en conflicto dentro del equipo. [Anexo #19](#_ANEXO_#20:_Especificación) .
2. Si el desglose de las especificaciones funcionales brinda una visión clara del proyecto para el Equipo de Desarrollo, se comienza con la planificación del sprint.
   1. Planificación de Sprints
3. Los sprints se planifican de forma iterativa, por lo que se debe planificar solo un sprint a la vez, y al finalizar la fase de puesta en producción, se debe planificar otro. El tiempo de cada sprint quedará a discreción del Equipo de Desarrollo bajo supervisión del Jefe de Desarrollo de Sistemas.

Equipo de desarrollo:

1. Selecciona las especificaciones funcionales con los puntos de función más altos en la EDT, se plantean y asignan las tareas puntuales para el cumplimiento de cada una de estas especificaciones entre los miembros del equipo. Se debe llevar el control de las tareas. [Anexo #22](#_ANEXO_#26:_Kanban) .
2. Se elabora la Planificación del sprint con las especificaciones funcionales y sus respectivas tareas para ser socializado con la Célula de Trabajo. [Anexo #21](#_ANEXO_#24:_Sprint).

Célula de Trabajo:

1. Se realiza una reunión con todos los miembros de la Célula de Trabajo, en la cual se analizan las especificaciones funcionales seleccionadas, validando la prioridad otorgada para su desarrollo en el sprint. Si existiesen dudas o conflictos de criterio respecto al Planificación del sprint entre los miembros de la Célula, el Propietario del Proceso tendrá voto decisivo.
2. Al validar la Planificación del sprint, se procede a acordar compromisos entre los miembros de la Célula de Trabajo.

Equipo de Desarrollo

1. Actualiza la EDT, en conjunto con el Jefe de Desarrollo, para la asignación de los responsables de cada tarea. Se hace la difusión pública de la EDT y se empieza la fase de diseño de la solución informática. ([Anexo #22](#_ANEXO_#23:_Planificación) ).
   1. Diseño de soluciones informáticas
2. El diseño de las soluciones informáticas será iterativo para agilizar el proceso de entrega continua. En caso de proyectos pequeños o puntuales, el diseño podrá hacerse en una sola iteración.

Célula de trabajo:

1. Se agenda y realiza una reunión para validar la propuesta del comportamiento del producto a entregar al final del sprint. Si la solución es clara y el comportamiento refleja visiblemente las necesidades del Propietario del Proceso, se acepta la propuesta y se procede con el prototipado. Caso contrario, se reevalúa y se realizan los ajustes pertinentes.

Desarrollador de Sistemas:

1. Realiza el desglose de la Planificación del sprint para realizar los diagramas necesarios, y representar la solución a desarrollar, su comportamiento deseado y los recursos a utilizar. Los diagramas desarrollados serán a discreción del Desarrollador de Sistemas.

Célula de Trabajo:

1. Revisa los diseños de la solución, si existen cambios en el análisis, se elabora una propuesta detallada con el Equipo de Desarrollo, misma que se presenta al dueño del producto para su aprobación y se envían nuevamente los Lineamientos del Proyecto. Se debe modificar el EDT y volver a planificar el sprint si amerita. Si no es necesario realizar cambios en el análisis y el diseño propuesto es correcto, empieza la fase de desarrollo.
2. Si es necesario para el desarrollo del proyecto, se solicita al Jefe de Infraestructura Tecnológica la creación de la base de datos en el ambiente de desarrollo y pruebas.

Desarrollador de Sistemas:

1. Elabora el diseño en base a la documentación entregada. Primero define los requerimientos que debe contener la base de datos, realiza el diseño entidad relación y luego genera el diseño físico de acuerdo con las bases de datos a utilizarse usando una herramienta de modelado de bases de datos. [Anexo #2](#_ANEXO_#3:_Herramientas).
2. Las tareas se dividen entre los miembros del equipo para realizar el trabajo en simultáneo, dependiendo del número de desarrolladores asignados se deben dividir para trabajar en el diseño de la base de datos y el prototipo.
3. Realiza las pruebas al diseño de la base de datos en conjunto con el Arquitecto de Sistemas, estas consisten en preparar un conjunto de casos con datos ficticios y verificar si el diseño soporta todos los casos y que tanto el prototipo como la base de datos no tengan inconsistencias entre ellos. Si la base de datos está obviando algún campo o columna que se está presentando en el prototipo, o viceversa, se debe analizar y corregir dichas inconsistencias.
4. Realizará los prototipos de las interfaces de usuarios, considerando el análisis previo, las especificaciones funcionales y los diagramas de proceso realizados. La herramienta para el prototipado se define en el [Anexo #12](#_ANEXO_#13:_Prototipado).
5. Analiza y valida la necesidad de crear nuevos servicios para desarrollar la solución, en caso de no ser necesario se omitiría la elaboración y revisión de arquitectura de software y hardware, caso contrario, debe elaborar el diseño de la arquitectura tanto de software como de hardware, las herramientas a utilizar para su elaboración quedan a discreción del Equipo de Desarrollo.
6. Ejecuta el script en la base de datos, este script es generado automáticamente usando una herramienta para el modelado de base de datos, [Anexo #2](#_ANEXO_#3:_Herramientas).
7. Genera el diccionario de datos, esto puede ser generado automáticamente usando una herramienta de modelado de base de datos, siempre y cuando los Desarrolladores de Sistemas Informáticos hayan documentado todas las tablas y campos de la base de datos.
8. Revisa los recursos existentes para su desarrollo para realizar el análisis de la solución, considera y decide si es necesario realizar nuevos diseños para los requerimientos del proyecto y decide. Si no es necesario, se planean ajustes a los recursos existente con los miembros del Equipo de Desarrollo y se empieza con la fase de desarrollo.
9. Analiza y valida la necesidad de realizar un prototipado con los recursos existentes, en caso de no ser necesario, realizará ajustes a los componentes existentes necesarios para el desarrollo.

Jefe de Desarrollo de Sistemas:

1. Analiza y valida, en conjunto con el Arquitecto de Sistemas y Equipo de Desarrollo, la correcta elaboración de la arquitectura de software y hardware procurando que el diseño permita una solución óptima, escalable, sostenible y seguro.
2. Valida si el diseño de la base de datos y el prototipo se encuentra conforme a lo establecido en las especificaciones funcionales, analiza si se pueden realizar optimizaciones o mejoras, y si se apega a los estándares y necesidades del Departamento de Desarrollo.

Jefe de Operaciones de Tecnología de la información:

1. Revisa los diseños de arquitectura de software y hardware para garantizar la disponibilidad y capacidad de los recursos necesarios para la elaboración de la solución propuesta por el Desarrollador.
   1. Desarrollo de soluciones informáticas.

Equipo de Desarrollo:

1. El desarrollador encargado deberá crear la rama de desarrollo en el sistema de control de versiones basado en la rama de producción y sobre ella desarrolla el requerimiento solicitado, conforme el [Anexo #3](#_ANEXO_#4:_Herramientas) en la sección de Sistema de Control de Versiones.
2. El desarrollador deberá realizar la construcción y configuración del pipeline para automatización de pruebas unitarias y despliegue.
3. El desarrollador procede a codificar los requerimientos aplicando los estándares de programación definidos en el [Anexo #7](#_￼ANEXO_#8:_Estándares), la arquitectura recomendada en el [Anexo #6](#_ANEXO_#7:_Arquitectura) y usando las herramientas para el desarrollo detalladas en el [Anexo #3](#_ANEXO_#4:_Herramientas);
4. Para la codificación del requerimiento se empleará la técnica de 'Pair Programming'. En este enfoque colaborativo, el desarrollador de front-end y el desarrollador de back-end trabajarán juntos en la implementación. El desarrollador de front-end se centrará en el análisis de la solución, interfaz de usuario y la experiencia del usuario, mientras que el desarrollador de back-end se enfocará en la lógica de negocio y la gestión de datos. Ambos Desarrolladores de Sistemas se alternarán en los roles de conductor y observador, colaborando estrechamente para garantizar la integración efectiva de los componentes front-end y back-end, y así entregar una solución completa y funcional que cumpla con los requisitos establecidos.
5. Ejecutar pruebas unitarias en función del plan de pruebas, para validar que sus propios desarrollos están correctos, previo al paso de las pruebas técnicas, si se trata de aplicaciones expuestas a internet deberá considerar las recomendaciones del [Anexo #9](#_ANEXO_#10:_Desarrollo) Además, es recomendable la utilización de las herramientas de análisis explicadas en el [Anexo #4](#_ANEXO_#5:_Herramienta) con el fin que estas herramientas nos ayuden a identificar posibles huecos de seguridad, vulnerabilidades, bugs, etc.
6. Durante el proceso de desarrollo, el equipo de desarrollo deberá registrar en el sistema de control de versiones cada finalización de una tarea, respetando los tiempos planificados:

* Los escenarios de pruebas utilizados para la ejecución de sus pruebas unitarias;
* La documentación del código fuente, conforme se establece en el [Anexo #7](#_￼ANEXO_#8:_Estándares);
* Toda la documentación generada de la solución informática, tanto técnica como normativa;

1. Las actualizaciones de los procedimientos almacenados utilizados para cumplir con el requerimiento, cuyo código también debe ser documentado.
2. Una vez validado el desarrollo con la célula de trabajo, el Equipo de Desarrollo elabora el expediente técnico de las soluciones informáticas que se han desarrollado en el área de Desarrollo de Sistemas. Referirse al [Anexo #5](#_ANEXO_#6:_Herramientas) “Herramientas de Documentación” del presente documento.
3. Se procederá a la creación de la bitácora de errores (Milestone) con el formato PT\_Nombre\_requerimiento, conforme al [Anexo #15](#_ANEXO__#16:), y en la misma se deberá adjuntar la documentación para la realización de las pruebas técnicas.
4. Se dará por culminado el desarrollo del sprint y se solicitará al Gestor de Control de Calidad la ejecución de pruebas técnica, cuyo formato se establece en la solicitud de ejecución de pruebas de usuario, detallado en el [Anexo #16](#_ANEXO_#17:_Solicitud).

Jefe de Desarrollo de Sistemas:

1. El Equipo de Desarrollo socializará con el Jefe de Desarrollo de Sistemas la culminación del desarrollo del Sprint asignado, el cual será el encargado de realizar las revisiones que considere necesarias previas a la presentación del sprint con la célula de trabajo y la solicitud de pruebas técnicas.

Célula de trabajo

1. Después de culminar el desarrollo del Sprint, este será socializado y presentado con la célula de trabajo, con la finalidad de presentar la solución al requerimiento y verificar si se ajusta o no a las necesidades de los Stakeholders. Si este no se ajustara a las necesidades o atributos de calidad previamente definidos, se deberá retornar a la fase de diseño.
   1. Pruebas Técnicas

Analista de Control de Cambios:

1. Después de terminar la fase de desarrollo, el Analista de Control de Cambios será el responsable de probar la solución informática para asegurarse de que el proyecto esté libre de errores y tenga una alta calidad.
2. Verifica que en el sistema de control de versiones se encuentre cargada todo el expediente técnico:
   * Toda la documentación generada en el desarrollo (historias de usuario, prototipos, manual de usuario actualizado, documentación del código fuente, diccionario de datos, matriz de parametrizaciones, instructivo para puesta en producción, diseños de las bases de datos).
   * Todos los procedimientos almacenados, scripts, etc., utilizados para cumplir con el requerimiento y listos para puesta en producción.
   * Los escenarios de pruebas que el Desarrollador de Sistemas informáticos usó para realizar sus pruebas unitarias.

Si encuentra alguna inconsistencia le debe comunicar al equipo de desarrollo responsable para su corrección, si todo se encuentra bien se comienza las pruebas técnicas.

1. Debe levantar sus escenarios de prueba adicionales al plan de pruebas proporcionado por el equipo de desarrollo, los cuales se crearán a partir de:
   * Los escenarios y casos de pruebas que el Desarrollador de Sistemas informáticos usó para sus pruebas unitarias.
   * La estructura desglosada de trabajo, en la cual se puede evidenciar todas las tareas que se deben completar para cumplir con los requisitos de alto nivel y las especificaciones funcionales.
   * La documentación generada durante el desarrollo del requerimiento solicitado, como es: prototipo, diseño de base de datos, diagramas de flujo, documento de requerimiento, etc.
2. Durante la etapa de revisión de código fuente y estándares, se deberá llevar a cabo una revisión exhaustiva del código fuente desarrollado para dar cumplimiento a las políticas de seguridad implementadas por la Cooperativa. El objetivo principal de esta revisión es evaluar la calidad y consistencia del código, así como asegurarse de que cumple con los estándares y directrices definidos por el equipo de desarrollo y la organización.

* Se deberá revisar cada componente del código fuente desarrollado, identificando posibles errores, inconsistencias o prácticas que no cumplan con los estándares definidos en la presente metodología, según el [Anexo #9](#_ANEXO_#9:_Desarrollo). Se verifica la legibilidad del código, la documentación, la estructura y se buscan posibles vulnerabilidades de seguridad.
* Se revisará que el expediente técnico elaborado por el equipo de desarrollo cumpla con la estructura del expediente técnico, según el [Anexo #5.](#_Herramientas_de_documentación)
* Además, se garantiza que se sigan las mejores prácticas de programación y se cumplen los estándares de codificación establecidos, lo que contribuye a la mantenibilidad y escalabilidad del software a lo largo del tiempo.

1. Se analizan las dependencias utilizadas en el proyecto para evitar dependencias con vulnerabilidades u obsoletas. Se puede utilizar las herramientas descritas en el [Anexo#6](#_ANEXO__#5:)
2. Se analiza el código fuente de las aplicaciones con el fin de identificar vulnerabilidades, huecos de seguridad, bugs u optimizaciones. Se puede utilizar herramientas las herramientas descritas en el [Anexo #4](#_ANEXO_#4:_Herramienta)**.**
3. Si la solución informática será expuesta en el Internet, se deberá adicionalmente validar las seguridades haciendo uso de la herramienta detallada en el [Anexo #4](#_ANEXO_#5:_Herramienta) (OWASP ZAP). Cerciorándose de que se cumpla con los controles necesarios que ayuden a evitar las vulnerabilidades descritas en el [Anexo #9](#_ANEXO_#10:_Desarrollo).
4. Las pruebas técnicas se deberán realizar sobre el ambiente de QA, solo en motivos de fuerza mayor donde el ambiente no esté disponible o no existan los recursos necesarios en este ambiente se podrá realizar las pruebas en otro ambiente, para lo cual se deberá publicar todo en este ambiente como si estuviera poniendo en producción.
5. En base al plan de pruebas elaborado, el encargado de realizar las pruebas técnicas, será el responsable de validar los escenarios de pruebas propuestos, si encuentra algún problema, deberá registrarlo en la bitácora de errores utilizada por el Equipo de Desarrollo.
6. Irá reportando cada problema detallando lo máximo posible cada uno, incluyendo los datos con los que probó, pantallas capturadas, explicación del problema, detalle de cada paso que se debe dar para que el Equipo de Desarrollo pueda replicar dicho problema y pueda corregirlo.
7. Una vez ejecutada esta etapa, todos los resultados de la revisión del código fuente, e incidencias encontradas serán reportadas en la bitácora de errores para que el equipo de desarrollo tome las medidas necesarias.
8. Socializa la culminación de las pruebas técnicas, una vez cerrados todos los problemas reportados notifica al Equipo de Desarrollo, Arquitecto de Sistemas y Jefe de Desarrollo, mediante correo electrónico, la culminación de las pruebas técnicas.

Equipo de Desarrollo:

1. El equipo de desarrollo respaldará la última versión, esta documentación siempre debe reposar en el sistema de control de versiones para el libre acceso de cualquier integrante del proyecto. Los escenarios y casos de pruebas levantados se reutilizarán durante las pruebas unitarias, técnicas y de usuario.
2. Para corregir los problemas reportados en la bitácora de errores se debe analizar, identificar, replicar y corregir los mismos; cada vez que corrija una incidencia se deberá cambiarle de estado a POR VALIDAR en el sistema de control de versiones y una vez culminado de corregir todos, deberá comunicarle al Analista de Control de Cambios para que valide la corrección.
3. Si existen conflictos de criterios el equipo de desarrollo analizará estos junto con el Jefe de desarrollo, en el caso que amerite su corrección se codificara los ajustes solicitados y actualizará el repositorio
4. Reporta errores corregidos en la bitácora de errores, luego el Analista de Control de Cambios verificara la seguridad y estándares en el código fuente.
5. Revisa si cada uno de los problemas en estado POR VALIDAR efectivamente han sido corregidos; de ser así, les cambia de estado ha CERRADO, caso contrario los pondrá como REABIERTO, detallando la fecha de la prueba, el problema encontrado y comunicará al Equipo de Desarrollo responsable para que vuelva a corregir los problemas reabiertos o nuevos.
6. Cada problema reabierto deberá constar con todos los sustentos necesarios para facilitar al Equipo de Desarrollo su reproducción y posterior corrección.

Célula de trabajo:

1. El Jefe de Desarrollo, en conjunto con el Arquitecto de Sistemas gestionan los errores reabiertos en los que, por temas técnicos, negocio, agentes externos, etc., el Equipo de Desarrollo no está de acuerdo con lo planteado por el encargado de las pruebas. En estos casos actúan como mediadores para que se desarrolle una corrección viable o en su defecto se justifique el cierre del error sin la corrección solicitada. En caso de ser necesario se solicitará la intervención de otros miembros de la célula de trabajo, con la finalidad de sanear el conflicto definitivamente.

* 1. [Pruebas de Usuario](#PruebaAlfa)

Analista de Control de Cambios:

1. Verifica que en el sistema de control de versiones se encuentre:
   * Todo el expediente técnico generado en el desarrollo (historias de usuario, prototipos,
   * manual de usuario actualizado, documentación del código fuente, diccionario de datos,
   * matriz de parametrizaciones, instructivo para puesta en producción, diseños de las
   * bases de datos).
   * Todos los procedimientos almacenados, scripts, etc., utilizados para cumplir con el requerimiento
   * Verificar el plan de pruebas.
   * Si encuentra alguna inconsistencia le debe comunicar al Desarrollador de Sistemas responsable para su corrección, si todo se encuentra bien se bloquea el sistema de control de versiones, para evitar que existan cambios de último movimiento por parte del Desarrollador de Sistemas informáticos.
2. Bloquea la rama del control de versiones para que el Desarrollador de Sistemas informáticos no pueda incluir ningún cambio de última hora, sin permiso del analista de control de cambios.
3. Preparara el entorno para las pruebas de usuario, en la cual se debe constatar el correcto funcionamiento de solución informática, en base al plan de pruebas, que podría ser actualizado por el Analista de Control de Cambios si así lo considera
4. Se planificará con el dueño del producto e integrantes de la célula de trabajo para definir el: lugar, la fecha y la hora en que se efectuará la exposición del sistema, en el cual, el Analista de Control de Cambios expondrá los cambios y/o mejoras definidas en la fase de análisis.
5. Revisa y documenta el resultado de las pruebas de usuario.
6. En caso de haber encontrado errores en la solución informática o la misma no cumpla los criterios de aceptación de la célula de trabajo, se registrarán en la Bitácora de Errores, lo más detallado posible, todas las observaciones realizadas por la célula de trabajo durante las pruebas de usuario, la idea es que el equipo de desarrollo pueda analizar estas observaciones y, si amerita, realizar las respectivas correcciones solicitadas. Una vez reportados los errores debe notificar vía correo electrónico al Jefe de Desarrollo de Sistemas, equipo de desarrollo y Arquitecto de Sistemas.
7. Desbloquear las ramas del sistema de control de versiones para que el equipo de desarrollo pueda incluir los cambios solicitados
8. Cuando todas las observaciones han sido solventadas, se bloqueará nuevamente las ramas del sistema de control de versiones
9. Si no se han registrado observaciones durante las pruebas de usuario, el Analista de Control de Cambios gestionará la configuración para el despliegue en producción y la aprobación del formulario de “Registro de puesta en producción” junto con la documentación de las pruebas. [Anexo #19](#_ANEXO_#19:_Registro), este documento se lo debe hacer firmar por el dueño del producto.

Célula de trabajo:

1. Luego de revisar y analizar los criterios del equipo de desarrollo y el Analista de control de Cambios, en base al producto mínimo viable requerido, se define si algún incidente pendiente de corregir es un error o un nuevo requerimiento y si se puede dejar para el siguiente sprint. Si se trata de un nuevo requerimiento, se gestiona con el dueño del producto el compromiso de su revisión e implementación para el próximo sprint. Si se obtiene la aceptación del dueño del producto, el desarrollo pasa a producción; por otro lado, si el incidente es un error se lo debe corregir inmediatamente. Si no se obtiene la aceptación de la célula de trabajo para la puesta en producción de la versión actual, o si depende de otro desarrollo, se da por cerrado el sprint y se retornará a la fase de diseño.

Dueño del producto:

1. De acuerdo con el plan de pruebas, el dueño del producto o su(s) delegado(s), en conjunto con la célula de trabajo, probarán la solución informática para certificar el correcto funcionamiento y el cumplimiento de las expectativas planteadas en los requerimientos. Estas pruebas deben ser lo más parecido a como se realiza el trabajo en el día a día. El Analista de Control de Cambios deberá guiar la ejecución de estas pruebas. El dueño del producto o su(s) delegado(s) deberán reportar al Analista de control de Cambios las observaciones identificadas en el sistema.
2. Si el dueño del producto o su(s) delegados(s) no identifican errores, pero consideran que el desarrollo no cumple con los criterios de aceptación definidos en la fase de análisis, procederán a solicitar las correcciones que consideren necesarias.
3. Firma el formulario de “Registro de puesta en producción” [Anexo #18](#_ANEXO_#19:_Registro) para dejar constancia de su participación en las pruebas de la solución informática.
   1. Puesta en Producción de las Soluciones Informáticas.

Analista de Control de Cambios:

1. Revisa el expediente técnico y código fuente.
2. Constata que todo se encuentre de acuerdo con la especificación de instalación y la estructura de la solución informática. Si la documentación no se encuentra de acuerdo con lo desarrollado en la solución informática solicita realizar los respectivos cambios al Equipo de Desarrollo.
3. Solicitará parametrización en caso de ser necesario.
4. Valida que el formulario de puesta en producción se encuentre legalizado.
5. Sincroniza el sistema de control de versiones del código y base de datos.

Jefe de Seguridad de la Información:

1. Si es necesario realizar una parametrización, debe enviar mediante correo electrónico la solicitud de parametrización al oficial de seguridad de la información, si no es necesario la parametrización debe entrega al Jefe de Infraestructura Tecnológica el formulario de “Registro de puesta en producción” [Anexo #18](#_ANEXO_#18:_Registro), para la generación de las bases de datos, tablas y procedimientos de las soluciones informáticas desarrollados o actualizados, en caso de que se requiera.
2. El jefe de Seguridad de la Información confirmará por medio de correo electrónico la ejecución de la parametrización.

Jefe de Operaciones de Tecnología de la Información:

1. Respalda la Base de datos de producción.
2. Ejecuta en el servidor de producción la última versión de los scripts y creación de bases de datos, tablas y procedimientos almacenados, de acuerdo a lo especificado en el formulario: “Registro de Puesta en Producción”.
3. Configura el proceso de construcción y despliegue automatizado.
4. Cambiar variables de entorno de desarrollo a producción.
5. Acepta el despliegue en producción
6. Proceder a realizar Monitoreo y revisión de logs con la finalidad de identificar si existió algún problema durante la publicación del sistema
7. Si se identifica algún error, el Jefe de Operaciones de TI deberá reversar los sistemas a la última versión estable y comunicar a la Célula de Trabajo sobre lo acontecido y al Equipo de Desarrollo para que realice las correcciones necesarias.

**Equipo de Desarrollo**

1. Analiza e identifica el problema y realiza las correcciones necesarias.

Analista de control de cambios:

1. Si la puesta en producción fue exitosa, el analista de control de cambios sincroniza la rama máster del proyecto de base de datos con los procedimientos almacenados desarrollados o actualizados para el desarrollo de la solución informática.
2. Comunica a la célula de trabajo el despliegue exitoso de la solución informática.
   1. Plan de entrenamiento a los usuarios

Célula de trabajo:

1. Define alcance y la modalidad que se usará para ejecutar el entrenamiento a los usuarios finales de la solución informática, pudiendo realizarse utilizando la modalidad más adecuada para el entrenamiento.

**Oficinista de tecnología**

1. Elabora el material referente al entrenamiento conforme al alcance definido y el manual de usuario de las soluciones informáticas de acuerdo con los estándares definidos en el presente documento.

**Célula de trabajo:**

1. Revisa el material desarrollado y constata que todo se encuentre de acuerdo a las especificaciones y alcance del proyecto. Si el manual, instructivo o plan de entrenamiento no se encuentra de acuerdo con lo desarrollado en la solución informática, solicita realizar los respectivos cambios al Oficinista de Tecnología.
2. Una vez listo el material, la célula asigna un representante que llevará a cabo el entrenamiento.
3. El representante de la célula llevará a cabo el entrenamiento, para lo cual convoca a los usuarios de la solución informática mediante correo electrónico y ejecuta el entrenamiento según la modalidad establecida.
4. El Analista de Control de Cambios publica la versión más reciente del manual de usuario en la solución informática para que pueda ser accedido por los usuarios.
5. Custodia el expediente técnico y el manual de usuario de la solución informática desarrollada
6. ANEXOS

## ANEXO #1: Herramientas para el análisis

El Desarrollador de Sistemas pueden hacer uso de diferentes técnicas que le permitan realizar de una mejor manera la recolección de información para la definición de las historias de usuario. Estas técnicas pueden ser:

**Entrevistas**

El Desarrollador de Sistemas se reúne con el/los usuarios para entablar un diálogo que le permitirá obtener información detallada de los requerimientos solicitados, las entrevistas pueden ser clasificadas en dos grandes grupos:

• Las entrevistas **cerradas,** donde el Desarrollador de Sistemas prepara un conjunto de preguntas antes del encuentro con el usuario, y se buscan respuestas para las preguntas formuladas.

• Las entrevistas **abiertas,** en las cuales no se preparan preguntas concretas, y, por el contrario, se discute con el usuario las expectativas que este tiene de la solución informática.

**Observación.**

Los métodos de observación involucran a dos participantes, el Desarrollador de Sistemas y el usuario de la solución informática:

El Desarrollador de Sistemas observa al usuario mientras trabaja y toma nota de las actividades que realiza cotidianamente en su trabajo. Esta técnica permite observar lo que se hace actualmente en un determinado proceso y reconocer ciertos subprocesos que pueden automatizarse dentro de la solución informática, además es posible que se puedan plantear mejoras al proceso actual, reconocer formatos, documentos, proveedores, sistemas involucrados, etc.

**Análisis de documentación.**

Algunos requerimientos solicitados se basan en nuevas normas, políticas o regulaciones internas y/o externas, por lo que el Desarrollador de Sistemas debe solicitar al usuario la documentación para ser analizada y complementar la historia de usuario.

**Prototipos.**

Esta técnica permite al usuario visualizar o experimentar la solución informática propuesta para cumplir con sus requerimientos, esto le permitirá tener una idea más clara de la solución informática a desarrollar, logrando aclarar o mejorar las especificaciones de requerimientos. Los prototipos puede ser una versión reducida o un conjunto de pantallas de la solución informática, no debe ser necesariamente funcional.

**Historia de Usuario**

Una vez recopilada la información necesaria, se procede a la realización de la historia de usuario, la cual contiene el detalle de los requerimientos del usuario y la forma que la solución informática los solventará, el formato de este documento se encuentra en el [Anexo #13](#_Historias_de_Usuario). Este documento podrá incluir al prototipo, referencias de documentos, correos adjuntos, etc. Una vez finalizado el Desarrollador de Sistemas deberá gestionar la aprobación de este documento al dueño del producto. El usuario al firmar acepta y aprueba el inicio de la etapa de desarrollo. Este entregable es gestionado y custodiado por el Desarrollador de Sistemas, si existiese una modificación se debe manejar un historial de versiones que deben ser autorizadas por el dueño del producto.

## ANEXO #2: Herramientas de diseño

**Diseño entidad – relación (modelo lógico)**

El diseño entidad – relación (ER) es una herramienta que permite describir de forma abstracta cómo se representan los datos de una solución informática de información. Este diseño expresa entidades relevantes para una solución informática de información, sus interrelaciones y propiedades.

Mediante el diseño ER se pretende visualizar los objetos que pertenecen a la base de datos como entidades, las cuales tienen unos atributos y se vinculan mediante relaciones.

**Entidades:** Una entidad es cualquier "objeto" discreto sobre el que se tiene información. Se representa mediante un rectángulo o "caja" etiquetada en su interior mediante un nombre. Ejemplos de entidades habituales en las soluciones informáticas son: factura, persona, empleado, etc.

Empleado

Cada ejemplar de una entidad se denomina instancia. Por ejemplo, Tesorería y Contabilidad pueden ser dos instancias distintas de la entidad "departamento". Las instancias no se representan en el diagrama. No obstante, se pueden documentar aparte porque son útiles para inicializar la [base de datos](http://es.wikipedia.org/wiki/Base_de_datos) resultante.

**Relaciones:** Una relación describe cierta interdependencia (de cualquier tipo) entre entidades. Se representa mediante un rombo etiquetado en su interior mediante un verbo. Además, dicho rombo debe unirse mediante líneas con las entidades que relaciona (es decir, los rectángulos).

Tiene

Una relación no tiene sentido sin las entidades que relaciona. Algunos ejemplos son: una persona (entidad) trabaja (relación) para un departamento (entidad).

**Atributos:** Los atributos son propiedades relevantes propias de una entidad y sólo una. Se representan mediante un círculo o elipse etiquetado mediante un nombre en su interior. Cuando un atributo es identificativo de la entidad se suele subrayar dicha etiqueta.

Por motivos de legibilidad, los atributos no suelen representarse en un diagrama entidad-relación, sino que se describen textualmente en otros documentos adjuntos.

Los atributos describen información útil sobre las entidades. En particular, los atributos identificativos son aquellos que permiten diferenciar a una instancia de la entidad de otra distinta. Por ejemplo, el atributo identificativo que distingue a un empleado de otro es su número de cédula.

Ejemplos de atributos de la entidad "persona":

* Número de cédula (identificativo)
* Nombre
* Apellidos
* Dirección
* Código postal

**Entidades Fuertes y Débiles:**

Cuando una entidad participa en una relación puede adquirir un papel fuerte o débil. Una entidad débil es aquella que no puede existir sin participar en la relación, es decir, aquella que no puede ser unívocamente identificada solamente por sus atributos. Una entidad fuerte es aquella que si puede ser identificada unívocamente. En los casos en que se requiera, se puede dar que una entidad fuerte "preste" algunos de sus atributos a una entidad débil para que, esta última, se pueda identificar.

Las entidades débiles se representan mediante un doble rectángulo, es decir, un rectángulo con doble línea.

**Cardinalidad de las Relaciones:**

Las relaciones, en principio binarias, pueden involucrar a un número distinto de instancias de cada entidad. Así, son posibles tres tipos de cardinalidades:

* Relaciones de uno a uno: una instancia de la entidad A se relaciona con una y solamente una instancia de la entidad B.
* Relaciones de uno a muchos: cada instancia de la entidad A se relaciona con varias instancias de la entidad B.
* Relaciones de muchos a muchos: cualquier instancia de la entidad A se relaciona con cualquier instancia de la entidad B.

El tipo de Cardinalidad se representa mediante una etiqueta en el exterior de la relación, respectivamente: **"1:1"**, **"1: N"** y **"N: M"**, aunque la notación depende del lenguaje utilizado, la que más se usa actualmente es el unificado. Otra forma de expresar la Cardinalidad es situando un símbolo cerca de la línea que conecta una entidad con una relación:

* "0" si la entidad no está obligada a participar en la relación.
* "1" si la entidad está obligada a participar en la relación y, además, cada instancia solamente participa una vez.
* "N", "M", ó "\*" si la entidad no está obligada a participar en la relación y cada instancia puede participar cualquier número de veces.

Ejemplos de relaciones que expresan Cardinalidad:

* Una factura (entidad) se emite (relación) a una persona (entidad) y sólo una, pero una persona puede tener varias facturas emitidas a su nombre. Es una relación 1:N.
* Un cliente (entidad) puede comprar (relación) varios artículos (entidad) y un artículo puede ser comprado por varios clientes distintos. Es una relación N: M.

Para el modelado de la base de datos se puede hacer uso de herramientas software que agilizarán este proceso, generalmente estas herramientas permiten: crear, modificar y/o editar un modelo de proceso de negocio, definir y usar las reglas de negocio, etc., sus principales elementos son:

* **Relación:** Es la relación básica del modelo relacional, se representa mediante una tabla.
* **Dominio:** Es el conjunto válido de valores que toma un atributo. Existen con independencia de cualquier otro elemento.
* **Atributo:** Representa las propiedades de la relación. Se representa mediante una columna.
* **Tupla:** Es una ocurrencia de la relación. Se representa mediante una fila.
* **Súper clave:** Es un subconjunto de atributos que permite distinguir unívocamente cada una de las entidades de un conjunto de entidades. Si se añade un atributo al anterior subconjunto, el resultado seguirá siendo una súper clave.
* **Clave candidata:** Dada una súper clave, si ésta deja de serlo quitando únicamente uno de los atributos que la componen, entonces ésta es una clave candidata.
* **Clave primaria:** Es una clave candidata, elegida por el diseñador de la base de datos, para identificar unívocamente las entidades en un conjunto de entidades.

Al finalizar el modelado de la base de datos, estas herramientas nos permiten obtener:

1. Restricciones y normalizaciones de la base de datos.
2. Diccionario de datos.
3. Código ejecutable para la creación, actualización, eliminación de componentes de la base de datos.

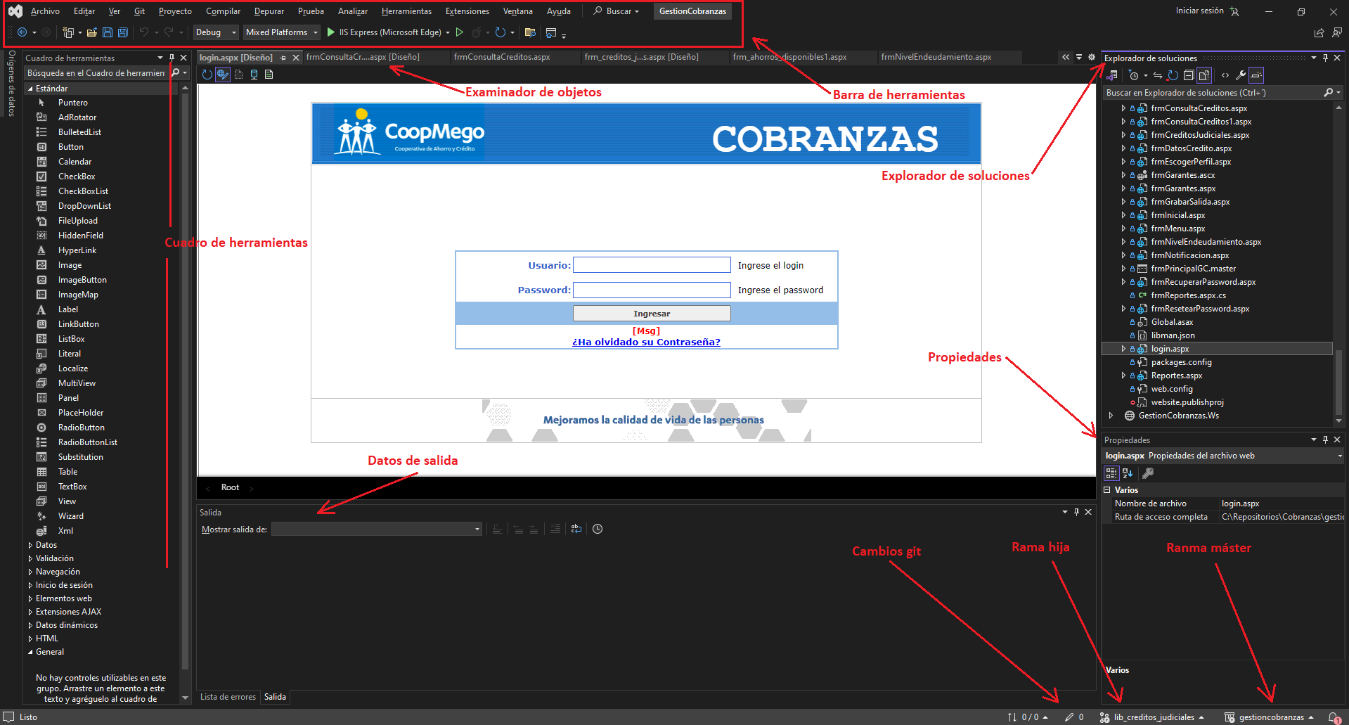
## ANEXO #3: Herramientas de desarrollo

**Herramienta de programación**

* **Microsoft Visual Studio**

Visual Studio es una herramienta de desarrollo eficaz que permite completar todo el ciclo de desarrollo en un solo lugar. Es un entorno de desarrollo integrado (IDE) completo que puede usar para escribir, editar, depurar y compilar el código y, luego, implementar la aplicación.

**Entorno de Visual Studio**

****

**Editor / Navegador**

El editor/navegador es la ventana de interfaz principal de Visual Studio .NET. En modo editor, nos muestra el código para editar y nos proporciona una interfaz gráfica para la ubicación de los controles. Podemos arrastrar y soltar para crear el diseño visual de nuestra aplicación. Hay dos opciones para pantalla del editor: modo Diseño y modo HTML:

* + **Modo Diseño**: En modo Diseño, el editor le permite mover los controles y los elementos gráficos por la ventana mediante una sencilla operación de arrastrar y soltar. Visual Studio .NET proporciona dos esquemas de posicionamiento de controles para diseñar páginas Web.
  + **Modo HTML**: En modo HTML, Visual Studio .NET destaca el código para que los distintos elementos, como los nombres de variables y las palabras clave, sean identificables al instante.
* **Examinador de objetos**

El Examinador de objetos es una herramienta que proporciona información sobre los objetos y sus métodos, propiedades, eventos y constantes.

* **Explorador de soluciones**

El Explorador de soluciones muestra la jerarquía de los archivos del proyecto. Desde esta ventana, se puede mover y modificar archivos. Los tipos de archivos que muestra el Explorador de soluciones incluyen:

* + **Referencias de proyectos** que listan las clases que utilizan la página y los controles Web. Todos los formularios Web Forms del proyecto. Todas las páginas de código subyacente que contienen la lógica que soporta los formularios Web Forms. Carpetas relacionadas con proyectos y sub-elementos.
  + **Propiedades:** Visual Studio .NET permite ajustar las propiedades de documentos, clases y controles utilizando una ventana común de Propiedades. Cuando creamos o seleccionamos un elemento, la ventana Propiedades muestra automáticamente las propiedades relacionadas.
* **Barra de herramientas**

La barra de herramientas se encuentra justo debajo del menú de navegación y contiene botones para acciones comunes como guardar, deshacer y rehacer, entre otros**.**

* **Cuadro de herramientas**

El Cuadro de herramientas permite arrastrar y soltar los controles que formarán nuestra aplicación. Las herramientas disponibles se agrupan por categorías en los siguientes menús:

* + **Datos:** Esta categoría contiene objetos que permiten a la aplicación conectarse y acceder a datos de una base de datos Microsoft SQL Server y otras bases de datos.
  + **Estándares** Esta categoría contiene un conjunto de controles de servidor que pueden agregarse a páginas Web.

**Ventajas:**

* Visual Studio provee una serie de herramientas para desarrollo, así como características de debugging, funcionalidad en base de datos y características innovadoras para la creación de aplicaciones en diferentes plataformas.
* Incluye un diseñador visual para desarrollo rápido, esto nos ayuda mucho para desarrollar en web, además nos provee de todas las herramientas y el [Framework](#Framework) para poder crear aplicaciones web con el soporte de ASP.NET, AJAX.
* Provee de un nuevo lenguaje de consultas integrado para el manejo de la información, (LINQ), para poder construir soluciones que analicen y actúen sobre la información.
* Provee al Desarrollador de Sistemas informáticos la habilidad de poder escoger entre múltiples versiones del [Framework](#Framework) con el mismo entorno de desarrollo.
* **Sistema de Control de Versiones.**

Un sistema de control de versiones pertenece a una categoría de herramientas de software que ayudan al equipo de desarrollo a gestionar los cambios realizados en el código fuente a lo largo del tiempo. El software de control de versiones realiza un seguimiento de cada modificación del código en un tipo especial de base de datos. Si se comete un error, el Desarrollador de Sistemas puede retroceder el tiempo y comparar versiones anteriores del código para ayudar a solucionar el error y minimizar la interrupción para todos los miembros del equipo.

**Ventajas de utilizar un sistema de control de versiones:**

* Ayuda al equipo a evitar la pérdida accidental de archivos.
* Mantiene un histórico de todo el desarrollo del proyecto, permitiendo realizar un seguimiento de las versiones anteriores de uno o más archivos.
* Añade trazabilidad al desarrollo de software, ya que se puede ver qué cambios se han hecho en el código en cada versión.
* Muestra mucha información estadística de cómo se está desarrollando el proyecto (principales autores, número de versiones, cambios, etc.).
* Facilita mucho el trabajo en equipo.
* Permite desarrollar varias versiones de un mismo programa a la vez.

La implementación de un sistema para el control de versiones permite alcanzar los siguientes objetivos:

* **General:** 
  1. Administrar el código fuente de proyectos de software con un alto rendimiento, seguridad y flexibilidad para el equipo de desarrollo.
* **Específicos:** 
  + Conocer el historial de diferentes cambios realizados en los archivos a lo largo del tiempo que permita solucionar problemas en versiones anteriores de software.
  + Administrar múltiples flujos de trabajo independientes entre sí, sin afectar el trabajo de los miembros del equipo de desarrollo.
  + Dar seguimiento a los cambios realizados en el software mediante el manejo de versiones publicadas en ambiente de producción.

**Roles utilizados:**

En el sistema de control de versiones se hará uso de tres roles, los cuales se describen a continuación:

* **Guest:** Rol sin acceso al código, un usuario con el permiso privado más bajo Guest solo puede hacer uso de los problemas en un proyecto, wiki pages y no tiene acceso de lectura al código.
* **Reporter:** Rol con las mismas capacidades del Guest pero con permisos de lectura del código pero no se permite cambios.
* **Developer:** El Desarrollador de Sistemas tienen asignado este rol, el cual tiene permisos de lectura y escritura dentro del repositorio, permitiéndole crear ramas dentro del repositorio, realizar cambios en el código y unir sus cambios a otras ramas. Cabe recalcar que un usuario con este rol no puede unir sus cambios a ramas que estén protegidas (como lo es la rama “Master”).
* **Maintainer:** Es el rol más alto a nivel de proyecto, ya que recibe permisos para hacer de todo, excepto llevar la administración del repositorio. Este rol tiene el permiso de unir cambios a la rama principal (rama Master).
* **Administrador:** El usuario con este rol tiene todos los permisos de lectura y escritura, y lleva a cabo la administración de todos los repositorios.

**Flujo de versionado de código para proyectos**



* **Ramas Base:** Ramas principales del proyecto.
  + **Master:** Rama maestra por defecto del sistema de control de versiones en producción, de la que dependen las demás ramas. En esta rama se deben incluir únicamente los cambios que ya se ponen en producción junto con la etiqueta que identifica la versión de la solución informática que se está lanzando. Esta rama a cargo del usuario del rol “Maintainer”.
  + **Soporte:** Rama que se utiliza para realizar soportes de las soluciones informáticas, se usa para hacer ajustes urgentes que se requiere poner en producción inmediatamente luego de las pruebas unitarias del Desarrollador de Sistemas. Esta rama se crea a partir de la rama master y una vez que se realizan los cambios solicitados y se verifica que el funcionamiento de la solución informática es el deseado, se procede a unir dichos cambios con la rama master. Esta rama está a cargo del usuario con el rol “Developer”.
  + **Liberación:** A esta rama se cargan los cambios realizados por el Desarrollador de Sistemas y que han sido validados por el Analista de Control de Cambios, una vez que se han cargado dichos cambios, el usuario con el rol Maintainer tiene la responsabilidad de quitarle los privilegios al Desarrollador de Sistemas para que no pueda realizar ningún cambio sobre ella y poder iniciar con las pruebas de usuario. Si las pruebas de usuario han sido exitosas y se pone en producción la solución informática, se deberá unir los cambios de esta rama a la “Master” con la etiqueta de la nueva versión. Esta rama está a cargo del usuario con el rol “Maintainer”.
  + **Desarrollo:** Rama creada a partir de la master para iniciar con el desarrollo de una nueva funcionalidad. De esta rama el Desarrollador de Sistemas creará sus propias ramas hijas las cuales deberá configurar con las direcciones de desarrollo debido a que el proyecto se baja con las direcciones que apuntan a producción. Una vez que el Analista de Control de Cambios ha validado que la solución informática funciona correctamente con los nuevos ajustes, el Desarrollador de Sistemas debe subir los cambios a la rama “Liberación” para iniciar con el proceso de puesta en producción. Esta rama está a cargo del usuario con el rol “Developer”.
* **Ramas hijas:** Son ramas adicionales, creadas para versionado de cambios. Estas ramas están a cargo del usuario con el rol “Developer”.
* **des\_nombre\_proyecto:** Esta rama se crea a partir de la rama “Desarrollo” y apunta al entorno de desarrollo. A partir de esta se crearán ramas hijas para iniciar con los cambios solicitados.
* **des\_nombre\_proyecto\_login:** Esta rama se crea a partir de la rama “des\_nombre\_proyecto” y en esta rama el Desarrollador de Sistemas realizará los cambios que han sido solicitados. Una vez realizados los cambios deberá subirlos a la rama “des\_nombre\_proyecto”.
* **lib\_nombreproyecto:** Esta rama se crea a partir de la rama “des\_nombreproyecto” ya lista con todos los entregables y referencias de producción para comunicar al analista de control de cambios que puede proceder a protegerla, lista para pruebas de usuario o transacciones controladas.
* **Estándares en los commits:**

En el proceso de desarrollo de software se debe mantener una estructura consistente y comprensible en los mensajes de commit para facilitar la colaboración, seguimiento y gestión del código. Por ello, se establecen los siguientes estándares para los commits, basados en la Convención Conventional Commits:

|  |  |
| --- | --- |
| **Tipo** | **Descripción** |
| feat | Se introduce una nueva característica con los cambios |
| fix | Se ha producido una corrección de errores |
| chore | Cambios que no se relacionan con una corrección o característica y no modifican archivos de SRC o de prueba (por ejemplo, actualización de dependencias) |
| refactor | Código refactorizado que no corrige un error ni añade una característica |
| docs | Actualizaciones de la documentación, como el archivo README u otros archivos de referencias |
| style | Cambios que no afectan al significado del código, probablemente relacionados con el formato del código, como espacios en blanco, punto y coma faltantes, etc. |
| test | Incluyendo pruebas nuevas o que corrigen pruebas anteriores |
| perf | Mejoras de rendimiento |
| ci | Integración continua relacionada |
| build | Cambios que afectan al sistema de compilación o dependencias externas |
| revert | Revierte una confirmación anterior |

Además, se recomienda proporcionar un mensaje de commit descriptivo y conciso que explique claramente los cambios realizados y su propósito.

**Herramienta de Base de Datos**

Para la administración de la base de datos se hace uso de sistemas software que facilitan el acceso a la información de la base de datos relacionales, en el mercado existen una gran variedad de estas herramientas por lo que se recomienda hacer uso de las que cumplan con las siguientes características:

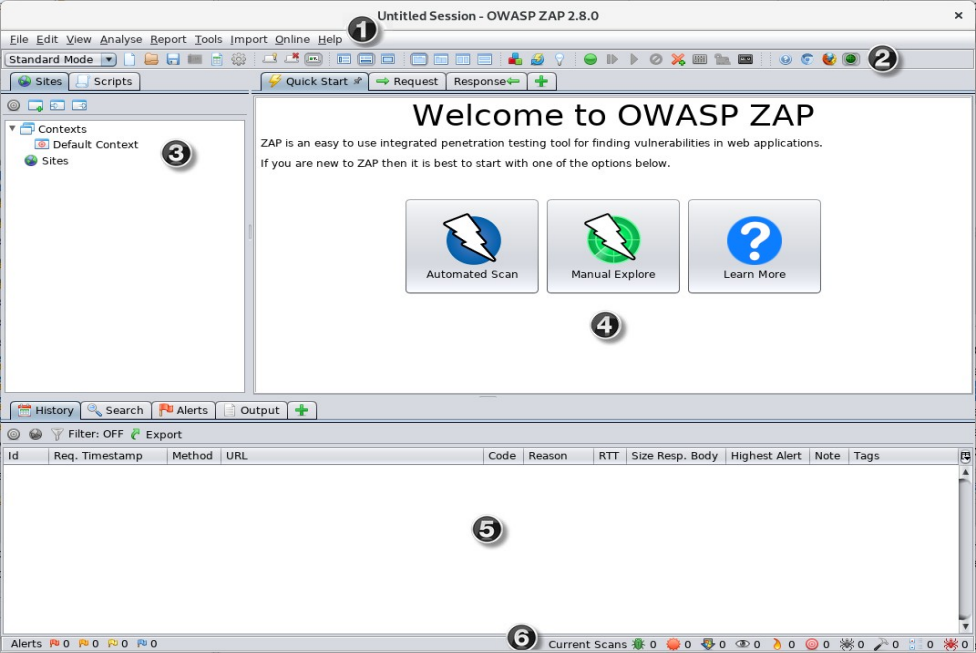
1. Software GNU de código libre.
2. Seguridad en los datos, sistemas reconocidos que sean avaluados, por temas de seguridad.
3. Interfaz gráfica que permita la navegación e interacción con los componentes de la base de datos y directorios.
4. Ayuda en la creación de scripts para ejecutarlos en las bases de datos.
5. Sistemas con administración de sesiones.
6. Soporte para Sybase, SQL Server, PostgreSQL y MongoDB.
7. Presentación de información en tablas dinámicas.

## ANEXO #4: Herramienta para pruebas

Para llevar a cabo las pruebas unitarias, técnicas y de usuario de las soluciones informáticas, se podrá utilizar herramientas que faciliten la validación de las mismas. Algunas herramientas que se recomiendan usar son:

**OWASP ZAP:**

OWASP ZAP (ZED ATTACK PROXY), permite la validación de los controles desarrollados para evitar las amenazas más comunes, descritas por OWASP Top 10, la pantalla principal es:



Es una de las herramientas más potentes, recomendadas para validar la seguridad de las aplicaciones web, fue desarrollada por OWASP. Su interfaz se compone de los siguientes elementos:

* + **Barra del menú:** proporciona acceso a varias herramientas automatizadas y manuales.
  + **Barra de herramientas:** incluyen accesos rápidos hacia las funciones de la herramienta.
  + **Árbol de ventanas:** muestra los sitios que se están analizando.
  + **Espacio de trabajo:** muestra las opciones para ejecutar el escáner manual o automático.
  + **Ventana de información:** muestra los detalles de la ejecución del escáner.
  + **Footer:** muestra un resumen de las alertas encontradas.

Entre sus principales características están:

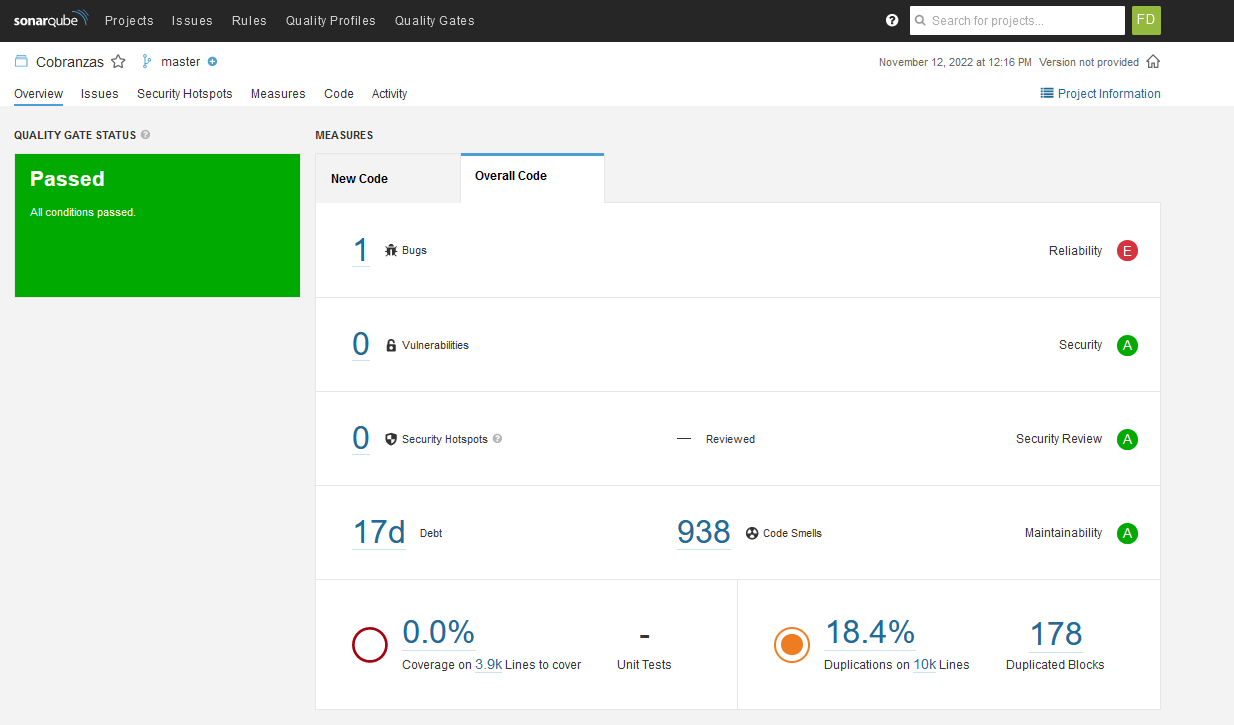
* + Es una herramienta totalmente gratuita y de código abierto.
  + Es multi-plataforma.
  + Fácil de instalar, dependiendo únicamente de Java 1.7 o superior.
  + Posibilidad de asignar una solución informática de prioridades.
  + Traducida a más de 12 idiomas, entre ellos, el español.
  + Excelente manual de ayuda y gran comunidad en la red.
  + Es intuitiva y fácil de utilizar.

La herramienta facilita la identificación de las vulnerabilidades que posee la página web desarrollada, las cuales se muestran en la pestaña de “Alertas” y se clasifican según su riesgo de la siguiente forma:

* **Alto.**
* **Medio.**
* **Bajo.**
* **Informativo.**
* **Falso Positivo.**

Se deberá resolver todas las alertas que se muestren con un riesgo “Alto”, las mimas que no sean producto de haber ejecutado el análisis en el ambiente de desarrollo y que consten el [Anexo #9](#_Desarrollo_de_Software).

**SonarQube:**

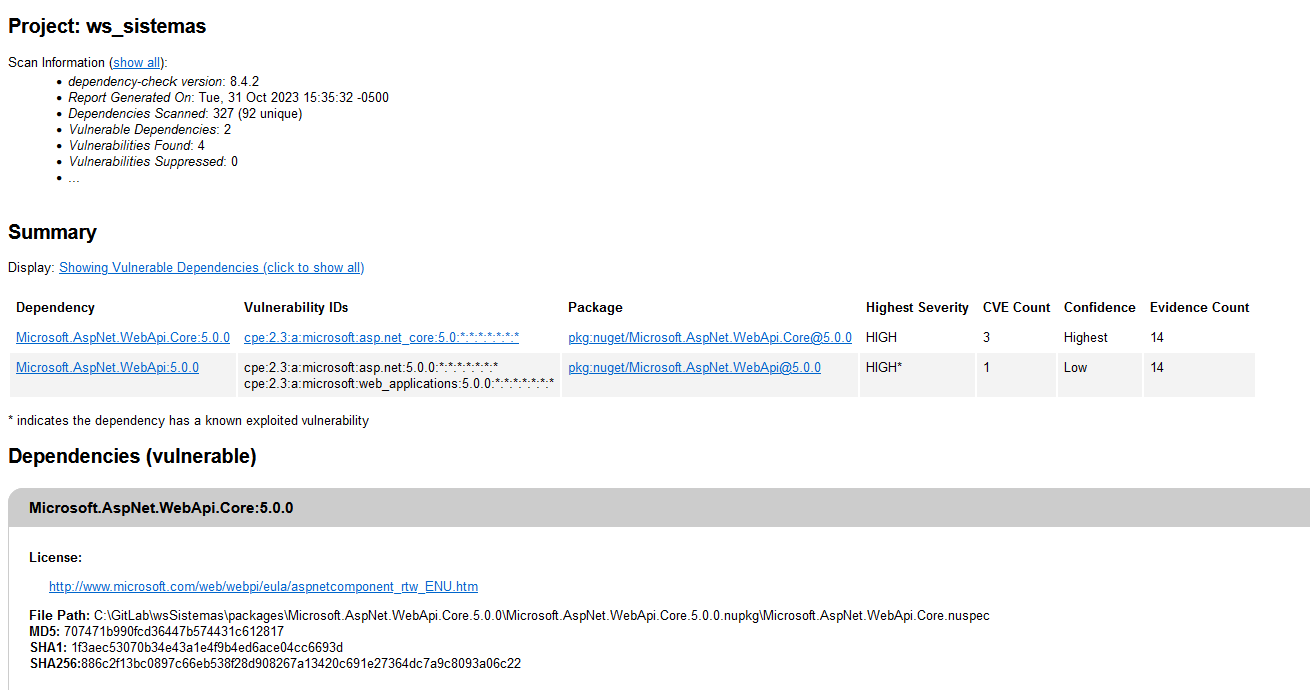


SonarQube es una plataforma de código abierto que permite realizar análisis estáticos de código para detectar problemas de seguridad, errores de programación, código duplicado, entre otros. Es una herramienta muy útil para evaluar la calidad del código y garantizar que cumpla con los estándares de seguridad y buenas prácticas de programación.

Principales características de SonarQube:

* Realiza análisis estáticos de código para identificar vulnerabilidades de seguridad y errores de programación.
* Proporciona métricas de calidad del código, como la cobertura de pruebas, la complejidad ciclomática y la duplicación de código.
* Permite definir reglas personalizadas para adaptarse a los estándares de codificación de cada proyecto.
* Ofrece integraciones con herramientas de construcción y control de versiones, facilitando su incorporación en el proceso de desarrollo.
* Proporciona informes detallados sobre el estado del código y las áreas que requieren atención.

**OWASP Dependency-Check:**



OWASP Dependency Check es una herramienta que se utiliza para identificar vulnerabilidades en las bibliotecas y dependencias utilizadas por una aplicación. Escanea el archivo de configuración de la aplicación y las dependencias declaradas para buscar vulnerabilidades conocidas en las bibliotecas utilizadas.

Principales características de OWASP Dependency Check:

* Escanea las dependencias de una aplicación en busca de vulnerabilidades conocidas en las bibliotecas utilizadas.
* Utiliza una base de datos de vulnerabilidades conocidas, incluyendo la base de datos de National Vulnerability Database (NVD) de NIST.
* Proporciona informes detallados sobre las vulnerabilidades encontradas, incluyendo información sobre la gravedad y las posibles soluciones.
* Es compatible con una amplia variedad de lenguajes de programación y entornos de desarrollo.
* Se integra fácilmente en los flujos de trabajo de desarrollo, permitiendo realizar escaneos automáticos como parte del proceso de construcción o despliegue de la aplicación.

Al utilizar SonarQube y OWASP Dependency Check junto con OWASP ZAP, se puede garantizar una cobertura más completa en la identificación y mitigación de vulnerabilidades de seguridad en las soluciones informáticas, tanto a nivel de código como de dependencias externas.

## ANEXO #5: Herramientas de documentación

Consideraciones generales para la documentación del desarrollo de aplicaciones informáticas:

Toda documentación que se genere para un proyecto debe poseer lo siguiente:

**A) Identificación del documento**

Este documento debe incorporar la siguiente información:

* Logotipo de la cooperativa;
* Nombre de la cooperativa;
* Lugar y fecha de elaboración;
* Unidades responsables de su elaboración, revisión y/o autorización;
* Objetivos del manual;
* Alcance;
* Responsables,
* Políticas;
* Procedimiento; y,
* Anexos.

**B) Estructura del manual de usuario.**

El manual de usuario tiene como objetivo instruir al usuario en el uso de la solución informática y dar respuesta a las dudas que puedan surgir en la operación del mismo. Este manual se publica con cada solución informática desarrollado en la Cooperativa y se actualiza cada vez que existan nuevas funcionalidades desarrolladas.

**C) Estructura del expediente técnico**

El expediente técnico es toda la documentación técnica generada del proceso de desarrollo de soluciones informáticas, va dirigido al equipo del desarrollo de sistemas, para facilitar el mantenimiento de la solución informática en caso de que se requiera; también puede ser revisado por los departamentos de control para verificar el cumplimiento de las políticas y procedimientos. El expediente técnico está conformado por los siguientes documentos:

|  |  |  |  |
| --- | --- | --- | --- |
| DOCUMENTO | FORMA ALMANCENAMIENTO | CONDICIÓN | REFERENCIA |
| **FASE ANALISIS** | | | |
| Requerimiento a la Dirección de Tecnología de Información | Física/Digital | Obligatorio | [**Anexo # 10**](#_Requerimiento_a_la) |
| Cronograma de trabajo | Digital | Obligatorio | [**Anexo # 11**](#_Cronograma_de_trabajo) |
| Prototipado con Penpot | Digital | Opcional | [**Anexo # 12**](#_Prototipo_de_pantallas) |
| Lineamientos del Proyecto | Digital | Obligatorio\* | [**Anexo # 13**](#_Historias_de_Usuario) |
| **FASE DISEÑO** | | | |
| Modelo entidad relación | Digital | Opcional | [**Anexo # 2**](#_Herramientas_de_Diseño) |
| Diccionario de datos | Digital | Opcional | [**Anexo # 5**](#_Herramientas_de_documentación) |
| **FASE DESARROLLO** | | | |
| Código fuente | Digital | Obligatorio | [**Anexo # 7**](#_Estándares_de_base) |
| Documentación del código fuente | Digital | Obligatorio | [**Anexo # 7**](#_Estándares_de_base) |
| Manual de usuario | Digital | Obligatorio | [**Anexo # 5**](#_Herramientas_de_documentación) |
| **FASE PRUEBAS Y CONTROL DE CALIDAD** | | | |
| Plan de pruebas | Digital | Opcional | [**Anexo # 14**](#_Plan_de_pruebas) |
| Bitácora de errores | Digital | Obligatoria | [**Anexo # 15**](#_Bitácora_de_errores) |
| Solicitud de ejecución de pruebas de técnicas | Digital | Obligatoria | [**Anexo # 16**](#_Solicitud_de_ejecución) |
| Incidentes del usuario | Digital | Opcional | [**Anexo # 17**](#_Incidentes_del_usuario) |
| Registro de puesta en producción | Física | Obligatoria | [**Anexo # 18**](#_Registro_de_Puesta) |

**\*** El documento Lineamientos del Proyecto podrá ser sustituido por otro mecanismo de análisis (prototipo, requerimiento, etc.).

**Estructura del Documento DICCIONARIO DE DATOS**

El diccionario de datos es una lista organizada de todos los datos pertinentes a la solución informática, con un conjunto de definiciones precisas y rigurosas para que tanto el Desarrollador de Sistemas como el usuario se entiendan.

En el Diccionario de Datos se:

* Describe el significado de los flujos y almacenes que muestran los Diagramas de Flujo de Datos.
* Describe la composición de los paquetes de datos que se mueven a través de los flujos de datos.
* Describe la composición de los paquetes de datos en los almacenes.
* Especifica los valores y unidades relevantes de piezas elementales de información entre los flujos de datos y los almacenes de datos.

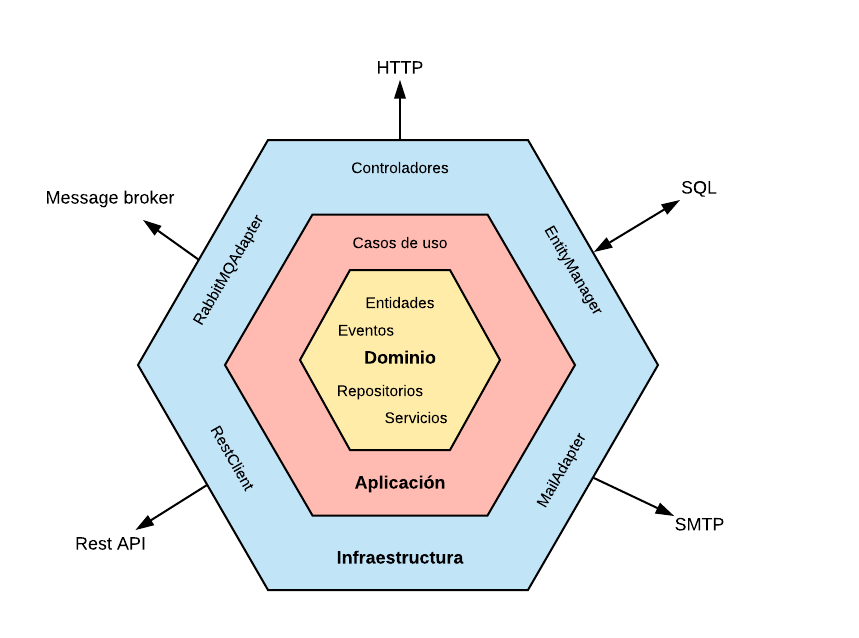
Notación: Se usa símbolos especiales para limitar el texto necesario empleado para describir las relaciones entre los datos y mostrar con claridad las relaciones estructurales.

La simbología empleada se describe a continuación:

|  |  |
| --- | --- |
| **Símbolo** | **Significado** |
| = | está compuesto de |
| + | Y |
| ( ) | opcional (puede estar presente o ausente) |
| { } | Iteración |
| [ ] | seleccionar una de varias alternativas |
| | | separador de alternativas en caso de selección |
| \*\* | Comentarios |
| @ | identificador en caso de almacenes |

## ANEXO #6: Arquitectura de desarrollo de software

**Arquitectura Hexagonal**



En la arquitectura hexagonal, las capas de Domain, Application, Infrastructure, y WebUI se organizan de manera específica para lograr una separación clara de las responsabilidades y para facilitar el mantenimiento y la evolución del sistema.

También conocida como arquitectura de puertos y adaptadores, ofrece varias ventajas significativas para el desarrollo de software. Una de las principales ventajas es su alta modularidad, que permite separar claramente la lógica de negocio del resto del sistema, facilitando así la escalabilidad y el mantenimiento. Además, al centrarse en interfaces y puertos, esta arquitectura promueve la independencia de las tecnologías utilizadas en cada capa, lo que facilita la integración de nuevas tecnologías o cambios en el futuro sin afectar otras partes del sistema. Esto también mejora la portabilidad del código y la reutilización de componentes. Otra ventaja es su capacidad para facilitar el testing, ya que permite realizar pruebas unitarias y de integración de forma más eficiente al aislar las diferentes capas del sistema.

A continuación, se proporciona una guía general sobre cómo se debería manejar estas capas:

1. **Domain**
   1. Propósito:
      * Define las entidades, agregados y servicios de dominio.
   2. Características:
      * Independiente de la infraestructura y la interfaz de usuario.
      * No debe tener dependencias hacia las capas superiores o inferiores.
      * Puede contener servicios de dominio que encapsulan la lógica de negocio.
   3. Ejemplos:
      * Cliente, Pedido, ServicioFacturación, etc.
2. **Application:**
   1. Propósito:
      * Contiene la lógica de aplicación que coordina las operaciones del dominio.
      * Utiliza servicios de dominio y orquesta interacciones entre ellos.
   2. Características:
      * Depende de la capa de dominio.
      * Define casos de uso específicos de la aplicación.
      * Puede manejar transacciones y coordinar operaciones complejas.
   3. Ejemplos:
      * CrearPedidoUseCase, FacturarPedidoUseCase, etc.
3. **Infrastructure:**
   1. Propósito:
      * Contiene implementaciones concretas de los servicios definidos en el dominio.
      * Maneja detalles de infraestructura como bases de datos, servicios externos, y frameworks.
   2. Características:
      * Depende de las capas de dominio y aplicación.
      * Implementa interfaces definidas en las capas superiores.
      * Puede incluir repositorios, servicios de mensajes, servicios de persistencia, etc.
   3. Ejemplos:
      * RepositorioCliente, ServicioCorreoElectrónico, etc.
4. **Presentación (Mismo nombre de la solución):**
   1. Propósito:
      * Contiene la interfaz de usuario y la lógica asociada.
      * Interactúa con la capa de aplicación para coordinar operaciones.
   2. Características:
      * Depende de las capas de aplicación, infraestructura y dominio.
      * Puede ser una aplicación web, una API REST, una aplicación móvil, etc.
   3. Ejemplos:
      * ControladorPedido, VistaCliente, etc.

## ANEXO #7: Estándares de programación y base de datos

**Estándares de base de datos:**

1. Todos los nombres de los objetos de las bases de datos como nombres de: base de datos, tablas, campos, disparadores, vistas, procedimientos almacenados, etc. deberán cumplir con las reglas de nomenclatura definidas en este Anexo.
2. Todos los objetos de la base como nombres de las bases de datos, de las tablas, campos, el código de los procedimientos almacenados, funciones, vistas, disparadores, etc. Se debe realizar en estilo lower\_case, es decir, todo en minúsculas y separando las palabras con guión bajo “\_”.
3. Todos los objetos de la base deben llevar un prefijo según se especificará más adelante, y, debe contener las siguientes reglas:  
   **Una sola Palabra:** El prefijo se conformará de la primera letra de la palabra, a esto se unirá la primera y segunda consonante que forma la palabra seguido de un guion bajo (\_) y finalmente para la conformación del nombre se adhiere la descripción. Ejemplo Una variable que tenga el tipo de dato integer el prefijo es int, Una tabla que debe tener el prefijo de su base de datos proveeduria es prv. En los casos en los que no se tengan 2 consonantes después de la primera letra se deberá usar las tres primeras letras para conformar el prefijo, por ejemplo, una variable de tipo de dato bit el prefijo es bit.  
   **Dos palabras**: El prefijo se establecerá tomando de la primera palabra la primera letra más la primera consonante y la primera letra de la segunda palabra, a esto se agregará el nombre descriptivo. Ejemplo el nombre de un campo que debe llevar el prefijo de su tabla formulario\_detalle el prefijo es frd.  
   **Tres palabras o más**: El prefijo estará conformado de la primera letra de cada una de las tres primeras palabras y a esto se adhiere el nombre descriptivo. Por ejemplo, el nombre de un campo que debe llevar el prefijo de su tabla asiento\_contable\_his el prefijo es ach..
4. Los nombres de las bases de datos creadas internamente por CoopMego deben comenzar con un prefijo de tres letras “meg\_” seguido de un guión bajo y a continuación el o los nombres que representen la aplicación que está siendo desarrollada.

|  |
| --- |
| **Nombre de bases de datos** |
| meg\_inversiones |
| meg\_proveeduria |
| meg\_facturacion\_electronica |

1. Las tablas deben llevar el nombre de acuerdo con la entidad que representen. Las tres primeras letras deben identificar el nombre de la base de datos a la que pertenecen seguida de guion bajo y luego el nombre de la tabla. Así mismo deben ir en plural haciendo referencia a que almacenan varios registros, excepto cuando sea una tabla dependiente (detalle de una cabecera) donde el nombre si puede ir en singular.

|  |  |
| --- | --- |
| **Nombre de bases de datos** | **Nombre de las tablas** |
| meg\_inversiones | inv\_certificados; inv\_cotizaciones |
| meg\_facturacion\_electronica | fce\_configuracion; fce\_comprobantes |
| meg\_nomina | nmn\_formularios; nmn\_formulario\_detalle |

1. Los nombres de los campos deben tener como prefijo tres letras que identifiquen el nombre de la tabla a la que pertenecen seguido de un guión bajo y luego el nombre del campo.

|  |  |  |
| --- | --- | --- |
| **Nombre de bases de datos** | **Nombre de las tablas** | **Nombre de los campos** |
| meg\_inversiones | inv\_usuarios | usr\_login; usr\_password |
| meg\_facturacion\_electronica | fce\_comprobantes | cmp\_id; cmp\_nro\_factura |
| meg\_nomina | nmn\_formulario\_detalle | frd\_id, frd\_valor |

1. Las llaves primarias deben comenzar con el mismo prefijo de los campos de la tabla, luego deben tener un sufijo “\_pk\_” y seguido el nombre del campo al cual pertenecen. Ejemplo: usr\_pk\_login
2. Las llaves foráneas deben comenzar con el mismo prefijo de los campos de la tabla, luego deben tener un sufijo “\_fk\_” y a continuación el nombre del campo. Ejemplo: usr\_fk\_departamento
3. El nombre de los procedimientos almacenados (storesprocedures) deben tener un prefijo de tres letras ([Tabla\_prefijos\_funciones)](#tabla_prefijos) seguido de un nombre que represente lo que realiza el procedimiento.
4. El diseño de la base de datos de la solución informática deberá contener:

* Documentación de la etapa del Análisis.
* Elaboración del Modelo Lógico de Datos.
* Elaboración del Modelo Físico de Datos.
* Normalización del Modelo Lógico de Datos.
* Scripts de creación de la base de datos.

**Tabla de prefijos para Funciones y Procedimientos Almacenados**

|  |  |  |
| --- | --- | --- |
| **Prefijo** | **Acción a realizar** | **Ejemplo** |
| get | Traer, obtener datos | get\_datos\_usuario |
| add | Insertar, almacenar datos | add\_datos\_compras |
| set | Establecer, actualizar datos | set\_oficina\_usuario |
| del | Borrar, eliminar datos | del\_pagos |

**Estándares de Programación:**

* Características de nomenclatura
* Los nombres de archivos, clases, métodos, variables, funciones y otros elementos de código deben ser descriptivos y proporcionar un contexto claro sobre su propósito y funcionalidad:
  + **Nombres Significativos:** Todos los nombres deben ser significativos y descriptivos, reflejando claramente la funcionalidad o el propósito del elemento al que hacen referencia.
  + **Utilizar Palabras Completas:** Preferir el uso de palabras completas en lugar de acrónimos o abreviaturas siempre que sea posible, a menos que el acrónimo sea universalmente reconocido y ampliamente aceptado.
  + **Contexto y Funcionalidad:** Los nombres deben proporcionar información clara sobre el contexto y la funcionalidad del elemento al que se refieren. Por ejemplo, un método debe describir qué hace y cómo se utiliza.
  + **Consistencia:** Mantener la consistencia en la nomenclatura en todo el proyecto.
  + **Longitud Adecuada:** Los nombres deben ser lo suficientemente descriptivos como para transmitir su propósito, pero también deben ser concisos y no excesivamente largos. Evitar nombres demasiado largos que puedan dificultar la legibilidad del código.
* Características del nombre del proyecto:
* El nombre de la solución debe tener relación directa con lo que hace la solución informática, siguiendo siempre las [reglas de nomenclatura](#Reglas_nomenclatura).
* Se debe anteponer el sufijo dependiendo del tipo de solución:

|  |  |  |
| --- | --- | --- |
| **NOMBRES PARA TIPOS DE SOLUCIONES** | | |
| **Tipo de solución** | **Prefijos** | **Ejemplos** |
| Servicio Web | ws | wsGestionCobranzas |
| Demonio | srv | srvGestionCobranzas |
| ApiGateway | apiGateway | apiGatewayGestionCobranzas |
| Interfaces Internas | interfaz | interfazGestionCobranzas |
| Interfaces Externas | interfazExterna | interfazExternaMegonline |
| Aplicaciones Web | appWeb | appWebMego |
| Aplicaciones Móviles | appMovil | appMovilGestiona |

* + Características de las capas de arquitectura:
* Para asegurar una estructura coherente y comprensible en la solución con arquitectura hexagonal, es crucial seleccionar nombres apropiados para las capas. Cada capa debe reflejar claramente su propósito y responsabilidades dentro del sistema. Idealmente, los nombres de las capas deben ser descriptivos y autoexplicativos, permitiendo a los desarrolladores entender rápidamente la funcionalidad de cada componente.
* El nombre de las capas en la arquitectura hexagonal son los siguientes:
  + **Domain.** Capa de dominio que contiene la lógica de negocio principal y los modelos de dominio.
  + **Applitacion.** Capa de aplicación que contiene los casos de uso y la lógica de aplicación específica del dominio.
  + **Infrastructure.** Capa de infraestructura que contiene la implementación concreta de los puertos y adaptadores, así como la lógica de acceso a datos y servicios externos.
  + **WebUI.** Contiene los controladores de la interfaz de usuario que manejan las solicitudes HTTP y la presentación de datos.
* Características de los textos alternativos.
* Todas las imágenes no decorativas y las que son susceptibles de ser seleccionadas (por ejemplo, vínculos) deben tener un texto alternativo asociado. Este texto alternativo se mostrará en navegadores sin soporte para imágenes. El texto alternativo (atributo alt) debe ser explicativo de lo que contiene o lo que representa la imagen.
* Se deben evitar colocar como texto alternativo el nombre de la imagen.
* Hay imágenes que solo tienen sentido visualmente, por ejemplo, viñetas. Estas imágenes deben tener texto alternativo vacío (alt ='''').
* Todas las imágenes que tengan texto deben tener un texto alternativo que contenga, como mínimo, el mismo texto que la imagen.
* Características de la estructura del código.
* El código de la aplicación debe estar totalmente documentado, con una descripción de lo que hace.
* Todo el código se debe estar estructurado por regiones y cada región debe llevar una breve descripción de lo que hace.
* Todas las funciones y procedimientos deben estar documentados siguiendo el formato establecido por el mismo lenguaje, para luego generar toda la documentación mediante la herramienta de desarrollo; ésta debe contener: una descripción general breve, una descripción de cada parámetro y, en caso de las funciones, del valor que retorna.
* Características de la declaración de variables
* Los nombres de las variables deben comenzar con un prefijo de tres letras representando el tipo de dato al que pertenecen seguido del nombre descriptivo de la variable. Por ejemplo: la variable real Percentil será: double dblPercentil. Variable de tipo string “texto” será: strTexto.
* Todas las variables, controles y clases en la definición de su nombre utilizarán un prefijo de 3 letras más el nombre descriptivo utilizando la notación camelCase. Para la definición del prefijo se toman en consideración lo siguiente:
* **Si el nombre del control, tipo de dato se compone de**:

**Una sola Palabra:** El prefijo se conformará de la primera letra de la palabra, a esto se unirá la primera y segunda consonante que forma la palabra seguido de un nombre descriptivo que permita identificar el objetivo de del objeto. Ejemplo Objeto de la clase Timer el prefijo es tmr; para Panel el prefijo es pnlPanelPrincipal, para double dblValor, para int intEdad.

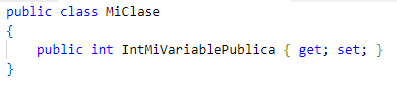
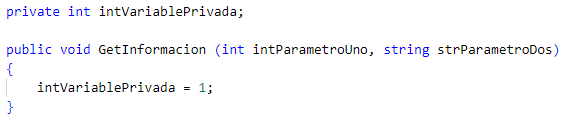
**Dos palabras:** El prefijo se establecerá tomando de la primera palabra la primera letra más la primera consonante y la primera letra de la segunda palabra, a esto se agregará el nombre descriptivo. Ejemplo CommandButton: cmb; PictureBox: pcb; Immage Button: imb.

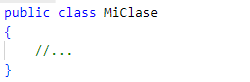
**Tres palabras o más:** El prefijo estará conformado de la primera letra de cada una de las tres primeras palabras y a esto se adhiere el nombre descriptivo. Por ejemplo, CheckedListBox: clbOpciones; DateTimePicker: dtpFechas

* El nombre de la variable debe ser la palabra completa y si la palabra es muy amplia debe limitarse a una parte de la misma de los datos que va a almacenar. Por ejemplo: Valor Agregado: dblValorAgrega.
* No se permite utilizar tildes, letras ñ y otros símbolos en los nombres de las variables. Por ejemplo: Valor Añadido: dblValorAniadido; Dirección: strDireccion
* Si se requiere numeración de variables, se debe hacer con números romanos incrementados al final del nombre de esta. Por ejemplo: Cantidad Pedido Segundo: dblCantidadPedidoII.
* Consideraciones para declaración de variables.

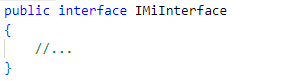
Para la declaración de variables se debe tener en cuenta las siguientes consideraciones:

|  |  |  |  |
| --- | --- | --- | --- |
| **NOMBRES DE VARIABLES** | | | |
| **Variables** | **Tipos de Datos** | **Prefijos** | **Ejemplos** |
| Cantidad de Pedido | sbyte | sby | sbyte sbyCantidadPedido; |
| Valor de Pedido | byte | byt | Byte bytValorPedido; |
| Valor Agregado | short | shr | Short shrValorAgregado; |
| Porcentaje de Pedido | unshort | ush | Unshort ushPorcentajePedido; |
| Sucursal | int | int | Int intSucursal; |
| Factor Externo | Long | lng | Long lngFactorExterno; |
| Valor Total | Float | flt | Float fltValorTotal; |
| Porcentaje de Venta | Double | dbl | double dblPorcentajeVenta; |
| Precio | decimal | dcm | Decimal dcmPrecio; |
| Cambio de Pedido | Boolean | bln | Boolean blnCambioPedido; |
| Sexo | Char | chr | Char chrSexo; |
| Nombre del Usuario | String | str | String strNombreUsuario; |

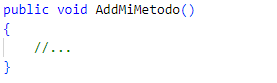
* Características de las funciones y procedimientos.
* Los nombres de las funciones deben comenzar con tres letras que representan la acción que ejecutan seguido de un nombre representativo de la función, para los procedimientos que no hace ninguna de las acciones definidas en la [tabla de funciones](#prefijos_proc_func) iría directamente el nombre representativo. Si el nombre estuviese formado por dos o más palabras, estas deberán estar después del prefijo en notación camelCase [reglas de nomenclatura](#Reglas_nomenclatura). Por ejemplo: Public string getSecuencial(); Private void ordenarPuntosVenta()
* Estándares de nomenclatura  
  **.NET (C#)**
  + Variables y miembros de clase
    - PascalCase para variables públicas y miembros de clase:  
      
    - camelCase para variables privadas, locales y parámetros de método:  
      
  + Clases
    - PascalCase para nombres de clases:



* + Interfaces
    - Prefijo con 'I' y PascalCase para nombres de interfaces:



* + Métodos
    - PascalCase para nombres de métodos:

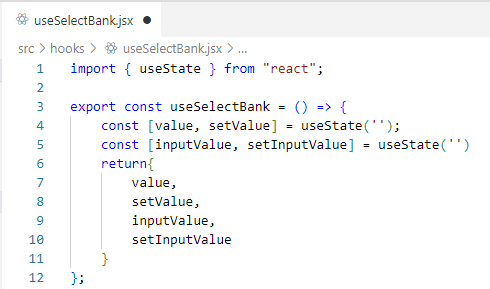
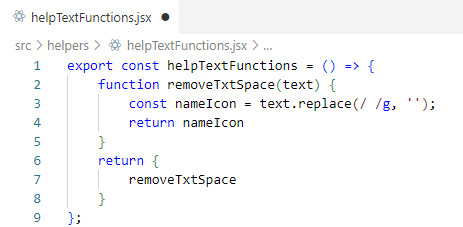


* + Propiedades
    - PascalCase para nombres de propiedades:  
      
  + Constantes
    - Mayúsculas con guiones bajos para constantes:  
      
  + Archivos
    - PascalCase para nombres de archivos y carpetas:  
      Ejemplo: MiClase.cs, MiCarpeta/
  + Soluciones
    - PascalCase para nombres de soluciones:  
      Ejemplo: MiSolucion.sln

**React (js)**

* + Componentes
    - PascalCase para nombrar componentes:  
      Ejemplo: *MyComponent*
    - Mantener el mismo nombre para el archivo y el componente si se está exportando un solo componente desde ese archivo.

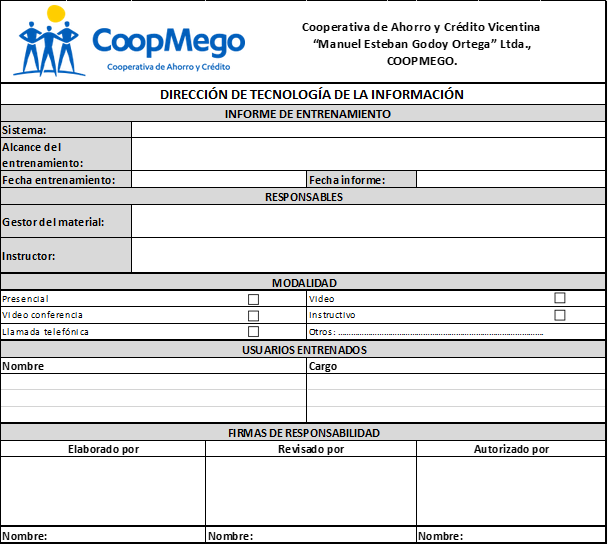


* + Hooks
    - Se debe usar el prefijo “use”:  
      
  + Funciones de eventos
    - Prefijo “handle” cuando una función realiza un evento (acción en la interfaz de usuario)  
      
  + Funciones de ayuda
    - Prefijo “help” cuando se crean funciones auxiliares.  
      ****

**Otros lenguajes de programación o frameworks.**

* + Se deben adaptar lo más posible a los estándares de los lenguajes especificados anteriormente.
  + Se pueden plantear estándares propios del framework si eso aporta fácil entendimiento e identificación de los objetos.
  + Si la tecnología que usamos nos obliga a utilizar alguna nomenclatura, se puede excepcionar los estándares.

## ANEXO #8: Informe de entrenamiento



## ANEXO #9: Desarrollo de software seguro

A continuación, se define el alcance del cumplimiento de OWASP que se debe respetar para el desarrollo de las aplicaciones informáticas que corresponda.

La versión más reciente de la lista de las vulnerabilidades más comunes e impactantes que aparecen en la aplicación de producción web OWASP se publicó en 2021.

**Control de acceso roto.**

* 1. Los sistemas de control de acceso están diseñados para garantizar que solo los usuarios legítimos puedan acceder a datos o funciones. Las vulnerabilidades de esta clase suelen permitir que los atacantes exploten los controles de acceso rotos, sobre todo esto es común en las aplicaciones web modernas. Un ejemplo para este caso puede ser que una aplicación web puede permitir que un usuario acceda a la cuenta de otro usuario cambiando una URL específica.

**Fallas criptográficas**

* 1. Las fallas criptográficas son la principal causa de la exposición de datos confidenciales, como contraseñas, números de tarjetas de crédito y cualquier otra información personal. Sin embargo, estos algoritmos criptográficos son invaluables para proteger la privacidad y la seguridad de los datos, pueden ser muy sensibles a errores de implementación o configuración. Las fallas criptográficas incluyen la imposibilidad de utilizar el cifrado, configuraciones incorrectas de los algoritmos criptográficos y una gestión de claves insegura. Como ejemplo en una organización se podría usar un algoritmo hash inseguro para el almacenamiento de contraseñas, no usar las contraseñas o usar la misma sal para todas las contraseñas de usuario almacenadas.

**Inyección**

* 1. Este tipo de vulnerabilidad es posible debido a que no se desinfectan adecuadamente las entradas del usuario antes de procesarlas, esto suele suceder cuando los datos y los comandos se entremezclan para que los datos proporcionados por el usuario, malintencionadamente puedan interpretarse como parte de un comando, dentro de una consulta, la entrada del usuario que contiene estos caracteres podría ser capaz de cambiar el comando que se está procesando, siendo frecuente esta acción en lenguajes como SQL.

**Diseño inseguro**

* 1. Vulnerabilidad que se puede introducir en el software durante el proceso de desarrollo, estas fallas en el diseño pueden afectar la seguridad del sistema, una aplicación que almacena y procesa datos confidenciales al no incluir en la aplicación un sistema de autenticación, entonces su implementación seguirá siendo insegura y no protegerá adecuadamente estos datos confidenciales.

**Mal configuración de seguridad**

* 1. La seguridad de una aplicación también se puede ver afectada por su configuración, un fabricante de software tendrá configuraciones predeterminadas para su aplicación, y los usuarios también podrán habilitar o deshabilitar varias configuraciones, estas configuraciones erróneas de seguridad podrían incluir habilitar aplicaciones o puertos innecesarios, dejar cuentas y contraseñas predeterminadas activas y sin cambios, o configurar mensajes de error para exponer demasiada información a un usuario.

**Componentes vulnerables y obsoletos**

* 1. Esta vulnerabilidad representa un peligro cuando los actores de amenazas han intentado insertar código malicioso o vulnerable en bibliotecas de uso común y dependencias de terceros. Si una organización carece de visibilidad del código externo que se utiliza dentro de su aplicación (incluidas las dependencias anidadas) y no lo escanea en búsqueda de dependencias, entonces puede ser vulnerable a la explotación.

Al no aplicar actualizaciones de seguridad a estas dependencias, se podría dejar una vulnerabilidad explotable abierta a ataques.

**Fallos de identificación y autenticación**

* 1. Varias aplicaciones y sistemas requieren alguna forma de identificación y autenticación, donde el usuario demuestre su identidad ante una aplicación o un servidor, que proporcione un certificado digital que verifique su identidad a un usuario al configurar una conexión cifrada con TLS. Estos fallos se producen cuando una aplicación se basa en procesos de autenticación débiles o no valida correctamente la información de autenticación. Al carecer de autenticación de múltiples factores (MFA) podría ser expuesta a ataque de relleno de credenciales, el atacante podría usar fácilmente combinaciones de nombre de usuario y contraseña de una lista de credenciales comunes y débiles.

**Fallas de integridad de datos y software**

* 1. La vulnerabilidad de fallas de integridad de datos y software, expone las debilidades en la seguridad de la canalización DevOps.

Esta clase de vulnerabilidad hace referencia a confiar en código de terceros de fuentes o repositorios que no son de confianza, no asegurar el acceso a la canalización de CI/CD y no validar adecuadamente la integridad de las actualizaciones aplicadas automáticamente. Un atacante podría reemplazar un módulo o dependencia confiable con una versión modificada o maliciosa, ocasionando que las aplicaciones creadas con esa dependencia podrían ejecutar código malicioso.

**Fallos de registro y monitoreo de seguridad**

* 1. Las fallas de registro y monitoreo de seguridad son la primera de las vulnerabilidades que han subido desde el décimo lugar en la iteración anterior de la lista. Estos incidentes de seguridad se han suscitado ya que una aplicación no logra registrar eventos de seguridad importantes, o que estos archivos de registro no se monitorean y manejan adecuadamente afectando directamente la visibilidad, las alertas de incidentes. Podría ocurrir que una aplicación no genere archivos de registro, que genere registros de seguridad que carezcan de información crítica o que estos archivos de registro solo estén disponibles localmente en una computadora. Estas fallas reducen la capacidad de una organización para detectar rápidamente un posible incidente de seguridad y responder de inmediato.

**Falsificación de solicitudes del lado del servidor**

* 1. La falsificación de solicitudes del lado del servidor (SSRF) se encuentra enumerada en la lista OWASP Top Ten, esta vulnerabilidad puede darse cuando una aplicación web no valida adecuadamente una URL proporcionada por un usuario al recuperar un recurso remoto ubicado en esa URL, permitiendo al atacante evadir los controles de acceso como un firewall, que bloquearía las conexiones directas del atacante a la URL de destino, pero que está configurado para proporcionar acceso a la aplicación web vulnerable. (CHECK POINT, s.f.).

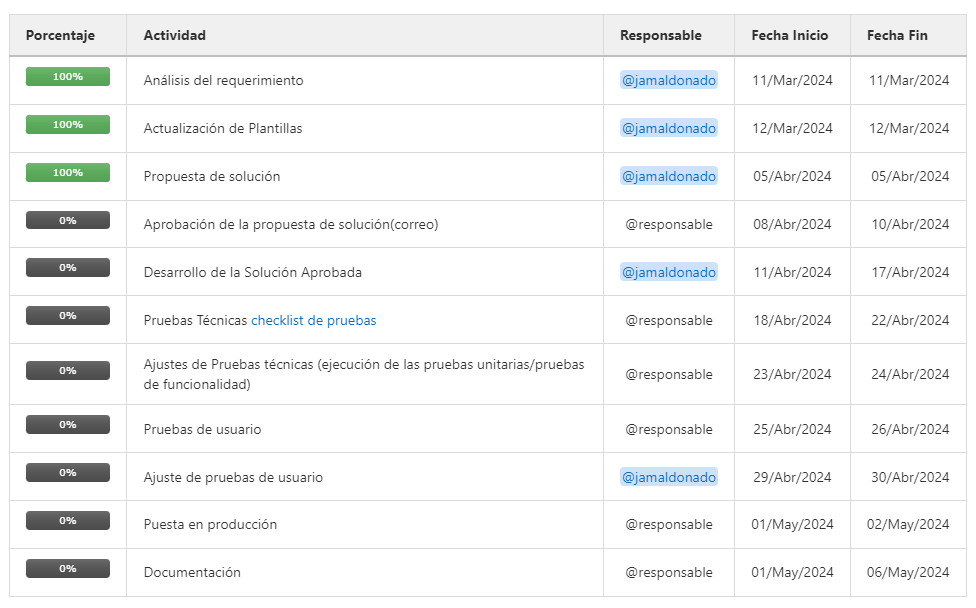
## ANEXO #10: Requerimiento a la Dirección de Tecnología de la Información

Se puede enviar un correo especificando el requerimiento o se puede enviar en físico el documento del requerimiento con el formato del ejemplo:

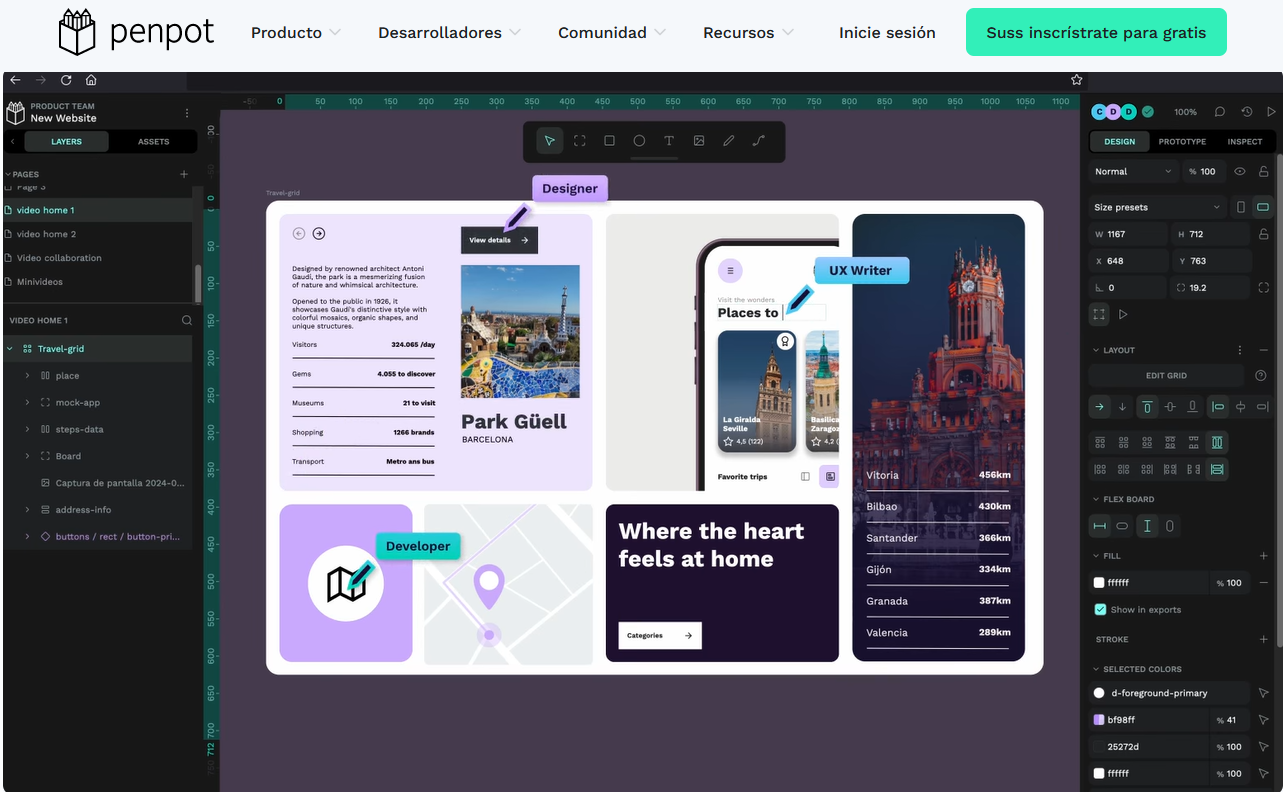


## ANEXO #11: Cronograma de trabajo

Ejemplo de cronograma:



## ANEXO #12: Prototipado con Penpot

****

**Penpot:** es una herramienta para diseño, de código abierto basada en la web, se puede trabajar de manera colaborativa entre un equipo de trabajo, también se puede usar desde cualquier dispositivo, independientemente del sistema operativo que se tenga: Linux, Windows, Mac.

Además, se puede instalar la herramienta en un servidor propio, utilizando Docker.

Mejora con la colaboración: cualquier persona puede contribuir al código de Penpot para arreglar bugs, añadir funcionalidades, hacer refactors, incluir mejoras en funcionalidades que ya existan, etc.

**Compartir diseños:**

Cuenta con un sistema similar al que se encuentra en Drive, lo que permite crear enlaces para compartir prototipos y asignar diferentes tipos de permisos, para facilitar el flujo de trabajo del equipo. Y por supuesto, se puede compartir diseños a través de un enlace y además se podrá deshabilitar el enlace en cualquier momento y dejar de compartir un diseño.

Una de las características que ofrece esta herramienta para ser más productivos, es el trabajo con componentes. Es decir, se puede crear, por ejemplo, el diseño de un botón y luego emplearlo en diferentes partes del proyecto. Modificar ese componente es tan simple como cambiar su tamaño, forma y colores, así como la tipografía usada.

## ANEXO #13: Lineamientos del Proyecto

Se envía un correo a la célula de trabajo informando lo que se estableció en la reunión.

**DETALLES DE LA REUNIÓN**

**Fecha:**

Jueves, xx de xxxx de 20xx

**Hora:**

15:00

**Lugar de la reunión:**

Matriz – Sala de reuniones del quinto piso

**Temas tratados:**

* [Listado de temas tratados]
* ...

**PROYECTO**

**Requerimiento:**

[NOMBRE DEL REQUERIMIENTO]

**Objetivos:**

[Especificar uno o varios objetivos a cumplirse duramente el proyecto según sea necesario]  
**Fecha tentativa finalización:**

xx de diciembre de 2023

**Célula de trabajo propuesta:**

* [Lista de personas que conformarán la célula de trabajo y el rol que van a cumplir o el cargo que desempeñan, ejemplo: Ana García (Jefe de xxxxxx)
* ...

**Especificaciones funcionales (EF):**

* **EF1.**
  + **Resultado:** [Describir qué se debe realizar]
  + **Procesos:** [Describir cómo se va a realizar]
  + **Justificación:** [Describir por qué o para qué se va a realizar]
* **EF2.**
  + ...

**Análisis de riesgos**

* **Amenazas Identificadas**:   
  Enumeración de las amenazas y vulnerabilidades identificadas durante el análisis de riesgos, organizadas por categorías (por ejemplo, seguridad de la información, riesgos técnicos, riesgos operativos, etc.).  
  Para cada amenaza identificada, se proporciona una descripción detallada de la amenaza, su probabilidad de ocurrencia, su impacto potencial en el proyecto y cualquier otro factor relevante.
* **Evaluación de Riesgos**:   
  Para cada amenaza identificada, se evalúa el riesgo asociado mediante la combinación de su probabilidad de ocurrencia y su impacto potencial en el proyecto.  
  Se asigna un nivel de riesgo a cada amenaza (por ejemplo, bajo, medio, alto) en función de esta evaluación.
* **Medidas de Mitigación**:   
  Para cada amenaza identificada, se proponen medidas de mitigación específicas diseñadas para reducir o eliminar el riesgo asociado y un plan detallado para su implementación.  
  Se describe cada medida de mitigación, incluyendo su efectividad esperada, los recursos necesarios para implementarla y el responsable de su ejecución.

**Observaciones:**

[Detallar cualquier información que se salga del formato]

## ANEXO #14: Plan de Pruebas

El plan de pruebas contempla los casos de prueba y los escenarios de prueba, que se registran en un documento para ser la guía para ejecutar las pruebas unitarias, técnicas y de usuario.

* **Casos de Prueba**

El diseño de casos de prueba fue creado debido a la imposibilidad de probar exhaustivamente el software.

Existen tres enfoques principales para el diseño de casos:

* **El enfoque estructural o de caja blanca**: Consiste en centrarse en la estructura interna del programa para elegir los casos de prueba. En este caso, la prueba ideal del software consistiría en probar todos los posibles caminos de ejecución, a través de las instrucciones del código, que puedan trazarse.
* **El enfoque funcional o de caja negra**: Consiste en estudiar la especificación de las funciones, la entrada y la salida para derivar los casos. Aquí, la prueba ideal del software consistiría en probar todas las posibles entradas y salidas del programa.

**Escenario de Prueba**

Los “escenarios de pruebas”, son las diferentes condiciones en las que el aplicativo deberá trabajar y por cada escenario es posible contar con uno o varios casos de pruebas.

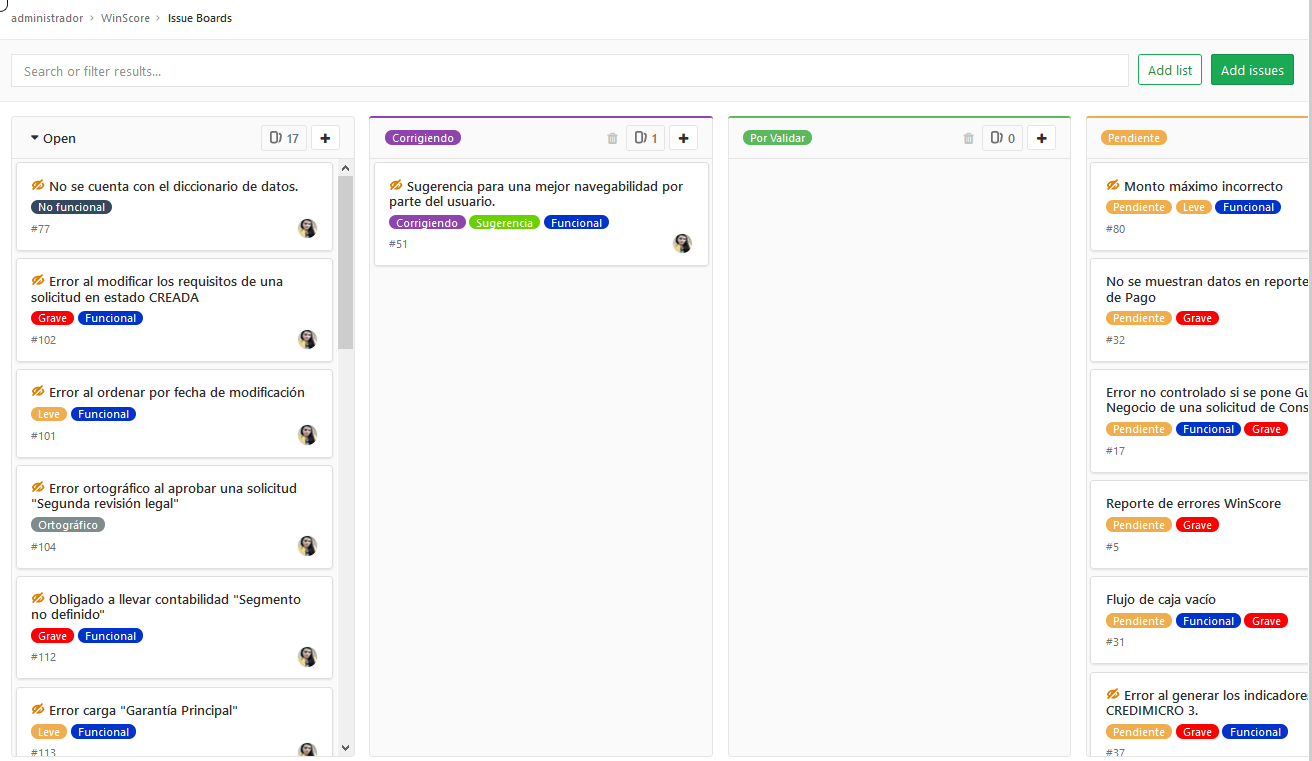
En <*situación*><*una expresión*> causa <*consecuencias deseables*>,

pero puede causar <*consecuencias indeseables*>

A partir de aquí se pueden analizar escenarios alternativos en donde distintas condiciones de la solución informática intentan obviar las consecuencias no deseadas.



## ANEXO #15: Bitácora de errores



## ANEXO #16: Solicitud de ejecución de pruebas técnicas

Se debe tomar en cuenta el siguiente formato:

**De:** (Desarrollador de Sistemas)   
**Para:** (Control de Cambios)  
**CC:** (Director de Tecnología de la Información); (Desarrollo de Sistemas); (Requerimientos TI);  
**Asunto:** Solicitud de pruebas [Nombre del sistema] para requerimiento..... (Tipo de atención)

**Adjunto:** (Requerimientos del Sistema, Documentación de análisis)

**Fecha cumplimiento QA:** (Fecha en la que se debería culminar las pruebas)

**~~Nombre del Sistema:~~** ~~(Nombre del sistema a Probarse)~~

**Descripción del Sistema:** (Descripción de los objetivos principales del sistema)

**Versión actual del Sistema:** (Definir la versión que está en producción) *[ V. X.X.X]*

**Versión nueva del Sistema**: (Definir la nueva versión creada o de la actualización) *[ V. X.X.X]*

**Fecha actualización:** (Fecha en el que se pone a producción)

**Manual actualizado:** (Informar si se adjunta el manual actualizado)

**Propietario del sistema:** (Definir el perfil del dueño del sistema)

**Programador principal:** (Nombre del Desarrollador de Sistemas responsable del sistema)

**Programador responsable de cambios:** (Nombre del Desarrollador de Sistemas responsable de cambios)

**Pruebas retroactivas:** (Definir la necesidad de realizar pruebas de procesos en producción)

**Listado de requerimientos/historias a probarse:**

* (Describir los requerimientos principales de la solución informática)
* (Detallar las historias en las que se basa la solución informática)

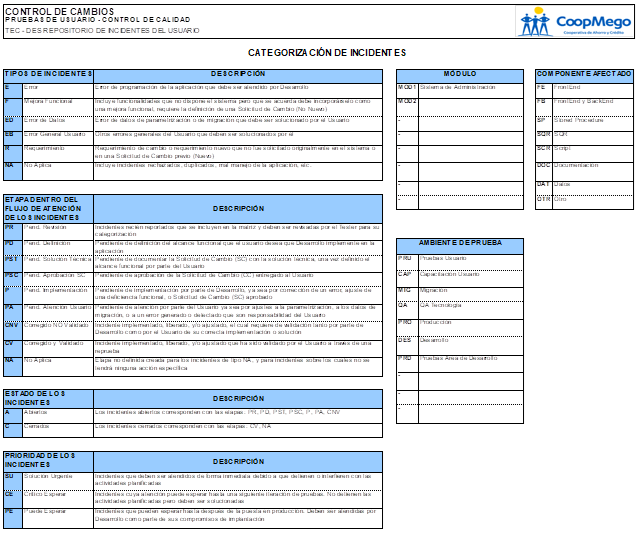
**Observaciones/Recomendaciones:**

* Sugerencias que se pueden realizar al analista de control de cambios para el proceso de pruebas de usuario y puesta en producción.

**URL Producción del sistema:** (Dirección de acceso en producción)

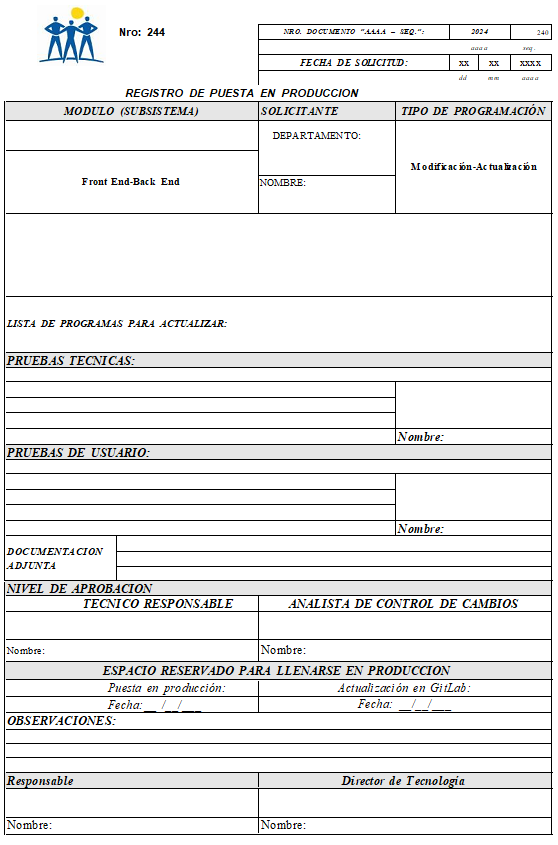
**URL Pruebas del sistema:** (Dirección de acceso en pruebas)

## ANEXO #17: Incidentes del usuario





## ANEXO #18: Registro de puesta en producción



## ANEXO #19: Especificación de requerimientos

Para realizar la especificación de requerimientos será indispensable utilizar una Estructura de Desglose de Trabajo.

Una EDT (Estructura de Desglose de Trabajo), es una descomposición jerárquica del alcance total del trabajo que debe llevar a cabo el equipo del proyecto para lograr los objetivos del proyecto y crear los entregables requeridos; se comienza con el resultado o producto deseado, que luego se desglosa o descompone en los entregables o tareas más pequeñas necesarios para crearlo.

Un EDT puede representarse de forma vertical como una lista, o a través de herramientas visuales como diagramas jerárquicos, mapas mentales, diagramas de Gantt, etc. La representación del EDT queda a discreción del Desarrollador de Sistemas, y este deberá estar disponible para visualización para todos los miembros de la cooperativa.

La Estructura de Desglose de Trabajo estará conformada por:

**Nivel 1 - Nombre del proyecto:**

El nivel más alto del EDT, se detallará el nombre del proyecto o solución que se va a desarrollar, los participantes del proyecto, y el dueño de la solución. Así mismo, tener de referencia el objetivo del proyecto.

**Nivel 2 - Requisitos de alto nivel RAN:**

Los requisitos de alto nivel “RAN”, son unidades de trabajo de alto nivel, que representan funcionalidades o características significativos en un proyecto de desarrollo de software. Tienden a ser grandes y complejas y pueden abarcar múltiples especificaciones funcionales.

Este nivel sirve para agrupar y organizar las especificaciones funcionales relacionadas, y permiten tener una visión general de las metas y objetivos del proyecto. Su redacción debe ser concisa y que describa la funcionalidad que se debe realizar, por ejemplo: “Reportes de gestiones y visitas”.

**Nivel 3 - Especificaciones funcionales EF:**

Las especificaciones funcionales “EF” son descripciones detalladas de funcionalidades o requisitos específicos desde la perspectiva del usuario. Están escritas en un lenguaje comprensible para los Stakeholders, los miembros de la Célula de Trabajo y el Equipo de Desarrollo. Se centran en las necesidades y expectativas de los usuarios finales y se realizarán utilizando la técnica del Círculo Dorado (¿Qué?, ¿Cómo?, ¿Por qué?).

Una especificación funcional deberá presentar de forma clara:

- El resultado esperado, una funcionalidad o característica dentro del requisito de alto nivel, puntual y específico.

- El proceso bajo el cual la necesidad será satisfecha, las acciones específicas a realizar, etapas o pasos del proceso a seguir, la estrategia o plan de acción, etc.

- El propósito de realizar la función, motivos y justificaciones que dejen el claro la intención del producto. Esto nos sirve para priorizar el trabajo durante la planificación de Sprints.

Un ejemplo de EF para la creación de un reporte:

- **Resultado esperado:** REPORTE DE CONTROL DE VISITAS A CLIENTES INCUMPLIDAS.

o El reporte será un documento que se generará periódicamente (semanal, mensual, etc.) dentro del módulo de COBRANZAS. Este informe proporcionará una visión clara de la situación actual de nuestras visitas a clientes y permitirá a la dirección y al equipo de ventas tomar decisiones informadas para mejorar la gestión de estas visitas y, en última instancia, fortalecer las relaciones con nuestros clientes.

- **Proceso:** El reporte de control de visitas a clientes incumplidas se creará como un nuevo reporte dentro del módulo de Reportes en el sistema de COBRANZAS.

o Se recopilarán datos sobre las visitas programadas y visitas incumplidas en nuestro sistema. Esto incluirá la fecha programada de la visita, el estado de la visita (cumplida o incumplida) y los detalles del cliente.

o Los datos recopilados se analizarán para identificar a los clientes que no han sido visitados según lo programado. Esto se hará comparando las fechas programadas con las fechas de visitas realizadas.

o Se creará un informe que muestre una lista de clientes incumplidos, incluyendo sus detalles y las fechas programadas de las visitas no realizadas. Además, se proporcionará información sobre las razones por las cuales no se completaron las visitas, si es posible.

o El reporte incluirá recomendaciones sobre las acciones correctivas necesarias, como la reprogramación de visitas, la asignación de recursos adicionales o la comunicación con los clientes afectados.

- **Propósito:** El propósito fundamental de este reporte es mejorar la gestión de nuestras visitas a clientes incumplidas. Esto es crucial para mantener relaciones sólidas con nuestros clientes y garantizar que estén satisfechos con nuestros servicios. Al identificar a los clientes que no han sido visitados según lo programado, podemos abordar los problemas, fortalecer la confianza y asegurarnos de que se cumplan sus necesidades. Esto, a su vez, ayudará a mantener y aumentar nuestra base de clientes y, en última instancia, aumentar nuestras ganancias.

Además, se debe considerar atributos de calidad como usabilidad, rendimiento, seguridad y mantenibilidad como parte de la EF.

Además, se debe detallar:

- Dificultad para realización de la EF

- Tiempo de desarrollo

- Sprint en el que se desarrolló

**Nivel 4 – Tareas:**

Las tareas son las abstracciones más pequeñas del EDT por lo que van en el último nivel. Son actividades puntuales y específicas que deben realizarse para completar una especificación funcional. Son acciones concretas que los miembros del Equipo de Desarrollo deben realizar, codificar funciones, procedimientos almacenados, crear servicios web, interfaces, etc.

Las tareas deben estar estructuradas de la siguiente manera, mediante el siguiente ejemplo:

- Nombre de la tarea: Consultas a base de datos.

- Descripción de la tarea: Crear consultas y procedimientos almacenados para analizar y recopilar los datos que permitan identificar a los clientes con visitas incumplidas por parte de los gestores de negocio y/o cobranzas, bajo los parámetros solicitados en la EF.

- Responsable: Desarrollador encargado

Al realizar los proyectos en un marco AGILE, se debe modificar el EDT para que se ajuste a la planificación y desarrollo iterativo, por lo que este se actualizará de forma constante a medida que se vayan realizando entregas de valor y se completen las tareas. Al principio del proyecto, mucha de la información quedará en blanco, por lo que es RESPONSABILIDAD del Desarrollador de Sistemas mantener actualizado el EDT durante cada iteración, inicio o finalización de un sprint.

## ANEXO #20: Estilo de programación en pares

Se debe aplicar un enfoque de capacitación continua, para la adquisición de habilidades y conocimientos en roles diferentes a la experiencia, también fomenta la colaboración y el intercambio de conocimientos entre los miembros del equipo de desarrollo.

Es un proceso flexible entre los miembros del Equipo de Desarrollo, donde:

1. Un Desarrollador de Sistemas es asignado al rol de Desarrollador Frontend DF, y otro de Desarrollador Backend DB, cada uno tiene experiencia en ese rol y se le asigna tareas específicas relacionadas.
2. Se establecen objetivos claros por cada sesión de trabajo, que estén alineados a la Planificación del sprint y a las tareas a realizar durante el sprint.
3. Es aconsejable que tenga proximidad en sus puestos de trabajo, o en caso de no ser posible, utilizar herramientas de comunicación continua y brindar respuestas inmediatas y que brinden claridad a los problemas que se presenten.

Este proceso mejora la comprensión mutua entre las complejidades de realizar tareas distintas al área de experticia del Desarrollador de Sistemas, dando flexibilidad al equipo y aumentando la resistencia de este frente a amenazas externas como el cambio de personal, lo que permite que los proyectos no se queden estancados durante períodos extensos de tiempo.

## ANEXO #21: Planificación del sprint

Debe contener la siguiente información:

**Número de Sprint:** [Número de iteración de sprint]

**Fecha de Creación:** [Fecha]

**Objetivo del Sprint**: [Breve descripción del objetivo principal del sprint]

**Duración del Sprint**: [Número de semanas/meses]

[Nombre o Código del Requisito de Alto Nivel]

**Especificaciones Funcionales Seleccionadas**

1. [Nombre o Código de la Especificación Funcional 1]
   * Puntos de Función: [Número de Puntos de Función]
   * Asignado a: [Nombre del Miembro del Equipo]
2. [Nombre o Código de la Especificación Funcional 2]
   * Puntos de Función: [Número de Puntos de Función]
   * Asignado a: [Nombre del Miembro del Equipo]
3. [Nombre o Código de la Especificación Funcional 3]
   * Puntos de Función: [Número de Puntos de Función]
   * Asignado a: [Nombre del Miembro del Equipo]

...

**Tareas Asignadas**

[Nombre de la Especificación Funcional 1]

1. Tarea 1: [Descripción de la Tarea]
   * Asignado a: [Nombre del Miembro del Equipo]
   * Estimación: [Número de Horas/Días]
2. Tarea 2: [Descripción de la Tarea]
   * Asignado a: [Nombre del Miembro del Equipo]
   * Estimación: [Número de Horas/Días]

...

[Nombre de la Especificación Funcional 2]

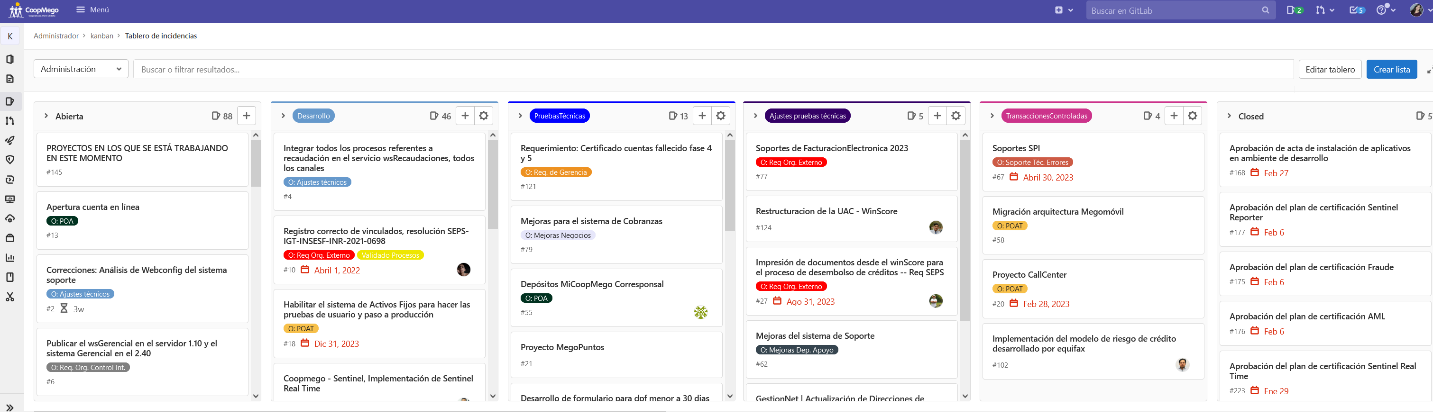
1. Tarea 1: [Descripción de la Tarea]
   * Asignado a: [Nombre del Miembro del Equipo]
   * Estimación: [Número de Horas/Días]
2. Tarea 2: [Descripción de la Tarea]
   * Asignado a: [Nombre del Miembro del Equipo]
   * Estimación: [Número de Horas/Días]

...

**Notas Adicionales**

* El progreso del sprint se evaluará en tres reuniones semanales de seguimiento con el equipo de desarrollo y al menos con una reunión semanal con la célula de trabajo o cuando exista un avance significativo del sprint.
* Cualquier impedimento o problema debe ser comunicado inmediatamente al Jefe de Desarrollo de Sistemas.
* La entrega del sprint está programada para el [Fecha de Entrega].

## ANEXO #22: Kanban



El Método Kanban es una forma de trabajar mediante la gestión visual para facilitar la gestión de proyectos dentro del desarrollo de software. Se basa en el principio de visualizar el flujo de trabajo y limitar la cantidad de trabajo en progreso (WIP) para optimizar la eficiencia y la entrega continua. A continuación, se detalla cómo se puede implementar el Método Kanban en el seguimiento de proyectos, facilitando la visualización del progreso de las etapas del proyecto en lugar de las tareas individuales:

1. **Creación del Tablero Kanban:**

Tablero de Etapas del Proyecto: Se maneja un tablero Kanban que represente las diferentes etapas del proyecto. Estas etapas pueden incluir "Pendiente (Abierto)", "Desarrollo", "Pruebas técnicas", “Ajustes de pruebas de usuario”, "Pruebas de usuario", “Ajustes de pruebas de usuario”, “Pruebas controladas”, “Producción (Cerrado)”, por ejemplo.  
Columnas de Etapas: Cada etapa del proyecto se representa como una columna en el tablero Kanban. Cada columna muestra el estado actual de las tareas en esa etapa.

1. **Definición de las Tareas del Proyecto:**

Cada tarea o actividad del proyecto debe poder accederse desde el tablero Kanban. Las Tarjetas de cada proyecto contienen información sobre cada tarea, incluyendo una descripción, estado actual o porcentaje de avance, desarrollador encargado, fechas estimadas de inicio y fin, y cualquier otra información relevante.

1. **Visualización del Progreso:**

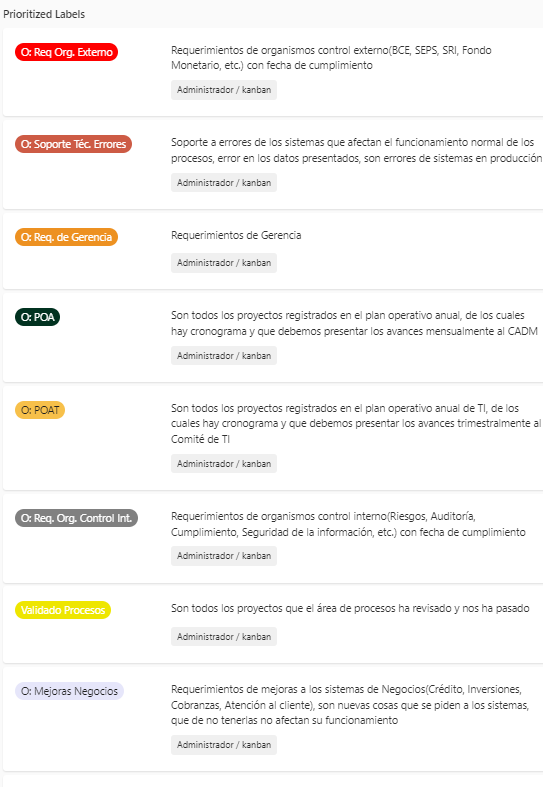
Movimiento de Tarjetas: A medida que las tareas progresan a través de las diferentes etapas del proyecto, las tarjetas se mueven de una columna a otra en el tablero Kanban. Esto proporciona una visualización clara del progreso del proyecto y de las tareas en curso.

1. **Actualización y Seguimiento Continuo:**

Reuniones de Seguimiento: Se sugiere realizar reuniones periódicas para revisar el tablero Kanban y el progreso del proyecto. Durante estas reuniones, se actualizan las tarjetas Kanban según sea necesario y se identifican posibles cuellos de botella o áreas de mejora.

1. **Priorización por etiquetas:**

Los proyectos deben ser segmentados según su origen. Se requiere un conjunto de etiquetas con un orden establecido para identificar su prioridad. Por ejemplo:



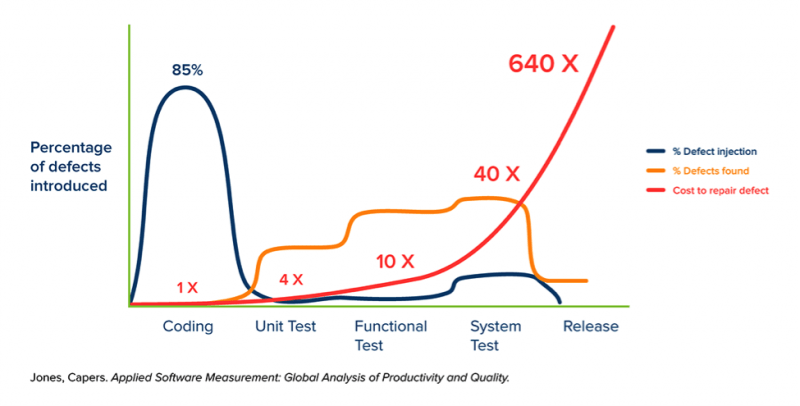
1. **Mejora Continua:**

Retrospectivas Regulares: Se analiza el proceso y se busca oportunidades de mejora. A partir de los comentarios recibidos, se ajusta y optimiza el tablero Kanban y el flujo de trabajo del proyecto.

La implementación del Método Kanban en el seguimiento de proyectos permite una gestión más visual y transparente del trabajo, lo que facilita la colaboración del equipo, mejora la eficiencia y ayuda a cumplir con los plazos y objetivos del proyecto de manera más efectiva.

## ANEXO #23: Seguridad en el desarrollo de Software.

La importancia del desarrollo de software seguro radica en la mitigación de posibles brechas de seguridad. Estas vulnerabilidades pueden comprometer los servicios ofrecidos por la cooperativa, dañar su reputación o exponer información sensible.



Abordar las vulnerabilidades detectadas tardíamente durante el ciclo de vida del desarrollo de software resulta significativamente más costoso que construir un software seguro desde el principio. Por lo tanto, es más económico invertir en seguridad durante el proceso de desarrollo que tratar las brechas de seguridad en un software ya finalizado.

Al aplicar seguridad en cada una de las etapas del ciclo de vida de desarrollo del software, se busca garantizar los principios fundamentales de la seguridad del software:

* **Integridad:** Se refiere a la capacidad del software para protegerse contra modificaciones maliciosas, destrucción o inserción de código malintencionado.
* **Confidencialidad:** Implica asegurar que el software restrinja el acceso no autorizado a su código fuente, contenidos y activos relacionados.
* **Disponibilidad:** El software debe estar disponible y funcional para todos los usuarios, servicios y procesos autorizados en la cooperativa.

Por este motivo, es importante aplicar las seguridades necesarias durante todo el proceso de desarrollo de soluciones informáticas. A continuación, se describe cada uno de ellos:

**Seguridad en el Análisis y Definición de Requerimientos**

Al igual que se definen los requerimientos funcionales, se debe definir los requerimientos de seguridad que garanticen la triada de la seguridad antes mencionada.

Al definir los requerimientos de seguridad, se deben considerar los siguientes aspectos:

* **Análisis de riesgos (**[**Anexo #13**](#_ANEXO_#13:_Lineamientos)**):** Identificar y evaluar los riesgos potenciales para la seguridad del software, y luego implementar medidas de seguridad proporcionales a estos riesgos y prioridades.
* **El servidor que alojará la solución:** Considerar la comunicación que tiene el servidor con otros (en caso de requerir peticiones), características de seguridad del servidor y protocolos de seguridad.
* **Tipo de autenticación de usuarios:** Tipo de autenticación que tendrán los usuarios (contraseñas, tokens, biométricos, etc.).
* **Registros generados por el sistema:** Definir los eventos y acciones que se registrarán en los registros de auditoría para monitorear la actividad del sistema y detectar posibles amenazas o incidentes de seguridad.
* **Perfiles para el acceso al sistema:** Establecer los diferentes roles de usuario y los privilegios asociados a cada perfil para garantizar que solo los usuarios autorizados tengan acceso a las funciones y datos correspondientes.
* **Tipos de acceso a los datos:** Especificar los niveles de acceso que tendrán los diferentes perfiles de usuario a los datos almacenados en el sistema, asegurando que se cumplan los principios de necesidad y mínimos privilegios.

**Seguridad en el Diseño del Sistema**

En esta etapa se define como se construirá la solución informática, cumpliendo con los requerimientos funcionales especificados y las seguridades necesarias que este necesite.

Algunos principios de seguridad que se debe tener en cuenta durante el diseño son los siguientes:

* **Minimizar la superficie de exposición:** Reducir el número de puntos de entrada posibles para un atacante, reduciendo así la exposición a posibles vulnerabilidades.
* **Criterio de menor privilegio:** Conceder a los usuarios o componentes del sistema únicamente los privilegios necesarios para llevar a cabo sus funciones.
* **Criterio de defensa en profundidad:** Implementar múltiples capas de seguridad en un sistema, de modo que, si una capa es comprometida, las demás capas puedan prevenir o mitigar el impacto del ataque.
* **Separación de privilegios:** Dividir las funciones y privilegios del sistema para que cada usuario o componente solo tenga acceso a los recursos y datos necesarios para llevar a cabo sus tareas.
* **Administración segura de información sensible:** Implementar controles adecuados como cifrado de datos, controles de acceso y auditorías de seguridad.
* **Diseño de Auditoria y Logs:** Definir las tramas que se registrarán en la auditoría, permitiendo rastrear y registrar eventos importantes del sistema, para facilitar la detección y respuesta a posibles ataques o violaciones de seguridad.

**Seguridad en el Desarrollo del Sistema**

Esta etapa va de la mano con los lineamientos de la etapa de diseño al igual el tiempo es proporcional a que tan bien esté definido el diseño.

En esta etapa se puede tener en cuenta los siguientes puntos:

* **Siempre validar los datos de entrada previo a procesarlos:** Verificar que los datos proporcionados por el usuario o externamente cumplan con los requisitos esperados antes de realizar cualquier tipo de procesamiento.
* **No confiar en que los datos recibidos son correctos:** Asumir que los datos proporcionados pueden ser maliciosos o incorrectos, y, por lo tanto, validarlos adecuadamente antes de utilizarlos.
* **Utilizar validación de datos entre capas:** Validar los datos en cada punto de entrada del sistema, incluyendo la interfaz de usuario, las APIs, y cualquier otro punto de acceso.
* **Controlar el tamaño de datos a recibir:** Limitar el tamaño y tipo de datos aceptados para evitar posibles ataques de denegación de servicio (DoS) o manipulación de datos.
* **Eliminar caracteres especiales** ([Anexo #7](#_ANEXO_#7:_Estándares))**:** Eliminar o neutralizar caracteres especiales que puedan ser utilizados para ataques de inyección de código, como por ejemplo SQL injection o Cross-Site Scripting (XSS).
* **Capturar errores de capas inferiores y no mostrarlos al usuario:** Manejar los errores de manera adecuada, capturándolos en las capas inferiores del sistema y presentando mensajes de error genéricos al usuario para evitar la divulgación de información sensible o potencialmente útil para un atacante.
* **Control de Acceso Basado en Parámetros:** Establecer controles de acceso basados en los parámetros del usuario y del contexto de la solicitud, limitando el acceso a funcionalidades y recursos sensibles según los roles y privilegios asignados.

**Pruebas de Seguridad**

Estas pruebas buscan confirmar que los criterios de seguridad definidos en la etapa de análisis y levantamiento de requerimientos fueron implementados en la etapa de diseño. Algunas pruebas de seguridad son:

**Pruebas de seguridad funcional**

* **Autenticación:** Evaluar la robustez del proceso de autenticación del sistema y resistencia a posibles ataques de suplantación de identidad.
* **Complejidad de contraseñas:** Evaluar que el sistema imponga una política de contraseña para que sea seguro el ingreso al sistema.
* **Bloqueo de cuentas:** Probar el bloqueo de cuenta después de ciertos intentos de acceso.
* **Restricciones de acceso en el diseño:** Impedir el acceso a funcionalidades que no tengan autorización ni el perfil sugerido.
* **Mecanismos de registros y logs:** Verificar como se está almacenando la información para que sea posible realizar auditoría sobre las operaciones realizadas por el usuario que tenga sesión activa.
* **Mensajes de error especificados:** Verificar si los mensajes de error que presentan los sistemas son los correctos y no revelan información delicada que pueda ser usada por atacante para vulnerar la seguridad.
* **Pruebas de seguridad basadas en riesgo:** Dependiendo del método seleccionado, se debe validar que las medidas propuestas han sido implementadas correctamente para mitigar los escenarios propuestos durante el análisis de riesgo.
* **Test de carga:** Permite determinar el límite de rendimiento del sistema y observar cómo responde ante picos de actividad o cargas de trabajo intensas.
* **Test de mutación de datos:** Técnica utilizada para evaluar la calidad y eficacia de las pruebas unitarias al introducir cambios deliberados en el código y observar cómo responden las pruebas a estos cambios.

**Seguridad en la Implementación**

Se verifica que la línea base de la configuración que se planteó sea la misma sobre la cual se hará el despliegue; para ello hay que tener en cuenta los siguientes puntos:

* **Proceso de Implementación:** La separación de ambientes ayuda a prevenir que problemas o vulnerabilidades en el entorno de desarrollo o pruebas afecten al entorno de producción, lo que reduce el riesgo de interrupciones o compromisos de seguridad.
* **Administración de implementación y mantenimiento: Releases, Parches y Firma de código:** Refiere a la gestión de la implementación y el mantenimiento continuo del software, liberando las mejoras y correcciones de vulnerabilidades que se detecten, asegurando que el software se mantenga seguro y funcional a lo largo del tiempo.
* **Validación de Parámetros:** Garantiza que los valores configurados en el sistema sean adecuados y estén alineados con los permisos y roles de usuario asignados.

## ANEXO #24: Registro de Logs

En todos los desarrollos realizados, se debe implementar un sistema de registro de logs que cumpla con los siguientes requisitos:

1. **Aplicación de la Norma ISO 20022:** El registro de logs del código fuente deben seguir los estándares establecidos en la Norma ISO 20022 en términos de formato, estructura y codificación. Esto incluye campos estandarizados, identificadores únicos y cualquier otra información requerida por la norma.
2. **Trazabilidad de servicios:** Se deben registrar las tramas de entrada y salida de los servicios, incluyendo detalles como la fecha y hora, el tipo de solicitud, el código de transacción, los parámetros enviados y las respuestas recibidas.
3. **Errores de base de datos:** Se debe registrar cualquier error relacionado con operaciones de base de datos, incluyendo consultas SQL fallidas, conexiones perdidas o problemas de integridad de datos.
4. **Excepciones:** Se deben registrar todas las excepciones generadas durante la ejecución del sistema, junto con información relevante como la ubicación en el código, el tipo de error y los datos involucrados.
5. **Modificaciones en la base de datos:** Se deben registrar las operaciones de inserción(Insert), actualización (Update) y eliminación (Delete) de registros en la base de datos, indicando el usuario responsable, el tipo de operación, los datos afectados, la fecha de transacción, el tipo de transacción, datos de origen (terminal, canal); en fin, la información almacenada en los logs deberá contestar a las preguntas de: ¿Quién ejecutó la acción?, ¿Qué acción ejecutó?, ¿Cuándo se ejecutó la acción?, ¿Desde donde se ejecutó la acción (IP, IMEI, MAC, georreferenciación, etc.)?; y, ¿Qué datos afectó la acción ejecutada?.
6. **Consulta de campos sensibles o críticos:** Se registrará el login del funcionario que realice consultas de campos sensibles o críticos definidos en el Manual de Seguridad de la información.

El registro de logs debe mantenerse de forma segura y accesible por el equipo de desarrollo solo en caso de soportes puntuales. Además, se deben establecer políticas claras sobre la retención y el almacenamiento de los registros, garantizando la confidencialidad y la integridad de la información registrada.