Control de Versiones

|  |  |  |  |
| --- | --- | --- | --- |
| **Versión** | **Fecha** | **Cambio** | **Responsable** |
| 1.0 | Junio 2022 | Elaboración | Dirección de Desarrollo Tecnológico |
| 1.1 | Agosto 2022 | Revisión | Dirección Coordinadora de Innovación y Desarrollo Tecnológica |

Contenido

[1. Objetivo 3](#_Toc110240866)

[2. Alcance 3](#_Toc110240867)

[3. Prerrequisitos para el desarrollo 3](#_Toc110240868)

[4. Metodología 5](#_Toc110240869)

[5. Proceso de Desarrollo Ágil Scrum 6](#_Toc110240870)

[5.1. Ideación o planteamiento 6](#_Toc110240871)

[5.2. Incepción 7](#_Toc110240872)

[5.3. Iteración o Construcción 8](#_Toc110240873)

[5.4. Puesta en Marcha o Liberación 11](#_Toc110240874)

[6. Roles 12](#_Toc110240875)

[1. Roles en el servicio 12](#_Toc110240876)

[1.1. Product Owner 12](#_Toc110240877)

[1.1.1. Actividades: 13](#_Toc110240878)

[7. Artefactos 16](#_Toc110240879)

[7.1. Product Backlog 16](#_Toc110240880)

[7.2. Sprint Backlog 16](#_Toc110240881)

[7.3. Historias de Usuario 16](#_Toc110240882)

[7.3.1. Elementos de las historias de Usuario 17](#_Toc110240883)

[8. Proceso Constructivo 18](#_Toc110240884)

[8.1. Diseño 18](#_Toc110240885)

[8.2. Construcción 21](#_Toc110240886)

[8.3. Pruebas 24](#_Toc110240887)

[9. Glosario 27](#_Toc110240888)

# **Objetivo**

Proporcionar una base para el desarrollo de los sistemas al interior de la Secretaría tanto por personal de esta, así como por proveedores externos.

En esta metodología se definen los lineamientos y procedimientos a observarse en el diseño, desarrollo, implementación, mantenimiento y atención de requerimientos de las aplicaciones Informáticas Institucionales.

# **Alcance**

Este documento aplica para todos los sistemas existentes de la SICT.

A la fecha en que se define esta metodología, ya existen sistemas que no se apegan a estos lineamientos, por lo que será esfuerzo compartido entre personal de la UTIC así como de los proveedores de desarrollo y mantenimiento de software, evolucionar las herramientas previamente existentes de forma gradual, para que todo sistema evolucione y cumpla con lo señalado en el presente documento.

# **Prerrequisitos para el desarrollo**

Con la finalidad de hacer un uso eficiente de los recursos asignados, lograr el mayor impacto posible con las limitantes existentes y dar soporte a la operación de los distintos sistemas existentes en la Secretaría, previo a la elaboración de nuevos sistemas, módulo o herramientas, se deberá de solicitar al dueño del proceso de la Unidad Administrativa, el requisitado y envío mediante oficio de la siguiente tabla. En este formato se relaciona la necesidad de contar con dicha solución, su impacto, beneficios tanto a la ciudadanía, como a la administración de esta Dependencia que permitan justificar y sustentar la inversión requerida para dicho sistema.

|  |  |  |  |
| --- | --- | --- | --- |
| **Preguntas** | **Datos** | **Limitaciones** | |
| **Nombre del sistema** |  |  | |
| **Unidad Administrativa solicitante** |  |  | |
| **Nombre del requirente autorizado (Product Owner)** |  |  | |
| **Maneja datos personales** |  | Sí / No | |
| **¿Como funciona el sistema?** |  | 200 palabras | |
| **Situación Actual (Problemática):** |  | 200 palabras | |
| **Alcance general**  **(¿Qué se espera lograr?)** |  | 200 palabras | |
| **Origen del requerimiento: (¿Cómo surge la necesidad?)** |  | 200 palabras | |
| **Que debe hacer el usuario para hacer uso del sistema** |  | 200 palabras | |
| **Usuarios del Sistema** |  | Internos (SICT), Externos Gobierno, Externos Ciudadanos, Todos | |
| **Unidades Administrativas o usuarios impactados por el cambio** |  | Nombre de la Unidad o usuario beneficiados | |
| **Cantidad de usuarios que utilizan el sistema** |  | Número aproximado anual | |
| **Riesgo identificado de no contar con la aplicación / solución** |  | 200 palabras | |
| **BENEFICIOS ESPERADOS DE LA APLICACIÓN/SOLUCIÓN** | | |
| 1. **Mejora la experiencia de la ciudadana** |  | Sí/No (Explique) | |
| 1. **Se alinea con la Austeridad Republicana** |  | Sí/No (Explique) | |
| 1. **Incrementa la productividad del personal** |  | Sí/No (Explique, defina métrica y cantidad esperada) | |
| 1. **Reduce costos a la SICT** |  | Sí/No (Explique, defina cantidad esperada) | |
| 1. **Mejora la toma de decisiones** |  | Sí/No (Explique) | |
| 1. **Reduce tiempos en los tramites** |  | Sí/No (Explique, defina métrica y cantidad esperada) | |
| 1. **Mejora la imagen y diseño de los sistemas** |  | Sí/No (Explique) | |

# **Metodología**

La definición de la metodología a utilizarse en los distintos proyectos de la Secretaría estará determinada por el tipo de proyecto y el proceso evolutivo que deberá observar durante la ejecución del mismo. Por lo anterior, será posible utilizar una de las siguientes: Metodología ágil mediante Scrum o Metodología híbrida Cascada - Ágil.

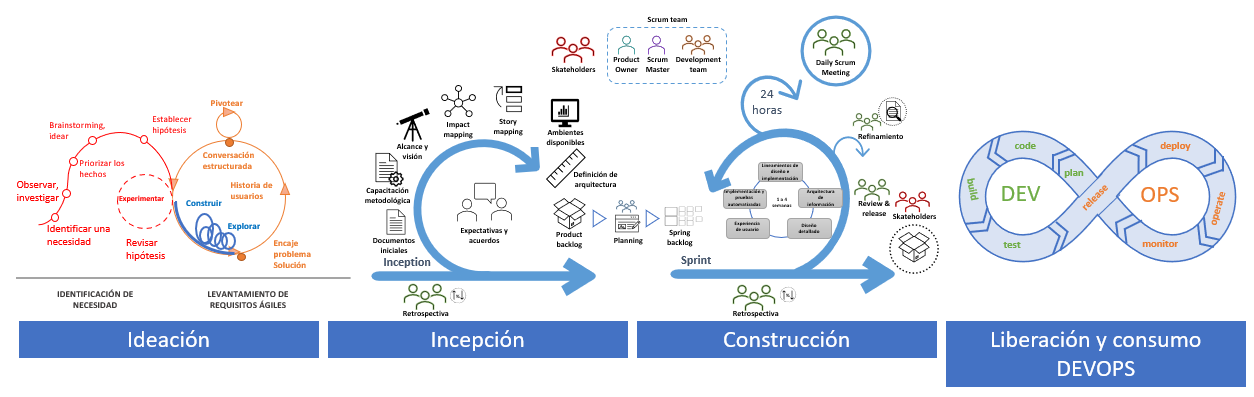
La metodología ágil es un enfoque de gestión de proyectos basado en la entrega de requisitos de forma iterativa e incremental a lo largo del ciclo de vida. Esta metodología se utilizará en la mayor cantidad de proyectos, donde se deberá de retroalimentar constantemente a los usuarios dueños de los procesos, para validar el resultado obtenido en cada paso, así como poder ajustar en el camino las nuevas necesidades o cambios en las reglas de negocio.

La metodología híbrida Cascada - Ágil será utilizada en los casos de Reingenierías de sistemas, donde ya existe una herramienta operativa y requiera una actualización completa. Se utilizará esta metodología cuando se tenga definido un alcance para su ejecución, con objetivo y funcionalidades claras, así como con un tiempo delimitado para su conclusión. Sin embargo, para permitir la evolución de estos sistemas, se utilizará en parte la metodología ágil con Scrum, lo que permitirá la evolución de funcionalidades y reglas de negocio, así como, para dar visibilidad al usuario dueño del proceso.

# **Proceso de Desarrollo Ágil Scrum**

El proceso de desarrollo ágil se utiliza para construir software, de calidad, y de forma eficiente, consta de 4 fases generales

* 1. Ideación o planteamiento
  2. Incepción Requerimientos priorizados
  3. Iteración o Construcción
  4. Puesta en Marcha o Liberación



# **Ideación o planteamiento**

Consiste en la Identificación de la necesidad, la identificación del problema a resolver y de los resultados esperados al terminar el desarrollo.

Responsable: **Product Owner (PO)**

1. Levantamiento de requisitos
2. Entrevista global workshop (lluvia de ideas)
   1. Junto con el personal del área usuaria el PO se encarga de:
      1. Entender la visión del área
      2. Las Expectativas de los usuarios
      3. Definir el alcance
      4. Identificar los principales usuarios y beneficiarios
      5. Comprender el Proceso sistematizado o a sistematizar
      6. Entender los problemas para resolver
      7. Responder:
         1. ¿Que se quiere?
         2. ¿Como se quiere?
         3. ¿Cuándo se quiere?
         4. ¿Porque se quiere?
      8. Definir la viabilidad operativa del proyecto
      9. Analizar los requisitos, validaciones y pruebas requeridas para garantizar que el desarrollo será funcional para asegurar que el desarrollo hace todo lo que se requiere y solo lo que se requiere.
   2. Construir las historias usuario HU - Historia

# **Incepción o Inicio**

Responsable: **Product Owner**

1. Refinar las historias de Usuario para asegurar que todas estas contengan
   1. Título de la Historia
   2. Descripción de la historia
   3. Boceto (un dibujo, reporte, manual, descripción del proceso, que permita describir la funcionalidad requerida)
   4. Los criterios de aceptación
   5. Casos para validación y escenarios de consumo
2. Que las historias estén priorizadas para ser atendidas por el equipo de desarrollo.

# **Iteración o Construcción**

Responsable: **Scrum Máster y el Equipo de Desarrollo**

El proceso de construcción se lleva a cabo dentro del tiempo que dura el Sprint y se ejecuta a través del Sprint Planing, Sprint Daily, Sprint Review y el Sprint Retrospective, sin embargo

Dentro de esta etapa debe existir

1. **Sprint Planning**
   1. Una vez que se tengan Product Backlog (historias de usuario) se procede a la Reunión de planificación.

En esta reunión vamos a planificar QUÉ es lo que vamos a hacer durante el Sprint y cómo lo vamos a hacer.

Durante la planificación interviene todo el Equipo Scrum, es decir, el Product Owner, el Scrum Máster y el Equipo de Desarrollo.

La estructura de la reunión está dividida de manera tal que conteste los siguientes dos interrogantes:

1. ¿QUÉ podemos entregar para lograr un nuevo Incremento de Producto durante este Sprint?

2. ¿CÓMO lo vamos a conseguir?

* 1. **Etapa 1**

Durante esta reunión el Product Owner debate con todo el equipo ¿Cuál es el Objetivo del Sprint que se quiere alcanzar? Explica y se asegura de su correcto entendimiento, de todos los detalles de los ítems del Product Backlog que deberían completarse para cumplir dicho objetivo.

* 1. **Etapa 2**

Una vez que se ha establecido el objetivo y seleccionado las historias para el Sprint, se decide y planifica como construirán estas funcionalidades para llegar a un Incremento de Producto terminado durante el Sprint.

En este punto se realizan las estimaciones mediante la metodología definida por la Dirección Coordinadora de Innovación y Desarrollo Tecnológica entre las que se pueden encontrar siendo enunciativos mas no limitativos:

* Puntos COSMIC
* Puntos de Historia Pocker Planing
* Puntos por función

Se empieza a descomponer las historias en partes más pequeñas a las que llamaremos tareas.

Las tareas son todas las actividades técnicas que tienen que completarse, al dividir los ítems en tareas se recomienda considerar que una tarea debe poder completarse en máximo un día de trabajo.

1. **Sprint Daily (Reunión Diaria)**

Una vez planificado el Sprint y las tareas para concluir durante el sprint se procede a la construcción del software, sistema (su incremento correspondiente al sprint).

El objetivo es lograr que el Equipo de Desarrollo se sincronice, para ello se planea el trabajo de las siguientes 24 horas

Durante esta reunión se espera que cada persona del Equipo de Desarrollo conteste las siguientes 3 preguntas en 2 a 3 minutos por integrante, a modo de agenda para optimizar la eficiencia del tiempo:

* ¿Qué hice ayer para ayudar a lograr el Objetivo del Sprint?
* ¿Qué voy a hacer hoy para ayudar a lograr el Objetivo del Sprint?
* ¿Existe algún impedimento que evite que el Equipo de Desarrollo o yo logremos el Objetivo del Sprint l?

1. **Sprint Review (Reunión de Revisión)**

Una vez terminado el tiempo del sprint y que las tareas se han concluido se lleva a cabo esta reunión donde se revisa el Incremento de Producto, es decir, lo que se realizó́ durante el Sprint.

El principal objetivo es obtener retroalimentación de los interesados para inspeccionar y evaluar el producto a fin de ajustar las Historias de Usuario.

Participan el Product Owner, el Scrum Máster, el Equipo de Desarrollo y los interesados clave, invitados por el Product Owner.

* El Product Owner y el equipo de desarrollo expone las historias de usuarios que se han terminado.
* El Equipo de Desarrollo realiza una exposición del Incremento de Producto.

En este punto el equipo de Desarrollo sube el incremento al ambiente de QA para que el Product Owner pueda revisarlo.

1. **Reunión de Retrospectiva del Sprint**

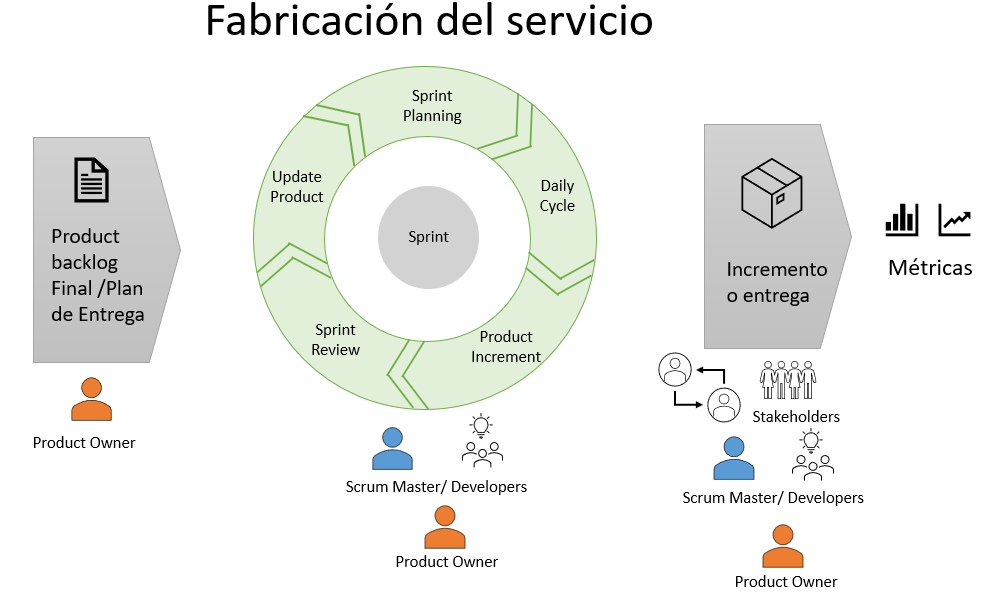
Participa el equipo Scrum completo

Posterior a esta reunión se puede tener la Reunión de Retrospectiva del Sprint. Una retrospectiva es una reunión donde el equipo participa después de completar un incremento de trabajo para inspeccionar y adaptar sus métodos y trabajo en equipo.

Según la Guía Scrum Oficial (2017) el propósito de la Retrospectiva de Sprint es:

* Inspeccionar cómo fue el último Sprint en cuanto a personas, relaciones, procesos y herramientas;
* Identificar y ordenar los elementos más importantes que salieron bien y las posibles mejoras; y,
* Crear un plan para implementar las mejoras a la forma en la que el Equipo Scrum desempeña su trabajo.

Este Ciclo se repite generando incrementos hasta lograr el producto objetivo.

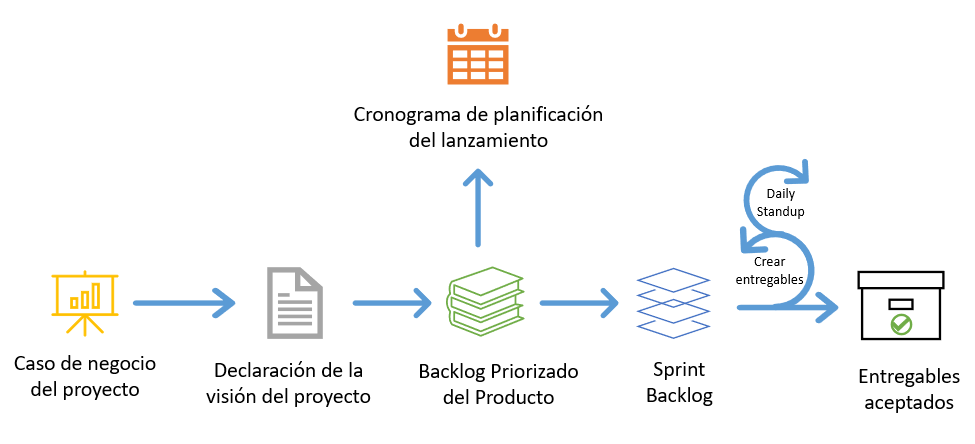


# **Puesta en Marcha o Liberación**

Una vez probado el incremento es Responsabilidad del Product Owner aprobar su implementación y liberación en producción.

Esto puede hacerse de 2 formas actualmente:

1. Mediante DevOps (Implementación continua) cuando el despliegue se hace en la plataforma de contenedores con la que cuenta la Secretaría (Open Shift).
   1. El Administrador del Contenedor de la UTIC notifica al Grupo de DevOps y realiza el Despliegue.
2. Mediante el envío de un archivo de despliegue enviado al Personal del Centro de Datos para su implementación.



# **Roles**

# **Roles en el servicio**

Los roles son funciones específicas desempeñadas por un recurso de acuerdo con su experiencia en un lugar o situación específica, independientemente del rango o perfil curricular, considerando las necesidades puntuales del proyecto en cada una de sus etapas.

De acuerdo con el marco de la metodología ágil, se definen los siguientes roles y responsabilidades, siendo enunciativos mas no limitativos:

# **Product Owner**

Es el representante de los intereses de la Unidad Administrativa de la SICT a la que pertenece y es designado por el Director General de la misma.

Entiende y transmite al Equipo de desarrollo las necesidades e intereses de los usuarios de la Unidad Administrativa a la que pertenece. Se asegura de reflejar las necesidades de su Unidad Administrativa en las historias de usuario.

# **Actividades:**

* Conoce los procesos de la Unidad Administrativa de su adscripción
* Tiene capacidad para tomar cualquier decisión que afecte al producto
* Define la visión del proyecto
* Crea, refina y registra épicas, historias de usuario y criterios de aceptación en el backlog
* Crea las Historias de Usuario
* Prioriza los elementos del backlog
* Orienta y aclara dudas al Equipo Scrum respecto de sus necesidades
* Acepta o rechaza los entregables
* Inicia y finaliza *sprints*
* Participa en las reuniones de retrospectiva del *sprint*

# **Scrum Máster**

Guía, facilita y enseña las prácticas de Scrum a todos los involucrados en el proyecto. Se asegura que todos los integrantes del equipo sigan correctamente los procesos de Scrum. Se asegura de que el desarrollo del producto avance sin contratiempos y que los integrantes del Equipo de desarrollo cuenten con todas las herramientas necesarias para realizar el trabajo.

# **Actividades:**

* Gestiona y asegura que el proceso Scrum se lleva a cabo correctamente
* Elimina progresiva y constantemente impedimentos que van surgiendo en el equipo scrum y que afectan la capacidad de este para entregar valor
* Facilita la creación de épicas e historias de usuario
* Ayuda al *Product Owner* en la creación del backlog priorizado del producto
* Organiza las reuniones del Equipo Scrum.
* Funge como líder que ayuda a motivar al equipo.
* Ayuda al Equipo Scrum a estimar el esfuerzo necesario para completar las tareas acordadas para el *sprint*.

# **Equipo de desarrollo.**

Responsable del desarrollo del producto, consiste en un grupo de personas que trabaja en las historias de usuario para crear los entregables del proyecto. Garantiza que los entregables del proyecto sean elaborados según los requerimientos.

# **Actividades:**

* Asegura una comprensión clara de las épicas e historias de usuario.
* Proporciona aportes al *Product Owner* en la creación de historias de usuario.
* Estima las historias de usuario
* Compromete historias de usuario a realizarse en un sprint
* Desarrolla una lista de tareas con base en las historias de usuario y dependencias acordadas
* Estima el esfuerzo de las tareas identificadas y de ser necesario, actualiza la lista de tareas.
* Elabora los entregables
* Identifica riesgos y ejecuta acciones de mitigación de riesgos, si los hay
* Discute los problemas que enfrenta cada miembro y busca soluciones para motivar al equipo
* Muestra los entregables completados al *Product Owner* para su aprobación
* Identifica oportunidades de mejora, si las hay, del Sprint actual y decide si está de acuerdo sobre las posibles mejoras viables para el próximo sprint
* Participa en la reunión de retrospectiva del proyecto

# **Miembro del Scrum Guidance Body (Líderes del Servicio UTIC)**

El Miembro del Scrum Guidance Body (SGB) “actúa como un consultor, está involucrado en la definición de los objetivos relacionados con la calidad, las regulaciones, la seguridad y otros parámetros claves de la organización. El SGB guía el trabajo llevado a cabo por el *Product Owner*, el Scrum Máster y el Equipo Scrum. También ayuda a captar las mejores prácticas que deben utilizarse en todos los proyectos de Scrum en la organización. Interactúa con los miembros del Equipo Scrum durante los procesos de Crear historias de usuario, Estimar tareas, Crear entregables y Refinamiento del Backlog Priorizado del Producto para ofrecer orientación y también proporcionar conocimientos según sea necesario.”

# **Actividades:**

* Participa en las reuniones que el Scrum Máster programe
* Registra historias de usuario correspondientes al ámbito tecnológico
* Registra los criterios de aceptación de las historias de usuario (tecnología)
* Puede Iniciar y cerrar sprints
* Puede solicitar cambio de prioridades en los requerimientos

El registro y gestión de solicitudes del servicio de mantenimiento se realizará a través de la herramienta de operación establecida por la UTIC, las solicitudes de mantenimiento evolutivo podrán ser susceptibles de cambios de alcance, cambios en los supuestos, planes o cambios en las condiciones bajo las cuales se hayan planteado originalmente. Estos cambios son responsabilidad del Product Owner Product Owner de la Unidad Administrativa responsable de la aplicación Informática, quien tendrá acceso a la herramienta para el registro correspondiente de la solicitud de mantenimiento evolutivo.  El Product Owner coordinará con el administrador del contrato de la UTIC, o con el personal de la UTIC designado por el administrador del contrato, el registro de las solicitudes en la herramienta.

# **Artefactos**

Corresponde a las evidencias que se registran a lo largo del proyecto, para soportar los tiempos y esfuerzos requeridos en la construcción del producto final (sistema). Los formatos para este propósito se definen dependiendo del tipo y alcance de cada contrato y/o proyecto.

# **Product Backlog**

El Product Backlog es una lista ordenada de todo lo que se conoce que es necesario en el producto. Es la única fuente de requisitos para cualquier cambio a realizarse en el producto. El Dueño de Producto (Product Owner) es el responsable de este, incluyendo su contenido, disponibilidad y ordenación.

# **Sprint Backlog**

Es el conjunto de elementos del Product Backlog seleccionados para el Sprint, más un plan para entregar el Incremento de producto y conseguir el Objetivo del Sprint. El Dueño de Producto (Product Owner) es responsable de asegurar que para cada Sprint se tengan suficientes elementos (historias de usuario) para obtener un incremento medible y sustancial.

# **Historias de Usuario**

Son representaciones de los requisitos, elaborados en una o dos frases y recogidas en lenguaje común y entendible por el área usuaria.

Estas Historias deben ser:

* Independientes: Debe ser independiente de otras historias.
* Negociable: Su alcance y criterios deben ser variables.
* Valorable: Deben aportar valor real al cliente, un incremento de producto completo.
* Estimable: Deben poder estimarse por el Equipo de Desarrollo por lo cual no deben ser demasiado grandes y debemos tener cierto conocimiento de esta a nivel negocio y técnico.
* Small (pequeña): Debe poder completarse dentro de un Sprint.
* Testable (comprobable): Debe ser posible verificar que la misma está completa una vez desarrollada. Para ello debe tener claros criterios de aceptación con los cuales verificamos que esté realmente lista.

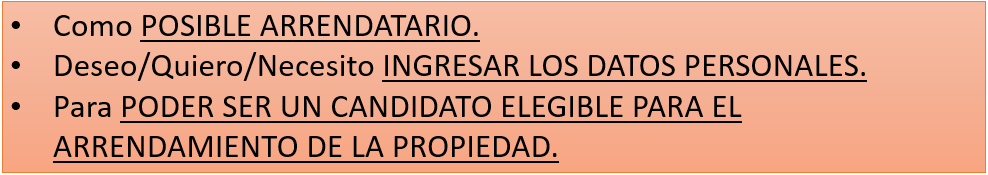
# **Elementos de las historias de Usuario**

Al ser el principal insumo para el desarrollo de sistemas, modulo y mejoras las historias de usuario deben tener los siguientes elementos:

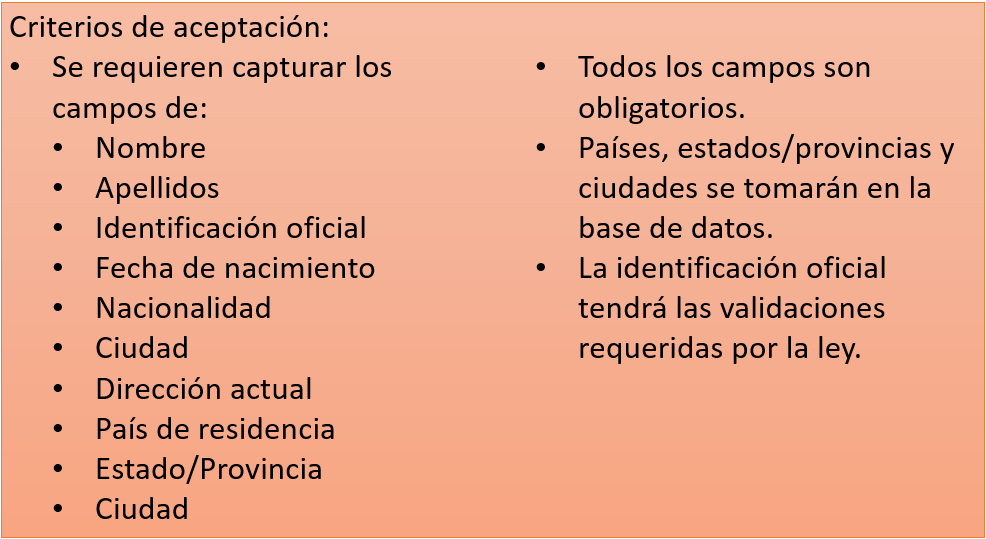
* **Título:** Tema de la historia de Usuario ejem:



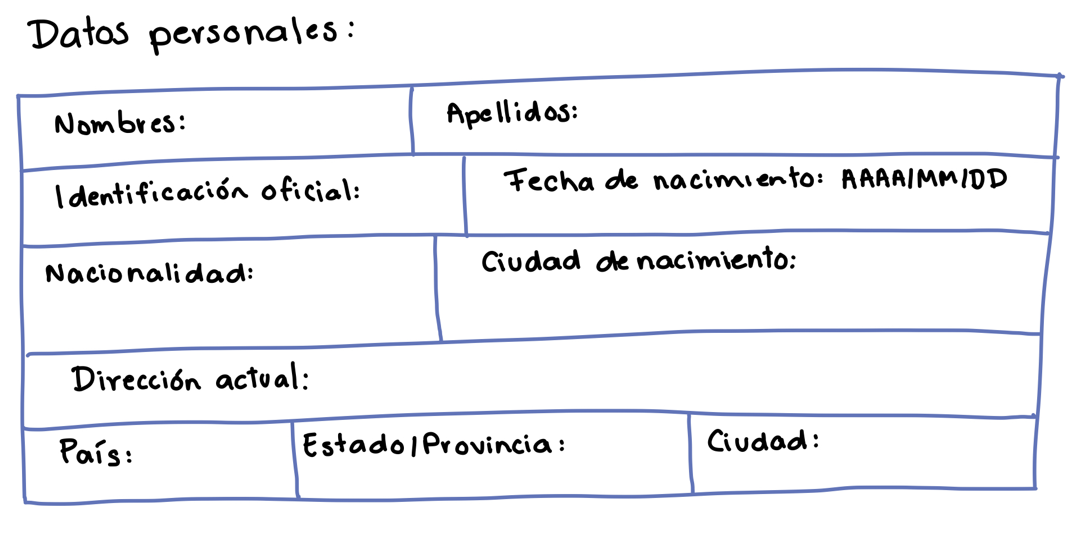
* **Descripción:** La descripción del requerimiento de lo que se requiere, para que se requiere y la visión de quien lo requiere.



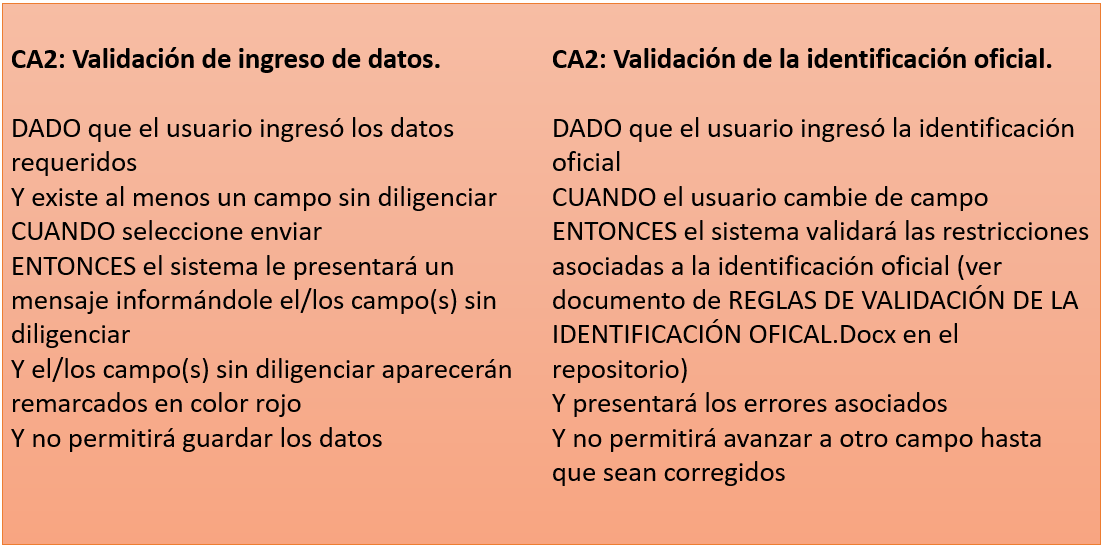
* **Criterios de aceptación:**

****

* **Bocetos:** Un dibujo – reporte, manual, descripción del proceso o cualquier documento que permita describir la funcionalidad o necesidad a realizar.

****

* **Escenarios y Validaciones:**



# **Proceso Constructivo**

El proceso constructivo se desarrolla dentro del tiempo del Sprint y se divide en las siguientes fases.

# **Diseño**

El equipo de desarrollo debe tener claro el objetivo, alcance, operatividad e interfaces requeridas para el desarrollo de la herramienta. Una vez clara esta información, se deberá de ejecutar en paralelo un análisis al interior del equipo, buscando:

Utilizar la programación orientada a objetos, donde se definan claramente las dependencias, herencias, polimorfismo, etc. Para lo cual se necesita tener claros los patrones de diseño a seguir y el principio SOLID. Favorecer la reutilización de componentes. Requiere la realización de diagrama de clases para el módulo a realizar.

Hacer la arquitectura de la solución, favoreciendo el funcionamiento mediante APIs. Diseñar y definir las interfaces internas al sistema a realizar o a modificar, considerando que se deben de seguir los principios SOLID. Requiere la realización de diagrama de interacción de los módulos Es importante que, para este apartado, se observe que el objetivo de la Secretaría es ir migrando las herramientas con el objetivo de poder ser utilizadas y explotadas mediante dispositivos móviles, por lo que todo deberá ir enfocado en simplificar, agilizar y permitir la explotación de información mediante dichos dispositivos.

Validar y proponer mejoras en el diseño de las interfaces propuestas por el usuario dueño del proceso, considerando como base, contar con interfaces ligeras, que reduzcan los clics a realizar para ejecutar una tarea o función. Se deberá de respetar la imagen institucional y estructurar las mismas mediante hojas de estilo, o metodologías que permitan la fácil actualización de estas interfaces. Validar la información suministrada por los usuarios antes de procesarla, teniendo en cuenta aspectos como: tipo de datos, rangos válidos, longitud, listas de caracteres aceptados, caracteres considerados peligrosos y caracteres de alteración de rutas, entre otros.

Se deberá de revisar, en caso de procesos ya existentes, la estructura de la base de datos, la calidad de la información ahí almacenada y proponer los evolutivos en los mismos, de forma tal que se normalice la base de datos, se depure la información obsoleta (tablas duplicadas) y se cuente con procesos internos (procedimientos almacenados, vistas, disparadores) que faciliten la explotación de la información.

La estructura de cualquier sistema debe estar diseñada de forma tal que permita la independencia de la UTIC de los programadores. Por lo que todo sistema deberá de considerar su administración por los mismos usuarios finales, basándose en roles y privilegios. Por lo que estos últimos podrán trabajar en independencia de la UTIC para procesos diarios. La UTIC serán los únicos que cuenten con privilegios de Super Administrador. El máximo perfil que podrá tener un usuario de una Unidad Administrativa es la de Administrador.

Todo sistema deberá ser configurable a través de un único archivo, desde donde se modificarán las variables para el correcto funcionamiento del sistema.

Contar con mecanismos de desconexión o cierre de sesión de los aplicativos (logout) que permitan terminar completamente la sesión o conexión asociada, las cuales deben encontrarse disponibles en todas las páginas protegidas por auto-conexión.

Proporcionar la mínima información de la sesión establecida, almacenada en cookies y complementos, entre otros.

Prevenir la revelación de estructura de directorios de los sistemas de información construidos

Certificar el cierre de la conexión a las bases de datos desde los aplicativos, tan pronto como estas no sean requeridas.

Proteger el código fuente de los aplicativos construidos, de tal forma de que no pueda ser descargado ni modificado por usuarios o personas no autorizadas.

Implementar medidas de seguridad al sistema en desarrollo, para satisfacer los objetivos de seguridad de la Secretaría y sus Unidades Administrativas.

Proteger la información utilizada por el sistema en desarrollo, mientras está siendo procesada, en tránsito o en almacenamiento

Usar únicamente los lenguajes, frameworks, base de datos, servicios en la nube con versiones autorizadas.

Implementar medidas de seguridad como el cifrado en tablas y datos de cualquier información confidencial.

Utilizar canales y protocolos de comunicación seguros que permita asegurar la confidencialidad, integridad, durante el procesamiento, envío y recepción de información dentro y fuera de la aplicación.

**Nota**: Los numerales anteriores deberán de aplicarse en los nuevos desarrollos. En los existentes se dará de forma gradual. Podrá haber excepciones, las cuales deberán ser autorizadas por la DCIDT de la UTIC.

# **Construcción**

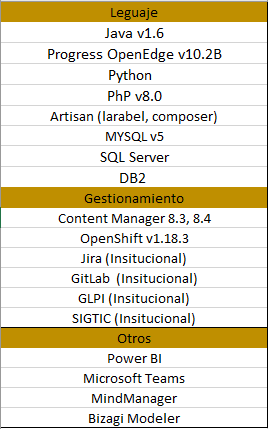
Esta elección dependerá tanto de las decisiones de diseño tomadas como del entorno en el que el software deba funcionar.

Al programar hay que hacer uso de las siguientes pautas:

* Evitar bloques de control no estructurados.
* Identificar correctamente las variables y su alcance.
* Elegir algoritmos y estructuras de datos adecuadas para el problema.
* Mantener la lógica de la aplicación lo más sencilla posible.
* Documentar y comentar adecuadamente el código de los sistemas.
* Facilitar la interpretación visual del código utilizando reglas de formato de código previamente consensuadas en el equipo de desarrollo.
* Los nombres de las funciones, de tablas o campos en estas, métodos, clases deben ser nombres entendibles, que den la idea de para qué sirve cada elemento, lo que facilitará el mantenimiento.
* El código debe estar correctamente identado, lo que simplificará la lectura de este para el personal de mantenimiento.
* En la parte inicial del archivo se deberá agregar el nombre del desarrollador que ha editado el documento, fecha y acción realizada.
* Se debe entregar código comentado, que indique puntualmente las acciones del código. Cuidando de ser concreto y claro.
* Evitar duplicidad en código, para lo cual se deben de utilizar funciones, librerías, etc., que ayuden en el proceso de mantenimiento.
* Se puede utilizar una misma interfaz gráfica para el proceso de alta, edición o borrado de registros.
* El borrado en todo sistema deberá de ser únicamente a nivel lógico, por lo que, en caso de requerir borrar un registro, únicamente se deberá de cambiar el estatus del registro de Activo (1) a inactivo (0). El objetivo es que todo sistema sea auditable, respetando la secuencia de los identificadores auto incrementales.
* Se evitarán condicionales o bucles anidados. Se buscará en caso de ser necesario, hacerlo en funciones independientes, las cuales reciban y retornen un valor que permita continuar el ciclo.
* En principio, no se permite el uso de plugins o herramientas de internet desarrolladas por terceros. Requieren autorización previa de la DCIDT de la UTIC.
* Privilegiar en todo momento el uso de APIs, lo anterior debido a que se debe ir orientando los desarrollos a un consumo abierto entre las distintas aplicaciones de la Secretaría, ciudadanía o su consumo mediante dispositivos móviles.
* A la par del desarrollo de la aplicación, en caso de aplicar, se considerará el testeo de forma automatizada.
* No mostrar mensajes de error con información sensible del sistema, identificadores de sesión, o información del usuario.
* En las BD las monedas se manejarán como numérico con 4 decimales y truncado.
* Todo desarrollo debe implementarse ya preparado para operar en IPV6 e IPV4 de manera indistinta
* Toda aplicación deberá contar con bitácora de movimientos, donde se registrarán las entradas y salidas de usuarios, sus ips, fechas y horas, así como de cada actividad realizada al interior de la aplicación. Con esto se podrá identificar qué usuario dio de alta, modificó, desactivó los registros. Idealmente, se deberá de concentrar esta información en formato json, donde se almacene también el cambio realizado, de forma que se pueda tener un histórico de lo existente antes de la modificación realizada.

También hay que tener en cuenta la adquisición de recursos necesarios para que el software funcione, además de desarrollar casos de prueba para comprobar el funcionamiento de este según se vaya programando.

Actualmente se utilizan las siguientes herramientas:



Los requerimientos que incluyan desarrollo implementado y nuevo; deben generarse en versiones actualizadas y consistentes, incluyendo el uso de diagramas de flujo para garantizar que el proceso del sistema esté organizado correctamente, como también la creación de manuales técnicos.

Todos los sistemas deberán ser 100% configurables, a través de un módulo, ningún parámetro deberá estar embebido en el código del sistema incluidos sin ser limitativos: IP’s, nombres de host, puertos, nombre de base de datos, etc.

Todo sistema en mantenimiento deberá contar con ambiente de Pruebas (QA), independiente de Producción. Se recomienda tener también ambiente de desarrollo.

Todo nuevo desarrollo deberá ejecutarse bajo una Arquitectura Orientada a Servicios (SOA) y privilegiando las tecnologías abiertas.

* El back-end deberá desarrollarse considerando como primera opción, el lenguaje Python
* El Font-end deberá desarrollarse considerando como primera opción VUE o PHP

Los anteriores en su versión más actual y estable disponible.

Siempre se debe mantener el registro de los proyectos mediante la herramienta de gestión “JIRA” en la UTIC.

Todo sistema o aplicativo debe contar con:

* Diagramas de base de datos
* Diagrama de aplicación
* Diagrama de Proceso
* Diagrama de bloques
* Requerimientos
* Registro de pruebas

Todo esto debe estar registrado en la herramienta de la UTIC para este fin, que en la actualidad es GitLab.

Las versiones del aplicativo o sistema se deberán guardar en 2 ramas

* Master o Principal (que será la rama Productiva)
* QA (que será la rama de Pruebas)

Cuando exista un mantenimiento se creará una rama nueva para el desarrollo, una vez concluido esta deberá cargarse a QA para poder probarse, y una vez aprobado se hará la unión (merge) con la rama Principal.

# **Pruebas**

Para cualquier proyecto dado, los objetivos de prueba incluyen:

**Prueba funcional:**

Prueba de Aceptación (UAT); se centra normalmente en el comportamiento y las capacidades de todo el sistema.

Los objetivos de la prueba de aceptación incluyen:

* Establecer confianza en la calidad del sistema en su conjunto.
* Validar que el sistema está completo y que funcionará como se espera.
* Verificar que los comportamientos funcionales y no funcionales del sistema sean los especificados.

Se pueden utilizar como base de prueba para:

* Requisitos de usuario
* Normativas, contratos legales y estándares.
* Casos de uso.
* Requisitos de sistema.
* Documentación del sistema o del usuario.
* Informes de análisis de riesgo.

**Prueba no funcional:**

* Pruebas de Estrés; determinan la solidez del software al evaluar el funcionamiento normal en condiciones de carga extremadamente pesada, garantizando así que el software no se bloquea en situaciones críticas.
* Pruebas de Calidad; procesos cuya ejecución permiten conocer la calidad de este, así como los posibles fallos que puedan existir a corto, medio o largo plazo.

Incluyen:

* + Detección y corrección de defectos de forma más eficiente y antes de la ejecución de la prueba.
  + Prevenir defectos en el diseño o la codificación descubriendo inconsistencias, ambigüedades, contradicciones, omisiones, inexactitudes y redundancias en los requisitos.
  + Defectos en los requisitos (por ejemplo, inconsistencias, ambigüedades, contradicciones, omisiones, inexactitudes y redundancias).
  + Defectos de diseño (por ejemplo, algoritmos o estructuras de base de datos ineficientes, alto acoplamiento, baja cohesión).
  + Defectos de codificación (por ejemplo, variables con valores no definidos, variables que nunca se utilizan, código inalcanzable, código duplicado, hard code).
  + Deficiencias o inexactitudes en la trazabilidad o cobertura de la base de prueba (por ejemplo, la falta de pruebas para un criterio de aceptación).
  + Especificaciones de interfaz incorrectas
  + Reducir el coste y el tiempo de la prueba
* Pruebas de Seguridad; verifican si el sistema está protegido contra ataques repentinos o deliberados de fuentes internas y externas.

Incluyen:

* + Vulnerabilidades de seguridad (por ejemplo, susceptibilidad a desbordamientos de la memoria).
  + Restringir el acceso (de personas de la organización y de las que no lo son) a los programas y archivos.
  + Asegurar que los involucrados puedan trabajar pero que no puedan modificar los programas ni los archivos que no correspondan (sin una supervisión minuciosa).
  + Asegurar que se utilicen los datos, archivos y programas correctos en/y/por el procedimiento elegido.
  + Que la información transmitida sea la misma que reciba el destinatario al cual se ha enviado y que no le llegue a otro.
  + Verificar a cada uno de los involucrados con claves distintas y permisos bien establecidos, en todos y cada uno de los sistemas o aplicaciones empleadas.
* Pruebas de Funcionalidad; incluye pruebas que evalúan las funciones que el sistema debe realizar. Los requisitos funcionales pueden estar descritos en especificaciones de requisitos de negocio, épicas, historias de usuario, casos de uso, o especificaciones funcionales, o pueden estar sin documentar. Las funciones describen "qué" debe hacer el sistema.
* Las Pruebas Unitarias deben poder cubrir casi la totalidad del código de nuestra aplicación. Una prueba unitaria será tan buena como su cobertura de código. La cobertura de código marca la cantidad de código de la aplicación que está sometido a una prueba. Por tanto, si la cobertura es baja, significará que gran parte de nuestro código está sin probar.
* La ejecución de una prueba no puede afectar la ejecución de otra. Después de la ejecución de una prueba el entorno debería quedar igual que estaba antes de realizar la prueba.
* Las diferentes relaciones que puedan existir entre módulos deben ser simulada para evitar dependencias entre módulos.
* Las pruebas unitarias tienen que poder repetirse tantas veces como uno quiera. Por este motivo, la rapidez de las pruebas tiene un factor clave. Si pasar las pruebas es un proceso lento no se pasarán de forma habitual, por lo que se perderán los beneficios que éstas nos ofrecen.
* Las pruebas unitarias se tienen que poder ejecutar sin necesidad de intervención manual. Esta característica posibilita que podamos automatizar su ejecución.

También hay que tener en cuenta la entrega de la Matriz de Pruebas que detallará cada una de las características y funcionalidades de cada uno de los módulos de los sistemas.

El objetivo de las matrices de pruebas es tener una lista de todas las funciones de cada módulo entregado y que el usuario aprobará cada uno de los elementos de la lista. Además, las matrices de pruebas servirán como evidencia de que fueron realizadas las pruebas, incluyendo las pruebas unitarias.

# **Glosario**

|  |
| --- |
| **Aplicación:** Es un sistema diseñado como una herramienta que permite simplificar tareas complejas |
| **Aplicativo:** Es un sistema diseñado como una herramienta que permite simplificar tareas compejas, utilizando dispositivos móviles |
| **Bug:** Error técnico |
| **Daily Scrum:** El Daily Scrum (o Scrum diario) es uno de los 5 eventos de Scrum con un bloque de tiempo de 15 minutos para que el Equipo de Desarrollo se sincronice. Para ello se planea el trabajo de las siguientes 24 horas. Se atiende cuando todos los miembros del equipo contestan las siguientes preguntas:  · ¿Qué hice ayer para ayudar a lograr el objetivo?· ¿Veo algún impedimento que evite que el Equipo de Desarrollo o yo logremos el objetivo? |
| **Defecto:** Funcionalidad solicitada no cumplida |
| **Equipo de Desarrollo:** El Equipo de Desarrollo (o Development Team) es uno de los tres roles en Scrum y se compone de todas las personas que se encargan de construir el Incremento de Producto en cada Sprint. |
| **Hallazgo:** Nuevo requisito |
| **Impedimento:** Situación externa del proyecto que no permite avanzar |
| **Plataforma:** Es un conjunto de sistemas que al interactuar entre sí, logran un objetivo general. |
| **Product Backlog:** La Lista de Producto es una lista ordenada de todo lo que se conoce que es necesario que un producto o servicio cumpla. |
| **Product Owner:** Es uno de los 3 roles del marco de trabajo. Su objetivo es asegurar que se entregue el producto «correcto» |
| **Retrospectiva:** Una retrospectiva es una reunión en la cual reflexionamos sobre la manera en la que trabajamos en un periodo de tiempo. Es una oportunidad para capitalizar aprendizajes y definir acciones de mejora a futuro. |
| **Rol:** Es una función desempeñada por un recurso en un lugar o situación específica, independientemente del rango o perfil curricular. |
| **Scrum Master:** El Scrum Máster es uno de los 3 roles del marco de trabajo Scrum. Es el responsable de la eficiencia del equipo Scrum y de que el equipo mejore de manera continua hacía su mejor versión. |
| **Sistema:** Conjunto de elementos inter relacionados entre sí, que permiten almacenar, procesar, guardar y exponer información |
| **Sprint Backlog:** El Sprint Backlog es la suma de todos los elementos del Product Backlog elegidos para el Sprint, más un plan de cómo crear el Incremento de Producto. |
| **Sprint Planning:** (planificación del Sprint) es uno de los cinco eventos de Scrum y es el primero que haremos al comenzar cada Sprint. En esta reunión vamos a planificar QUÉ es lo que vamos a hacer durante el Sprint y CÓMO lo vamos a hacer. |
| **Sprint Review:** (Revisión del Sprint) se realiza al final del Sprint. se revisa el Incremento de Producto, es decir, lo que se realizó durante el Sprint y se analizan los cambios que tuvo el Product Backlog. |
| **Sprint:** Ciclo de ejecución para construir un incremento |
| **Tablero Kanban:** El tablero Kanban es una herramienta compuesta por columnas para representar el estado de una tarea y filas que representan diferentes tipos de actividades. Cada tablero de Kanban tiene al menos tres columnas con estados base:·  «To Do» / Por hacer (punto de entrada de una tarea)  ·  «W.I.P» / Trabajo en proceso  ·  «Done» (Terminado)\* |