

**PLAN DE GESTIÓN DE LA CONFIGURACIÓN DEL SOFTWARE**

1. **Introducción**
2. **Situación de la empresa**

En BigTech contamos con un equipo de profesionales altamente capacitados y apasionados por la tecnología, que se dedican a brindar soluciones de software innovadoras y efectivas para nuestros clientes.

Nos esforzamos por mantener altos estándares de calidad en todos los proyectos en los que trabajamos, y buscamos ofrecer soluciones personalizadas y efectivas para satisfacer las necesidades de nuestros clientes. Además, nos enfocamos en la investigación y el desarrollo de nuevas tecnologías y en estar siempre al día con las últimas tendencias en el mercado.

Nos tomamos en serio nuestra responsabilidad social y ambiental, y nos esforzamos por tener un impacto positivo en nuestras comunidades y en el medio ambiente.

Pero como cualquier empresa, estamos experimentando una serie de problemas con nuestro sistema de control de versiones. Hemos notado que hay una falta de protocolos claros establecidos para el uso del control de versiones, lo que ha llevado a algunos miembros del equipo a hacer cambios que no se ajustan a los estándares de la empresa o que no son coherentes con los objetivos del proyecto.

Además, hemos notado dificultades para compartir y colaborar en el código fuente y otros artefactos del proyecto debido a que los miembros del equipo no están utilizando el mismo control de versiones o no están siguiendo los mismos protocolos. También hemos experimentado problemas de integración debido a que se están utilizando múltiples herramientas de control de versiones en diferentes partes de la empresa.

La pérdida de datos ha sido otro problema para nosotros en BigTech Consulting. Hemos descubierto que si no se realiza correctamente el seguimiento de las versiones, se puede perder información importante. Por ejemplo, se han encontrado casos en los que se ha sobrescrito una versión anterior con una nueva versión sin guardar la anterior, lo que ha resultado en la pérdida de toda la información contenida en la versión anterior.

También hemos estado experimentando conflictos de fusión cuando dos o más personas trabajan en el mismo archivo al mismo tiempo y realizan cambios diferentes. Si no se resuelven estos conflictos adecuadamente, se pueden perder cambios importantes o incluso corromper el archivo.

Finalmente, hemos notado que hay una falta de control de acceso, lo que ha llevado a que cualquier persona con acceso al repositorio pueda realizar cambios. Esto ha provocado errores y la eliminación de archivos importantes. Estamos trabajando para establecer protocolos claros para el uso del control de versiones y mejorar la colaboración y el trabajo en equipo en el proyecto.

A continuación hablaremos sobre los problemas que presentamos y sus consecuencias para nuestra empresa:

1. Falta de protocolos claros: Si no hay protocolos claros establecidos para el uso del control de versiones, los miembros del equipo pueden hacer cambios que no se ajusten a los estándares de la empresa o que no sean coherentes con los objetivos del proyecto.
2. Dificultades para compartir y colaborar: Si los miembros del equipo no están utilizando el mismo control de versiones o no están siguiendo los mismos protocolos, puede haber dificultades para compartir y colaborar en el código fuente y otros artefactos del proyecto.
3. Problemas de integración: Si se están utilizando múltiples herramientas de control de versiones en diferentes partes de la empresa, puede ser difícil integrar y gestionar el código fuente y otros artefactos del proyecto.
4. Pérdida de datos: Si no se realiza correctamente el seguimiento de las versiones, se puede perder información importante. Por ejemplo, si se sobrescribe una versión anterior con una nueva versión sin guardar la anterior, se pierde toda la información contenida en la versión anterior.
5. Conflictos de fusión: Si dos o más personas trabajan en el mismo archivo al mismo tiempo y realizan cambios diferentes, puede haber conflictos al fusionar los cambios. Si no se resuelven estos conflictos adecuadamente, se pueden perder cambios importantes o incluso corromper el archivo.
6. Falta de control de acceso: Si no se establecen permisos de acceso adecuados, cualquier persona con acceso al repositorio puede realizar cambios, lo que puede provocar errores o incluso la eliminación de archivos importantes.
7. **Propósito del plan**

El propósito del Plan es establecer procesos y procedimientos para la gestión efectiva de la configuración del software en la empresa, lo que garantiza la calidad y control en los cambios realizados en el software. Además, este plan busca asegurar que los cambios se implementen de manera controlada y efectiva, lo que garantiza que el software funcione correctamente y cumpla con los requisitos del cliente y del negocio. En general, el objetivo es establecer un enfoque estructurado y coherente para la gestión de la configuración del software, lo que mejora la eficiencia y efectividad del proceso de desarrollo de software en la empresa.

1. **Benchmarking de herramientas** Se realizará un análisis comparativo de las herramientas disponibles en el mercado para la gestión de la configuración del software. Se evaluarán las herramientas según su capacidad para gestionar la configuración del software, su facilidad de uso, su compatibilidad con otros sistemas y herramientas utilizadas en la empresa y su costo. Se seleccionará la herramienta más adecuada para la empresa y se definirán los procedimientos para su implementación y uso.

* **MERCURIAL**

En el desarrollo de software, el control de versiones es esencial para colaborar en proyectos y asegurarse de que no se pierda información o se creen conflictos en el código fuente. Una de las herramientas más populares para el control de versiones es Mercurial.

***Características:***

* Sistema de control de versiones distribuido.
* Facilidad de uso, aprendizaje y uso sencillo.
* Velocidad y rendimiento en el manejo de grandes repositorios.
* Flexibilidad y personalización a través de una gran cantidad de herramientas y plugins.
* Compatibilidad multiplataforma (Windows, Linux, Mac OS X).
* Permite una gestión eficiente de ramas y fusión de ramas.
* Ofrece soporte para control de acceso y permisos.
* Cuenta con una arquitectura modular y escalable.
* Ofrece una integración sencilla con otras herramientas de software.

***Uso en el desarrollo de software:***

Mercurial es ampliamente utilizado en el desarrollo de software debido a su facilidad de uso, velocidad y flexibilidad. Los desarrolladores pueden trabajar en equipo y colaborar en un proyecto sin temor a perder información o a crear conflictos en el código fuente. Mercurial también es compatible con múltiples plataformas, incluyendo Windows, Linux y Mac OS X.

***Ventajas:***

* Fácil de aprender y de utilizar.
* Velocidad y rendimiento en el manejo de grandes repositorios.
* Sistema de control de versiones distribuido, lo que permite que cada desarrollador tenga su propia copia del repositorio.
* Permite la gestión eficiente de ramas y la fusión de ramas.
* Compatibilidad multiplataforma.
* Amplia gama de herramientas y plugins que pueden ser utilizados para personalizar y mejorar su funcionalidad.
* Ofrece soporte para control de acceso y permisos.
* Arquitectura modular y escalable.
* Integración sencilla con otras herramientas de software.

**Desventajas:**

* Puede ser menos conocido que otras herramientas de control de versiones como Git.
* Curva de aprendizaje para los desarrolladores que no están familiarizados con el uso de sistemas de control de versiones distribuidos.
* Puede ser difícil de manejar para proyectos muy grandes o complejos.
* Requiere un servidor centralizado para compartir cambios entre desarrolladores, lo que puede aumentar la complejidad en entornos de equipo muy grandes.}
* **APACHE SUBVERSION**

Apache Subversion, también conocido como SVN, es una herramienta de control de versiones de código abierto, desarrollada por la Apache Software Foundation. SVN se ha utilizado en proyectos de desarrollo de software de diferentes tamaños, desde pequeños hasta grandes corporaciones.

SVN fue creado para ser una alternativa a los sistemas de control de versiones centralizados como CVS. En lugar de tener un servidor centralizado que controle la versión del código, SVN utiliza una arquitectura de servidor-cliente, permitiendo a cada desarrollador tener una copia local del repositorio. Los cambios se sincronizan con el servidor central cuando es necesario.

***Características:***

* Sistema de control de versiones centralizado, que utiliza una arquitectura cliente-servidor.
* Fácil de usar, especialmente para usuarios que han utilizado sistemas de control de versiones centralizados.
* Soporte para control de acceso y permisos, permitiendo a los desarrolladores trabajar en diferentes áreas del proyecto.
* Registro completo de cambios y revisiones.
* Amplia documentación y una comunidad de usuarios activa.

***Uso en el desarrollo de software:***

Apache Subversion es ampliamente utilizado en el desarrollo de software para mantener un control efectivo sobre el código fuente y las versiones del proyecto. Los desarrolladores utilizan SVN para colaborar en el desarrollo de un proyecto y mantener un registro completo de los cambios realizados en el código fuente. SVN permite a los desarrolladores trabajar en diferentes áreas del proyecto de forma aislada, sin afectar el trabajo de los demás desarrolladores. Además, SVN ayuda a controlar el acceso y los permisos de los usuarios, lo que garantiza que solo las personas autorizadas puedan realizar cambios en el código fuente. También permite la creación de ramas, lo que permite que se trabajen diferentes versiones del mismo proyecto sin afectar a la versión principal.

***Ventajas:***

* Fácil de aprender y utilizar, especialmente para usuarios que tienen experiencia con sistemas de control de versiones centralizados.
* Amplia documentación y una comunidad de usuarios activa.
* Ofrece una excelente integración con otros sistemas de software.
* Soporte para control de acceso y permisos.
* Registro completo de cambios y revisiones.

**Desventajas:**

* No es tan rápido como otros sistemas de control de versiones distribuidos, como Git.
* No es tan flexible para la creación de ramas y fusiones como los sistemas de control de versiones distribuidos.
* Requiere un servidor centralizado, lo que puede aumentar la complejidad en entornos de equipo muy grandes.
* **BAAZAR**

Bazaar es una herramienta de software de control de versiones distribuido (DVCS, por sus siglas en inglés) que permite a los desarrolladores colaborar en proyectos de software de forma remota. Fue desarrollado por Canonical Ltd. y lanzado en el año 2005 como una alternativa al popular software de control de versiones Git.

***Características:***

* La gestión de múltiples ramas
* La capacidad de trabajar offline
* La integración con otros sistemas de control de versiones
* Una interfaz de línea de comandos fácil de usar

***Uso en el desarrollo de software:***

Bazaar se utiliza en el desarrollo de software para mantener un registro de todos los cambios realizados en el código fuente y para facilitar la colaboración entre desarrolladores. Permite a los equipos trabajar de forma remota en el mismo código fuente, realizar cambios y enviarlos de vuelta al repositorio central para su revisión y aprobación.

***Ventajas***

* Facilidad de uso
* Su capacidad para trabajar con proyectos grandes y complejos
* Su flexibilidad para adaptarse a diferentes flujos de trabajo.
* Es fácil de instalar y configurar
* Tiene una gran comunidad de usuarios y desarrolladores que ofrecen soporte y recursos

***Desventajas***

* Velocidad en operaciones grandes
* Su pobre rendimiento en proyectos muy grandes
* Su dependencia de una red para operaciones básicas
* Su falta de popularidad y recursos en comparación con otras herramientas de control de versiones como Git o Mercurial.
* **GIT**

Git es un sistema de control de versiones distribuido que se utiliza para rastrear cambios en archivos y coordinar el trabajo entre múltiples personas en un proyecto de desarrollo de software. Fue creado por Linus Torvalds en 2005 y se ha convertido en una de las herramientas más populares en la industria del desarrollo de software.

***Características***

* La gestión de múltiples ramas
* La capacidad de trabajar offline
* La integración con otros sistemas de control de versiones
* Una interfaz de línea de comandos fácil de usar

***Uso en el desarrollo de software***

Git se utiliza en el desarrollo de software para controlar versiones, facilitar la colaboración en equipo, manejar ramas de desarrollo, deshacer cambios, gestionar conflictos y trabajar con repositorios remotos. Estas características hacen que Git sea una herramienta esencial para el desarrollo eficiente y organizado de software.

***Ventajas***

* Control de versiones
* Colaboración eficiente
* Ramificación y fusión flexibles.
* Deshacer cambios y revertir a versiones anteriores
* Velocidad y rendimiento

***Desventajas***

* Curva de aprendizaje inicial
* Comandos de línea de comandos
* Resolución de conflictos
* Tamaño del repositorio

**Comparación de Herramientas**

| **CARACTERÍSTICAS** | **MERCURIAL** | **APACHE SUBVERSION** | **BAAZAR** | **GIT** |
| --- | --- | --- | --- | --- |
| Manejo centralizado del repositorio |  | **X** |  | **X** |
| Control detallado de acceso y permisos | **X** | **X** | **X** | **X** |
| Multiplataforma | **X** | **X** | **X** | **X** |
| Historial de archivos |  | **X** |  | **X** |
| Gratuito | **X** | **X** | **X** | **X** |
| Mantenimiento de repositorios | **X** | **X** | **X** | **X** |
| Soporte de auto-reparación de repositorios |  | **X** |  | **X** |
| Gestión y fusión de rama |  |  |  | **X** |

Al realizar el análisis de Benchmarking de herramientas se ha decidido utilizar Git como la herramienta de control de versiones preferida en lugar de opciones alternativas como Mercurial, Bazaar y Apache Subversion por las siguientes razones:

* **Amplia adopción y comunidad:** Git es ampliamente utilizado y cuenta con una gran comunidad de desarrolladores que respaldan su desarrollo y proporcionan soporte continuo. Esto se traduce en una amplia gama de recursos disponibles, documentación y soluciones a problemas comunes. La comunidad activa también fomenta la creación y mejora de herramientas y complementos que mejoran la experiencia de uso de Git.
* **Rendimiento y escalabilidad**: Git está diseñado para ser rápido y eficiente, incluso en repositorios grandes con una historia extensa. Su enfoque distribuido y su capacidad para almacenar cambios de manera incremental hacen que las operaciones, como la navegación en la historia del repositorio y la búsqueda de cambios específicos, sean rápidas y eficientes. Esto es especialmente beneficioso en entornos de desarrollo ágiles y colaborativos.
* **Flexibilidad y gestión avanzada de ramas:** Git ofrece una gestión avanzada de ramas, lo que facilita la creación y fusión de ramas para trabajar en paralelo en diferentes características, experimentar con nuevas ideas y colaborar eficientemente en equipos. Esto proporciona una mayor flexibilidad y permite adaptarse a diferentes flujos de trabajo y estrategias de desarrollo.
* **Amplio ecosistema de herramientas y servicios:** Git cuenta con un amplio ecosistema de herramientas y servicios que facilitan la colaboración, la revisión de código y la integración con otros sistemas y servicios. Plataformas populares como GitHub, GitLab y Bitbucket ofrecen una interfaz gráfica amigable, características adicionales como solicitudes de extracción y sistemas de control de acceso, y promueven una colaboración más efectiva y un seguimiento mejorado de los cambios en el proyecto.
* **Capacidad de auto-reparación del repositorio:** Git proporciona mecanismos y herramientas que pueden ayudar a solucionar problemas y restaurar un repositorio en caso de daños o errores. Si bien no tiene una función automática de auto-reparación, la capacidad de verificar la integridad del repositorio, revertir a versiones anteriores o clonar nuevamente el repositorio desde una fuente remota contribuye a la recuperación eficiente en situaciones problemáticas.

Basándonos en estas consideraciones, se ha seleccionado Git como la herramienta de control de versiones principal para garantizar un control eficiente y efectivo de la configuración del software, fomentar la colaboración en equipo y facilitar el desarrollo y mantenimiento de proyectos de manera ágil y escalable.

1. **Identificación**
   1. **Clasificación de Ítem**

La clasificación de ítems es un proceso importante en la gestión de configuración y en la organización de los elementos y documentos dentro de un proyecto. La clasificación se utiliza para categorizar y agrupar los ítems de acuerdo con características similares, lo que facilita su identificación, seguimiento y control a lo largo del ciclo de vida del proyecto.

Existen diferentes enfoques para clasificar los ítems, y la elección de la clasificación adecuada dependerá del tipo de proyecto y de los requerimientos específicos. A continuación, se presentan tres categorías comunes de clasificación de ítems:

* Ítems por evolución: Estos son los ítems que están sujetos a cambios y evolución a lo largo del tiempo. Incluyen documentos y archivos ejecutables que se modifican o actualizan a medida que avanza el proyecto.
* Ítems fuente: Estos son los ítems relacionados con el código fuente y los archivos necesarios para compilar una aplicación o software. Incluyen documentos de diseño y construcción de la arquitectura, así como los archivos de código fuente. Estos ítems suelen estar asociados con el desarrollo y la programación del software.
* Ítems de soporte: Estos son los ítems relacionados con los sistemas operativos, el software base y los requisitos del proyecto. Incluyen documentos de especificación de requisitos, análisis de requisitos, especificación de la base de datos y otros documentos que respaldan el desarrollo y la implementación del proyecto.
  + 1. **Clasificación de los Ítems de la Configuración (CI).**

| **Tipo**  **(E= Evolución, F=Fuente, S=Soporte)** | **Nombre del Ítem (CI)** | **Nomenclatura** | **Extensión** | **Proyecto** |
| --- | --- | --- | --- | --- |
| E | ***Plan de Proyecto (PROJECT CHARTER)*** | SVCE-PC | .DOCX | SVCE |
| E | *Cronograma del Proyecto* | SVCE-CP | .XLS | SVCE |
| E | *Documento de Especificación de Requisitos 1* | SVCE-DER-1 | .DOCX | SVCE |
| E | *Documento de Especificación de Requisitos 2* | SVCE-DER-2 | .DOCX | SVCE |
| E | *Documento de Especificación de Requisitos 3* | SVCE-DER-3 | .DOCX | SVCE |
| E | *Documento de Especificación de Requisitos 4* | SVCE-DER-4 | .DOCX | SVCE |
| E | *Documento de Especificación de Requisitos 5* | SVCE-DER-5 | .DOCX | SVCE |
| E | *Documento de Especificación de Requisitos 6* | SVCE-DER-6 | .DOCX | SVCE |
| E | *Documento de Especificación de Requisitos 7* | SVCE-DER-7 | .DOCX | SVCE |
| E | *Documento de Especificación de Requisitos 8* | SVCE-DER-8 | .DOCX | SVCE |
| E | *Documento de Especificación de Requisitos 9* | SVCE-DER-9 | .DOCX | SVCE |
| E | *Documento de Análisis de Especificaciones de Requisitos* | SVCE-DAER | .DOCX | SVCE |
| E | *Documento de Construcción y Análisis de Arquitectura* | SVCE-DCAA | .DOCX | SVCE |
| E | *Documento del Diseño del Software 1* | SVCE-DDS-1 | .DOCX | SVCE |
| E | *Documento de Especificación de la BD* | SVCE-DBD | .DOCX | SVCE |
| F | *Documento de Verificación de hito 1* | SVCE-DVH-1 | .DOCX | SVCE |
| E | *Documento del Diseño del Software 2* | SVCE-DDS-2 | .DOCX | SVCE |
| E | *Documento del Diseño del Software 3* | SVCE-DDS-3 | .DOCX | SVCE |
| F | *Documento de Verificación de Hito 2* | SVCE-DVH-2 | .DOCX | SVCE |
| F | *Documento de Verificación de Hito 3* | SVCE-DVH-3 | .DOCX | SVCE |
| E | *Documento de Revisión y Validación de Hitos* | SVCE-DRVH | .DOCX | SVCE |
| F | *Acta de Cierre del Proyecto* | SVCE-ACP | .DOCX | SVCE |

* 1. **Definición de Nomenclaturas**

La Nomenclatura de los Ítems de la Configuración (CI) es un conjunto de reglas y convenciones utilizadas para nombrar y etiquetar los elementos de la configuración de un sistema. Estos elementos pueden ser cualquier cosa que forme parte de la configuración, como hardware, software, documentación, procedimientos, scripts, entre otros.

"La nomenclatura de los ítems de configuración debe ser concisa, descriptiva y fácilmente comprensible por cualquier miembro del equipo de desarrollo" (Pressman & Maxim, 2015, p. 420).

La CI puede variar según la organización o el proyecto específico, pero generalmente sigue una estructura jerárquica que refleja la estructura del sistema.

A continuación se evidenciará la nomenclatura establecida por la empresa para los ítems en evolución:



Ejemplo 1:

Proyecto: Sistema de Ventas de Componentes Electrónicos

Documento: Plan del Proyecto

Nomenclatura: SVCE-PP.docx

Observaciones:

* La extensión será de acuerdo al tipo de archivo.
* Si existen documentos que coinciden con la nomenclatura se agregará la siguiente letra de la palabra.

Documento: Plan del Proyecto

Nomenclatura: SVCE-PP.docx

Documento: Plan de Prueba

Nomenclatura: SVCE-PPR.docx

* Si hay archivos los cuales tienen relación en la temática y necesitan numeración se le agregará “-” + “numeración .

Documento: Especificación de Requerimiento 1

Nomenclatura: SVCE-ER-1.docx

Documento: Especificación de Requerimiento 2

Nomenclatura: SVCE-ER-2.docx

* **Número de versión:**

La primera parte del número de versión se refiere a la versión principal del software o documento. Un cambio en la versión principal generalmente significa una revisión importante o un cambio fundamental en el documento.

Ejemplo 2:

Documento: Plan del Proyecto

Nomenclatura: SVCE-PP.docx

Versión: 1.0

Después de un cambio significativo:

Documento: Plan del Proyecto

Nomenclatura: SVCE-PP.docx

Versión: 2.0

* **Número de revisión:**

La segunda parte del número de versión indica una revisión menor o una actualización del documento. Es importante porque ayuda a los desarrolladores y usuarios a identificar rápidamente qué versión de un componente o parte del software están utilizando y cuántas revisiones o cambios se han realizado en ese componente o parte en particular. También puede ayudar a los desarrolladores a rastrear el historial de cambios y revisiones de un componente o parte del software, lo que puede ser útil para fines de mantenimiento y resolución de problemas.

Ejemplo 3:

Documento: Plan del Proyecto

Nomenclatura: SVCE-PP.docx

Versión: 1.0 (revisión 0)

Versión: 1.1 (revisión 1)

Versión: 1.1 (revisión 2)

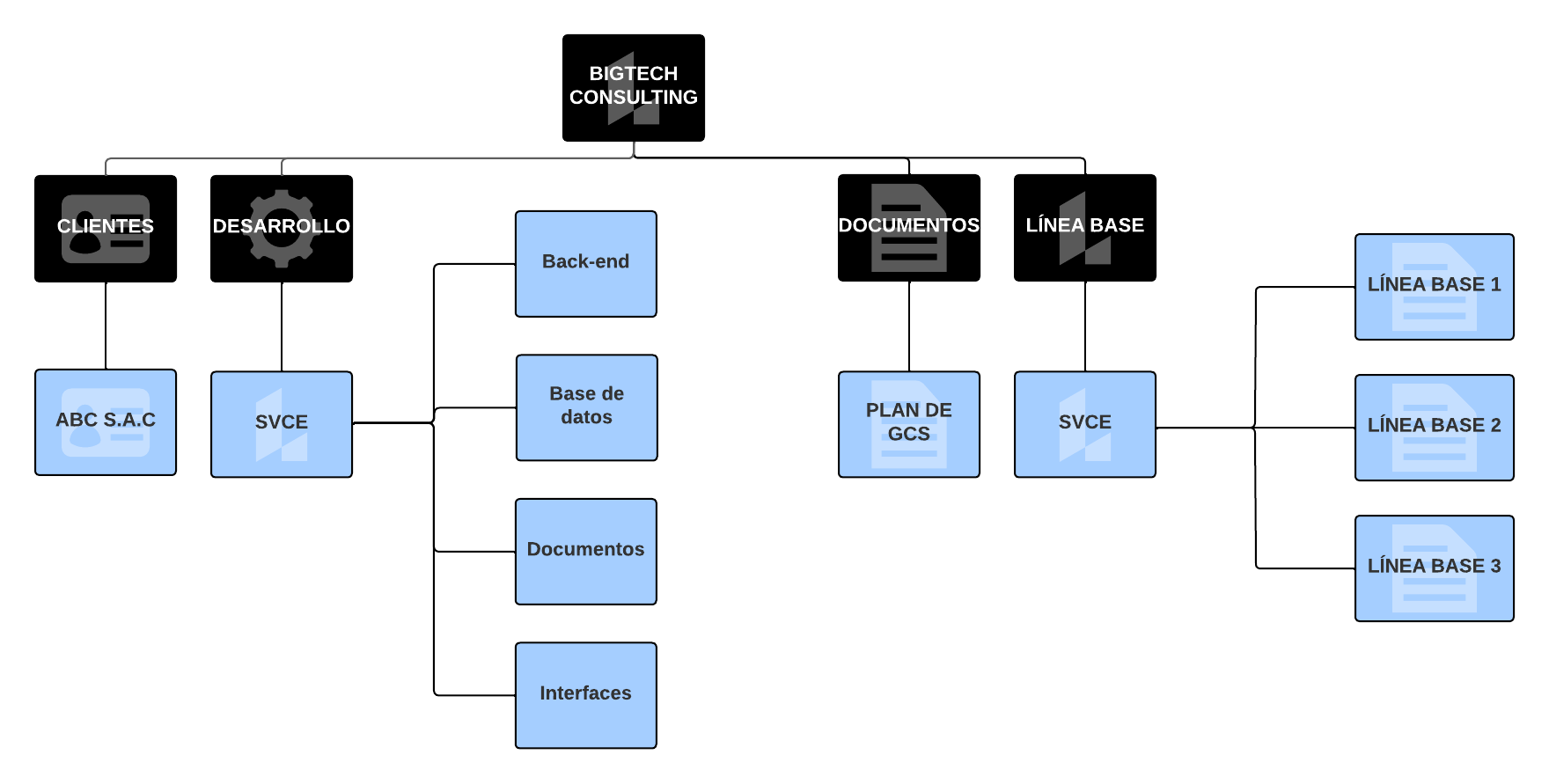
Entonces para saber en qué revisión se encuentra basta con observar el número después del punto.

**En síntesis:**



La gestión de la configuración del software mantendrá un registro de estos documentos y sus respectivas versiones y revisiones a medida que el proyecto avance. Además, se seguirán las reglas de nomenclatura establecidas para garantizar la coherencia y comprensibilidad en la identificación de los ítems de configuración en el proyecto.

* 1. **Diseño de la estructura del repositorio (Descripción de las librerías)** En este apartado se puede encontrar graficado la estructura del repositorio de Bigtech Consulting, correctamente organizado. Esta estructura organizada es necesaria y fundamental ya que permite que el mantenimiento y comunicación en el desarrollo de software, de esta forma, se puede desarrollar y tener una idea clara de todos los proyectos que han sido desarrollados o están en proceso, por parte de la empresa Bigtech Consulting

****

* 1. **Definición de Líneas Base**

Definición de Líneas Bases

Hito 1:

Para facilitar la colaboración y el control del proyecto, se creará un repositorio centralizado que servirá como fuente única de verdad para los documentos y el código fuente. A medida que avanzamos en el Hito 1, cada miembro del equipo asumirá la responsabilidad de especificar requisitos específicos. Esto implicará la elaboración de especificaciones claras y detalladas para cada uno de ellos, como el catálogo de productos correctamente clasificados, la gestión de la base de datos, la búsqueda de clientes, el filtrado de productos, la selección y compra de artículos, la edición del listado de compras, los medios de pago, el reporte de compra y el soporte al cliente.

Además, el equipo de desarrollo contará con la contribución del DBA para diseñar una base de datos robusta y eficiente que respalde las funcionalidades del software. Asimismo, los diseñadores UX y el Arquitecto de Software colaborarán para crear un diseño detallado del software, teniendo en cuenta los requisitos establecidos. Una vez que todas estas actividades estén completas, se realizará un análisis y verificación exhaustivos del Hito 1 para asegurarnos de que se hayan alcanzado los objetivos y los entregables correspondientes. Este análisis y verificación nos permitirá garantizar la calidad y el cumplimiento de las metas establecidas.

Hito 2:

En este hito se da comienzo a la fase de programación, además la metodología que se ha adoptado es una ágil, la cual es la metodología Scrum , la cual nos proporcionará facilidades a la hora de trabajar en el proyecto.  
Respecto a los entregables tenemos: La implementación de los requisitos

* Requisito 1: Catálogo de productos correctamente clasificados.
* Requisito 2: Gestión de base de datos de los productos
* Requisito 3: Búsqueda de cliente para generar un reporte.

donde el Arquitecto de Software será el encargado de implementar los requisitos especificados en el Hito 1; luego tenemos el Diseño de Software 2 donde los diseñadores UX y el Arquitecto de Software colaborarán para crear el diseño detallado del software, teniendo en cuenta los requisitos establecidos

* Requisito 4: Búsqueda y filtrado de productos.
* Requisito 5: Selección y compra de artículos mediante un carrito de compras.
* Requisito 6: Edición del listado de compras durante la   
   compra.  
  siguiendo con el apartado de los requisitos, tenemos la Implementación de los Requisitos 4, 5 y 6 (los cuales fueron especificados anteriormente) en donde el Arquitecto de Software será el encargado de implementar los requisitos especificados en el Hito 1; mencionado anteriormente el diseño de software, continuamos con este apartado con el Diseño de Software 3, en el cual los diseñadores UX y el Arquitecto de Software colaborarán para crear el diseño detallado del software, teniendo en cuenta los requisitos establecidos
* Requisito 7: Medios de pago que confirmen la compra.
* Requisito 8: Reporte de compra que permita al usuario

solicitarlo.

* Requisito 9: Soporte al cliente para resolver cualquier   
   problema.

y para finalizar, tenemos el Análisis y Verificación del Hito 2, en este documento se realizará un análisis para asegurar que el diseño del software y la base de datos cumplan con los requisitos establecidos.

Hito 3:

En el siguiente hito del proyecto, nos enfocaremos en la implementación de los requisitos

* Requisito 7: Medios de pago que confirmen la compra.
* Requisito 8: Reporte de compra que permita al usuario

solicitarlo.

* Requisito 9: Soporte al cliente para resolver cualquier   
   problema.

asegurando que se lleven a cabo de manera efectiva. Además, realizaremos un análisis detallado y exhaustivo del hito 3 para garantizar que el diseño y la implementación cumplan con todos los requisitos establecidos en esta fase. Posteriormente, procederemos a llevar a cabo una validación integral de los hitos 1-3, revisando minuciosamente cada uno de ellos para verificar que se hayan cumplido todos los objetivos y entregables correspondientes. Finalmente, el Jefe del Proyecto se encargará de elaborar el acta de cierre del proyecto, documentando los resultados finales y cerrando de manera formal esta etapa del proyecto.

1. **Control**Dentro de los sistemas que viene realizando BigTech Consulting se encuentra el Sistema de Venta de Componentes Electrónicos, el cual se viene desarrollando para la empresa TechSales.   
   Es por ello que se realizará el control con el objetivo de garantizar la integridad, trazabilidad y consistencia del software a lo largo de su ciclo de vida. El control de configuración es esencial para gestionar los cambios y versiones del software, asegurando que los componentes y elementos del sistema estén correctamente identificados, controlados y documentados.

Software:

* 1. **ETAPAS DE ÍTEMS**

**ITERACIÓN 1**

| **SISTEMA DE VENTAS DE COMPONENTES ELECTRÓNICOS (SVCE)** | | | | |
| --- | --- | --- | --- | --- |
| **FECHA DE CONTROL: 07/06/2023** | | | | |
| **ETAPAS** | **ITEM** | **NOMENCLATURA** | **INICIO** | **FIN** |
| **ACEPTADO** | *Documento de Especificación de Requisitos 1* | SVCE-DER-1.DOCX | 12/04/2023 | 21/04/2023 |
| *Documento de Especificación de Requisitos 2* | SVCE-DER-2.DOCX | 12/04/2023 | 21/04/2023 |
| *Documento de Especificación de Requisitos 3* | SVCE-DER-3.DOCX | 12/04/2023 | 21/04/2023 |
| *Documento de Especificación de Requisitos 4* | SVCE-DDS-4.DOCX | 12/04/2023 | 21/04/2023 |
| *Documento de Especificación de Requisitos 5* | SVCE-DDS-5.DOCX | 12/04/2023 | 21/04/2023 |
| *Documento de Especificación de Requisitos 6* | SVCE-DDS-6.DOCX | 12/04/2023 | 21/04/2023 |
| *Documento de Especificación de Requisitos 7* | SVCE-DDS-7.DOCX | 12/04/2023 | 21/04/2023 |
| *Documento de Especificación de Requisitos 8* | SVCE-DDS-8.DOCX | 12/04/2023 | 21/04/2023 |
| *Documento de Especificación de Requisitos 9* | SVCE-DDS-9.DOCX | 12/04/2023 | 21/04/2023 |
| *Documento de Análisis de Especificaciones de Requisitos* | SVCE-DAER.DOCX | 22/04/2023 | 22/04/2023 |
| *Documento de Construcción y Análisis de Arquitectura* | SVCE-DCAA.DOCX | 23/04/2023 | 27/04/2023 |
| *Documento del Diseño del Software 1* | SVCE-DDS-1.DOCX | 28/04/2023 | 02/05/2023 |
| *Documento de Especificación de la BD* | SVCE-DBD.DOCX | 28/04/2023 | 02/05/2023 |
| *Documento de Verificación de hito 1* | SVCE-DVH-1.DOCX | 03/05/2023 | 03/05/2023 |

**ITERACIÓN 2**

| **SISTEMA DE VENTAS DE COMPONENTES ELECTRÓNICOS (SVCE)** | | | | |
| --- | --- | --- | --- | --- |
| **FECHA DE CONTROL: 07/06/2023** | | | | |
| **ETAPA** | **ITEM** | **NOMENCLATURA** | **INICIO** | **FIN** |
| **ACEPTADO** | Implementación de Requisito 1- Catálogo de productos correctamente clasificados. | - | 04/05/2023 | 14/05/2023 |
| Implementación de Requisito 2 - Gestión de base de datos de los productos | - | 04/05/2023 | 14/05/2023 |
| Implementación de Requisito 3 - Búsqueda de cliente para generar un reporte. | - | 04/05/2023 | 14/05/2023 |
| Diseño de Software 2 | SVCE-DDS-2.DOCX | 15/05/2023 | 18/05/2023 |
| Implementación de Requisito 4 - Búsqueda y filtrado de productos. | - | 19/05/2023 | 03/06/2023 |
| Implementación de Requisito 5 - Selección y compra de artículos mediante un carrito de compras. | - | 19/05/2023 | 03/06/2023 |
| Implementación de Requisito 6 - Edición del listado de compras durante la compra. | - | 19/05/2023 | 03/06/2023 |
| Diseño de Software 3 | SVCE-DDS-3.DOCX | 20/05/2023 | 03/06/2023 |
| **BORRADOR** | Análisis y Verificación del Hito 2 | SVCE-DVH-2.DOCX | 20/05/2023 | 07/06/2023 |

* 1. **Proceso de Control de Cambios**El Proceso de Control de Cambios es una parte integral del Plan de Gestión de Configuración del software y tiene como objetivo administrar y controlar los cambios que ocurren durante el ciclo de vida del software. Este proceso asegura que los cambios se realicen de manera controlada, minimizando riesgos y maximizando la calidad del software entregado. 
     1. **Formato de Solicitud de Cambio**

El formato de solicitud de cambio es un documento estructurado que se utiliza para solicitar de manera formal la implementación de modificaciones en un proyecto de software. Este documento recopila la información necesaria sobre el cambio propuesto, facilitando su evaluación y gestión adecuada.

**Tabla 1: Solicitud de cambio 1**

| **ID** | SC-01: Encriptamiento de usuario y contraseña |
| --- | --- |
| **PROYECTO** | SVCE (Sistema de Ventas de Componentes Electrónicos) |
| **FECHA** | 19/06/2023 |
| **FUENTES** | Eros Sánchez Anaya - Analista Senior |
| **AUTORES** | Jefferson Callupe Arias |
| **DESCRIPCIÓN** | En cambio, consiste en encriptar el usuario y la contraseña.  A medida que aumentan las preocupaciones sobre la seguridad de los datos, es esencial que fortalezcamos la protección de las credenciales de nuestros usuarios, especialmente sus contraseñas.  **Motivo:**  Actualmente, nuestro software almacena las contraseñas de los usuarios en un formato legible en la base de datos MySQL. Esto representa un riesgo significativo de seguridad, ya que en caso de una brecha de seguridad, las contraseñas podrían ser expuestas y utilizadas maliciosamente. Por lo tanto, es imperativo que implementemos un encriptamiento sólido para garantizar la confidencialidad de las contraseñas de los usuarios.  **Propósito:**  El objetivo principal de este cambio es implementar un algoritmo de encriptación de usuario y contraseña en el proceso de login, para convertir las contraseñas en un formato irreconocible y prácticamente imposible de descifrar. Esto asegurará que incluso si un atacante obtiene acceso a la base de datos, no puedan utilizar las contraseñas en su forma original.  **Recursos para la implementación:**   * Investigación y selección de un algoritmo de encriptación adecuado, como bcrypt o Argon2, que sean compatibles con Java NetBeans y MySQL. * Modificación del código fuente existente para incorporar la encriptación de las contraseñas durante el proceso de registro y autenticación. * Actualización del esquema de la base de datos para almacenar las contraseñas encriptadas. * Revisión y actualización de las políticas y prácticas de seguridad relacionadas con el manejo de las contraseñas.   **Tiempo estimado:**  Se estima que la implementación de este cambio tomará aproximadamente tres semanas, incluyendo el tiempo requerido para la investigación, desarrollo y pruebas exhaustivas.  **Estado:**  Este cambio se encuentra en estado de solicitud y requiere una revisión y aprobación por parte del Comité del Control de Cambios. Una vez aprobado, se procederá con la implementación y se actualizará el estado en consecuencia. |
| **JUSTIFICACIÓN** | El encriptamiento de usuario y contraseña en el login de nuestro software es esencial para proteger los datos sensibles, cumplir con las regulaciones, generar confianza en los usuarios y prevenir ataques. Al encriptar las contraseñas, evitamos su exposición, garantizamos la confidencialidad, fortalecemos la seguridad, cumplimos con requisitos normativos y prevemos ataques de fuerza bruta. |

**Tabla 2: Solicitud de cambio 2**

| **ID** | SC-02: Búsqueda para generar un reporte del cliente |
| --- | --- |
| **PROYECTO** | SVCE (Sistema de Ventas de Componentes Electrónicos) |
| **FECHA** | 19/06/2023 |
| **FUENTES** | Equipo de desarrollo |
| **AUTORES** | Marco Antonio Lopez Salinas |
| **DESCRIPCIÓN** | **Motivo:**  En el sistema de gestión de usuarios y compras en línea, se propone una mejora que consiste en volver a solicitar los datos personales al usuario, como nombre, apellido, dirección, etc., una vez que ha iniciado sesión y ha realizado una compra.  **Propósito:**  El objetivo principal es poder almacenar toda la información relacionada con el usuario y su compra en una misma tabla, lo que simplificará la codificación y permitirá generar reportes más eficientes.  **Tiempo estimado:**  Se estima que la implementación de este cambio tomará aproximadamente 3 semanas, incluyendo el tiempo necesario para el análisis, diseño, desarrollo y pruebas.  **Estado:**  Esta solicitud de cambio se encuentra en estado de revisión y aprobación por parte del equipo de desarrollo. Una vez aprobada, se procederá con la implementación y se actualizará el estado en consecuencia. |
| **JUSTIFICACIÓN** | La justificación de este cambio se basa en la necesidad de contar con una base de datos más organizada y centralizada. Actualmente, los datos del usuario y los detalles de la compra se almacenan en tablas separadas, lo que dificulta la generación de reportes que relacionen directamente el nombre del usuario con la compra realizada. Al volver a solicitar los datos del usuario al finalizar la compra, se podrá guardar toda esta información en una sola tabla, eliminando la necesidad de realizar consultas complejas y jalar datos de diferentes tablas. Además de simplificar la codificación, esta mejora facilitará la generación de reportes que involucren el nombre del usuario junto con los detalles de su compra. Al tener todos los datos en una misma tabla, se podrán realizar consultas más sencillas y eficientes para extraer la información requerida en los reportes. Cabe destacar que esta solicitud de cambio no implica modificaciones significativas en la arquitectura del sistema existente. Se trata principalmente de una mejora en la estructura de la base de datos y en la forma en que se gestionan los datos del usuario y las compras. Implementar este cambio permitirá simplificar la codificación, mejorar el rendimiento del sistema y facilitar la generación de reportes más completos y precisos. |

**Tabla 3: Solicitud de cambio 3**

| **ID** | SC-03:Incorporación de API de pagos con tarjeta |
| --- | --- |
| **PROYECTO** | SVCE (Sistema de Ventas de Componentes Electrónicos) |
| **FECHA** | 19/06/2023 |
| **FUENTES** | Eros Sánchez Anaya - Analista Senior |
| **AUTORES** | Chesney Taichi Marchena Tejada |
| **DESCRIPCIÓN** | **Motivo:**  Con el objetivo de brindar una experiencia de usuario más conveniente y ampliar las opciones de pago, es crucial que el sistema cuente con la capacidad de aceptar pagos mediante tarjetas de crédito o débito. En la actualidad, se carece de esta funcionalidad, lo que limita las opciones de pago y puede afectar negativamente el nivel satisfacción de los usuarios.  **Propósito:**  El propósito principal de este cambio es integrar una API de pagos con tarjeta en el sistema, lo que permitirá procesar transacciones de manera segura y eficiente. Al ofrecer esta funcionalidad, mejorará significativamente la experiencia de compra para los usuarios y se brindará un método de pago confiable y conveniente.  **Recursos para la implementación:**   * Investigación y selección de una API de pagos con tarjeta confiable y compatible con las necesidades del proyecto. * Integración de la API seleccionada en el sistema existente, asegurando una comunicación segura y confiable con el proveedor de servicios de pagos. * Desarrollo de la interfaz de usuario para permitir a los usuarios ingresar los detalles de su tarjeta y procesar los pagos de manera segura. * Implementación de las funciones necesarias para manejar la respuesta de la API de pagos y actualizar el estado de los pagos en la base de datos.   **Tiempo estimado:**  Se estima que la implementación de este cambio tomará aproximadamente 4 semanas, teniendo en cuenta el tiempo requerido para la investigación, desarrollo y pruebas.  **Estado:**  Este cambio se encuentra en estado de solicitud y requiere una revisión y aprobación por parte del Comité del Control de Cambios. Una vez aprobado, se procederá con la implementación y se actualizará el estado en consecuencia. |
| **JUSTIFICACIÓN** | La justificación para la integración de una API de pagos con tarjeta se basa en los siguientes puntos:   1. Mejora de la experiencia del usuario al ofrecer una forma conveniente y ampliamente utilizada de completar transacciones en línea. 2. Ampliación de opciones de pago para adaptarse a las preferencias y necesidades de los usuarios, lo que puede aumentar la tasa de conversión y fidelidad de los clientes. 3. Garantía de seguridad de los datos financieros mediante el uso de una API confiable que maneje de forma segura la comunicación y el almacenamiento de información de tarjetas. 4. Integración fluida con proveedores de servicios de pago establecidos para facilitar la gestión de transacciones y la reconciliación financiera. 5. Mantenimiento de la competitividad al alinearse con los estándares y tendencias de la industria en términos de opciones de pago.   Por lo tanto, la incorporación de una API de pagos con tarjeta en el sistema de software brindará beneficios significativos al mejorar la experiencia de los usuarios, ampliar las opciones de pago y fortalecer la seguridad de los datos financieros. Esto permitirá que el sistema se mantenga a la vanguardia y cumpla con las expectativas de los usuarios y los estándares de la industria. |

**Tabla 4: Solicitud de cambio 4**

| **ID** | SC-04: Incorporación de editor de compras en carrito |
| --- | --- |
| **PROYECTO** | SVCE (Sistema de Ventas de Componentes Electrónicos) |
| **FECHA** | 19/06/2023 |
| **FUENTES** | Equipo de desarrollo |
| **AUTORES** | Cristhian Paolo Atuncar Yataco |
| **DESCRIPCIÓN** | **Motivo:**  En el sistema de gestión de usuarios y compras en línea, se propone una mejora que consiste en incorporar un editor de compras luego de agregar producto a la canasta virtual de compras y acceder al apartado de carrito. Al mostrar los productos que se encuentran en proceso de confirmación, dar opción a modificar la cantidad o remover el producto.  **Propósito:**  El objetivo principal es poder ofrecer mayor facilidad al usuario al momento de interactuar con el sistema, lo que supone además, un correcto y cómodo proceso de compra.  **Tiempo estimado:**  Se estima que la implementación de este cambio tomará aproximadamente 2 semanas, incluyendo el tiempo necesario para el análisis, diseño, desarrollo y pruebas.  **Estado:**  Esta solicitud de cambio se encuentra en estado de revisión y aprobación por parte del equipo de desarrollo. Una vez aprobada, se procederá con la implementación y se actualizará el estado en consecuencia. |
| **JUSTIFICACIÓN** | La justificación de este cambio corresponde en primera instancia, a satisfacer de manera correcta las necesidades del cliente, además de ofrecer un servicio de calidad. Teniendo en cuenta que las decisiones a tomar de un cliente, en elegir y/o preferir un sistema de compras, se basa en cómo el sistema ofrece sus funcionalidades. Además, como todo sistema de ventas, adoptar un apartado de edición de compras, ayudará de manera directa a la empresa, a que no se produzcan errores y/o reclamos en el proceso de compra, ofreciendo la mejor experiencia al usuario. Cabe destacar que esta solicitud de cambio no implica modificaciones significativas en la arquitectura del sistema existente. Se trata principalmente de una mejora en la estructura del carrito. |

**Tabla 5: Solicitud de Cambio 5**

| **ID** | SC-05: Compatibilidad con múltiples plataformas |
| --- | --- |
| **PROYECTO** | SVCE (Sistema de Ventas de Componentes Electrónicos) |
| **FECHA** | 19/06/2023 |
| **FUENTES** | Equipo de desarrollo |
| **AUTORES** | Kevin Tupac Agüero |
| **DESCRIPCIÓN** | **Motivo:**  Agregar la capacidad multiplataforma a nuestra aplicación nos permitirá ampliar nuestro alcance, ofrecer una experiencia de usuario consistente, mejorar la eficiencia en el desarrollo, adaptarnos a futuras tendencias tecnológicas y escalar nuestro negocio de manera flexible.  **Propósito:**  Nuestro propósito al agregar la capacidad multiplataforma es ofrecer una experiencia de usuario excepcional, expandir nuestro alcance, optimizar nuestros procesos de desarrollo y adaptarnos a las necesidades cambiantes del mercado.  **Tiempo estimado:**  Se estima que la implementación de este cambio tomará aproximadamente 3 meses, incluyendo el tiempo necesario para el análisis, diseño, desarrollo y pruebas.  **Estado:**  Esta solicitud de cambio se encuentra en estado de revisión y aprobación por parte del Comité de Control de Cambios. Una vez aprobada, se procederá con la implementación y se actualizará el estado en consecuencia. |
| **JUSTIFICACIÓN** | La justificación de esta decisión estratégica de implementar la capacidad multiplataforma en nuestra aplicación traerá beneficios significativos tanto para nuestra empresa como para nuestros clientes. Al adoptar esta funcionalidad, expandiremos nuestro alcance en el mercado, mejoraremos la experiencia del usuario y optimizaremos nuestros procesos de desarrollo y mantenimiento. Además, estaremos preparados para adaptarnos a las tendencias tecnológicas y fomentar el crecimiento y la escalabilidad de nuestro negocio. |

**Tabla 6: Solicitud de Cambio 6**

| **ID** | SC-06: Generar un documento externo respecto al reporte de compra |
| --- | --- |
| **PROYECTO** | SVCE (Sistema de Ventas de Componentes Electrónicos) |
| **FECHA** | 19/06/2023 |
| **FUENTES** | Equipo de desarrollo |
| **AUTORES** | Eros Anaya Sánchez |
| **DESCRIPCIÓN** | **Motivo:**  Con el objetivo de brindar una experiencia de usuario mucho más rica, se dará mayores facilidades a la hora de dar el reporte de venta en cierto formato.  **Propósito:**  El propósito principal de este cambio es que a la hora de generar el reporte de compra, este se descargará como un documento aparte, para que el cliente pueda tenerlo en todo momento sin necesidad de acceder al software constantemente.  **Recursos para la implementación:**   * Bibliotecas y herramientas de generación de documentos, se puede utilizar bibliotecas o herramientas específicas que te permitan generar documentos en formato PDF, Word u otros formatos comunes. * Se pueden diseñar plantillas de documentos con un formato predefinido que se ajuste a los requerimientos del reporte de compra. * Utilizando la información existente en la tabla de reporte de compra, puedes generar dinámicamente el contenido del documento. Esto implica extraer los datos necesarios de la base de datos o sistema de almacenamiento del proyecto y colocarlos en la plantilla de documento, reemplazando los campos variables con los valores correspondientes. * Interfaz de usuario y experiencia de usuario, la cual debe ser clara y fácil de usar para que el usuario pueda generar y descargar el documento..   **Tiempo estimado:**  Se estima que la implementación de este cambio tomará aproximadamente 4 semanas, teniendo en cuenta el tiempo requerido para la investigación, desarrollo y pruebas.  **Estado:**  Este cambio se encuentra en estado de solicitud y requiere una revisión y aprobación por parte del Comité del Control de Cambios. Una vez aprobado, se procederá con la implementación y se actualizará el estado en consecuencia. |
| **JUSTIFICACIÓN** | La justificación para realizar el cambio de generar un documento aparte que el usuario pueda descargar se basa en las siguientes razones:   1. Mejora de la experiencia del usuario: Al proporcionar la opción de descargar un documento aparte, se brinda comodidad al usuario, ya que no será necesario que acceda constantemente al software para consultar el reporte de compra. Esto agiliza el proceso y permite al usuario tener acceso rápido a la información que necesita. 2. Portabilidad de la información: Al contar con un documento descargable, el usuario puede tener acceso a los datos del reporte de compra en cualquier momento, incluso sin conexión a Internet. Esto facilita el seguimiento de las compras, la revisión de información importante o la presentación de informes a terceros, sin depender exclusivamente de la aplicación en línea. 3. Facilidad de archivo y compartición: Descargar el informe de compra como un documento permite al usuario guardar una copia localmente en su dispositivo o sistema de archivos. Esto simplifica el archivo y la organización de los informes, y también facilita su compartición con otras personas o departamentos que pueden requerir acceso al informe. 4. Flexibilidad y adaptabilidad: La capacidad de generar un documento aparte brinda flexibilidad y adaptabilidad al sistema. En el futuro, podrían surgir nuevas necesidades o requerimientos relacionados con los informes de compra, y tener la capacidad de generar documentos descargables permitirá realizar modificaciones o mejoras adicionales de manera más sencilla.   En resumen, implementar la generación de un documento aparte descargable en el proyecto mejora la experiencia del usuario, brinda portabilidad de la información, facilita el archivo y compartición de los informes y proporciona flexibilidad para futuras mejoras. |

**Tabla 7: Solicitud de cambio 7**

| **ID** | SC-07: Incorporación de un proceso de verificación |
| --- | --- |
| **PROYECTO** | SVCE (Sistema de Ventas de Componentes Electrónicos) |
| **FECHA** | 19/06/2023 |
| **FUENTES** | Eros Sánchez Anaya - Analista Senior |
| **AUTORES** | Liñan Paredes Saul Alexander |
| **DESCRIPCIÓN** | **Motivo:**  En el contexto de nuestro sistema de ventas de componentes electrónicos, se propone una mejora significativa: la implementación de un proceso de verificación de seguridad adicional durante las transacciones. Este proceso adicional tiene como objetivo garantizar la autenticidad y legitimidad de las compras realizadas, generando un ambiente de confianza tanto para nuestros usuarios como para el sistema en sí.  **Propósito:**  El propósito principal de esta mejora es fortalecer la seguridad en todo el proceso de compra en línea, protegiendo la información financiera sensible de nuestros usuarios y evitando cualquier posible incidente de fraude.  **Recursos para la implementación:**   * Investigación de proveedores de servicios de pago: Realizar una investigación exhaustiva para identificar proveedores de servicios de pago que ofrezcan opciones de verificación de seguridad para tarjetas de crédito. Evaluar la reputación, la seguridad y la integración técnica de los proveedores para tomar una decisión informada. * Modificación del código fuente existente para incorporar la verificación de seguridad de tarjetas de crédito * Actualización del esquema de la base de datos para almacenar la información relevante de las transacciones de tarjetas de crédito de manera segura. * Realizar pruebas exhaustivas del sistema para garantizar que la verificación de seguridad de tarjetas de crédito funcione correctamente   **Tiempo estimado:**  Para la implementación de este cambio, se estima un plazo aproximado de 2 semanas, considerando el tiempo necesario para realizar un análisis detallado de los requisitos, diseñar e implementar el proceso de verificación de seguridad y realizar pruebas exhaustivas.  **Estado:**  Esta solicitud de cambio se encuentra en estado de revisión y aprobación por parte de la Comisión de Control de Cambio (CCC). Una vez aprobada, se procederá con la implementación y se actualizará el estado en consecuencia. |
| **JUSTIFICACIÓN** | La justificación de este cambio se centra en mejorar la experiencia del cliente y brindar un servicio de alta calidad en nuestro sistema de ventas de componentes electrónicos. Al implementar la verificación de seguridad para tarjetas de crédito, estamos abordando una necesidad fundamental en el proceso de compra en línea, ya que este método de pago es ampliamente utilizado y demandado por nuestros clientes.  Al incorporar un proceso de verificación de seguridad para las tarjetas de crédito, garantizamos que los datos sensibles de los clientes estén protegidos de posibles amenazas y fraudes. Esto fortalece la seguridad del sistema y aumenta la confianza de los clientes al realizar sus compras. |

* + 1. **Estado de SC**
    2. **Proceso**

1. **Estado**
2. **Entrega y Gestión de Release**

**REFERENCIAS**

**Apache Subversion. (s. f.). Apache Subversion Introduction. Apache Software Foundation. Recuperado el 23 de abril de 2023, de** [**https://subversion.apache.org/docs/introduct.html**](https://subversion.apache.org/docs/introduct.html)

**Mackall, M. (2005). Mercurial: The Definitive Guide. O'Reilly Media, Inc.**

**Mercurial (n.d.). In Mercurial. Retrieved September 15, 2021, from** [**https://www.mercurial-scm.org/**](https://www.mercurial-scm.org/)

**Leon, A., & Leon, M. (2015). Software Configuration Management Handbook. Taylor & Francis. https://www.taylorfrancis.com/books/9781317676946**

**Wheeler, D. A. (2009). Subversion versus git: Myths and facts. In Proceedings of the 2009 3rd IEEE International Conference on E-Learning in Industrial Electronics (pp. 7-11). IEEE.**

**Pressman, R. S., & Maxim, B. R. (2015). Ingeniería del software: Un enfoque práctico (7a ed.). McGraw-Hill Education.**[**http://cotana.informatica.edu.bo/downloads/ld-Ingenieria.de.software.enfoque.practico.7ed.Pressman.PDF**](http://cotana.informatica.edu.bo/downloads/ld-Ingenieria.de.software.enfoque.practico.7ed.Pressman.PDF)

***5. Gestión de la Configuración del Software (GCS) 5.1. La Configuración del Software*. (n.d.).** [**https://w3.ual.es/~rguirado/posi/Tema5-Apartado5.pdf**](https://w3.ual.es/~rguirado/posi/Tema5-Apartado5.pdf)