“Año de la recuperación y consolidación de la economía peruana”

Identificación en la Configuración del Software

Luis Sebastian Nuñez Fuentes 2022-119035
Sebastian Tomas Linares Liendo 2022-119010
Luciana Nikole Huertas Condori 2022-119020

Ingeniería en informática y sistemas, Universidad Nacional Jorge Basadre Grohmann

Ingeniería de Software II

Mgr. Gianfranco Alexey Málaga Tejada

04 de Junio del 2025

Tacna, Perú

Identificación en la Configuración del Software

Tabla de Contenidos

A. Introducción

El desarrollo de software moderno se caracteriza por una creciente complejidad, múltiples interdependencias entre componentes, equipos trabajando de forma paralela y un ritmo constante de cambios. En este contexto, la ausencia de mecanismos de control adecuados expone a los proyectos a riesgos significativos, como inconsistencias, defectos recurrentes o dificultades para reproducir versiones anteriores del producto (Computer, s.f.).

La Gestión de la Configuración de Software (GCS) se posiciona como una disciplina fundamental para afrontar estos desafíos. Su objetivo es mantener la coherencia funcional y física del software a lo largo de su ciclo de vida, brindando visibilidad y control sobre cada componente desarrollado. Su relevancia ha crecido en entornos modernos que adoptan metodologías ágiles, prácticas DevOps e integración/entrega continua (CI/CD), donde la gestión eficiente de cambios y versiones es indispensable para asegurar calidad, colaboración y fiabilidad (Larion, s.f.)

Dentro de la GCS, el proceso de identificación en la configuración del software representa el paso inicial y estructural. Este proceso consiste en definir y organizar sistemáticamente los elementos de configuración que serán controlados, facilitando su trazabilidad, recuperación y control a lo largo del desarrollo. Una identificación deficiente puede comprometer la eficacia de las actividades posteriores de la GCS, como el control de cambios, la contabilidad del estado o las auditorías.

Esta monografía tiene como propósito ofrecer un marco teórico riguroso sobre la identificación en la configuración del software, apoyado en fuentes reconocidas como la Guía SWEBOK. Asimismo, se presenta un estudio de caso aplicado al desarrollo de un Sistema de Gestión de Clínica Dental, que permitirá ilustrar la aplicación práctica de los principios abordados, los beneficios obtenidos y los desafíos encontrados.

B. Marco Teórico

a. Gestión de la Configuración de Software (GCS)

La Gestión de la Configuración (GC) se define como la disciplina encargada de establecer y mantener la coherencia de las características funcionales y físicas de un producto a lo largo de todo su ciclo de vida. La Gestión de la Configuración de Software (GCS) es

la aplicación de estos principios específicamente a los artefactos de software (IEEE Computer Society, 2024, p. 8-1).

La GCS es un proceso de ingeniería, que indica procedimientos técnicos para el control y mejora de la calidad del software. Este proceso se estructura en diversas actividades clave, entre las cuales se incluyen: la identificación de los elementos de configuración, la gestión de cambios, el control de versiones, la auditoría de la configuración y la generación de informes de estado. Cada una de estas fases incorpora criterios definidos para garantizar un control riguroso, permitiendo integrar de forma coherente todas las tareas asociadas al desarrollo desde sus etapas iniciales. Asimismo, la GCS asigna responsabilidades específicas a los integrantes del equipo, con el objetivo de facilitar el mantenimiento y la evolución eficiente del sistema software (Pressman, 2010).

b. Identificación en la Configuración de Software

i. Definición y propósito de la identificación (SWEBOK)

La identificación en la configuración del software es una actividad clave dentro de la gestión de la configuración del software, cuyo propósito es definir claramente qué elementos del software deben ser gestionados y controlados durante todo el ciclo de vida del proyecto. Según el SWEBOK, esta actividad consiste en identificar y establecer esquemas únicos para los ítems de configuración, lo cual incluye no solo el código fuente sino también la documentación, herramientas, datos y otros artefactos asociados (IEEE Computer Society, 2024, p. 8-6).

El propósito de esta identificación es garantizar la trazabilidad, el control y la integridad de los productos de software. Esto permite controlar los cambios, mantener versiones y baselines estables, y facilitar la coordinación entre los diferentes equipos que participan en el desarrollo y mantenimiento. En esencia, la identificación proporciona la base para que las demás actividades de la gestión de configuración (como el control de cambios, el estado de la

configuración y las auditorías) se realicen de forma ordenada y efectiva (IEEE Computer Society, 2024, p. 8-6).

Esta práctica resulta fundamental para asegurar que el software evoluciona de manera controlada y predecible, minimizando riesgos asociados a modificaciones no autorizadas o pérdida de información crítica. Además, contribuye a la mejora continua al facilitar la evaluación de los impactos de los cambios y la documentación sistemática del estado del producto.

ii. Ítems a controlar (Software Configuration Items - SCI)

Los ítems a controlar dentro de la gestión de la configuración se denominan Software Configuration Items (SCIs) y representan cualquier componente del software o sus artefactos relacionados que deben ser gestionados como unidad (IEEE Computer Society, 2024, p. 8-6). Estos ítems pueden incluir una amplia variedad de elementos, tales como el código fuente, código ejecutable, documentación de diseño, especificaciones, planes de prueba, datos, herramientas de desarrollo y manuales de usuario.

La correcta selección de los SCIs es crítica porque implica un balance entre la visibilidad y control suficientes para garantizar la integridad del software, y la eficiencia en la gestión evitando una excesiva fragmentación que podría generar complicaciones administrativas (IEEE Computer Society, 2024, p. 8-6).

Además, los SCIs suelen tener relaciones estructurales entre ellos que deben ser consideradas para facilitar la trazabilidad y el análisis de impacto de cambios. Por ejemplo, un cambio en un módulo puede afectar documentación asociada o pruebas relacionadas, por lo que el esquema de identificación debe contemplar estas interdependencias para asegurar una gestión integral (IEEE Computer Society, 2024, p. 8-7).

iii. Relaciones entre los elementos de configuración

En la gestión de la configuración del software, las relaciones entre los elementos de configuración, son fundamentales para

garantizar la integridad, coherencia y trazabilidad del sistema. Estas relaciones permiten entender cómo se interconectan los diferentes artefactos, como el código fuente, la documentación, los casos de prueba y las librerías, y cómo los cambios en un elemento pueden afectar a otros. Según los marcos normativos y las buenas prácticas reconocidas, la correcta identificación y registro de estas relaciones es esencial para permitir un análisis de impacto eficiente y facilitar auditorías técnicas y de calidad. Los esquemas de relación como lo de sucesión, derivación o dependencia, permiten trazar una red estructurada entre los elementos, ayudando a prevenir errores de integración y a mantener la consistencia del sistema a lo largo de sus ciclos de vida (IEEE Computer Society, 2024, p. 8-7).

La documentación de estas relaciones no solo mejora la trazabilidad, sino que también facilita la toma de decisiones cuando se evalúan cambios en el software. Diversos autores señalan que la falta de un esquema de relaciones bien definido, puede generar problemas en el control de versiones, retrasos en la liberación de productos y dificultades en la validación de requisitos. Como indica la literatura, la inclusión de mecanismos de trazabilidad entre los SCIs, junto con el soporte de herramientas específicas, permite a los equipos de desarrollo detectar conflictos, evaluar riesgos y garantizar la coherencia en todo el proceso de gestión de la configuración (IEEE Computer Society, 2014, p. 8-10).

iv. Versiones de software y variantes

El control de versiones es un proceso clave dentro de la gestión de la configuración, ya que permite gestionar la evolución del software a lo largo del tiempo de manera estructurada y controlada. Cada versión corresponde a un estado específico del sistema, identificado por un conjunto consistente de SCIs que han sido aprobados para su liberación. Este proceso de versionado asegura que cada entrega del producto sea rastreable, reproducible y verificable, lo que es especialmente relevante en entornos de desarrollo colaborativos o distribuidos. La creación de baselines o líneas base es un mecanismo

formal que define un conjunto de elementos de configuración aprobados en un momento específico, sirviendo como referencia para futuras modificaciones y desarrollos. Por lo que el control de versiones y la definición de líneas base permiten mantener configuraciones estables a lo largo del tiempo, mejorando la trazabilidad y la reproducibilidad del producto software (Sommerville, 2011, p.738-740).

Por otro lado, la gestión de variantes permite adaptar el software a diferentes entornos, requisitos de clientes o configuraciones específicas. Estas variantes se pueden manejar mediante estrategias de ramificación, como el uso de ramas de sistema de control de versiones o la aplicación de mecanismos como feature toggles. Según diversos enfoques descritos en la literatura, la correcta gestión de variantes reduce la complejidad y los errores en la personalización del producto, permitiendo mantener un código base estable mientras se desarrollan adaptaciones específicas. La implementación de estas prácticas requiere una combinación de control de versiones, integración continua y documentación clara en artefactos como el documento de la descripción de versión, facilitando la sincronización de cambios y la trazabilidad de versiones (Hass, A. M., 2008, p.26-27).

v. Líneas base (Baselines)

Un concepto esencial dentro de la GCS es el de línea base, la cual representa una versión validada y controlada de un componente del sistema. Esta versión se emplea como referencia estable para evaluar el avance del proyecto y garantizar consistencia a lo largo del desarrollo (LARION, s.f.).

Las líneas base representan lo que el proyecto acordó en momentos importantes: un documento, código o elemento de configuración que se aprobó como consecuencia de una revisión técnica. Una vez establecida la línea base, los cambios en los elementos de esa línea base necesitan una aprobación formal, y el sistema de GCS a menudo impide que se modifiquen directamente (Pressman, 2010).

Se pueden definir distintos tipos de línea base dependiendo del grado de complejidad del sistema y de las necesidades de la organización (SWEBOK Wiki, s.f.). Las tres líneas base formales que suelen crearse y coinciden con las etapas del ciclo de vida son:

- Línea Base Funcional: incluye información sobre los requisitos funcionales y la estructura del diseño de los componentes principales (Larion, s.f.; SWEBOK Wiki, s.f.).
- Línea Base Asignada: representa la asignación de requisitos a los elementos principales del sistema, define las expectativas de cada componente en términos de funcionalidad y rendimiento (SWEBOK Wiki, s.f.).
- Línea Base de Producto: Define la versión final del producto de software, lista para la entrega al cliente o para su lanzamiento (Larion,s.f.).

vi. Proceso de adquisición de los SCIs

La adquisición de los Software Configuration Items (SCIs) es un proceso fundamental dentro de la Gestión de la Configuración del Software (SCM), que asegura la coherencia y el control de los elementos del software a lo largo de su ciclo de vida. No se trata solo de la compra o la incorporación inicial de componentes, sino de un proceso continuo de identificación, formalización y control dentro de un sistema de gestión.

La identificación de estos elementos implica un análisis detallado para determinar su relevancia dentro del proyecto, su susceptibilidad a cambios durante el desarrollo y su uso posterior. Los elementos que no tienen un impacto significativo o que son temporales generalmente no se consideran SCIs, lo que permite focalizar los esfuerzos en los componentes esenciales que afectarán el producto final (CMS.gov, 2024).

El proceso de adquisición de SCIs consta de varias fases clave, que incluyen la planificación, donde se definen las necesidades y los criterios de aceptación, y la definición del producto, que implica la validación de los requisitos funcionales y no funcionales (Standish

Group, 1995). La determinación del enfoque es otro aspecto crítico, donde se decide si los elementos serán desarrollados internamente, comprados o utilizados desde soluciones de código abierto. Asimismo, la identificación y evaluación de proveedores es esencial, particularmente cuando se incorporan componentes de software de terceros, lo que implica garantizar que dichos elementos estén debidamente registrados y controlados en la Base de Datos de Gestión de la Configuración (CMDB) (Miami University, 2024).

Una característica esencial de este proceso es que no se trata de un evento aislado. A medida que el software evoluciona, surgen nuevos componentes, se modifican los existentes y aparecen dependencias externas, lo que requiere que el proceso de adquisición de SCIs sea dinámico y se adapte continuamente a los cambios. Este enfoque es especialmente relevante en entornos ágiles y de integración continua (CI/CD), donde los cambios son frecuentes y de pequeña escala, lo que demanda una revisión y formalización constante de los SCIs (Micro Focus, 2024).

c. Biblioteca de software

En la Gestión de la Configuración del Software (SCM), una biblioteca de software actúa como un repositorio centralizado, fundamental para gestionar y almacenar todos los artefactos relacionados con el software, como el código compilado, las bibliotecas, las dependencias y la documentación. Su propósito es asegurar que todos los componentes del software sean accesibles y controlados por versiones durante todas las etapas del ciclo de vida del desarrollo, garantizando la integridad y la fiabilidad de las versiones. Además, la biblioteca funciona como el "backbone" del proceso de construcción, siendo esencial tanto para el desarrollo como para la entrega de las versiones finales del software (Computer.org, 2024).

Existen tres tipos principales de bibliotecas dentro de la SCM: la biblioteca dinámica, que alberga elementos recién creados o modificados; la biblioteca controlada, que gestiona las líneas base y requiere autorización para modificar los elementos; y la biblioteca

estática, que guarda las versiones finales y estables de los elementos de configuración para su uso general. Estas bibliotecas permiten un control eficaz de las versiones y son esenciales para asegurar la trazabilidad de los componentes y las decisiones de desarrollo, lo que es crucial para analizar el impacto de los cambios propuestos (CMS.gov, 2024).

Una gestión eficiente de la biblioteca de software no solo permite el acceso y control de versiones, sino que también asegura que haya una "única fuente de verdad", eliminando silos de información y evitando la dependencia de la memoria de los desarrolladores. Esto facilita la colaboración y permite una mayor consistencia en el desarrollo, independientemente de los cambios de equipo o de los proyectos transitorios. Además, la biblioteca favorece la reutilización de componentes, lo que ofrece beneficios significativos como la reducción de costos, la mejora de la calidad y la fiabilidad del software, y la aceleración del tiempo de comercialización (CMS.gov, 2024).

C. Caso práctico: Implementación de la Identificación de la Configuración en el Proyecto del Sistema de Gestión de Clínica Dental

a. Descripción general del proyecto

Este proyecto corresponde al desarrollo e implementación de un sistema de gestión clínica enfocado en las necesidad del “Centro Dental Especializado Loayza”. Se tiene como objetivo principal proporcionar una solución software adaptada a los procesos operativos y clínicos de la empresa, y que esto permita reemplazar los métodos tradicionales de gestión con herramientas tecnológicas modernas y eficientes sin perder la facilidad de manejo y uso.

El centro dental abarca distintas especialidades de la odontología, tales como la ortodoncia, implantes dentales, blanqueamiento dental, periodoncia, cirugía oral, prótesis dentales, toma de radiografías, endodoncia, entre otros. Resultando en que la variedad de tratamientos requiera un control riguroso y organizado de la información clínica de cada paciente, así como de los procedimientos que han sido realizados o se han planificado.

Por lo cual se han planteado diversas características base. Entre ellas se encuentra el registro detallado de las citas realizadas, anotaciones clínicas, procedimientos aplicados. Este historial debe permitir una trazabilidad completa del tratamiento del paciente desde su primera vista.

También se tiene la implementación de un modelo gráfico de la cavidad bucal del paciente en el que se representen las piezas dentales individualmente, con la posibilidad de marcar tratamientos realizados, condiciones clínicas, observaciones relevantes y registro de placas radiográficas.

De manera adicional, la clínica requiere de una forma de agendar citas, donde se puedan automatizar tareas recurrentes, integrar la agenda con el historial clínico, y eliminar la dependencia en terceros que puedan ofrecer este servicio.

La naturaleza del proyecto involucra una implementación gradual y controlada, aplicando los principios de la ingeniería de software como lo son la gestión de la configuración, el control de versiones, la documentación técnica (aunque en menor escala dada la metodología ágil aplicada) y la validación con usuarios reales. En este contexto, la identificación de la configuración cumple un rol fundamental para garantizar que todos los artefactos del sistema (requerimientos, módulos, interfaces, datos, documentación, entre otros) sean trazables, auditables y capaces de ser modificados de manera controlada de acuerdo al avance del ciclo de vida del software.

El proyecto es parte de una actividad para ganar experiencia y conocimiento para los integrantes del equipo de desarrollo, quienes deben demostrar su capacidad para abordar una situación de la vida real, elaborar una solución efectiva, gestionarla bajo estándares técnicos, y entregar un producto funcional, útil y de valor para el cliente.

b. Aplicación de la identificación de software en el proyecto

i. Selección de SCIs en el sistema

Este punto representa un paso estratégico crucial para garantizar un control eficaz y ordenado del software. La selección debe fundamentarse en un análisis exhaustivo del sistema completo y sus

componentes, considerando tanto aspectos técnicos como las necesidades específicas de gestión del proyecto (IEEE Computer Society, 2024, p. 8-6).

Para este proyecto, el SWEBOK recomienda que la selección de SCIs alcance un equilibrio adecuado: los ítems a controlar deben ser lo suficientemente detallados para que su gestión y trazabilidad sean efectivas, pero sin llegar a un nivel excesivo que genere carga administrativa innecesaria y ralentice el proceso. Por ejemplo, se pueden identificar como SCIs independientes a los módulos funcionales principales, la base de datos, la documentación técnica, los scripts de pruebas automatizadas y los archivos de configuración del servidor, en lugar de controlar cada archivo individualmente (IEEE Computer Society, 2024, p. 8-6).

Adicionalmente, el esquema de identificación para los SCIs debe contemplar la evolución del software durante todo su ciclo de vida. Esto implica que cada ítem pueda manejar versiones, variantes y relaciones estructurales claras, permitiendo no solo el mantenimiento de líneas base estables, sino también una evaluación eficiente del impacto que cada cambio pueda tener en componentes interrelacionados, como la sincronización entre el módulo de citas y el historial clínico. Esta práctica garantiza que los cambios sean controlados y documentados adecuadamente, facilitando la estabilidad y calidad del sistema (IEEE Computer Society, 2024, p. 8-7).

a. Código fuente

Tabla

Elemento de Configuración	**Descripción**	**Importancia**
Controladores (ts)	Gestionan rutas y lógica de interacción cliente-servidor.	Esenciales para el flujo del sistema.
Servicios (ts)	Contienen la lógica de negocio.	Clave para el funcionamiento del sistema.
Módulos (ts)	Agrupaciones lógicas que organizan componentes.	Aportan escalabilidad y organización.
DTO (Data Transfer Objects)	Transportan datos entre	Garantizan estructura e

	capas.	integridad.

b. Documentación de Requisitos

Tabla

Elemento de Configuración	Descripción	Importancia
Requisitos Funcionales	Funciones y características requeridas.	Guía esencial del desarrollo.
Requisitos No Funcionales	Rendimiento, seguridad, usabilidad, etc.	Establecen restricciones de calidad.
Historias de Usuario	Funcionalidades descritas desde el punto de vista del usuario.	Aseguran valor entregado.

c. Diseño del Sistema

Tabla

Elemento de Configuración	Descripción	Importancia
Diseño General	Interacción entre módulos.	Clave para una arquitectura coherente.
Diseño UX/UI	Interfaces y experiencia del usuario.	Vital para la usabilidad.
Diseño de Base de Datos	Estructura relacional de los datos.	Base del almacenamiento de datos.
Diseño de Bajo Nivel	Clases, algoritmos, métodos.	Detalla la implementación técnica.

d. Planes y Gestión del Proyecto

Tabla

Elemento de Configuración	Descripción	Importancia
Plan General del Proyecto	Alcance, cronograma, recursos, responsables.	Base organizativa del proyecto.
Reglamento Interno del Equipo	Normas de trabajo.	Orden y productividad del equipo.
Contrato con el Cliente	Acuerdo formal de condiciones y entregables.	Establece responsabilidades.

e. Pruebas y Validación

Tabla

Elemento de Configuración	Descripción	Importancia
Casos de Prueba por Módulo	Pruebas documentadas por funcionalidad.	Garantizan que cada módulo funciona.
Scripts de Prueba Automatizados	Ejecutan pruebas sistemáticas.	Aseguran repetibilidad y confiabilidad.

f. Despliegue y Configuración

Tabla

Elemento de Configuración	Descripción	Importancia
Scripts de Despliegue	Automatizan instalación y configuración.	Evitan errores manuales.
Base de Datos (Script)	Estructura y carga de datos inicial.	Asegura coherencia entre entornos.
Configuración de Logging	Registra eventos y errores.	Crucial para mantenimiento.
Configuraciones de Entorno	Variables, tokens, conexiones.	Habilitan el funcionamiento del sistema.
URLs Backend y Frontend	Direcciones de acceso al sistema.	Esenciales para la comunicación y acceso.
Servicios Externos	APIs integradas.	Impulsan funcionalidades clave.

g. Documentación Técnica y de Usuario

Tabla

Elemento de Configuración	Descripción	Importancia
Manual de Usuario General	Uso global del sistema.	Facilita la adopción del sistema.
Manual por Módulo	Instrucciones específicas.	Guía para el uso de funcionalidades concretas.

h. Herramientas de Desarrollo

Tabla

Elemento de Configuración	Descripción	Importancia
NestJS	Framework backend basado en Node.js.	Estructura el backend.
React (o biblioteca frontend)	Framework para la interfaz de usuario.	Desarrollo del frontend moderno.

ii. Establecimiento de relaciones entre los SCIs (cómo se vinculan en el sistema)

Tabla

Categoría	Código Fuente	Documentación de Requisitos	Diseño del Sistema	Gestión del Proyecto	Pruebas y Validación	Despliegue y Configura-ción	Documenta-ción Técnica	Herramientas de Desarrollo
Código Fuente	-	Utiliza	Utiliza	-	Incluye	Extiende	Ocasiona	Utiliza
Documen-tación de Requisitos	Dirige	-	Guía	Es parte de	-	Ayuda a medir	Referencia	-
Diseño del Sistema	Guía	Utiliza	-	-	-	-	Es parte de	-
Gestión del Proyecto	-	Incluye	-	-	Controla	-	Incluye	Gestiona
Pruebas y Validación	Evalúa	-	-	Controla do por	-	Extiende	Es parte de	-
Despliegue y Configura ción	Requie re	Se mide por	-	-	Incluye	-	Genera	-
Documen-tación Técnica	Incluye	Incluye	Incluye	Es parte de	Incluye	Reporta	-	-
Herra-mientas de Desarrollo	Facilita	-	-	Controla do por	-	-	-	-

iii. Definición y control de versiones

Para el control de versiones en el proyecto del sistema de gestión de clínica dental, se adoptó la herramienta GitHub como repositorio centralizado y sistema de control distribuido. Esta selección respondió a su accesibilidad, integración con herramientas de automatización, y soporte para flujos de trabajo colaborativos. Se

definió una política de versionada basada en el esquema SemVer, estructurando cada versión en formato MAJOR.MINOR.PATCH, lo que permitió distinguir fácilmente entre nuevas funcionalidades, mejoras incrementales y correcciones menores. Cada modificación al código fue acompañada por una descripción clara en el mensaje de commit, alineada con el artefacto de requisitos y documentación técnica correspondiente.

Se utilizó la estrategia GitFlow como técnica de control de versiones, que permitió estructurar las ramas en desarrollo (develop), producción (main), características nuevas (feature/), correcciones (hotfix/) y versiones de prueba (release). Esta estructura facilitó una gestión ordenada del código fuente, reduciendo conflictos y favoreciendo una integración continua y más limpia. Cada liberación del sistema se marcó mediante tags en GitHub, acompañados de la versión del documento de descripción de versiones que especificaba los SCIs incluidos, cambios aplicados y artefactos relacionados. Esta práctica permitió establecer líneas base verificables en cada hito del proyecto, asegurando la trazabilidad y la consistencia entre código, documentación y pruebas.

iv. Establecimiento de líneas base para el proyecto

1. Línea Base Funcional

Antes de presentar la tabla con los elementos correspondientes a la Línea Base Funcional, es importante entender que esta línea base establece las funcionalidades principales que debe tener el sistema. En el contexto de un sistema de gestión para una clínica dental, la Línea Base Funcional incluye los módulos y características esenciales que aseguran que el sistema cumpla con los requisitos funcionales definidos por el cliente, como la gestión de citas, el control de historiales médicos y la facturación. Esta línea base es clave para tener claridad sobre las expectativas y para garantizar que las funcionalidades críticas estén presentes desde el inicio del proyecto.

Tabla

Documento de Requisitos

Artefacto	Documento de requisitos
Objetivo	Obtener una relación validada de los requisitos a partir de información facilitada por el usuario, a fin de obtener un catálogo detallado de requisitos, mediante el cuál se pueda comprobar que los productos generados en las actividades de modelización se ajustan a los requisitos.
Producto a obtener	Documento de requisitos funcionales, de rendimiento, de seguridad. Historias de Usuario.
Participantes	Stakeholders y Analistas
Técnicas	Entrevistas y análisis de documentación existente

Nota. Esta tabla muestra los elementos necesarios para validar y obtener un catálogo de requisitos detallados, esenciales para asegurar que el sistema de gestión de la clínica dental cumpla con los requisitos del cliente.

Tabla

Modelo de Datos

Artefacto	Modelo de datos
Objetivo	Representar estructuralmente los datos que el sistema de gestión de la clínica dental debe almacenar, procesar y relacionar. Este modelo permite visualizar las entidades clave del sistema, sus atributos y las relaciones entre ellas.
Producto a obtener	Modelo conceptual de Datos Modelo Lógico de Datos
Participantes	Analista de Datos, Equipo de desarrollo
Técnicas /Prácticas	Diagramas de Entidad-Relación Uso de Herramientas CASE Normalización

Nota. Esta tabla describe la representación estructural de los datos que el sistema debe manejar, proporcionando un modelo conceptual y lógico que facilita la creación de la base de datos.

Tabla

Modelo de Procesos del Sistema

Artefacto	Modelo de procesos del Sistema
Objetivo	Elaborar un documento estructurado que contenga los diagramas de procesos clave del sistema de gestión para una clínica dental. Estos diagramas describen los flujos operativos de cada módulo
Producto a obtener	Modelo conceptual de Datos Modelo Lógico de Datos
Participantes	Analistas, usuarios clave, product owner
Técnicas /Prácticas	Diagramas de UML

Nota. Aquí se definen los flujos operativos del sistema de gestión de la clínica dental, a través de diagramas de procesos clave que representan los módulos del sistema.

Tabla

Descripción de Interfaces

Artefacto	Descripción de Interfaces
Objetivo	Documentar las interfaces internas del sistema, enfocadas en las interacciones entre los distintos perfiles de usuario (odontólogos, administrativos y pacientes) y el sistema, a través de pantallas, formularios y formatos de salida (como reportes o recetas). Estas interfaces garantizan la usabilidad y correcta operación del sistema de gestión clínica.
Producto a obtener	Especificación de Interfaz de Usuario Descripción de interfaces de usuario (UI) para cada módulo
Participantes	Analistas, usuarios clave, product owner
Técnicas /Prácticas	Prototipos y wireframes ágiles

Nota. En esta tabla se documentan las interfaces entre los diferentes perfiles de usuario y el sistema, asegurando la usabilidad a través de prototipos y wireframes ágiles.

Tabla

Especificación Formal de Requisitos de Software

Artefacto	Especificación Formal de Requisitos de Software
Objetivo	Contar con una especificación detallada y validada de los requisitos de software que guiarán el desarrollo y la entrega incremental del sistema, permitiendo al equipo de desarrollo entender claramente las funcionalidades y restricciones para cada sprint.
Producto a obtener	Documento de Especificación de Requisitos de Software actualizado Historias de usuario claras y aceptadas Criterios de aceptación para cada requisito Documentación ligera y flexible para ajustes durante el desarrollo ágil
Participantes	Analistas, Product Owner, equipo de desarrollo, usuarios clave y stakeholders
Técnicas /Prácticas	Revisión continua y refinamiento de requisitos Priorización basada en valor y riesgos

Nota. Este artefacto presenta los requisitos detallados del software, fundamentales para guiar el desarrollo y asegurar que el sistema cumpla con las funcionalidades definidas.

Tabla

Aprobación de Requisitos por Parte del Cliente

Artefacto	Aprobación de requisitos por parte del cliente
Objetivo	Obtener la validación y aprobación formal por parte del cliente y principales interesados sobre los requisitos definidos, incluyendo restricciones, limitaciones de diseño y criterios de aceptación, para asegurar un entendimiento claro y compartido antes de iniciar o continuar el desarrollo de los sprints.
Producto a obtener	Acta de Aprobación de Requisitos Registro de acuerdos y observaciones sobre requisitos y restricciones
Participantes	Jefe de Proyecto Cliente
Técnicas /Práctica	Presentación y exposición de los requisitos y criterios definidos

s	Discusión abierta para aclaración de dudas y acuerdos

Nota. Esta tabla garantiza que los requisitos del sistema sean validados y aprobados formalmente por el cliente, asegurando un entendimiento claro antes de continuar con el desarrollo.

2. Línea Base Asignada

La Línea Base Asignada es aquella en la que se determinan los recursos, tiempos y responsables para cumplir con las funcionalidades definidas. En este caso, se asignan los recursos del equipo de desarrollo y se establecen los plazos para cada una de las funcionalidades del sistema de gestión. La tabla a continuación refleja cómo cada función será asignada a los miembros del equipo, las herramientas que utilizarán y el tiempo estimado para su implementación, lo cual es crucial para el control de progreso durante el ciclo de desarrollo.

Tabla

Especificación Formal de Requisitos de Software

Artefacto	Diseño de la interfaz del usuario
Objetivo	Definir la estructura visual, comportamiento interactivo y la experiencia de usuario de las interfaces de los módulos del sistema, asegurando una navegación intuitiva y coherente que facilite la interacción eficiente de los usuarios con el sistema.
Producto a obtener	Diagramas de flujo de navegación entre pantallas Especificaciones de elementos visuales Guía de estilo visual como colores, tipografía e iconografía
Participantes	Diseñadores UX/UI / Frontends Stakeholders
Técnicas /Prácticas	Revisión y retroalimentación iterativa con usuarios finales Prototipado rápido con herramientas Figma Revisión de Estándares de Accesibilidad

Nota. Define la estructura visual, el comportamiento interactivo y la experiencia de usuario de las interfaces del sistema, asegurando una navegación intuitiva y eficiente.

Tabla

Modelo Físico de Datos

Artefacto	Modelo Físico de Datos
Objetivo	Definir la estructura física y almacenamiento de los datos que soportan los módulos del sistema, asegurando la integridad, eficiencia y escalabilidad de la base de datos para el correcto funcionamiento de la aplicación.
Producto a obtener	Esquemas de base de datos físicos. Scripts de creación y configuración de la base de datos Documentación de cambios y versiones.
Participantes	Desarrollador Backend DBA Analistas
Técnicas /Prácticas	Modelado entidad-relación (ER) adaptado a nivel físico Revisión y validación con el equipo de desarrollo

Nota. Establece la estructura física de la base de datos, garantizando la eficiencia, integridad y escalabilidad para el correcto funcionamiento del sistema.

Tabla

Especificación Técnica del Plan de Pruebas

Artefacto	Especificación Técnica del Plan de Pruebas
Objetivo	Definir el enfoque, criterios, tipos y niveles de pruebas que se aplicarán al sistema durante su desarrollo, con el fin de garantizar la calidad funcional, técnica y de rendimiento del software desde las primeras fases del ciclo de vida.
Producto a obtener	Documento del plan de pruebas por módulo Casos de prueba técnicos como unitarias y de integración
Participantes	Equipo de QA Desarrolladores Backend y Frontend Product Owner
Técnicas /Prácticas	Diseño de casos de prueba basados en historias de usuario Priorización de pruebas por riesgos y valor del módulo Uso de framework como Jest

Nota. Define el enfoque y los criterios de pruebas para garantizar la calidad técnica, funcional y de rendimiento del sistema desde sus primeras fases de desarrollo.

Tabla

Manuales de Implantación

Artefacto	Manuales de Implantación
Objetivo	Documentar de forma detallada los procedimientos, configuraciones y recursos necesarios para desplegar correctamente el sistema en los entornos de prueba y/o producción asegurando su correcto funcionamiento.
Producto a obtener	Manual de instalación técnica del sistema(backend y frontend) Manual de configuración inicial del software Guía de despliegue en entorno local Procedimientos de backup y recuperación Requisitos de hardware, software y red
Participantes	Equipo de desarrollo (backend y frontend)
Técnicas /Prácticas	Uso de contenedores Docker Documentación detallada en Markdown

Nota. Documenta los procedimientos, configuraciones y recursos necesarios para desplegar el sistema en los entornos de prueba y producción, asegurando su correcto funcionamiento.

Tabla

Documento de Aprobación del Diseño

Artefacto	Documento de aprobación del diseño
Objetivo	Contar con la validación y aprobación formal del diseño del sistema de gestión para la clínica dental, asegurando que la arquitectura, interfaces, modelos de datos y demás componentes de diseño cumplen con los requisitos establecidos por el cliente y están alineados con los objetivos del proyecto.
Producto a obtener	Acta de Aprobación del Diseño del Sistema Firma de conformidad del cliente Validación del diseño de interfaz de usuario, modelo físico de datos y estructura funcional Confirmación del cumplimiento de restricciones de diseño, usabilidad y rendimiento.
Participantes	Scrum Máster y cliente

Técnicas /Prácticas	Revisión formal del diseño mediante presentación estructurado

Nota. Valida y aprueba el diseño del sistema, asegurando que cumpla con los requisitos establecidos por el cliente y los objetivos del proyecto.

3. Línea Base de Producto

La Línea Base de Producto establece la versión inicial del software que incluye tanto el código funcional como los componentes adicionales, como la documentación y la integración de sistemas. En el caso del sistema de gestión de clínica dental, esta línea base define el producto que se entregará al cliente en una versión estable, completa y lista para su implementación. La tabla de la Línea Base de Producto proporciona detalles sobre los artefactos finales, las versiones de los módulos y cómo estos se integran en el sistema para asegurar su funcionamiento completo.

Tabla

Base de Datos Física

Artefacto	Base de datos física (base_de_datos.sql)
Objetivo	Contener el script completo para la creación y configuración de la base de datos final, incluyendo datos iniciales, para asegurar que el sistema almacena y gestiona la información correctamente en producción y pruebas.
Producto a obtener	Archivo SQL con scripts de creación de tablas, índices, datos iniciales y configuraciones específicas del gestor de base de datos.
Participantes	Equipo de desarrollo backend, DBA (Administrador de Base de Datos).
Técnicas	Diseño físico de base de datos, normalización, pruebas de integridad y rendimiento.

Nota. Contiene el script completo para la creación y configuración de la base de datos final, asegurando que el sistema gestione la información correctamente en producción y pruebas.

Tabla

Código Fuente del Sistema

Artefacto	Código fuente del sistema
Objetivo	Contar con el código completo y organizado que implementa todas las funcionalidades del sistema, facilitando mantenimiento y futuras modificaciones.
Producto a obtener	Repositorios de código fuente con estructura modular. Documentación requirements.txt.
Participantes	Equipo de desarrollo frontend y backend.
Técnicas /Prácticas	Programación modular. Control de versiones con Git. Revisiones de código.

Nota. Presenta el código completo y organizado que implementa todas las funcionalidades del sistema, permitiendo mantenimiento y futuras modificaciones.

Tabla

Bibliotecas y Herramientas Utilizadas

Artefacto	Bibliotecas y herramientas utilizadas
Objetivo	Documentar las librerías, frameworks y herramientas utilizadas, asegurando la replicabilidad del entorno de desarrollo y compatibilidad futura.
Producto a obtener	Listado detallado de tecnologías empleadas: React, shadcn, TailwindCSS (frontend); NestJS, TypeORM, Jest (backend); Visual Studio Code, Docker, Git, GitHub (herramientas).
Participantes	Equipo de desarrollo backend y frontend
Técnicas	Gestión de dependencias. Control de versiones. Documentación técnica.

Nota. Documenta las librerías, frameworks y herramientas utilizadas, asegurando la replicabilidad del entorno de desarrollo y la compatibilidad futura.

Tabla

Pruebas Unitarias, de Integración y de Sistema

Artefacto	Pruebas unitarias, de integración y de sistema

Objetivo	Garantizar la calidad del producto final mediante la ejecución y documentación de pruebas en todos los niveles.
Producto a obtener	Reportes. Documentación de resultados de pruebas unitarias, de integración y de sistema.
Participantes	Equipo de QA. Desarrolladores Backend y Frontend. Product Owner.
Técnicas /Prácticas	Automatización de pruebas con Jest. Ejecución en entornos controlados.

Nota. Garantiza la calidad del producto mediante la ejecución y documentación de pruebas en todos los niveles del sistema.

Tabla

Scripts de Despliegue

Artefacto	Scripts de Despliegue (deploy.sh)
Objetivo	Automatizar la instalación, configuración y despliegue del sistema para minimizar errores humanos y agilizar la puesta en producción.
Producto a obtener	Scripts ejecutables que realizan el despliegue completo en entornos definidos (desarrollo, prueba, producción).
Participantes	Desarrolladores Backend y Frontend
Técnicas /Prácticas	Scripting bash. Uso de Docker y contenedores. Pruebas de despliegue.

Nota. Automatiza la instalación, configuración y despliegue del sistema para agilizar la puesta en producción y minimizar errores humanos.

Tabla

Manual del Usuario Final

Artefacto	Manual del Usuario Final
Objetivo	Proporcionar instrucciones claras y detalladas para que los usuarios puedan manejar el sistema de forma

	autónoma.
Producto a obtener	Documento de usuario con guías, tutoriales, FAQs y descripción de funcionalidades.
Participantes	Equipo de documentación, analistas, usuarios finales.
Técnicas /Prácticas	Redacción técnica. Revisión con usuarios.

Nota. Proporciona instrucciones claras y detalladas para que los usuarios puedan manejar el sistema de forma autónoma, incluyendo guías, tutoriales y FAQs.

v. Herramientas y técnicas usadas para gestión y control (por ejemplo, uso de repositorios Git, herramientas de SCM, etc.)

Para garantizar una correcta gestión y control del desarrollo del sistema de la clínica dental, se ha hecho uso de herramientas ampliamente reconocidas en el ámbito del desarrollo ágil del desarrollo software, como lo son Jira y Git, principales aliadas a lo largo del ciclo de vida del proyecto. Estas herramientas han permitido establecer un flujo de trabajo ordenado, asegurar la trazabilidad de las actividades y mantener la calidad del producto software que se ha desarrollado.

Jira ha sido la plataforma elegida para la gestión integral de los módulos y las pruebas del sistema. Aunque no es una herramienta nativa para testing como lo es TestLink, su flexibilidad permite configurar un entorno que cubra los aspectos esenciales del control de calidad. Se ha trabajado con una estructura que permite vincular directamente los casos de prueba con las historias de usuario y los módulos del sistema. En este esquema, cada módulo del sistema ha sido representado por un Epic, y dentro de ellos se han registrado funcionalidades específicas bajo el tipo de New Feature. Las mejoras sobre funcionalidades ya existentes han sido documentadas como Mejoras, mientras que los errores identificados durante la ejecución de pruebas han sido gestionados como Bugs, manteniendo su relación directa con las funcionalidades correspondientes.

Adicionalmente, se han definido tipos de issues personalizados para registrar casos de prueba, permitiendo planificar y documentar pruebas directamente dentro del entorno de Jira. A través de filtros, tableros y exportación de reportes a Excel o PDF, se ha llevado a cabo un monitoreo constante de la calidad del sistema, controlando tanto la cantidad de pruebas realizadas como los errores encontrados y su respectiva solución.

Por otro lado, para el control de versiones se ha utilizado Git, asegurando una gestión eficiente del código fuente. Además, gracias a su enfoque distribuido, permitió mantener múltiples ramas activas para distintas etapas del desarrollo, esto sin comprometer la estabilidad del sistema principal. Cada cambio ha sido registrado y versionado, lo cual permite regresar regresar a estados anteriores del proyecto si fuese necesario, esto ha facilitado la colaboración entre los distintos miembros del equipo. Las plataformas como Github han sido claves para facilitar la revisión de código mediante pull requests y para llevar un seguimiento adicional de las tareas técnicas asociadas a cada cambio.

En conjunto, ambas herramientas han sido aplicadas por la gestión de la configuración del software (SCM), entre ellas, la identificación y trazabilidad de todos los elementos de configuración (SCI), abarcando desde el código fuente y los scripts de despliegue, hasta la documentación de diseño y los manuales de usuario. Cada elemento ha sido controlado, versionado y vinculado a su funcionalidad o módulo correspondiente dentro del sistema que se está desarrollando.

Estas herramientas y técnicas han sido fundamentales para sostener una metodología de trabajo colaborativa, ordenada y centrada en la calidad, permitiendo además responder de manera eficiente a cambios o incidencias durante el desarrollo.

c. Beneficios y desafíos encontrados

La implementación de la identificación y gestión de la configuración en el proyecto del Sistema de Gestión de Clínica Dental nos trajo muchos

beneficios claros, aunque también tuvimos algunos desafíos al principio, como suele pasar al adoptar cosas nuevas.

La adopción de un sistema de control de versiones distribuido como Git permitió a los desarrolladores trabajar en paralelo en diferentes módulos (por ejemplo, pacientes, citas) sin conflictos de sobreescritura, porque Git nos daba un lugar controlado para unir el código, esto generó mayor productividad en el equipo.

Además que nos dió la capacidad de rastrear cada cambio, esto hizo que fuera más fácil auditar, encontrar errores y comprender cómo evolucionó el sistema. Al controlar estos cambios nos dió la capacidad de asegurar que todo fuera consistente con las líneas base.

No obstante también se enfrentaron algunos desafíos como la resistencia al cambio, ya que algunos miembros del equipo se sentían más cómodos con herramientas tradicionales y tardaron en adaptarse al nuevo flujo de trabajo. Para lograr una gestión adecuada del control de cambios, fue necesario establecer convenciones claras para el uso de Git. Aunque algunos integrantes encontraron difícil aprenderlas inicialmente, estas prácticas resultaron fundamentales para mejorar la trazabilidad y el orden en el desarrollo del proyecto.

d. Conclusiones del caso práctico

La aplicación del proceso de identificación de la configuración en el proyecto del sistema de gestión para una clínica dental demostró ser un componente esencial para el éxito del desarrollo. Desde la definición de SCIs hasta el establecimiento de líneas base, el enfoque sistemático permitió asegurar que cada cambio fuera controlado, documentado y validado. La selección adecuada de elementos de configuración como módulos de funcionalidad, scripts de base de datos, documentación técnica y manuales de usuarios, posibilitó un control exhaustivo del producto en cada etapa del ciclo de vida. A su vez, el establecimiento de relaciones entre SCIs fortaleció la trazabilidad entre requisitos, código y pruebas, facilitando el análisis de impacto ante cada cambio.

El uso de GitHub y GitFlow permitieron gestionar múltiples líneas de desarrollo con eficacia, integrando funcionalidades específicas para distintos tipos de usuario como administradores, odontólogos y pacientes, sin

comprometer la estabilidad del sistema principal. Las líneas base establecidas antes de cada entrega funcionaron como puntos de referencia sólidos, minimizando el riesgo de errores de producción. Sin embargo el proyecto también enfrentó desafíos, como la necesidad de capacitar al equipo en prácticas de gestión de configuración, o la dificultad inicial para definir correctamente todos los SCIs relevantes. A pesar de ellos, los beneficios superaron ampliamente las dificultades, se incrementó la calidad del producto, se redujo el número de incidencias post-entrega, y se logró una mayor cohesión en el equipo de trabajo. El caso evidencia que la implementación rigurosa de gestión de configuración no sólo es factible en proyectos medianos, sino también altamente recomendable.

D. Conclusiones generales

E. Referencias bibliográficas

CMS.gov. (2024). Configuration Management Processes. Recuperado de https://www.cms.gov/tra/Application_Development/AD_0520_Configuration_Management_Processes.htm

Computer.org. (2024). Software Configuration Management. Recuperado de https://www.computer.org/resources/software-configuration-management

IEEE Computer Society. (2024). Software configuration management. En H. Washizaki (Ed.), Guide to the Software Engineering Body of Knowledge (SWEBOK Guide), version 4.0 (pp. 8-1 – 8-17). Los Alamitos, CA: IEEE Computer Society.

Hass, A. M. (2008). Configuration Management: Principles and Practice. Addison-Wesley.

LARION. (s.f.). Practical guide to software configuration management. https://larion.com/practical-guide-to-software-configuration-management/

Miami University. (2024). Configuration Item Owner. Recuperado de https://miamioh.teamdynamix.com/TDClient/2385/ITSM/KB/ArticleDet?ID=109256

Micro Focus. (2024). Configuration item relationships. Recuperado de https://docs.microfocus.com/SM/9.51/Hybrid/Content/configuration/concepts/configuration_item_relationships.htm

Pressman, R. S. (2010). Ingeniería del software: Un enfoque práctico (7.ª ed.). McGraw-Hill.

SEBOK Wiki. (s.f.). Configuration baselines. Guide to the Systems Engineering Body of Knowledge (SEBoK). https://sebokwiki.org/wiki/Configuration_Baselines

Standish Group. (1995). CHAOS Report

Sommerville, I. (2011). *Ingeniería de software* (9.ª ed.). Pearson Educación.