Sistema MinGO
Utilizando Proceso Unificado de Desarrollo

Sistema MinGO
Plan de Gestión de la Configuración del Software

Versión 1

Historia de Revisión

Fecha	Versión	Descripción	Autores
14/1/2026	1.0	Versión inicial del Plan de Gestión de la Configuración para el sistema MinGO.	Marcos Escobar, Mateo Sosa, Fernando Tipán

Tabla de Contenidos

1. Introducción

1.1 Propósito del plan de gestión de la configuración

El documento presentado está dirigido al equipo de desarrollo de sistema MinGO, líder del proyecto, responsables de la gestión de la configuración y para el responsable del aseguramiento de la calidad del software; su principal objetivo es establecer y estandarizar los requisitos, políticas, estándares y procedimientos para la gestión de los artefactos generados por el producto software.

Dentro del documento también se detallan las actividades que permitirán mantener la integridad de los artefactos que se obtienen a lo largo del desarrollo del producto software, de esta manera garantizando que los cambios sean formalmente controlados, teniendo en cuenta la participación de todos los interesados en el desarrollo del sistema.

1.2 Alcance

El Plan de Gestión de la Configuración se aplicará a ciclo de vida completo del sistema MinGO, desde el comienzo del desarrollo hasta el despliegue en producción. Esto incluye los siguientes componentes:

- Backend API (NestJS + PostgreSQL)
- Aplicación móvil (Flutter)
- Base de datos y migraciones (Prisma)
- Documentación previa de la fase de análisis.

1.3 Definiciones y Acrónimos

Acrónimo	Significado
SQA	Aseguramiento de calidad de software (Software Quality Assurance)
GC	Gestión de la configuración
ECS	Elemento de configuración de software
PGC	Plan de gestión de la configuración
API	Application Programming Interface
LSEC	Lengua de Señas Ecuatoriana
CI/CD	Continuous Integration / Continuous Deployment
DTO	Data Transfer Object
ORM	Object-Relational Mapping
JWT	JSON Web Token

1.4 Referencias

- IEEE Computer Society. Software Engineering Technical Commitee. IEEE Standard for Software Configuration Management ANSI-IEEE 828-1990.
- https://forja.molinux.info/frs/download.php/104/PLN_GC.pdf
- SÁNCHEZ María Isabel, Gestión de la Configuración, Politécnica de Madrid, 2006
- Pressman Ingeniería de Sw Un enfoque práctico Pressman Roger S 5ta Edic.

2. Especificaciones de Gestión

2.1 Organización

El proyecto MinGO será desarrollado siguiendo una arquitectura Clean Architecture con Domain-

Driven Desing. Para la estructura organizacional del sistema MinGO se tomó en cuenta los siguientes principios:

- Separación de responsabilidades entre el Backend y Frontend.
- Versionamiento de código mediante el uso de Git y GitHub como plataforma central.

La estructura propuesta busca generar agilidad en la ejecución de las actividades de gestión de la configuración durante el ciclo de vida del software. Todos los participantes del proyecto deben estar atentos a los puntos de partida que se establecen en las líneas base.
Los subprocesos de control de versiones y control de cambios tendrán soporte con herramientas computacionales, lo cual permitirá que todas las dependencias involucradas tengan a su alcance la información que requieran, de manera oportuna.

2.2 Responsabilidades

Rol	*Funciones*	*Responsables*
Líder del proyecto	Se encarga de organizar las diferentes acciones que se llevarán a cabo durante el proceso de desarrollo. Además, verificará el cumplimiento de los procedimientos descritos para el control de cambios.	Marcos Escobar
Gestor de la configuración del software	Tiene la responsabilidad de definir los procesos para la Gestión de la configuración del Software, para esto proporciona estándares para manejar repositorios Git y cómo se deben gestionar las ramas, merges, con el fin de mantener la integridad de las líneas base.	Mateo Sosa
Comité de Control de Cambios	Se encarga de la toma de decisiones ante solicitudes o necesidad de cambios, priorizando que el impacto de los cambios no afecte a los demás artefactos para ello aprobara o rechazara cambios críticos.	Mateo Sosa, Marcos Escobar
Responsable de SQA	Se encarga de realizar auditorías sobre la Gestión de la Configuración del Software, es decir, se encargará de verificar el cumplimiento de estándares establecidos y la revisión del código y documentación, que estos cumplan con los registros de cambios al momento.	Fernando Tipán
Bibliotecario	Se encarga de controlar que se realicen los cambios sobre las últimas versiones de la documentación. Además, se encarga de mover, agrupar y gestionar los artefactos almacenados en las diferentes carpetas especializadas.	Marcos Escobar

2.3 Herramientas de Soporte

Para el subproceso de control de versiones, se utiliza la herramienta Git como sistema distribuido que permite gestionar los cambios que se realizan en el código y artefactos a lo largo del desarrollo. El repositorio de documentación y código se encuentra en alojado en GitHub, el cual es una plataforma que facilita el trabajo colaborativo mediante sus diferentes funcionalidades. Como principal estrategia para el desarrollo, se utiliza GitFlow, lo que nos permitirá tener ramas claramente diferenciadas para la versión estable de sistema MinGO.
Por otro lado, para la gestión de cambios el proyecto hace uso de Jira como herramienta principal para el seguimiento y control de tareas y solicitudes de cambios. Mediante esta plataforma se registran errores y funcionalidades a desarrollar, permitiendo asignar responsables, prioridades y visualizar el estado de una actividad. También se hace uso de los tableros que ofrece Jira para facilitar la planificación de fechas y organización de la carga de trabajo. De esta manera todas las decisiones tanto

organizacionales como las actividades completadas quedan registradas y documentadas desde su planteamiento hasta su cierre.

En lo que respecta a la integración y el despliegue, el proyecto hace uso principalmente de servicios en la nube para evitar costos de adquisición de equipos de almacenamiento y servidores locales, teniendo de esta manera una ejecución estable y fácilmente escalable. La base de datos es gestionada mediante Supabase que es una alternativa open-source a Firebase y que provee un motor PostgreSQL, además de facilitar servicios de autenticación y seguridad. La API de sistema en cambio está desplegada en Render, el cual es un servicio de hosting especializado en desplegar aplicaciones backend de forma continua y automatizada. Render se integra junto con el repositorio GitHub por lo que cada cambio aprobado y enviado a la rama principal es desplegado automáticamente garantizando la consistencia de versiones.

Finalmente, en lo relacionado a la comunicación entre miembros del equipo y documentación del sistema, se utiliza archivos de tipo Markdown para mantener una documentación constantemente actualizada y versionada dentro del mismo repositorio, incluyendo archivos README y el historial de cambios. La API REST se documente mediante OpenAPI/Swagger, proporcionando una entrada clara y accesible para su uso y consumo. Adicionalmente, se utilizan diagramas realizados en Visual Paradigm para representar el sistema desde diferentes perspectivas.

3. Definición de Gestión de la Configuración

3.1 Identificación de la Configuración

3.1.1 Selección de los Elementos de Configuración del Software (ECS)

Los ECS de MinGO están organizados según la arquitectura del sistema y el ciclo de vida del desarrollo.

Disciplinas Básicas	*Código*		*Nombre ECS*
Requisitos	MCU		Modelo de casos de uso, el cual está compuesto por:
		DCU	Diagrama de casos de uso
	ERS		Especificación de requerimientos de software
Análisis	MA		Modelo de Análisis
		DCA	Diagrama de clases de análisis
Diseño	MD		Modelo de diseño
		DPD	Diagrama de patrón de diseño
		DC	Diagrama de clases
		DCO	Diagrama de componentes
		DAS	Descripción de la arquitectura del software
		DER	Diagrama entidad relación
Implementación	CF		Código fuente
	CE		Código ejecutable
	SBD		Script de implementación de la base de datos
Pruebas	PP		Plan de pruebas
	ECP		Especificación casos de prueba
Implantación	MI		Manual de instalación
	MU		Manual de usuario

Disciplinas de Gestión	***Código***	***Nombre ECS***
Gestión del proyecto	PDP	Plan de desarrollo del proyecto
Gestión de configuración y cambio	PGC	Plan de gestión de la configuración
Gestión de la calidad de software	PSQA	Plan de gestión de la calidad de software

3.1.2 Esquema de identificación

Elementos de configuración del software:
Los ECS serán identificados mediante la siguiente información:

- Código del ECS
- Nombre completo del ECS
- Autor(es) del artefacto
- Ruta en el repositorio Git
- Línea base a la que pertenece
- Tipo de ECS (código fuente, documentación, configuración, etc.)
- Fecha de creación
- Última fecha de modificación
- Hash del commit Git

Línea Base:
Para este proyecto se han definido las líneas base que se describen a continuación, una por cada disciplina de la metodología Proceso Unificado de Desarrollo.

Código	***Nombre Línea Base***
LBR	Requisitos
LBA	Análisis
LBD	Diseño
LBC	Implementación / Construcción
LBP	Pruebas
LBI	Implantación
LBGP	Gestión del proyecto
LBGC	Gestión de configuración y cambio
LBQA	Gestión de la calidad de software

Versiones y Variantes:
Se aplicará el siguiente esquema de identificación de versiones y variantes para todos los ECS que se han identificado en la sección anterior, de tal forma que se tenga en todo momento una tabla actualizada con la información correspondiente a las mismas.

- Código del ECS.
- Descripción del ECS
- Número de versión o variante, el cual será secuencial
- Fecha de creación
- Autor o autores.
- Ubicación
- Observación, se indican los cambios respecto de la versión anterior.

3.1.3 Relaciones Existentes entre ECS

Se puede considerar que los ECS son objetos y están conectados con otros ECS mediante relaciones.
Equivalencia:
Se refiere a los artefactos ECS que se encuentran almacenados en múltiples ubicaciones pero que representan el mismo artefacto.

ECS 1	***Ubicaciones***	***Descripción***
DCU	• Se encuentra en LBR • Se encuentre en la carpeta de Diseños /	Se trata del diagrama de casos de uso extendido, pero se encuentra distribuido dentro de la línea base y la carpeta de

	Diagrama de Casos de Uso	diseños.

Composición:
Se trata de ECS que están construidos en base a otros ECS, pero en este caso no existen artefactos documentales que se encuentren compuestos por otros.

Dependencia:
Esta relación se refiere a que un ECS que requiere de otro para funcionar o comprenderse de manera correcta.

ECS 1	*ECS2*	*Descripción*
DCU	ERS	Es necesario revisar la documentación de especificación de requisitos para comprender el diagramado de casos de uso.
DPD	DC	El diagrama de patrones de diseño parte principalmente del diagrama de clases por lo que es necesario revisar primero el diagramado de clases.
DC	DAS	El diagrama de clases es una simplificación a nivel de clases por lo que es importante revisar el diagrama completo de la arquitectura.

Derivación:
Un ECS se origina partir de otro.

ECS 1	*ECS2*	*Descripción*
ERS	DCU	Los casos de uso se derivan de los requisitos funcionales del SRS
DAS	DC	Las clases del diagrama de clase nacen de la implementación de las diferentes carpetas del diagrama de arquitectura.
ERS	CF	El código fuente de aplicación proviene del desarrollo de los diferentes requisitos funcionales y la aplicación de los no funcionales del SRS.

Sucesión:
Historia de cambios sobre un elemento (commits de Git). Cada commit representa una revisión sucesiva.

3.1.4 Definición y Establecimiento de Bibliotecas Software

Una biblioteca de software es una colección controlada de software y documentación relacionada,

donde su principal objetivo es ayudar al desarrollo y mantenimiento del sistema MinGO.
Se establecen las siguientes bibliotecas:

1. **Biblioteca de Trabajo**
   Es la biblioteca inicial por definición del proyecto y comprende un área de trabajo donde los desarrolladores elaboran el código y documentos primaria del sistema. Una vez que todos los documentos dentro de este apartado sean aprobados por los participantes serán trasladados a la Biblioteca de Soporte.
   Control de Acceso:
   Lectura/Escritura completa para el desarrollador propietario.
   Contenido:
   - Elicitación
     - Línea Base
       - LBA
       - LBC
       - LBD
       - LBR
     - Especificación RS
     - Cronograma
     - Matriz IREB
     - Historias de Usuario
     - Actas de Reunión
     - Backlog
     - Pruebas Unitarias
     - Reporte de Errores
   - Perfil del Proyecto
   - Diseños
     - Patrón de Diseño
     - Diseño de Arquitectura
     - Casos de Uso Extendido
     - Diagrama de Clases
     - Diagrama de Componentes
     - Diagrama ER

   Ubicación:
   - Repositorio GitHub de proyecto
   - La ruta empieza en la raíz del repositorio ./Biblioteca\de\Trabajo
2. **Biblioteca de Soporte**
   En este apartado se almacenan los ECS aprobados y transferidos desde la Biblioteca de Trabajo.
   Control de Acceso:
   Lectura para todos, Escritura mediante Pull Request aprobado por al menos 1 revisor.
   Contenido:
   - Elicitación
     - Línea Base
       - LBA
       - LBC
       - LBD
       - LBR
     - Especificación RS
     - Cronograma
     - Matriz IREB
     - Historias de Usuario
     - Actas de Reunión
     - Backlog
     - Pruebas Unitarias
     - Reporte de Errores
   - Perfil del Proyecto
   - Diseños
     - Patrón de Diseño
     - Diseño de Arquitectura

- Casos de Uso Extendido
- Diagrama de Clases
- Diagrama de Componentes
- Diagrama ER

Ubicación:

- Repositorio GitHub de proyecto
- La ruta empieza en la raíz del repositorio ./Biblioteca\de\Soporte

3. **Biblioteca Maestra**

Se utiliza para almacenar ECS completos y aprobados destinados a su entrega al cliente o para su distribución. Los artefactos están sujetos a control de cambios formal y estricto.

Control de Acceso:

Lectura para todos. Escritura SOLO mediante merge aprobado por: Líder del proyecto, Gestor de la Configuración del Software y Responsable de SQA.

Contenido:

- Elicitación
  - Línea Base
    - LBA
    - LBC
    - LBD
    - LBR
  - Especificación RS
  - Cronograma
  - Matriz IREB
  - Historias de Usuario
  - Actas de Reunión
  - Backlog
  - Pruebas Unitarias
  - Reporte de Errores
- Perfil del Proyecto
- Diseños
  - Patrón de Diseño
  - Diseño de Arquitectura
  - Casos de Uso Extendido
  - Diagrama de Clases
  - Diagrama de Componentes
  - Diagrama ER

Ubicación:

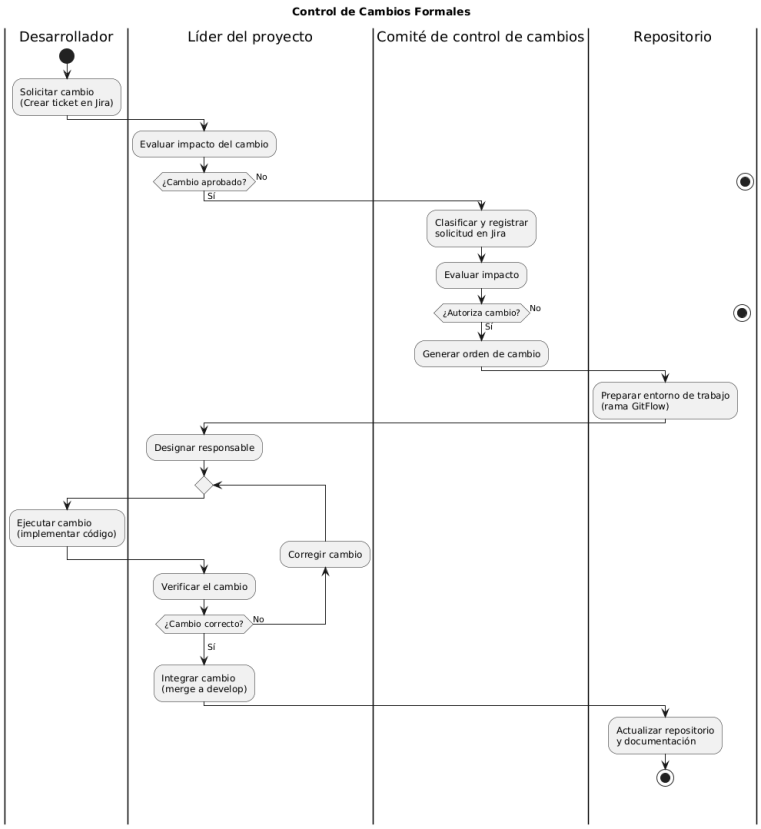
- Repositorio GitHub de proyecto
- La ruta empieza en la raíz del repositorio ./Biblioteca\Maestra

3.2 Configuración y control de cambios

El control de los cambios del proyecto recae sobre el Gestor de Configuración y del Jefe de Proyecto, que garantizan que las modificaciones efectuadas al sistema se realicen de forma organizada y sistemática. Con la finalidad de garantizar que los cambios sean documentados se utiliza Jira, herramienta que permite registrar, priorizar y dar un seguimiento real a las solicitudes de cambios, que nos otorga una trazabilidad desde la apertura hasta el cierre de un cambio.

Si se requiere un cambio se documenta mediante un ticket en Jira, donde se describe entre algunas cosas su alcance, prioridad y siguiente versión. Previo a su incorporación, el cambio es revisado para identificar los artefactos afectados o los cambios realizados en cuestión. La aprobación depende de la criticidad del cambio y puede requerir validación adicional en casos que afecten la estructura del sistema.

Por otra parte, para a incorporación de los cambios se realiza siguiendo las estrategias que nos ofrece GitFlow, utilizando ramas especificas si es el caso para realizar cambios importantes dentro del código del sistema, y para el caso de la documentación serán revisados por el jefe del proyecto y el Gestor de la Configuración para ser incorporados a las diferentes Bibliotecas de trabajo generadas dentro del repositorio.



3.3 Contabilidad del Estado de la Configuración

El objetivo de esta tarea es mantener a los usuarios, gestores y desarrolladores al tanto del estado de la configuración y como sigue evolucionando el sistema y su documentación. Se tiene los siguientes informes:

1. Inventario de ECS

Ofrece visibilidad sobre el contenido de la biblioteca de trabajo al proyecto.

Código	*Nombre ECS*	*Versión*	*Línea Base*	*Autor*	*Estado*
DCU	Diagrama de casos de uso	1.0	LBR	Marcos Escobar	Activo
DPD	Diagrama de patrón de diseño	1.0	LBD	Mateo Sosa	Activo
DC	Diagrama de clases	1.0	LBD	Fernando Tipán	Activo
DCO	Diagrama de componentes	1.0	LBD	Mateo Sosa	Activo
DAS	Diagrama de arquitectura del software	1.0	LBD	Marcos Escobar	Activo
DER	Diagrama entidad relación	1.0	LBD	Marcos Escobar	Activo
CF	Código fuente	0.5	LBI	Marcos Escobar, Mateo Sosa, Fernando Tipán	Activo

2. Inventario de Versiones

Contiene las versiones generadas hasta la fecha.

Código ECS	*Versión*	*Fecha*	*Observaciones*
DCU	V1.0	2026-01-15	
DPD	v1.0	2026-02-15	
DC	v1.0	2026-01-15	
DCO	v1.0	2026-01-15	
DAS	v1.0	2026-01-15	
DER	v1.0	2026-01-15	
CF	V0.5	2026-01-15	

3. Inventario de Líneas Base

Contiene información correspondiente a cada línea base identificada.

Código LB	*Nombre*	*Fecha*	*ECS Incluidos*
LBR	Línea Base Requisitos	2026-12-15	ERS aprobado con 25 requisitos funcionales y 20 no funcionales
LBA	Línea Base Análisis	2026-12-15	
LBD	Línea Base Diseño	2026-12-15	DPD, DC, DCO, DAS, DER diagramados
LBC	Línea Base de Construcción	2026-12-15	API v0.5, APP v0.5, DB-SCHEMA v1.0

4. Inventario de Relaciones entre ECS

Contiene información sobre las relaciones de dependencia y derivación.

Relaciones de Dependencia:

ECS 1	*ECS2*	*Descripción*
DCU	ERS	Es necesario revisar la documentación de especificación de requisitos para comprender el diagramado de casos de uso.
DPD	DC	El diagrama de patrones de diseño parte principalmente del diagrama de clases por lo que es necesario revisar primero el diagramado de clases.
DC	DAS	El diagrama de clases es una simplificación a nivel de clases por lo que es importante revisar el diagrama completo de la arquitectura.

Relaciones de Derivación:

ECS 1	*ECS2*	*Descripción*
ERS	DCU	Los casos de uso se derivan de los requisitos funcionales del SRS
DAS	DC	Las clases del diagrama de clase nacen de la

		implementación de las diferentes carpetas del diagrama de arquitectura.
ERS	CF	El código fuente de aplicación proviene del desarrollo de los diferentes requisitos funcionales y la aplicación de los no funcionales del SRS.

3.4 Auditoría de la Configuración

Con el fin de evaluar la conformidad del producto software con respecto a especificaciones, estándares, acuerdos contractuales u otros criterios, se realizarán auditorías de la configuración al final de cada iteración y antes de crear una línea base.

PLAN DE AUDITORÍAS DE LA CONFIGURACIÓN								
ECS	***10-01-2026***	***11-01-2026***	***12-01-2026***	***13-01-2026***	***14-01-2026***	***15-01-2026***	***18-01-2026***	***20-01-2026***
ERS								
DCU								
DPD								
DC								
DCO								
DAS								
DER								
CF								

4. Glosario

Término	***Definición***
Versión	Es una instancia actual o previa de un artefacto dentro del repositorio GitHub.
Revisión	Es la acción de controlar que cada nueva versión que aparece sea aprobada y descartada
Línea Base	Son aquellos ECS que fueron aprobados formalmente por el equipo, y sirven como un punto de partida para las siguientes versiones. Una vez que fue aprobada y establecida es necesario realizar un proceso de control formal si se requiere una modificación.
Pull Request (PR)	Se refiere a una solicitud formal dentro de GitHub para integrar cambios de una rama a la rama principal del proyecto.
Commit	Se refiere a una captura en el tiempo de los elementos bajo versiona miento, este es registrado en el historial del Git y posee un hash único para su identificación.
GitFlow	Es una estrategia que propone Git para trabajar en diferentes ramas antes de la rama principal y luego combinar mediante PR
Clean Architecture	Es un patrón arquitectónico que divide las responsabilidades en diferentes capas, la lógica de negocio se desarrolla en la capa domain.
Domain-Driven Design (DDD)	Enfoque de diseño que pone el dominio del negocio y la lógica de dominio en el centro del desarrollo.
Use Case	Clase que encapsula una operación de negocio específica. Ejemplo: RegisterUser, CompleteLesson.
Entity	Objeto del dominio con identidad única. Ejemplo: User, Lesson,

	Activity.
Value Object	Objeto del dominio sin identidad, definido por sus atributos. Ejemplo: Email, Password, UUID.
Repository Pattern	Patrón que abstrae el acceso a datos mediante interfaces, separando la lógica de persistencia.
DTO (Data Transfer Object)	Objeto simple para transferir datos entre capas o sistemas, sin lógica de negocio.
BLoC (Business Logic Component)	Patrón de Flutter para gestión de estado, separando UI de lógica de negocio.
Prisma	ORM (Object-Relational Mapping) para Node.js que facilita interacción con base de datos.
Migration	Script que modifica el schema de base de datos de forma versionada y controlada.
Smoke Testing	Prueba rápida que verifica que las funcionalidades críticas funcionan correctamente.
Code Coverage	Métrica que indica qué porcentaje del código está cubierto por tests automatizados.