Sistema de Gestión de Contenidos para Optimizar la Creación de Aplicativos Web en Entornos Educativos

Juan Diego Gutierrez Venegas

No Ficha 2848530-B

Servicio Nacional De Aprendizaje – SENA

Bogotá D.C. Septiembre 2025

Contenido

2. Resumen

El presente proyecto se centra en el diseño e implementación de un Sistema de Gestión de Contenidos (CMS) orientado a optimizar la creación de aplicativos webs en entornos educativos. La iniciativa surge ante la necesidad de las instituciones académicas de contar con herramientas que permitan desarrollar plataformas digitales de manera ágil, escalable y adaptable a diferentes contextos pedagógicos. El CMS propuesto busca simplificar la gestión de contenidos, integrar módulos reutilizables y ofrecer una interfaz intuitiva que facilite el trabajo tanto de administradores como de docentes, reduciendo las barreras técnicas y acelerando los procesos de publicación y actualización de información.

La solución se fundamenta en un enfoque modular que garantiza la personalización y la extensión de funcionalidades, permitiendo a los usuarios construir aplicativos webs de acuerdo con las necesidades de enseñanza-aprendizaje. Asimismo, incorpora principios de usabilidad, seguridad y rendimiento, con el fin de asegurar un sistema confiable y eficiente. El proyecto contempla además la implementación de buenas prácticas de desarrollo de software, siguiendo metodologías ágiles que favorecen la colaboración y la mejora continua.

Entre los principales beneficios del CMS se destacan la reducción del tiempo de desarrollo de aplicaciones educativas, la optimización de recursos tecnológicos, la mejora en la experiencia de usuario y la posibilidad de integrar nuevas tecnologías como repositorios de aprendizaje, entornos colaborativos y herramientas interactivas. En conclusión, este sistema se presenta como una solución estratégica para fortalecer la innovación educativa a través de la digitalización, aportando una plataforma versátil que se adapta a los requerimientos actuales de la educación moderna.

3. Abstract

The CMS Builder project focuses on the design and development of a content management system aimed at facilitating the creation and administration of websites in an efficient and scalable manner. Its main objective is to provide a tool that allows users without advanced programming knowledge to manage pages, menus, posts, users, and configurations through an intuitive and adaptable interface.

The system was developed following the Model-View-Controller (MVC) architectural pattern, which enhances code organization, maintainability, and the integration of new features. A relational database was implemented to support structured data management and ensure consistency and reliability in the information stored.

Key features of the solution include dynamic content customization, role and permission management, and a simplified editor designed to improve the user experience. Additionally, the modular design ensures that the CMS can scale, integrate with other systems, and adapt to different types of web projects.

From a technical perspective, security in data handling, usability in navigation, and efficiency in internal processes were prioritized to guarantee stable performance in production environments. The CMS Builder is mainly targeted at small and medium-sized businesses, independent developers, and projects requiring a flexible, customizable, and easy-to-implement management system.

4. Introducción

La transformación digital ha impactado de manera significativa el ámbito educativo, generando la necesidad de herramientas tecnológicas que faciliten la gestión, creación y difusión de contenidos de manera dinámica y eficiente. En este contexto, el desarrollo de un Sistema de Gestión de Contenidos (CMS) se convierte en una estrategia clave para optimizar la creación de aplicativos webs en entornos académicos, permitiendo a las instituciones fortalecer sus procesos formativos y responder a las demandas de la educación moderna.

El presente documento expone el diseño y la implementación de un CMS enfocado en entornos educativos, resaltando su relevancia como solución integral para mejorar la administración de información, promover la innovación pedagógica y reducir las barreras técnicas asociadas al desarrollo de plataformas digitales. A través de un enfoque modular, escalable y adaptable, el sistema propuesto busca facilitar el trabajo de docentes, estudiantes y administradores, promoviendo la autonomía en la gestión de contenidos y favoreciendo experiencias de aprendizaje más interactivas y personalizadas.

Asimismo, se abordan los fundamentos metodológicos, técnicos y prácticos que respaldan la construcción del proyecto, destacando la aplicación de buenas prácticas de ingeniería de software y metodologías ágiles orientadas a la calidad y mejora continua. De esta forma, el documento no solo presenta la solución tecnológica, sino que también ofrece un marco de referencia que evidencia su impacto potencial en la digitalización educativa.

5. Planteamiento del Problema

5.1 Descripción del Problema

En el ámbito educativo, las instituciones enfrentan limitaciones para desarrollar y mantener aplicativos webs que apoyen procesos pedagógicos y administrativos. La creación de estas plataformas suele requerir altos conocimientos técnicos, inversión de tiempo y recursos que muchas veces no están disponibles. Además, los sistemas existentes carecen de flexibilidad y escalabilidad, lo que dificulta su adaptación a diferentes contextos académicos y necesidades de enseñanza. Esta situación genera obstáculos en la digitalización de los entornos educativos y limita la innovación pedagógica.5.2 Objetivo General

Diseñar y desarrollar un sistema de gestión de contenidos (CMS Builder) que permita la creación, administración y personalización de sitios web de manera flexible, segura y eficiente, facilitando la autogestión digital a usuarios sin conocimientos avanzados en programación mediante una interfaz intuitiva y un diseño modular y escalable.

5.2 Objetivo General

Diseñar e implementar un Sistema de Gestión de Contenidos (CMS) que optimice la creación de aplicativos webs en entornos educativos, permitiendo una administración eficiente, escalable y adaptable a las necesidades pedagógicas de las instituciones.

5.3 Objetivos Específicos

- Analizar los requerimientos funcionales y no funcionales para la construcción del CMS.
- Desarrollar un sistema modular que permita la integración y reutilización de componentes.
- Implementar una interfaz intuitiva que facilite la gestión de contenidos por parte de docentes y administradores.
- Garantizar la seguridad, rendimiento y usabilidad del sistema mediante buenas prácticas de desarrollo.
- Evaluar el impacto del CMS en la optimización de procesos educativos digitales.

5.4 Justificación del Proyecto

La implementación de un CMS educativo responde a la necesidad de modernizar los procesos académicos mediante herramientas digitales flexibles y accesibles. Este proyecto permitirá reducir los tiempos de desarrollo, optimizar recursos tecnológicos y mejorar la experiencia de aprendizaje al facilitar la creación de entornos digitales interactivos. Asimismo, el CMS fomenta la autonomía de los usuarios en la gestión de información y contribuye a la transformación digital de las instituciones educativas.

5.5 Alcance del Proyecto

El sistema abarcará la gestión de contenidos académicos, la integración de módulos reutilizables, el control de usuarios y permisos, así como la posibilidad de extender funcionalidades según las necesidades de cada institución. No se contemplan en esta fase integraciones con sistemas externos de gran escala, pero se dejarán bases para su futura incorporación.

5.6 Matriz de Riesgo

Riesgo	***Probabilidad***	***Impacto***	***Estrategia de Mitigación***
Retrasos en el desarrollo por falta de recursos	Media	Alta	Planificación con metodologías ágiles y control de cronograma
Baja adopción del sistema por falta de capacitación	Alta	Media	Capacitación a docentes y administradores, documentación de usuario
Problemas de seguridad y pérdida de datos	Baja	Alta	Implementación de protocolos de seguridad y respaldos periódicos
Dificultades técnicas en la escalabilidad del sistema	Media	Media	Diseño modular y pruebas de carga desde etapas tempranas
Cambios en los requerimientos institucionales	Alta	Media	Flexibilidad en el diseño y revisiones periódicas con stakeholders

6. ELICITACIÓN DE REQUISITOS

6.1 Identificación de Procesos

Para el desarrollo del CMS se identificaron los siguientes procesos clave dentro del entorno educativo:

- Gestión y publicación de contenidos académicos.
- Administración de usuarios, roles y permisos.
- Creación de aplicativos web modulares y reutilizables.
- Configuración de parámetros institucionales y personalización del sistema.
- Almacenamiento seguro y consulta eficiente de información.
- Interacción entre docentes y estudiantes mediante módulos colaborativos.

6.2 Recolección de la información del software a construir de acuerdo con las necesidades del cliente

6.2.1 Elección de la técnica de recolección de la información

Se seleccionaron técnicas mixtas para garantizar una visión integral de las necesidades:

- Entrevistas a docentes y administradores académicos.
- Encuestas dirigidas a estudiantes para identificar necesidades de interacción.
- Observación directa de procesos actuales de gestión de contenidos.
- Análisis documental de plataformas educativas ya implementadas.

6.2.2 Diseño de los formatos, según la técnica o técnicas de recolección de la información seleccionada(s)

- Formatos de entrevista estructurada para administradores y docentes.
- Formularios digitales de encuesta para estudiantes.
- Listas de verificación para observaciones de procesos.
- Plantillas de análisis comparativo para la revisión de plataformas existentes.

6.2.3 Aplicación de la técnica de recolección de la información

- Se realizaron entrevistas semiestructuradas con 5 administradores y 10 docentes.
- Se aplicaron encuestas a una muestra de 50 estudiantes.
- Se observaron prácticas de publicación de contenidos en aulas virtuales.
- Se analizaron documentos institucionales y manuales de software educativo.

6.2.4 Organización de la información recolectada

La información obtenida fue organizada en matrices de requisitos funcionales y no funcionales, priorizando aspectos como:

- Facilidad de uso e interacción.
- Seguridad en el acceso y almacenamiento de datos.
- Escalabilidad y adaptabilidad del sistema.
- Eficiencia en los procesos de creación y publicación de contenidos.
- Integración con herramientas y recursos educativos existentes.

6.3 ESPECIFICACIÓN DE REQUERIMIENTOS

6.3.1 Requerimientos Funcionales

RF01	El sistema debe permitir el registro, edición y eliminación de usuarios con roles diferenciados
RF02	El sistema debe ofrecer gestión de roles y permisos para controlar el acceso a funcionalidades específicas.
RF03	El sistema debe permitir la creación de aplicativos web modulares y reutilizables
RF04	El sistema debe ofrecer plantillas o módulos predefinidos para acelerar el desarrollo de aplicativos.
RF05	El sistema debe permitir la personalización de proyectos, integrando componentes reutilizables.
RF06	El sistema debe permitir el versionamiento básico de proyectos
RF07	El sistema debe tener una arquitectura MVC para un mejor orden

6.3.2 Requerimientos Funcionales

RNF01	El CMS debe ser distribuido como código abierto y contar con documentación para su instalación y personalización.
RNF02	El sistema debe incluir un instalador automático que configure la base de datos, los usuarios iniciales y la estructura básica del proyecto.
RNF03	Debe estar diseñado de manera portable, permitiendo su instalación en distintos entornos de desarrollo
RNF04	El código debe estar organizado bajo estándares de buenas prácticas (PSR en PHP o equivalentes según el lenguaje elegido).
RNF05	El sistema debe ser ligero y no depender de configuraciones externas complejas.
RNF06	El CMS debe garantizar seguridad en sus componentes iniciales (hash de contraseñas, validación de formularios, control de sesiones).

6.3.3 Propuesta Técnica

CMS será un framework instalable de código abierto que proporcionará automáticamente la estructura inicial de un proyecto web, incluyendo:

Gestión de usuarios y roles (administrador y usuarios básicos).

Administrador de vistas con plantillas preconfiguradas.

Módulo de autenticación (login, registro, recuperación de contraseñas).

Configuración inicial de base de datos mediante el instalador.

El proyecto estará desarrollado bajo una arquitectura MVC (Modelo-Vista-Controlador) para asegurar orden, escalabilidad y fácil mantenimiento. La instalación se realizará a través de un asistente que cree automáticamente tablas en la base de datos, genere las credenciales del administrador y configure el entorno del sistema.

La propuesta contempla el uso de tecnologías de código abierto como PHP/Laravel o Node.js, con soporte para bases de datos MySQL/PostgreSQL. Al ser un CMS enfocado en aprender a programar, el código estará bien documentado y comentado, de manera que los usuarios puedan modificarlo, extenderlo o adaptarlo para construir proyectos como sistemas POS, blogs, e-commerce o portales educativos.

7. REGLAS DEL NEGOCIO

RN-01: Todo usuario registrado debe autenticarse mediante un sistema de login seguro para acceder a las funcionalidades del CMS.

RN-02: El sistema debe contar con al menos un usuario administrador creado automáticamente durante la instalación, encargado de gestionar roles, permisos y configuraciones globales.

RN-03: El administrador es el único que puede crear, editar y eliminar usuarios con privilegios avanzados.

RN-04: Cada usuario debe tener un rol asociado que determine los permisos y accesos a las funcionalidades del CMS.

RN-05: Toda contraseña debe almacenarse de manera cifrada mediante algoritmos de hash seguros.

RN-06: El CMS debe generar automáticamente la estructura base del proyecto (gestión de usuarios, login, vistas iniciales) al finalizar la instalación.

RN-07: El instalador debe verificar la disponibilidad de la base de datos y crear las tablas necesarias de forma automática.

RN-08: Las vistas del sistema deben ser administrables mediante plantillas reutilizables y configurables.

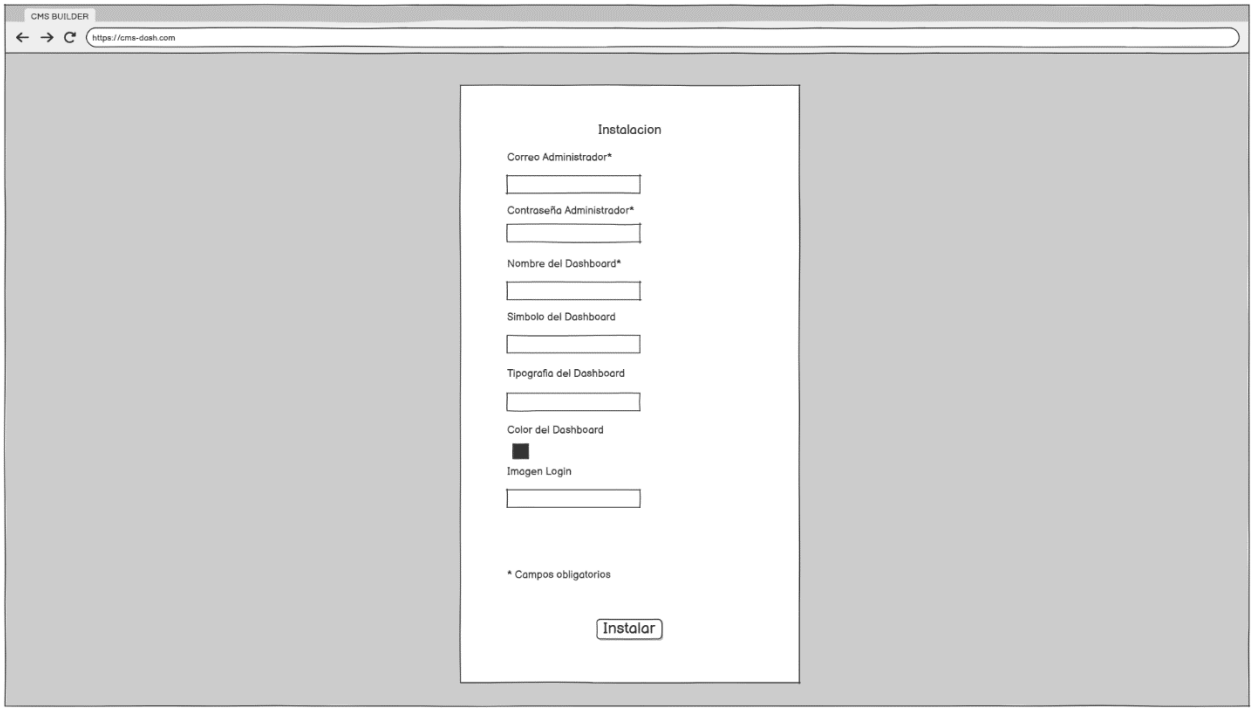
RN-09: Todo acceso a secciones restringidas debe validar sesión activa y rol correspondiente.

RN-10: El CMS debe permitir la extensión mediante módulos adicionales sin alterar la estructura base.

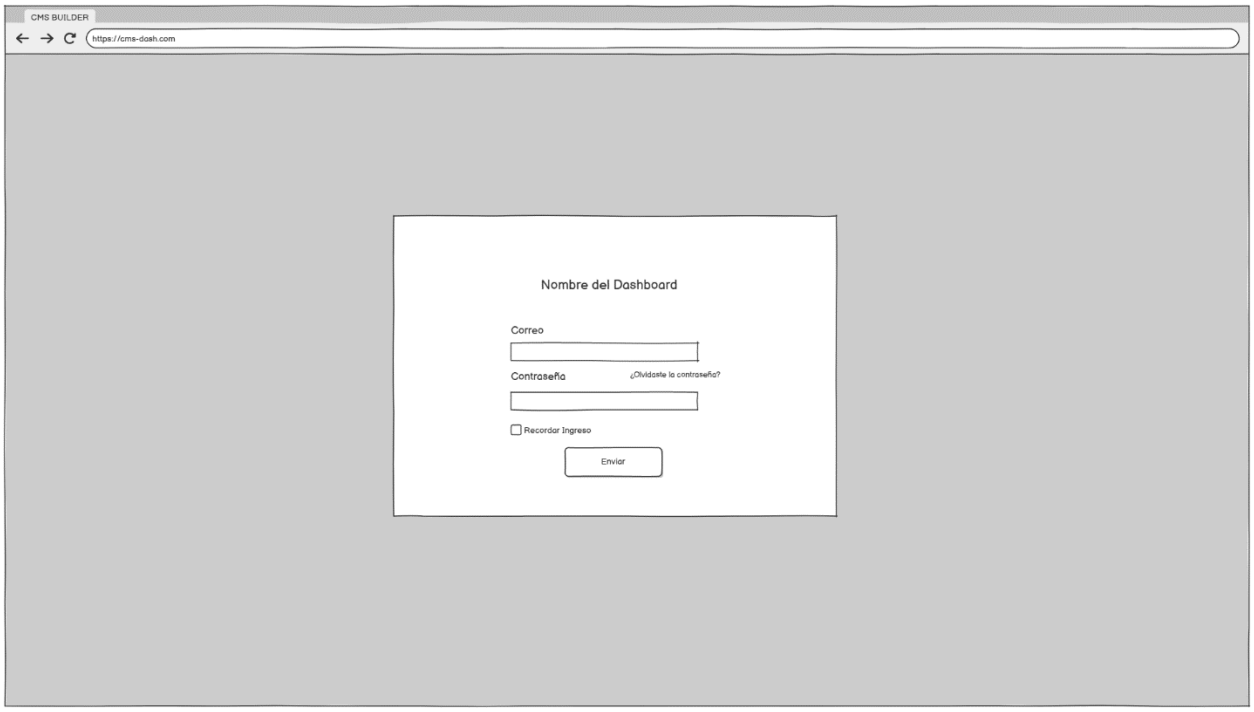
8. ANÁLISIS DE LA ESPECIFICACIÓN DE REQUISITOS DEL SOFTWARE

8.1ALTERNATIVAS DE SOLUCIÓN (PROTOTIPO O PROTOTIPOS DEL SISTEMA, MOCKUPS)

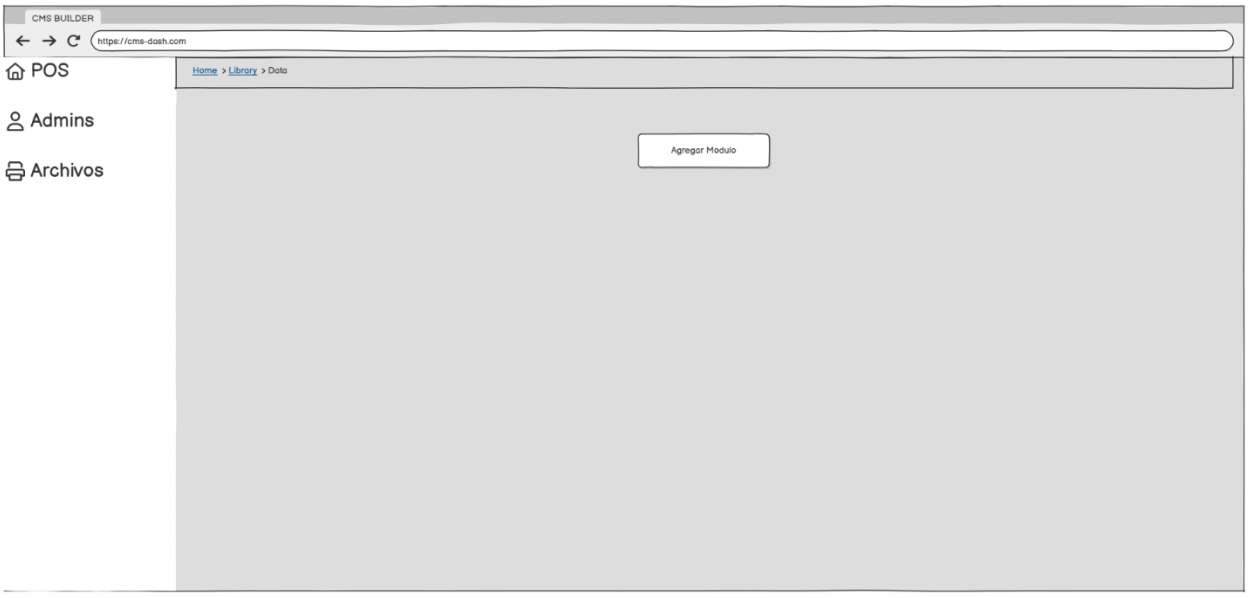
8.1.1 Instalador



8.1.2 Login

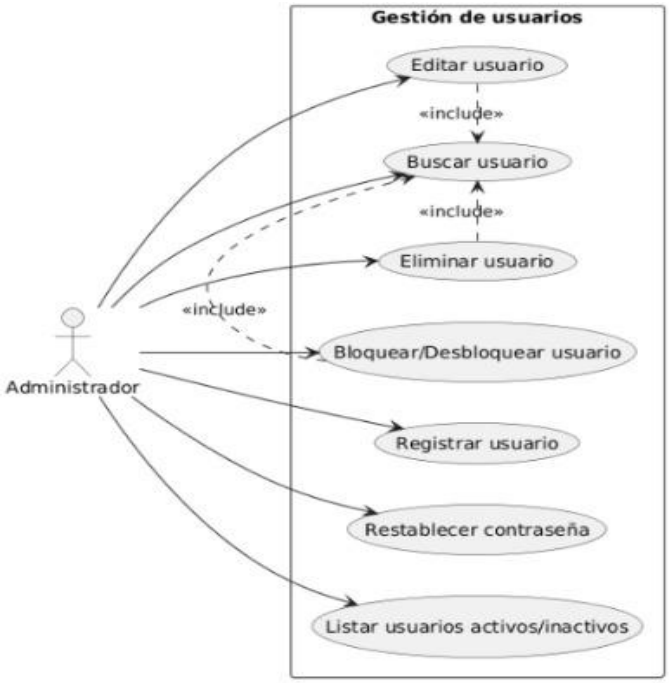


8.1.3 Dashboard

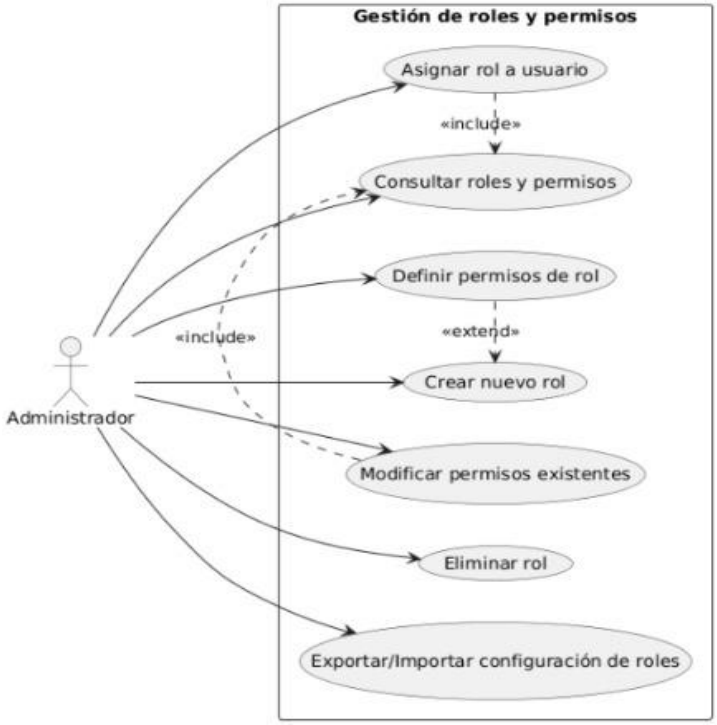


8.2 DIAGRAMA DE CASOS DE USO Y EXTENSIBILIDAD DE ESTOS

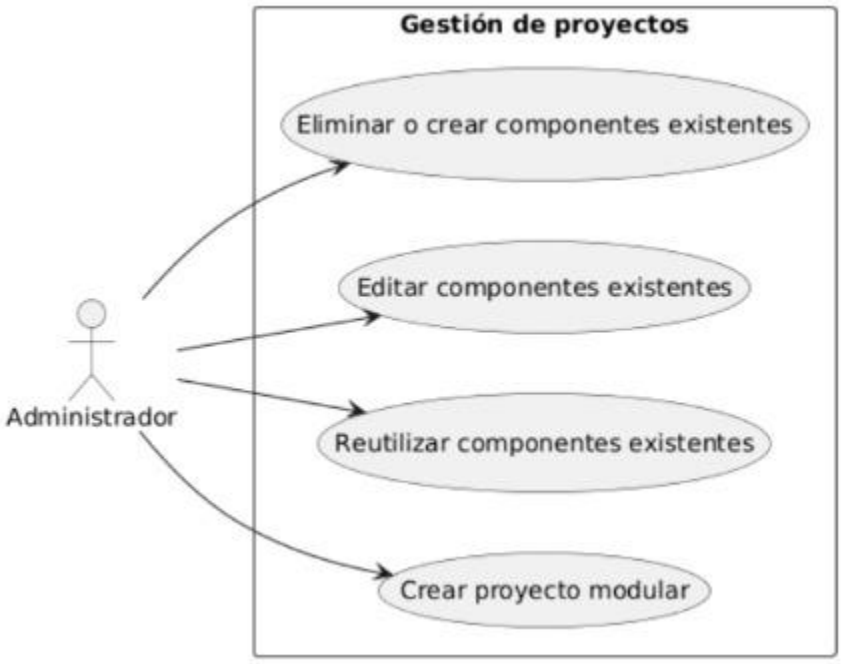
8.2.1 Gestion de usuarios



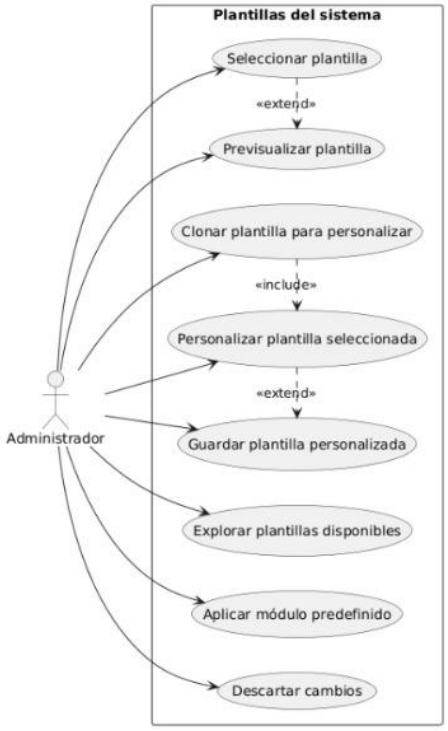
8.2.2 Gestion de usuarios



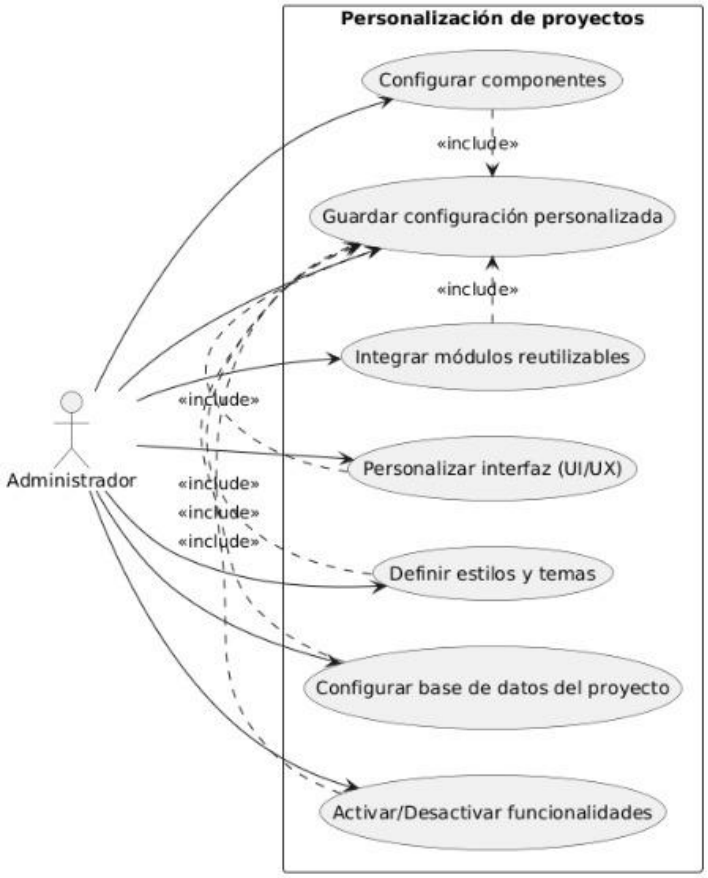
8.2.3 Gestion de Proyectos



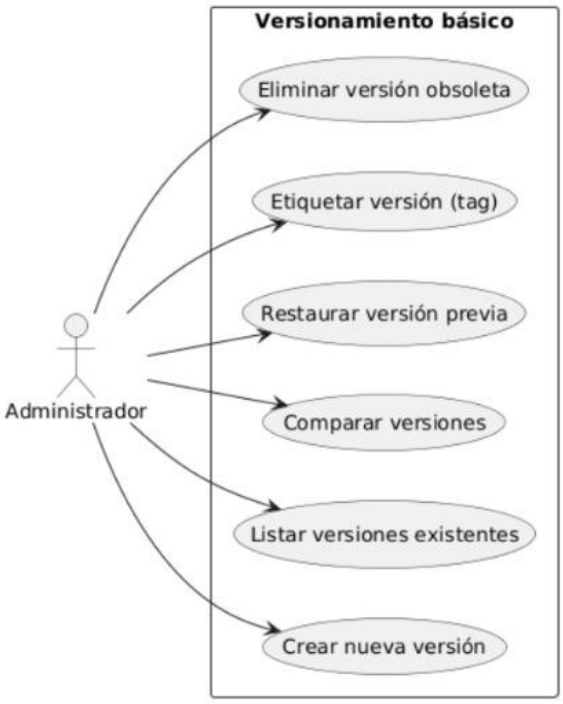
8.2.4 Plantillas del Sistema



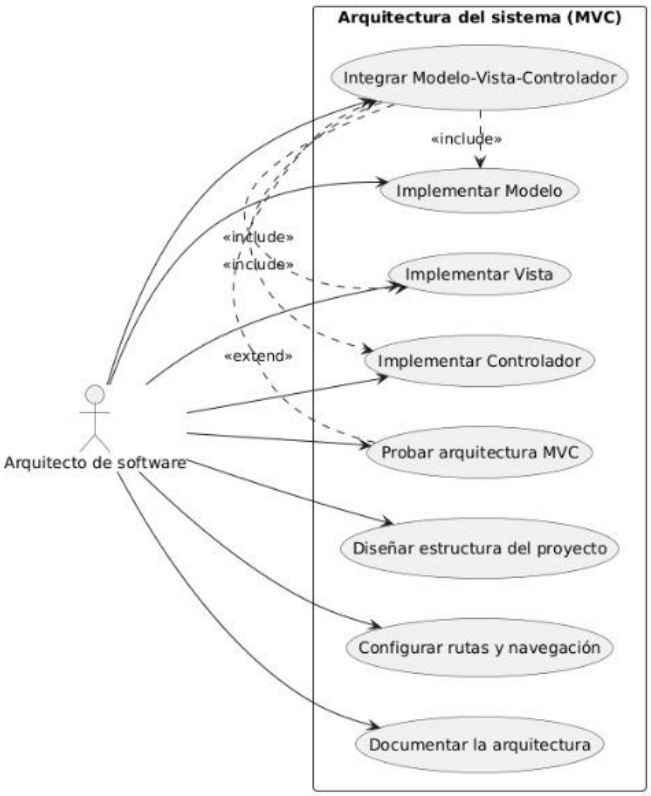
8.2.5 Personalizacion de proyectos



8.2.6 Versionamiento Basico

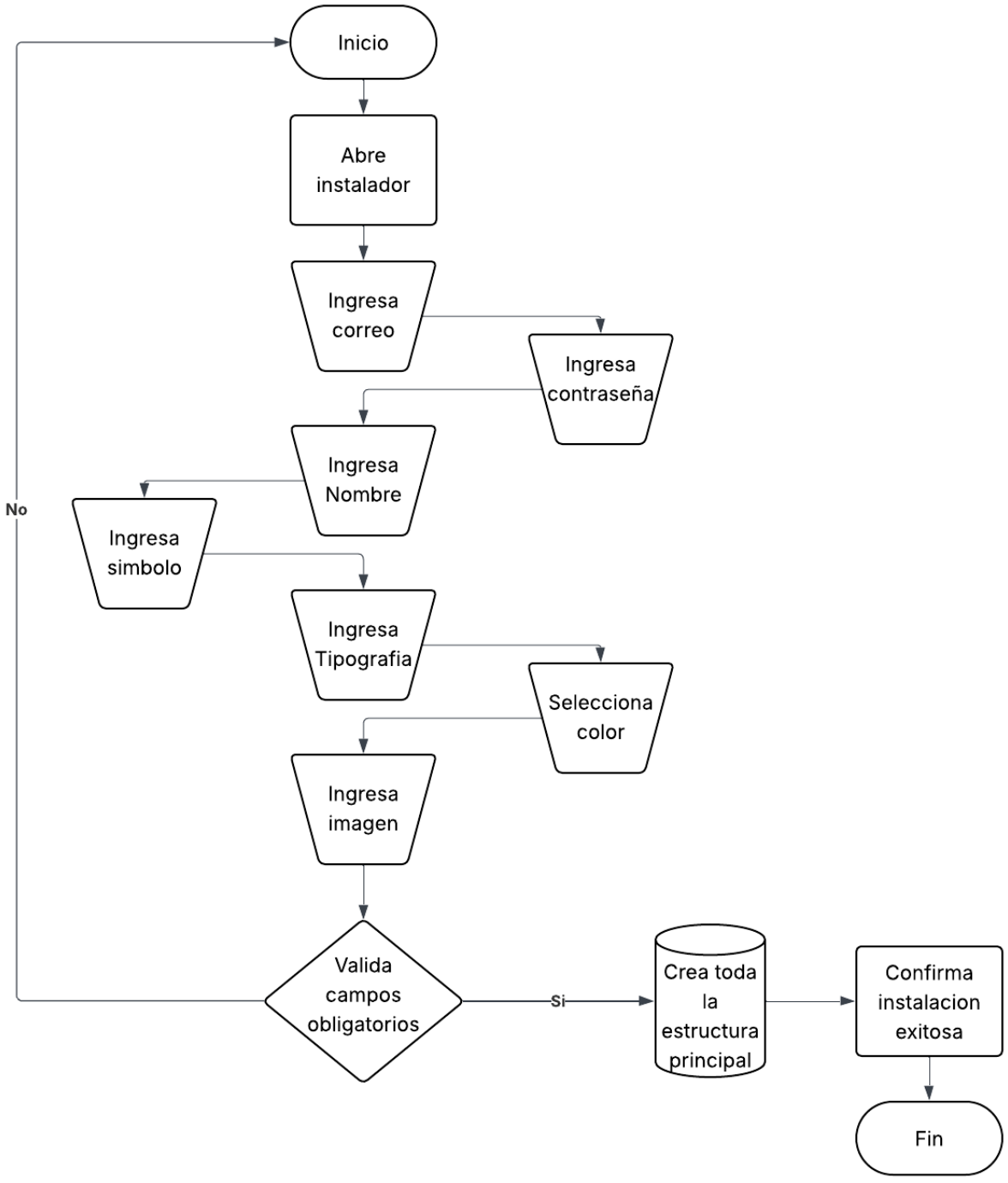


8.2.7 Arquitectura del sistema (MVC)

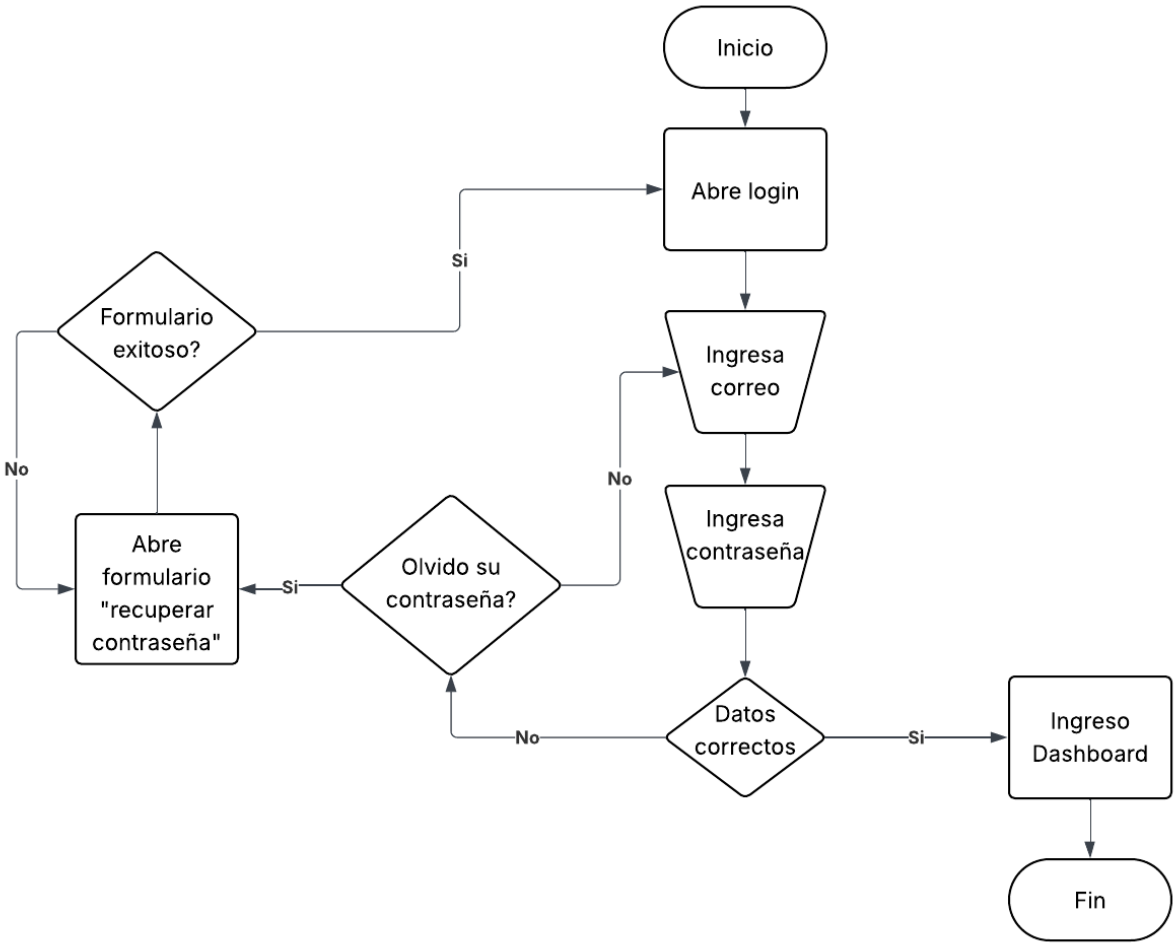


8.3 DIAGRAMAS DE ACTIVIDADES, Y SECUENCIAS

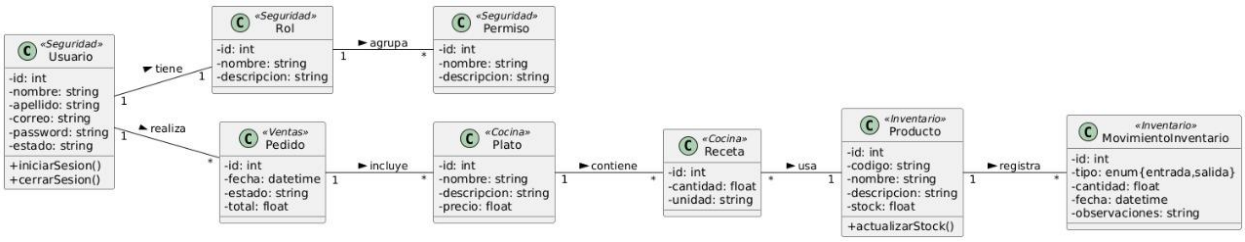
8.3.1 Instalación



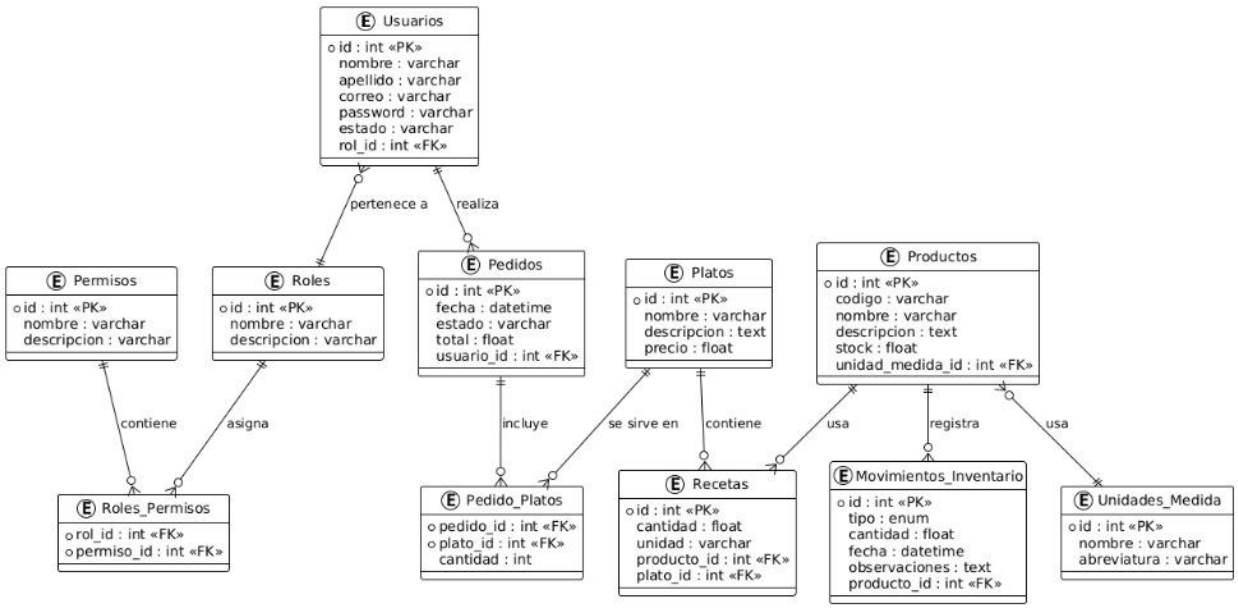
8.3.2 Login



8.4 CONSTRUIR EL MODELO DE DOMINIO DEL SISTEMA (DIAGRAMA DE CLASES).



8.5 ELABORAR EL MODELO ENTIDAD RELACIÓN.



9. DISEÑO DE LA SOLUCIÓN DEL SOFTWARE DE ACUERDO CON LOS PROCEDIMIENTOS Y REQUISITOS TÉCNICOS

9.1 ARQUITECTURA DEL SOFTWARE (DIAGRAMA DE COMPONENTES), Y PATRONES DE DISEÑO DE SOFTWARE.

Nuestro CMS está pensado para ser modular y fácil de mantener. Para lograrlo, usamos una arquitectura **Modelo-Vista-Controlador (MVC)**, lo que significa que la lógica, la información y la interfaz gráfica están separadas.

El controlador maneja la lógica (ejemplo: qué pasa cuando un usuario crea una página).

- El modelo se conecta con la base de datos.
- La vista es lo que el usuario ve en pantalla.

  Además, aplicamos patrones de diseño como:

- **Singleton**, que asegura que la conexión a la base de datos siempre sea única.
- **Factory**, para generar de forma más sencilla formularios o módulos.

9.2DIAGRAMA DE DESPLIEGUE.

9.3DISEÑO FRONT-END - INTERFACES GRÁFICAS DE USUARIO (HTML.CSS Y JS)

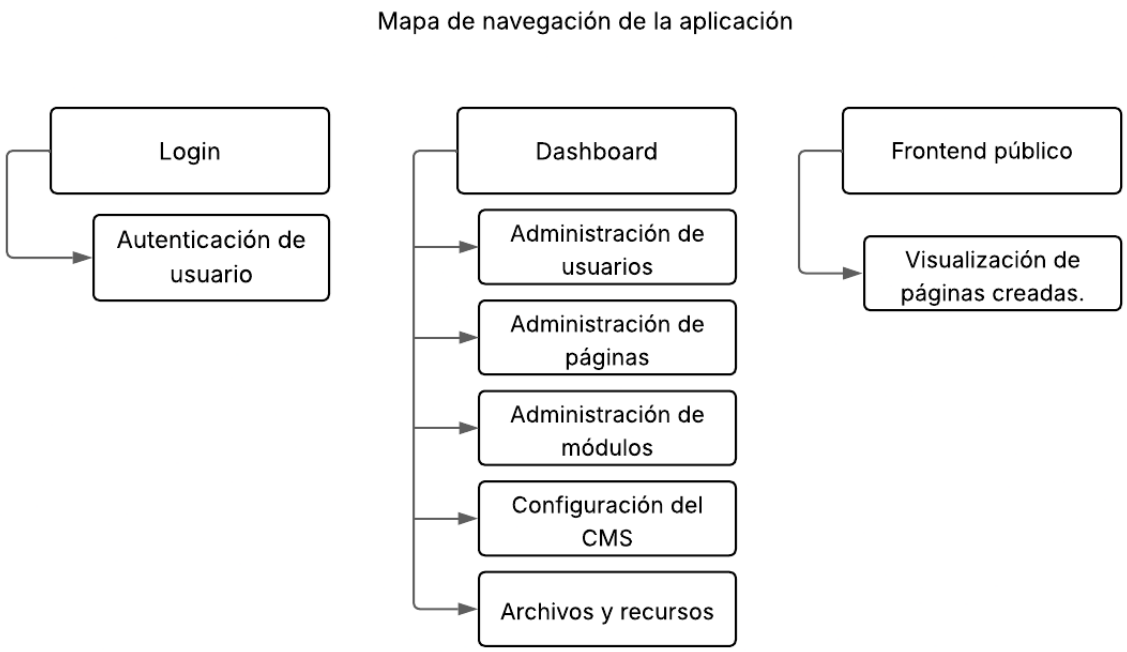
El aspecto visual está hecho con **HTML, CSS y JavaScript**, asegurando que el sistema sea sencillo de usar.

- HTML estructura las pantallas.
- CSS se encarga de los estilos, colores y distribución.
- JavaScript (junto con AJAX) da dinamismo, como actualizar tablas o cargar formularios sin necesidad de recargar toda la página.

9.4 INTERFACES GRÁFICAS DE USUARIO MÓVILES.

- El CMS está diseñado con un enfoque responsive, lo que significa que se adapta automáticamente a distintos tamaños de pantalla.
- Un administrador puede gestionar el contenido desde un celular, una tablet o un computador sin perder usabilidad.
- Incluso, en el futuro, este sistema podría convertirse en una aplicación móvil con tecnologías como PWA o Ionic.

9.4 MAPA DE Navegación



9.5 DETERMINAR TIPOS DE BASES DE DATOS.

- Para este CMS se utiliza principalmente **MySQL/MariaDB**, que es una base de datos relacional muy conocida y confiable.
- Además, se puede integrar un sistema de caché (como Redis) para acelerar las consultas en proyectos grandes.

9.6 MODELO DE DATOS DIAGRAMA ER (NORMALIZACIÓN BASE DE LA DATOS (Mínimo tercera forma normal)

9.7 DICCIONARIO DE DATOS.

9.7.1Tabla admins

Contiene la información de los administradores del sistema

Campo	*Tipo de dato*	*Descripción*
Id_admin	INT (PK, AI)	Identificador único del administrador.
email_admin	TEXT	Correo electrónico del administrador
password_admin	TEXT	Contraseña encriptada del administrador.
rol_admin	TEXT	Rol asignado (ej. superadmin, admin, editor).
permissions_admin	TEXT	Permisos del administrador en formato JSON.
token_admin	TEXT	Token de autenticación.
token_exp_admin	TEXT	Fecha de expiración del token.
status_admin	INT (1 = activo, 0 = inactivo)	Estado del administrador.
title_admin	TEXT	Título o nombre para mostrar del dashboard.
symbol_admin	TEXT	Símbolo del sistema.

font_admin	TEXT	Fuente utilizada en la interfaz.
color_admin	TEXT	Color para el sistema
back_admin	TEXT	Fondo personalizado.
scode_admin	TEXT	Código de seguridad adicional.
chatgpt_admin	TEXT	Configuración de integración con ChatGPT.
date_created_admin	DATE	Fecha de creación del registro.
date_updated_admin	TIMESTAMP	Última fecha de actualización.

9.7.2 Tabla pages

Almacena las páginas creadas

Campo	*Tipo de dato*	*Descripción*
id_page	INT (PK, AI)	Identificador único de la página.
title_page	TEXT	Nombre de la página.
url_page	TEXT	URL amigable de la página.
icon_page	TEXT	Ícono asociado a la página (Bootstrap Icons).

type_page	TEXT	Tipo de página (ej. modules, custom).
order_page	INT	Orden de aparición en el menú.
date_updated_page	Date	Fecha de creación.

9.7.3 Tabla Modules

Define los módulos que se insertan en cada página.

Campo	*Tipo de dato*	*Descripción*
id_module	INT (PK, AI)	Identificador único del modulo
id_page_module	TEXT	Página a la que pertenece el módulo.
type_module	TEXT	Tipo de módulo (ej. tables, breadcrumbs).
title_module	TEXT	Nombre del módulo.
suffix_module	TEXT	Sufijo identificador.
content_module	TEXT	Contenido del módulo.
width_module	INT (por defecto 100)	Ancho del módulo en % de la página.

editable_module	INT (1 = editable, 0 = no)	Indica si el módulo es editable.
date_created_module	Date	Fecha de creación.

9.7.4 Tabla Columns

Campo	*Tipo de dato*	*Descripción*
id_column	INT (PK, AI)	Identificador único de la columna.
id_module_column	INT (FK)	Módulo al que pertenece la columna.
title_column	TEXT	Nombre real del campo en la BD.
alias_column	TEXT	Nombre visible en la tabla.
type_column	TEXT	Tipo de dato mostrado (text, email, password, boolean, select, object).
matrix_column	TEXT	Opciones (en caso de tipo select).
visible_column	INT (1 = visible, 0 = oculto)	Indica si se muestra en la interfaz.
date_created_column	DATE	Fecha de creación.

date_updated_column	TIMESTAMP	Última fecha de actualización.

9.8 POLÍTICAS DE SEGURIDAD DE LOS DATOS.

El CMS garantiza la protección de la información mediante la aplicación de políticas de seguridad orientadas a preservar la **confidencialidad, integridad y disponibilidad** de los datos.

1. **Control de acceso**
    - Autenticación mediante credenciales únicas por usuario.
    - Políticas de contraseñas seguras (longitud mínima, complejidad y caducidad periódica).
    - Gestión de roles y permisos para limitar el acceso a información sensible.
2. **Protección de la información**
    - Encriptación de contraseñas y datos sensibles en reposo.
    - Uso de protocolos seguros (HTTPS/TLS) para la transmisión de datos.
    - Respaldo periódico de la base de datos y almacenamiento en entornos seguros.
3. **Monitoreo y auditoría**
    - Registro de actividades relevantes (accesos, modificaciones y operaciones críticas).
    - Análisis periódico de los registros para la detección de accesos no autorizados o comportamientos anómalos.
4. **Gestión de vulnerabilidades**
    - Actualización constante del software y dependencias del CMS.
    - Aplicación de parches de seguridad en el menor tiempo posible.
    - Evaluaciones periódicas de riesgos y pruebas de penetración.

5. **Disponibilidad y recuperación**
    - Implementación de planes de contingencia ante fallos críticos.
    - Procedimientos de recuperación de información en caso de pérdida o incidente de seguridad.

10. CONSTRUCCIÓN DEL SOFTWARE

10.1 Base de Datos para el Software a partir del Modelo de Datos

A partir del modelo entidad-relación planteado para el CMS, se diseñó la base de datos en un motor relacional (ejemplo: PostgreSQL o MySQL). La base de datos contempla las tablas principales:

Usuarios: almacena los datos de acceso y perfil de cada usuario.

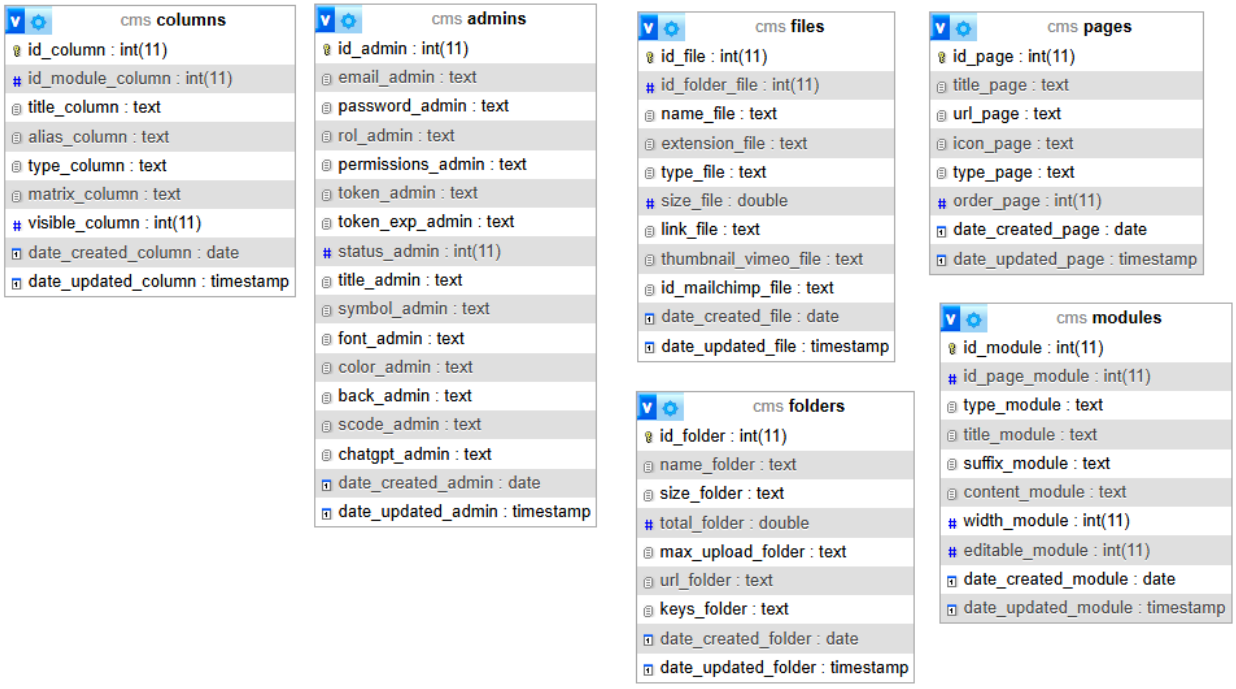
Roles: define los perfiles de uso (Administrador, Editor, Visitante).

Usuario Rol: tabla intermedia para relación N:M.

Configuraciones: guarda los parámetros iniciales definidos en la instalación (nombre del dashboard, tipografía, color, imagen de login, etc.).

Vistas: representa las páginas/plantillas gestionadas por el CMS.

Contenidos: almacena publicaciones, proyectos o recursos creados.



10.1.1 Objetos de la Base de Datos

Procedimientos almacenados

sp_crearUsuario(correo, contraseña, rol): registra un usuario con su rol.

sp_crearContenido(id_usuario, titulo, cuerpo): inserta un nuevo contenido ligado a un usuario.

sp_configuracionInicial(datos_instalador): guarda la configuración del dashboard en la instalación.

Vistas

vw_UsuariosConRoles: retorna usuarios con su(s) rol(es).

vw_ContenidoPublico: lista los contenidos marcados como públicos.

Disparadores (Triggers)

trg_logUsuario: al crear/modificar un usuario, se guarda un log de auditoría.

trg_actualizaFechaContenido: al actualizar un contenido, se cambia automáticamente la fecha de modificación.

10.1.2 Esquemas de Seguridad de los Datos

El CMS implementa un modelo de seguridad basado en roles y permisos:

Roles y privilegios

Administrador: acceso total (gestión de usuarios, contenidos y configuraciones).

Editor: creación y edición de contenidos y vistas.

Visitante: acceso limitado a vistas públicas.

Mecanismos de seguridad

Cifrado de contraseñas con algoritmos hash

Control de acceso basado en roles.

Validaciones en procedimientos almacenados para evitar inyecciones SQL.

Backups automáticos y restricciones de acceso a la base de datos por IP.

10.2 CODIFICACIÓN DEL SOFTWARE DE ACUERDO CON EL DISEÑO ESTABLECIDO.

10.2.1 Front-End (PHP, HTML, CSS, JavaScript, Responsive)

El front-end del CMS fue implementado principalmente en PHP, integrando código HTML5, CSS3 y JavaScript para la construcción de las vistas dinámicas. PHP se utiliza como lenguaje de servidor para:

- Generación dinámica de formularios e interfaces.
- Gestión de sesiones (login/logout).
- Renderizado de vistas del dashboard y módulos del sistema.
- Manejo de validaciones de campos en conjunto con JavaScript.

Las tecnologías utilizadas son:

- PHP 8+: procesamiento de lógica de negocio en el servidor y renderizado de vistas.
- HTML5: estructura semántica del contenido.
- CSS3: diseño visual con Grid y Flexbox para un layout responsivo.
- JavaScript (ES6+): validación de formularios, interactividad en dashboard y AJAX para comunicación asincrónica.
- Responsive Design: aplicación de *Media Queries* y pruebas en múltiples resoluciones.

Los componentes principales del Front-End:

Instalador: formulario dinámico para registrar administrador y personalizar dashboard.

Login: validación de credenciales con retroalimentación visual en caso de error.

Dashboard: panel administrativo con gestión de usuarios, contenidos y configuraciones.

Módulo de Vistas y Contenidos: interfaces amigables para la creación y edición.

10.2.2 Estándar de Codificación

Se adoptaron estándares de desarrollo PHP/HTML/JS para garantizar calidad y mantenibilidad:

1. Documentación del código

- Uso de PHPDoc para comentar funciones, clases y métodos.
- Comentarios estructurados en bloques para separar lógica de presentación (// y /* ... */).
- Convención de nombres en inglés y en *camelCase* ($userEmail, $dashboardName).

2. Buenas prácticas

- Separación entre lógica y presentación: uso de plantillas .php para vistas y controladores PHP para la lógica.
- Principio DRY (Don’t Repeat Yourself).

- Validaciones tanto en frontend (JS) como en backend (PHP).
- Manejo seguro de datos: sanitización con filter_input() y consultas preparadas (PDO o MySQLi).
- Uso de Composer para gestión de dependencias y librerías externas.

3. Aplicabilidad del estándar

- Todo código PHP deberá cumplir con el estándar PSR-12 (PHP-FIG).
- Revisión de código mediante *pull requests* antes de integración.
- Archivos organizados en arquitectura MVC

10.2.3 Código fuente de los módulos del software web y móvil

El código fuente del CMS se organiza en una arquitectura **MVC (Modelo-Vista-Controlador)**:

Web (PHP + HTML, CSS, JS):

/api/models/ → clases de acceso a datos y lógica de negocio (ej. User.php, Content.php).

/cms/controllers/ → controladores que procesan las peticiones HTTP (ej. AuthController.php, ContentController.php).

/cms/views/ → plantillas dinámicas con HTML/PHP para formularios, dashboard y login.

/cms/views/assets → archivos públicos (index.php, assets CSS/JS).

Móvil (Cliente de referencia en Flutter/React Native):

Estructura en módulos (pantallas: login, dashboard, contenidos).

Consumo de servicios REST del CMS a través de peticiones HTTP (JSON).

Cada módulo se implementa con código documentado en **PHPDoc** para el backend y **JSDoc** para scripts JavaScript.

10.2.4 Servicios Web

El CMS expone una **API RESTful** para la integración de aplicaciones externas (web o móviles).

Características:

- **Formato de datos:** JSON.
- **Autenticación:** JWT (JSON Web Tokens).

- **Métodos principales:**

RECURSO	METODO	DESCRIPCION
LOGIN	POST	Autenticación de Usuario
USUARIOS	GET	Listar usuarios.
USUARIOS	POST	Crear nuevo usuario.
CONTENIDOS	GET	Consultar contenidos.
CONTENIDOS	POST	Crear contenido.
CONTENIDOS/{ID}	PUT	Actualizar contenido.
CONTENIDOS/{ID}	DELETE	Eliminar contenido.

10.2.5 Control de versiones

El proyecto se administra con **Git** como sistema de control de versiones.

- **Repositorio:** GitHub/GitLab privado.
- **Ramas de trabajo:**
  - main: código estable y probado.
  - develop: rama activa para integración de nuevas funcionalidades.
  - feature/*: ramas específicas para nuevas características.
  - hotfix/*: correcciones urgentes sobre producción.
- **Estrategia:** *Git Flow*.
- **Integración continua (CI/CD):** mediante GitHub Actions o GitLab CI para ejecutar pruebas automáticas y despliegues controlados.

11. PLAN DE DESPLIEGUE (HOSTING, DOMINIO, SUBDOMINIO, COSTOS, HERRAMIENTAS, PLATAFORMA, SEGURIDAD)

- **Dominio**: entre **$40.000 – $80.000 COP/año**

- **Hosting VPS básico**: entre **$40.000 – $120.000 COP/mes**.

- **Certificado SSL**:

  Gratuito con Let's Encrypt.

  Pago: entre **$200.000 – $400.000 COP/año**.

- **Escalamiento en nube (AWS, Azure, GCP)**: desde **$200.000 – $600.000 COP/mes**, según el consumo de CPU, RAM y almacenamiento.

- **Herramientas adicionales** (monitorización, backups externos, correo empresarial, etc.): entre **$50.000 – $150.000 COP/mes**.

- **Ejemplo de escenario económico (mínimo viable)**

  Dominio .com → **$50.000 COP/año**.

  VPS de 2GB RAM / 50GB SSD (Hostinger o DigitalOcean) → **$60.000 COP/mes**.

  SSL gratuito con Let's Encrypt → **$0 COP**.

  Backups manuales → **incluidos en hosting**.

  - Costo mensual estimado: **$60.000 COP**.

    Costo anual total (incluyendo dominio): **~$770.000 COP**.

12. IMPLANTACIÓN DEL SOFTWARE.

12.1 PLAN DE IMPLANTACIÓN (PLATAFORMAS TECNOLÓGICAS)

El sistema se implementará en un entorno cliente-servidor web con las siguientes características:

- Servidor de aplicaciones
    - Apache o Ngix
    - PHP 8.x.
    - Extensiones PDO MySQL
- Servidor de base de datos
    - MySQL/MariaDB 10.x en un VPS o servicio gestionado (ej: AWS RDS, Azure MySQL, InfinityFree en fase inicial).
- Sistema operativo recomendado:
    - Ubuntu Server 22.04 LTS o Windows Server 2022.
- Plataforma de despliegue:
    - Opciones iniciales: InfinityFree (entorno de pruebas).
    - Escalamiento: AWS (EC2 + RDS) o DigitalOcean (Droplets + Managed DB).

- Herramientas de versionamiento y CI/CD:
    - GitHub/GitLab para control de versiones.
    - GitHub Actions o Jenkins para automatización de despliegues.

12.2 PLAN DE CAPACITACIÓN DE USUARIOS DEL SISTEMA.

El plan de capacitación de usuarios del sistema tiene como finalidad garantizar que todos los actores involucrados en el uso de la aplicación adquieran los conocimientos y habilidades necesarios para operar de manera eficiente, segura y autónoma las funcionalidades del software. La capacitación se enfocará en la práctica, mediante talleres guiados y materiales de apoyo, para asegurar la apropiación tecnológica y minimizar errores durante la operación diaria.

La capacitación se estructurará en tres fases:

Inducción general: Presentación de la interfaz, objetivos del sistema, principales módulos y flujos de trabajo. Se busca que el usuario se familiarice con la lógica general de la herramienta.

Capacitación funcional: Entrenamiento por roles, donde se abordan las tareas específicas que cada tipo de usuario debe realizar. Se incluyen ejercicios prácticos sobre registro, consultas, reportes y administración del sistema.

Soporte y retroalimentación: Etapa destinada a resolver dudas, registrar dificultades y reforzar los temas que presenten mayor complejidad, asegurando la transferencia del conocimiento y la adaptación a los procesos reales de la organización.

El plan contempla el uso de **manuales de usuario impresos y digitales**, tutoriales audiovisuales y sesiones presenciales o virtuales, según disponibilidad de los usuarios. Al finalizar, se aplicará una **evaluación de desempeño práctico** para medir el nivel de comprensión y uso efectivo del sistema, generando actas de asistencia y resultados que garanticen la trazabilidad de la capacitación.

12.3 MANUAL DEL USUARIO (PDF, MÓDULO AYUDA)

Encuéntrese en (link)

13. ADOPCIÓN DE BUENAS PRÁCTICAS EN EL PROCESO DE DESARROLLO DE SOFTWARE.

El CMS se desarrolla siguiendo un conjunto de **buenas prácticas de ingeniería de software** que permiten garantizar la calidad, seguridad, mantenibilidad y escalabilidad del sistema.

13.1 Gestión del Ciclo de Vida

- Uso de **metodologías ágiles** (Scrum/Kanban) para mejorar la iteración y la retroalimentación continua.
- Definición de **entregables incrementales** y planificación de sprints.
- Documentación de requerimientos y cambios mediante un sistema de gestión (Jira, Trello, GitHub Projects).

13.2 Control de Versiones

- Implementación de **Git** como sistema de control de versiones.
- Flujo de trabajo basado en ramas (main, develop, feature, hotfix).
- Uso de **pull requests y revisiones de código** para garantizar calidad en los cambios.

13.3 Calidad del Código

- Aplicación de **estándares de codificación** (PSR en PHP, PEP8 en Python, ESLint en JavaScript).

- Integración de herramientas de **análisis estático** (SonarQube, ESLint, PHPStan).
- Inclusión de **pruebas unitarias y de integración** en cada release.

13.4 Seguridad en el Desarrollo

- Principio de **mínimo privilegio** en el acceso a datos y recursos.
- Validación y sanitización de entradas para prevenir ataques de inyección.
- Uso de librerías seguras y actualizadas.
- Escaneo periódico de vulnerabilidades en dependencias.

13.5 Automatización y Despliegue

- Implementación de **CI/CD** para pruebas, compilación y despliegue automático.
- Uso de **contenedores Docker** para garantizar entornos consistentes.
- Integración de pipelines de seguridad (DevSecOps).

13.6 Documentación y Trazabilidad

- Documentación técnica y de usuario actualizada de manera iterativa.
- Registro de cambios en un **changelog** oficial.
- Uso de **diagramas de arquitectura** (UML, C4 Model) para mantener claridad en el diseño del sistema.

13.7 Monitoreo y Mantenimiento

- Monitoreo continuo del rendimiento y disponibilidad del sistema.
- Implementación de alertas para incidencias críticas.
- Políticas de mantenimiento preventivo y correctivo.

14. CONCLUSION

El desarrollo del Sistema de Gestión de Contenidos (CMS) permitió consolidar una solución tecnológica orientada a la optimización de procesos educativos digitales, integrando principios de escalabilidad, seguridad y usabilidad. A lo largo del proyecto se abordaron de manera

estructurada las etapas de análisis, diseño, construcción, despliegue e implantación, garantizando un producto adaptable a diferentes contextos institucionales y pedagógicos.

El CMS resultante no solo facilita la creación y administración de aplicaciones web mediante una interfaz intuitiva y modular, sino que también contribuye a la reducción de barreras técnicas, promoviendo la autonomía de los usuarios en la gestión de contenidos. Asimismo, la incorporación de buenas prácticas de desarrollo de software y metodologías ágiles asegura un entorno robusto, mantenible y preparado para futuras ampliaciones.

En conclusión, este proyecto constituye un aporte significativo a la transformación digital educativa, al proporcionar una herramienta estratégica que fortalece la innovación, optimiza los recursos tecnológicos y se alinea con las demandas actuales de la enseñanza-aprendizaje en entornos digitales.

15. GLOSARIO

Aplicativo Web

Programa o software accesible a través de un navegador, diseñado para ofrecer funcionalidades específicas en línea (ejemplo: sistemas de gestión académica, blogs, e-commerce).

Arquitectura MVC (Modelo-Vista-Controlador)

Patrón de diseño que separa la lógica del negocio (Modelo), la presentación (Vista) y la interacción del usuario (Controlador), lo que facilita la escalabilidad y el mantenimiento del software.

Backups (Respaldos)

Copias de seguridad de los datos y configuraciones del sistema, necesarias para restaurar la información en caso de fallos, ataques o pérdida de datos.

Base de Datos Relacional

Sistema estructurado de almacenamiento de datos que organiza la información en tablas relacionadas entre sí. En este proyecto se usan MySQL/MariaDB.

CMS (Content Management System / Sistema de Gestión de Contenidos)

Plataforma que permite crear, gestionar y modificar contenidos digitales sin necesidad de conocimientos avanzados en programación.

Control de Versiones

Mecanismo que registra y organiza los cambios realizados en el código o proyectos, permitiendo restaurar versiones anteriores y trabajar de manera colaborativa (ejemplo: Git/GitHub).

CRUD (Create, Read, Update, Delete)

Conjunto básico de operaciones que permiten la gestión de datos: crear, leer, actualizar y eliminar información en una base de datos.

Dominio Web

Nombre único que identifica un sitio en Internet (ejemplo: www.cmsprogramacion.com).

Escalabilidad

Capacidad de un sistema para crecer y adaptarse a una mayor demanda de usuarios, datos o funcionalidades sin perder rendimiento.

Framework

Conjunto de herramientas, bibliotecas y convenciones que facilitan el desarrollo de aplicaciones. Ejemplo: Laravel (PHP) o Express (Node.js).

Hosting

Servicio que provee la infraestructura para almacenar y ejecutar sitios web o aplicativos en Internet.

Interfaz de Usuario (UI)

Medio visual e interactivo mediante el cual el usuario se comunica con el sistema (pantallas, menús, formularios).

Metodologías Ágiles

Enfoques de desarrollo de software que promueven la colaboración, la flexibilidad y la entrega continua de resultados (ejemplo: Scrum, Kanban).

Módulo

Componente independiente y reutilizable dentro del CMS, que agrega funcionalidades específicas (ejemplo: módulo de usuarios, módulo de foros, módulo de login).

Patrón de Diseño

Solución general y reutilizable a un problema recurrente en el desarrollo de software. Ejemplo: Singleton, Factory, Observer.

Plantilla (Template)

Modelo prediseñado que sirve como base para crear aplicativos o páginas rápidamente, evitando comenzar desde cero.

Repositorio

Almacén digital donde se guarda el código fuente del proyecto, generalmente gestionado con herramientas de control de versiones como GitHub o GitLab.

Roles y Permisos

Sistema de control de acceso que define lo que cada usuario puede o no hacer dentro del CMS. Ejemplo: administrador, docente, estudiante.

Seguridad Informática

Conjunto de medidas para proteger los datos y procesos del sistema contra accesos no autorizados, ataques y pérdidas de información (ejemplo: cifrado de contraseñas, SSL).

Subdominio

Sección independiente de un dominio principal que organiza accesos o funcionalidades (ejemplo: admin.cmsprogramacion.com).

Usabilidad

Facilidad con la que un usuario puede interactuar con un sistema, comprendiendo su funcionamiento y cumpliendo sus tareas sin complicaciones.