# DISEÑO Y DESARROLLO DE SOFTWARE EDUCATIVO II

Control Versions

Yurgen Antonio Hoyos Aldana
José Carlos Negrete Hoyos
Johana Patricia Díaz Padilla

**Documento de Propuesta de Diseño de Software I, II y II**

Bienvenido a la documentación del Módulo de Control de Versiones. Este componente es una herramienta esencial para gestionar y rastrear cambios en el código fuente de un proyecto, facilitando la colaboración entre desarrolladores y asegurando la integridad y consistencia del software.

# Etapa 1 Diseño de la Aplicación y Análisis de Requisitos

# 1. Introducción

El Control de Versiones es un sistema que registra y controla las alteraciones realizadas en el código fuente de un proyecto a lo largo del tiempo. Permite a los desarrolladores trabajar en paralelo, fusionar cambios y revertir a versiones anteriores de manera eficiente. Además, proporciona un historial detallado de todos los cambios realizados, lo que resulta fundamental para la identificación y corrección de errores.

## Propósito del Documento

El presente documento tiene como finalidad documentar el proceso de diseño, análisis e implementación de software de tipo educativo, comercial, OVA, componente o módulo de aplicaciones.  Se divide en tres etapas para facilitar el entendimiento y aplicación a gran escala en la asignatura de diseño de software.

- Etapa 1 Diseño de la Aplicación y Análisis de Requisitos

Esta etapa cumple la tarea de recoger todas las competencias desarrolladas en todas las áreas de formación del currículo de la licenciatura en Informática y Medios Audiovisuales y ponerlas a prueba en el diseño y análisis de un producto educativo que se base en las teorías de aprendizaje estudiadas, articule las estrategias de enseñanza con uso de TIC y genere innovaciones en educación con productos interactivos que revelen una verdadera naturaleza educativa. Estos productos deben aprovechar las fortalezas adquiridas en las áreas de

tecnología e informática, técnicas y herramientas, medios audiovisuales y programación y sistemas, para generar productos software interactivos que permitan a los usuarios disfrutar de lo que aprenden, a su propio ritmo. Todo esto en el marco de un proceso metodológico(metodologías de desarrollo de software como MODESEC, SEMLI, etc.) que aproveche lo aprendido en la línea de gestión y lo enriquezca con elementos de la Ingeniería de Software.

- Etapa 2: Persistencia de Datos con Backend – Servidor

En la etapa 2 se continúa con los lineamientos de la etapa 1, para seguir adicionando elementos de diseño e implementación de software, enfocados en el desarrollo de APIs, servidores o microservicios que permitan soportar aplicaciones cliente del software educativo; en este sentido, el curso presenta los conceptos de los sistemas de bases de datos, su diseño lógico, la organización de los sistemas manejadores de bases de datos, los lenguaje de definición de datos y el lenguaje de manipulación de datos SQL y NoSQL; de tal manera que los estudiantes adquieran las competencias para analizar, diseñar y desarrollar aplicaciones para gestionar y almacenar grandes cantidades de datos, mediante el uso de técnicas adecuadas como el diseño y modelo lógico y físico de base datos, manejo de los sistemas de gestión de bases de datos, algebra relacional, dominio del lenguaje SQL como herramienta de consulta, tecnología cliente / servidor; igualmente, se definirán los elementos necesarios para el acceso a dichas bases de datos, como la creación del servidor API, utilizando tecnologías de vanguardia como node.js, express, Nest.js, Spring entre otros; para, finalmente converger en el despliegue de la API utilizando servicios de hospedaje en la nube, preferiblemente gratuitos. También podrá implementar servidores o API’s con inteligencia artificial o en su defecto crear una nueva capa que consuma y transforme los datos obtenidos de la IA.

El desarrollo del curso se trabajará por proyectos de trabajo colaborativo que serán evaluados de múltiples maneras, teniendo en cuenta más el proceso que el resultado.

- Etapa 3: Consumo de Datos y Desarrollo Frontend – Cliente

La etapa 3 el estudiante está en capacidad de establecer la mejor elección de herramientas de consumo de datos y técnicas en aras de lograr el mejor producto a nivel de software o hardware acorde a los requerimientos funcionales y no funcionales del problema a solucionar. En este punto el estudiante puede consumir los datos a través de un cliente que puede ser una aplicación

de celular, una aplicación de escritorio, una página web, IoT(internet de las cosas) o incluso, artefactos tecnológicos.

El diseño gráfico es de los requisitos esenciales en la capa de presentación, por lo tanto, se requieren los cursos de diseño gráfico vistos previamente. Los elementos anteriores nos permiten elegir el paradigma y tecnología para desarrollar nuestras aplicaciones, teniendo en cuenta que podríamos desarrollar aplicaciones de tipo cliente.

Esta documentación tiene como objetivo proporcionar una guía completa para el uso efectivo del Módulo de Control de Versiones en el contexto de nuestro proyecto. Aquí encontrarás instrucciones detalladas sobre la configuración inicial, comandos básicos, estrategias de ramificación y fusiones, entre otros aspectos fundamentales. El propósito de la documentación del Módulo de Control de Versiones es proporcionar a los usuarios una guía completa y detallada sobre cómo utilizar eficazmente esta herramienta en el contexto de su proyecto de desarrollo de software

## Alcance del Proyecto del módulo de control de versiones

El alcance del proyecto abarca el diseño, desarrollo e implementación del Módulo de Control de Versiones, una herramienta integral para la gestión y seguimiento de cambios en el código fuente de los proyectos de software. Este módulo se integrará en el flujo de trabajo de desarrollo de la organización, permitiendo a los equipos colaborar de manera eficiente y mantener un registro detallado de todas las modificaciones realizadas en el código.

## Definiciones y Acrónimos

● Repositorio: Un espacio donde se almacena y gestiona el código fuente de un proyecto.

● Commit (Confirmación): Un conjunto de cambios realizados en el código fuente que se registra en el historial del repositorio.

● Rama (Branch): Una línea independiente de desarrollo que permite trabajar en funcionalidades o correcciones sin afectar la rama principal.

● Fusión (Merge): La integración de los cambios de una rama en otra, combinando el código de ambas.

● Control de Versiones (Version Control): Un sistema que registra y gestiona cambios en el código fuente de un proyecto a lo largo del tiempo.

● API: Interfaz de Programación de Aplicaciones (Application Programming Interface).

● DBMS: Sistema de Gestión de Bases de Datos (Database Management System).

● SQL: Lenguaje de Consulta Estructurada (Structured Query Language).

● HTTP: Protocolo de Transferencia de Hipertexto (Hypertext Transfer Protocol).

● REST: Transferencia de Estado Representacional (Representational State Transfer)

● JSON: Notación de Objetos de JavaScript (JavaScript Object Notation).

● JWT: Token de Web JSON (JSON Web Token).

● CRUD: Crear, Leer, Actualizar y Borrar (Create, Read, Update, Delete). ORM: Mapeo Objeto-Relacional (Object-Relational Mapping). 9

● MVC: Modelo-Vista-Controlador (Model-View-Controller). API RESTful: API que sigue los principios de REST.

● CI/CD: Integración Continua / Entrega Continua (Continuous Integration / Continuous Delivery).

● SaaS: Software como Servicio (Software as a Service).

● SSL/TLS: Capa de sockets seguros/Seguridad de la Capa de Transporte (Secure Sockets Layer/Transport Layer Security).

● HTML: Lenguaje de Marcado de Hipertexto (Hypertext Markup Language).

● CSS: Hojas de Estilo en Cascada (Cascading Style Sheets).

● JS: JavaScript.

● DOM: Modelo de Objeto del Documento (Document Object Model).

● UI: Interfaz de Usuario (User Interface).

● UX: Experiencia del Usuario (User Experience).

● SPA: Aplicación de Página Única (Single Page Application).

● AJAX: Asincrónico JavaScript y XML (Asynchronous JavaScript and XML).

● CMS: Sistema de Gestión de Contenido (Content Management System).

● CDN: Red de Distribución de Contenido (Content Delivery Network).

● SEO: Optimización de Motores de Búsqueda (Search Engine Optimization).

● IDE: Entorno de Desarrollo Integrado (Integrated Development Environment).

● CLI: Interfaz de Línea de Comandos (Command Line Interface).

● PWA: Aplicación Web Progresiva (Progressive Web App

# 2. Descripción General

## Objetivos del Sistema

El módulo de Control de Versiones tiene como objetivo principal proporcionar una plataforma robusta para la gestión y seguimiento de cambios en el código fuente de proyectos de software. Busca facilitar la colaboración eficiente entre equipos de desarrollo, mantener un registro detallado de modificaciones, optimizar el flujo de trabajo mediante la creación, gestión y fusión de ramas, promover la integridad y disponibilidad del código, facilitar la revisión y aprobación de código, posibilitar la etiquetación y versionado de releases, integrarse con sistemas de despliegue continuo y fomentar buenas prácticas de desarrollo colaborativo

## Funcionalidad General

- Crear y gestionar múltiples ramas de desarrollo para facilitar la colaboración en paralelo.
- Realizar commits detallados, registrando información sobre los cambios, como autor y fecha.
- Visualizar el historial completo de cambios para identificar y resolver problemas.
- Fusionar ramas de manera controlada y resolver conflictos de manera eficiente.
- Etiquetar y versionar releases siguiendo prácticas de versionado semántico.
- Facilitar la revisión y aprobación de código para mejorar la calidad y seguridad.
- Optimizar el flujo de trabajo de desarrollo y garantizar la integridad del código fuente.

## Restricciones

Se debe implementar un riguroso sistema de autenticación y autorización para garantizar que solo usuarios autorizados tengan acceso al sistema. Además, se deben tomar medidas para proteger la integridad y confidencialidad de los datos almacenados en el sistema de Control de Versiones. Esto incluye la encriptación de la información y la adopción de buenas prácticas de seguridad.

## Usuarios del sistema

| funcionalidad | docente | alumno | administrador |
|---|---|---|---|
| Save new version | | | |
| Get history versions | | | |
| Get previous version | | | |
| Get next version | | | |
| Apply versión to source | | | |
| Get initial version | | | |
| Get latest version | | | |
| Apply initial version | | | |
| Apply latest version | | | |

# 3. Requisitos funcionales

**Gestión de Versiones:**

- Permitir la creación y guardado de nuevas versiones.
- Facilitar el acceso al historial completo de versiones para un seguimiento detallado.

**Comparación de Versiones:**

- Proporcionar herramientas para comparar diferentes versiones de un documento o proyecto.
- Permitir la visualización de cambios específicos entre versiones.

**Acceso y Navegación:**

- Facilitar la recuperación de versiones anteriores y siguientes.

- Ofrecer una interfaz de fácil navegación para explorar el historial de versiones.

**Aplicación Selectiva de Versiones:**

- Permitir la aplicación selectiva de cambios a partir de versiones específicas.
- Garantizar la integración sin problemas de versiones seleccionadas en el trabajo actual.

**Restauración y Reversión:**

- Habilitar la restauración rápida de versiones anteriores.
- Permitir la reversión a una versión específica si es necesario.

**Notificaciones y Comentarios:**

- Facilitar la comunicación entre estudiantes, profesores y administradores sobre cambios en las versiones.
- Posibilitar la adición de comentarios a versiones específicas para contextualizar los cambios.

**Seguridad y Privacidad**:

- Garantizar la seguridad de las versiones almacenadas.
- Gestionar los permisos de acceso para diferentes roles de usuarios.

**Exportación e Importación**:

- Ofrecer la posibilidad de exportar e importar versiones para facilitar la colaboración.

**Integración con Otras Herramientas:**

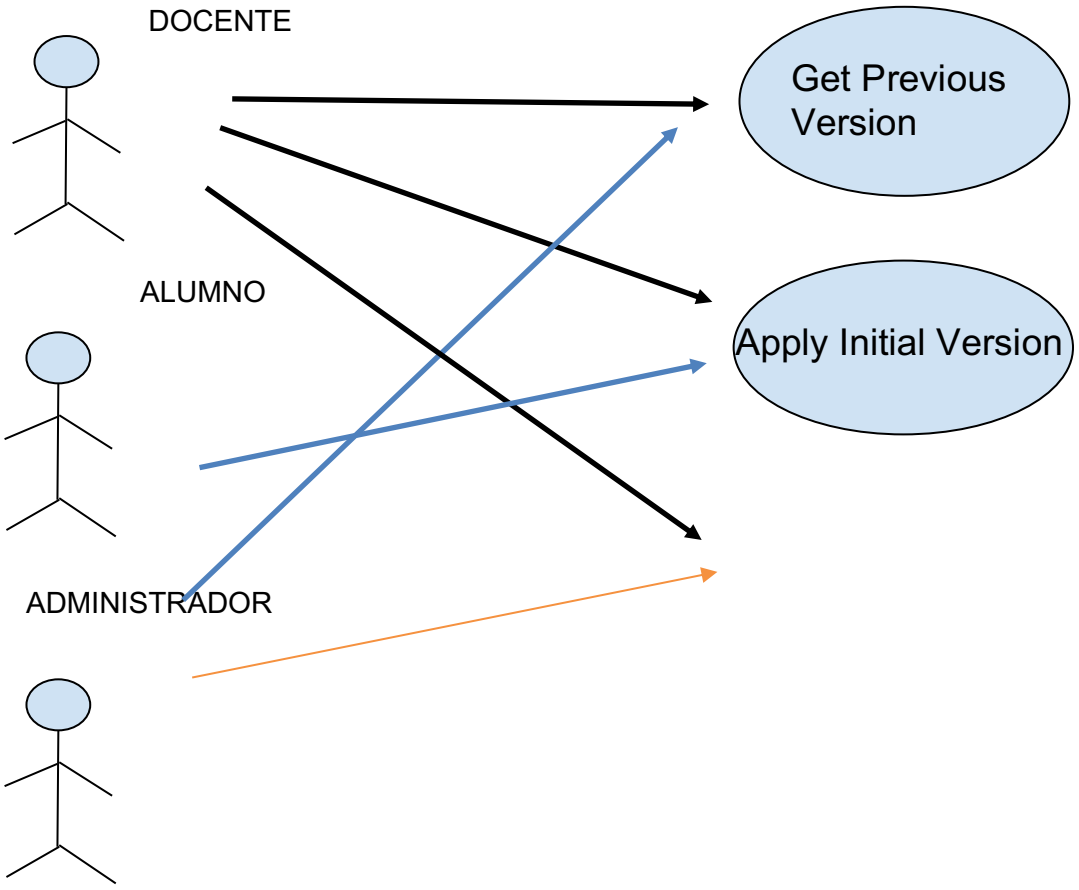
- Integrarse con herramientas educativas existentes en el entorno, como plataformas de aprendizaje o sistemas de gestión escolar.

**Registro de Actividades:**

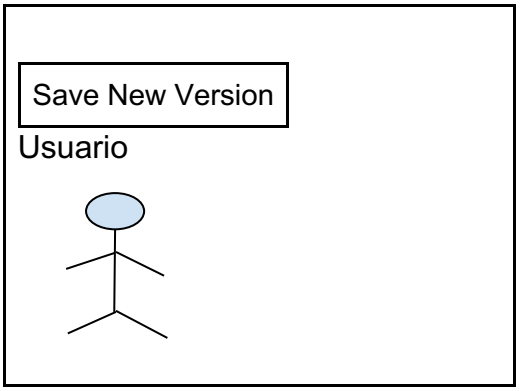
- Mantener un registro detallado de todas las acciones realizadas en relación con las versiones.

# CASOS DE USO

**CONTROL VERSIONS**




CASO DE USO 1
SAVE NEW VERSION



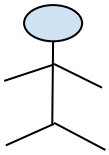

1.Save New Version

Flujo: Guardar nueva versión
+

Prioridad: Alta
Complejidad: Alta

CASO DE USO 2
GET VERSIONS

Get Versions



Administrador
Docente
Alumno

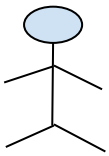
Prioridad: Requerido
Versión: Indefinida(?)
Tamaño: Mediano
Complejidad: Media

2.Get Versions

Flujo: Obtener versiones

CASO DE USO 3
APPLY VERSION

Apply Version



Administrador
Docente
Alumno

Prioridad: Requerido
Versión: Indefinida(?)
Tamaño: Mediano
Complejidad: Media

5.Apply Version
Flujo: Aplicar la version

**CASO No. 1 SAVE NEW VERSION**

| | | |
|---|---|---|
| **ID** | CU#1 | |
| **Nombre** | Save New version | |
| **Actores** | Usuario (Docente, alumno, administrador) | |
| **Objetivo** | Guardar una nueva versión de un objeto JSON en el sistema de control de versiones (CVS). | |
| **Urgencia** | 5 | |
| **Esfuerzo** | 4 | |
| **Precondiciones** | ● El usuario debe estar autenticado.<br>● El usuario debe tener permisos para crear. | |
| **Flujo básico** | **Usuario** | **Sistema** |
| | Guarda un nuevo contenido. | |
| | | Selecciona el objeto JSON que se desea guardar. |
| | | El sistema de control de versiones almacena el objeto de orígen en el sistema de control de versiones. |
| | | El sistema de control de versiones actualiza el historial de versiones del objeto JSON con la nueva versión. |
| | | El sistema de control de versiones retorna el estado de la nueva versión |
| **Postcondiciones** | ● El objeto JSON de destino tiene una versión guardada en el sistema de control de versiones.<br>● El historial de versiones del objeto JSON se ha actualizado con la nueva versión.<br>● Se debe retorna el estado actual de la versión | |

**CASO No. 2 GET VERSIONS**

| | | |
|---|---|---|
| **ID** | CU#2 | |
| **Nombre** | Get Versions | |
| **Actores** | Usuario (Docente, alumno, administrador) | |
| **Objetivo** | Obtener la lista de versiones del historial disponibles para un objeto en el sistema de control de versiones. | |
| **Urgencia** | 4 | |
| **Esfuerzo** | 5 | |
| **Precondiciones** | ● Debe estar autenticado en el sistema.<br>● Debe tener acceso al objeto del cual desea obtener el historial de versiones.<br>● Debe haber al menos dos versiones registradas del objeto en el sistema. | |
| **Flujo básico** | **Usuario** | **Sistema** |
| | Presiona el icono en el objeto de obtener el historial de versiones. | |
| | | El sistema de control de versiones presenta al sistema la lista de versiones para ese objeto. |
| | Visualiza información de las propiedades de cada versión y se resaltan los cambios entre versiones. | |
| **Flujo alternativo** | ● El usuario selecciona el objeto para el cual desea obtener el historial de versiones.<br>● No se muestra el icono informativo indicando que hay versiones disponibles. | |
| **Postcondiciones** | ● El usuario obtiene la lista de versiones disponibles para el objeto seleccionado.<br>● El sistema de control de versiones presenta las propiedades del historial de versiones. | |

| | |
|---|---|

**CASO No. 3 APPLY VERSIONS**

| | | |
|---|---|---|
| **ID** | CU#3 | |
| **Nombre** | Apply Versions | |
| **Actores** | Usuario (Docente, administrador) | |
| **Objetivo** | Aplicar una versión específica de un objeto a la fuente actual en el sistema de control de versiones. | |
| **Urgencia** | 4 | |
| **Esfuerzo** | 4 | |
| **Precondiciones** | ● El usuario debe estar autenticado en el sistema de control de versiones.<br>● El usuario debe tener acceso y permisos para editar la fuente a la que se aplicará la versión.<br>● Deben existir versiones registradas del objeto en el sistema. | |
| **Flujo básico** | **Usuario** | **Sistema** |
| | CU#2 | |
| | Selecciona la versión específica que desea aplicar a la fuente actual. | |
| | Aplica la versión seleccionada | |
| | | El sistema de control de versiones actualiza la versión fuente. |
| **Flujo alternativo** | ● El usuario no aplica la versión seleccionada.<br>● El sistema de control de versiones mantiene la versión fuente. | |
| **Postcondiciones** | ● La fuente se actualiza con los cambios de la versión especificada por el usuario.<br>● El sistema de control de versiones registra la acción de aplicar la versión en el historial del objeto.<br>● El usuario recibe una confirmación de que la versión se ha aplicado con éxito. | |

| | URGENCIA | | | | | |
|---|---|---|---|---|---|---|
| IMPACTO | | 1-Baja | 2-Menor | 3-Moderada | 4-Alta | 5-Obligatoria |
| | 5-Muy alto | | | | | CU#1 |
| | 4-Alto | | | | CU#3 | CU#2 |
| | 3-Medio | | | | | |
| | 2-Bajo | | | | | |
| | 1-Muy bajo | | | | | |

# 4. REQUISITOS NO FUNCIONALES

**Rendimiento:**

- La plataforma debe ser eficiente incluso con grandes cantidades de datos y usuarios simultáneos.

**Escalabilidad:**

- Debería ser capaz de manejar un aumento gradual en el número de usuarios y datos sin degradación significativa del rendimiento.

**Disponibilidad:**

- El sistema debe estar disponible de manera confiable durante períodos críticos, como épocas de presentación de proyectos o evaluaciones.

**Seguridad:**

- Garantizar la seguridad de los datos almacenados, con medidas como cifrado y autenticación adecuados.

**Compatibilidad:**

- Ser compatible con diversos navegadores y dispositivos para asegurar un acceso sin problemas.

**Usabilidad:**

- Proporcionar una interfaz de usuario intuitiva y fácil de usar para usuarios con diversos niveles de habilidad técnica.

**Mantenimiento:**

- Facilitar actualizaciones y mantenimiento sin afectar la continuidad del servicio.

**Cumplimiento Normativo:**

- Cumplir con regulaciones de privacidad y seguridad de datos pertinentes al ámbito educativo.

**Documentación:**

- Disponer de documentación clara y completa para usuarios y administradores.

**Tiempo de Respuesta**:

- Garantizar tiempos de respuesta rápidos para la visualización de versiones y realización de acciones relacionadas.

**Costo:**

- Mantener costos operativos y de mantenimiento dentro de los límites establecidos.

**Adaptabilidad:**

- Ser capaz de adaptarse a cambios en las tecnologías y requerimientos educativos con el tiempo.

**Integración:**

- Integrarse de manera eficiente con otras aplicaciones y sistemas utilizados en el entorno educativo.

**Resiliencia:**

- Ser resistente a fallos para evitar pérdida de datos o interrupciones prolongadas.

## Requisitos de Desempeño:

1. **Tiempo de Respuesta:**
    - La plataforma debe proporcionar tiempos de respuesta rápidos al acceder a versiones y realizar acciones relacionadas.
2. **Escalabilidad:**
    - La aplicación debe ser capaz de manejar eficientemente un aumento en el número de usuarios y la cantidad de datos almacenados.
3. **Carga Concurrente**:
    - Debería ser capaz de manejar múltiples usuarios realizando operaciones concurrentes sin degradación significativa del rendimiento.
4. **Optimización de Búsquedas:**
    - Las búsquedas y recuperación de versiones específicas deben ser optimizadas para mejorar la eficiencia.
5. **Gestión de Grandes Volúmenes de Datos:**
    - Debe manejar grandes volúmenes de datos de versiones sin afectar negativamente el rendimiento.

## Requisitos de Seguridad:

1. **Cifrado:**
    - Implementar cifrado para proteger la confidencialidad de los datos almacenados, especialmente versiones anteriores.
2. **Autenticación y Autorización:**
    - Garantizar un sólido sistema de autenticación y autorización para controlar el acceso a diferentes versiones según los roles definidos.
3. **Auditoría y Registro de Actividades:**
    - Mantener registros detallados de todas las actividades relacionadas con las versiones para facilitar la auditoría y la identificación de posibles problemas de seguridad.
4. **Resiliencia ante Ataques:**

- Implementar medidas de seguridad para resistir intentos de ataques, como inyección de código o intentos de manipulación de datos.

5. **Gestión de Vulnerabilidades:**

   - Realizar análisis de vulnerabilidades periódicos y aplicar parches de seguridad de manera oportuna.

6. **Políticas de Contraseñas:**

   - Establecer políticas de contraseñas robustas y fomentar su uso entre los usuarios.

7. **Restricciones de Acceso Físico y Lógico**:

   - Limitar el acceso físico y lógico a los servidores que almacenan las versiones.

8. **Respaldo y Recuperación:**

   - Implementar un sistema de respaldo regular y eficiente para garantizar la disponibilidad de datos incluso en casos de incidentes de seguridad.

9. **Cumplimiento Normativo:**

   - Asegurar que la plataforma cumple con las regulaciones de privacidad y seguridad de datos pertinentes al entorno educativo.

10. **Capacitación en Seguridad:**

    - Proporcionar capacitación en seguridad para usuarios y administradores, fomentando prácticas seguras en el manejo de versiones.

11. **Gestión de Sesiones:**

    - Implementar gestión de sesiones segura para prevenir accesos no autorizados.

## Requisitos de Usabilidad:

1. **Interfaz Intuitiva:**

   - La interfaz de usuario debe ser intuitiva y fácil de usar, incluso para usuarios con niveles variados de habilidades técnicas.

2. **Navegación Sencilla**:

   - La plataforma debe proporcionar una navegación clara y lógica, facilitando a los usuarios encontrar y gestionar versiones de manera eficiente.

3. **Instrucciones y Ayuda Contextual:**

- Ofrecer instrucciones claras y ayuda contextual para guiar a los usuarios en la utilización de las funciones del control de versiones.

4. **Personalización de la Interfaz:**

    - Permitir a los usuarios personalizar su experiencia, como la visualización de información relevante y la configuración de preferencias.

5. **Compatibilidad con Dispositivos y Navegadores:**

    - Asegurarse de que la plataforma sea compatible con una variedad de dispositivos y navegadores para una accesibilidad óptima.

6. **Retroalimentación Inmediata:**

    - Proporcionar retroalimentación inmediata sobre acciones realizadas, como la confirmación de la creación de nuevas versiones o la aplicación de cambios.

7. **Accesibilidad:**

    - Cumplir con estándares de accesibilidad para garantizar que la plataforma sea utilizada por personas con discapacidades.

8. **Capacitación y Recursos:**

    - Ofrecer recursos de capacitación, tutoriales y documentación para ayudar a los usuarios a aprovechar al máximo el control de versiones.

## Requisitos de Escalabilidad:

1. **Manejo Eficiente de Datos:**

    - Escalar de manera eficiente al manejar grandes cantidades de datos relacionados con las versiones sin afectar el rendimiento general.

2. **Escalabilidad Horizontal:**

    - Ser capaz de escalar horizontalmente al agregar más recursos o servidores según sea necesario.

3. **Gestión de Usuarios Concurrentes**:

    - Escalar para manejar un número creciente de usuarios concurrentes sin degradación significativa del rendimiento.

4. **Optimización de Recursos:**

    - Utilizar eficientemente los recursos disponibles, como memoria y capacidad de procesamiento, para garantizar una escalabilidad sostenible.

5. **Adaptabilidad a la Carga:**

    - Ser capaz de adaptarse rápidamente a picos de carga, como durante períodos de presentaciones o evaluaciones.
6. **Monitoreo y Ajuste Automático**:

    - Implementar sistemas de monitoreo para identificar automáticamente necesidades de escalabilidad y realizar ajustes según sea necesario.
7. **Planificación de Capacidad:**

    - Desarrollar planes de capacidad que permitan anticipar y abordar las demandas futuras en términos de usuarios y datos.
8. **Pruebas de Escalabilidad:**

    - Realizar pruebas regulares de escalabilidad para garantizar que la plataforma pueda crecer de manera efectiva.

# 5.MODELO E/R

## Diagrama Entidad- Relación

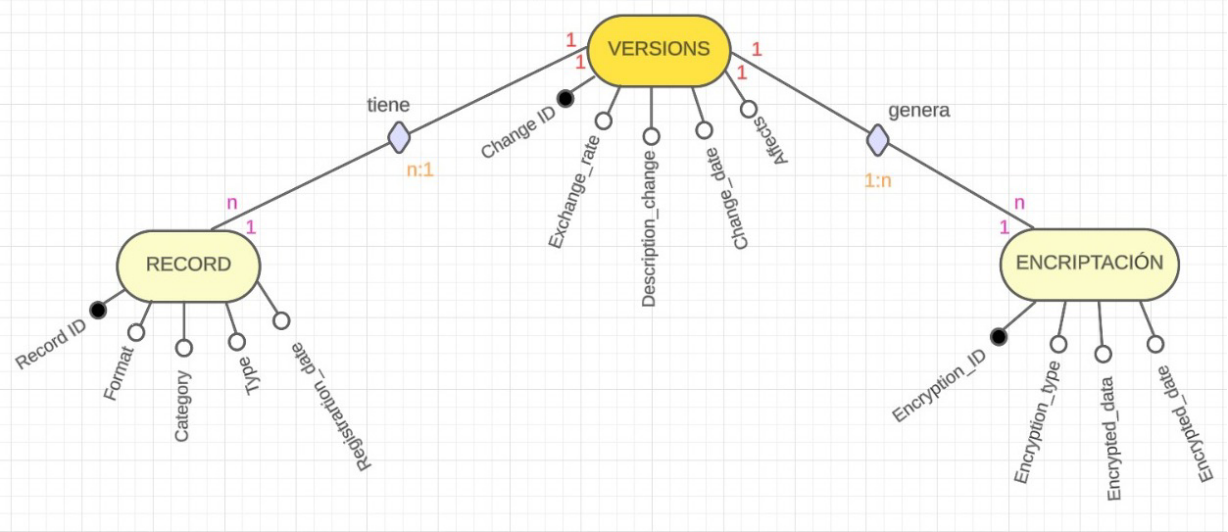
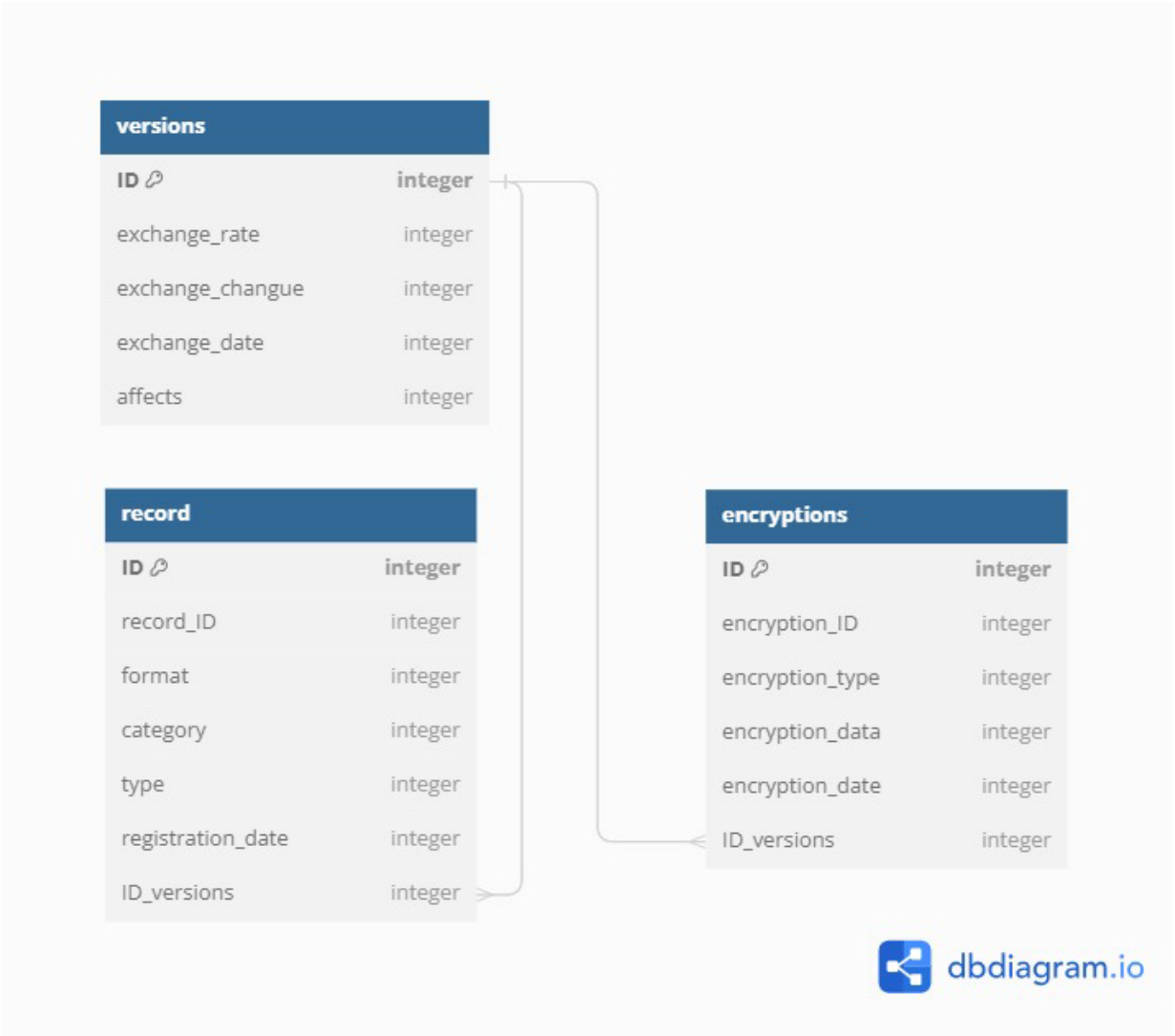


Diagrama relacional



## Script de modelo relacional

https://dbdiagram.io/

```
Table versions {
  ID integer [primary key]
  exchange_rate integer
  exchange_changue integer
  exchange_date integer
  affects integer
}

Table encryptions {
  ID integer [primary key]
  encryption_ID integer
  encryption_type integer
  encryption_data integer
  encryption_date integer
  ID_versions integer [ref: > versions.ID]
}

Table record {
  ID integer [primary key]
  record_ID integer
  format integer
  category integer
  type integer
  registration_date integer
  ID_versions integer [ref: > versions.ID]
}
```

# 6. Anexos

# Etapa 2: Persistencia de Datos con Backend

## 7. Introducción

En esta fase, nos enfocaremos en establecer un sólido sistema de almacenamiento y recuperación de datos respaldado por un backend eficiente. La correcta implementación de la persistencia de datos es esencial para garantizar la integridad y accesibilidad de las versiones en nuestro sistema de control de versiones educativo.

**Propósito de la Etapa**

El propósito principal de esta etapa es diseñar, implementar y probar la infraestructura de persistencia de datos que respalde las funcionalidades clave del módulo de control de versiones. Buscamos crear una base robusta que facilite la gestión eficiente de versiones, garantizando la seguridad y la disponibilidad de los datos en todo momento**.**

**Alcance de la Etapa**

**Diseño del Esquema de Base de Datos:**

- Definir la estructura de la base de datos que albergará la información relacionada con las versiones, usuarios y actividades del sistema.

**Implementación del Backend:**

- Desarrollar y configurar el backend necesario para gestionar operaciones de persistencia, incluyendo la creación, actualización, y recuperación de datos.

**Integración con la Lógica de Negocio:**

- Asegurar una integración eficiente entre la capa de persistencia de datos y la lógica de negocio del módulo de control de versiones.

**Seguridad de Datos:**

- Implementar medidas de seguridad robustas para proteger la integridad y confidencialidad de la información almacenada.

**Pruebas de Persistencia:**

- Realizar pruebas exhaustivas para verificar la eficacia y confiabilidad del sistema de persistencia, identificando y corrigiendo posibles problemas.

**Definiciones y Acrónimos**

- BD: Base de Datos.
- API: Interfaz de Programación de Aplicaciones.
- ORM: Mapeador Objeto-Relacional.
- SQL: Lenguaje de Consulta Estructurado.
- CRUD: Crear, Leer, Actualizar, Eliminar (Operaciones básicas en sistemas de información).
- ACL: Lista de Control de Acceso.
- TLS: Capa de Seguridad de Transporte.
- API REST: Interfaz de Programación de Aplicaciones basada en Transferencia de Estado Representacional.

# 8. Diseño de la Arquitectura de Backend

Descripción de la Arquitectura Propuesta

Componentes del Backend

Diagramas de Arquitectura

# 9. Elección de la Base de Datos

Evaluación de Opciones (SQL o NoSQL)

Justificación de la Elección

Diseño de Esquema de Base de Datos