

DISEÑO Y DESARROLLO DE SOFTWARE EDUCATIVO II

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

Profesor:
Alexander Toscano

UNIVERSIDAD DE CÓRDOBA
FACULTAD DE EDUCACIÓN Y CIENCIAS HUMANAS
LICENCIATURA EN INFORMÁTICA
MONTERÍA-CÓRDOBA
2023

Etapas 1 Diseño de la Aplicación y Análisis de Requisitos

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

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.

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: Asíncrono 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.

Usuarios del Sistema

Desarrolladores de Software

- Realizar commits y gestionar cambios en el código fuente.
- Gestionar y fusionar ramas para colaboración en paralelo.
- Participar en la revisión y aprobación de código.
- Etiquetar y versionar releases siguiendo prácticas de versión.

Líderes de Equipo y Gestores de Proyecto

- Supervisar y coordinar actividades de desarrollo.
- Gestionar ramas y resolver conflictos.
- Obtener información sobre el estado y progreso del desarrollo.

Administradores de Sistemas

- Configurar y mantener el entorno del sistema de Control de Versiones.

Testers y QA

- Acceder a versiones específicas para realizar pruebas de calidad.

Otros Colaboradores

- Acceder a la documentación y revisar cambios relevantes para su área.

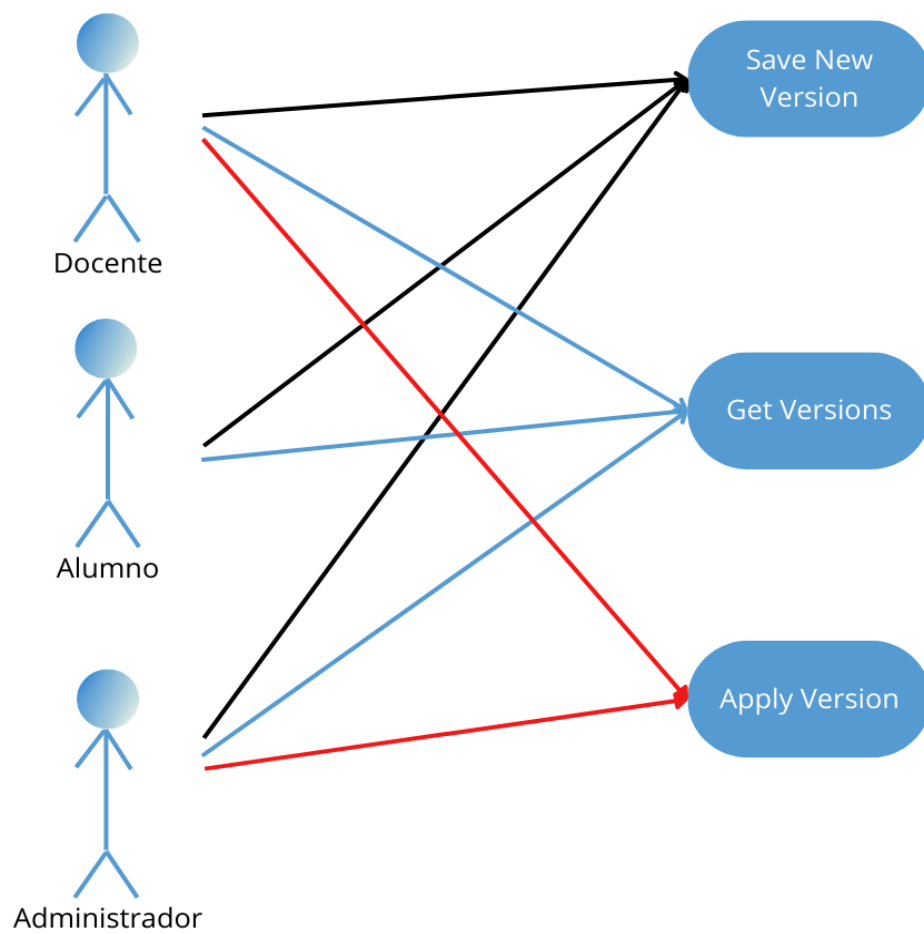
Restricciones

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.

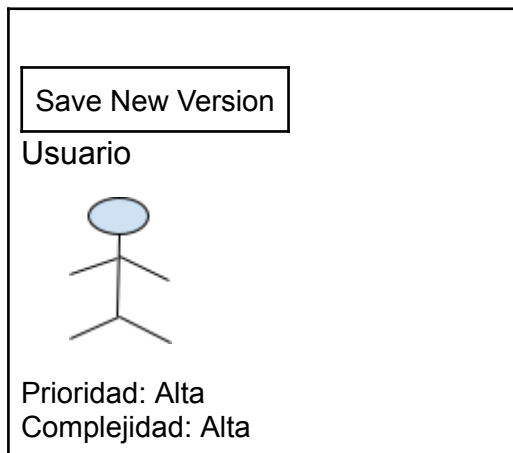
3. REQUISITOS FUNCIONALES

CASOS DE USOS CONTROL VERSIONS

CASO DE USO 1 SAVE NEW VERSION



CASO DE USO 1 SAVE NEW VERSIONS

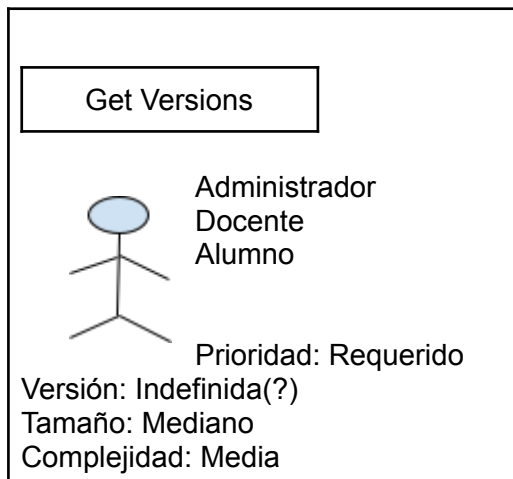


1.Save New Version

Flujo: Guardar
nueva versión
+

UML Use Case Diagram for '1.Save New Version': A rectangular box contains the title '1.Save New Version' at the top. Below the title, the flow 'Flujo: Guardar nueva versión' is listed, followed by a '+' symbol.

CASO DE USO 2 GET VERSIONS

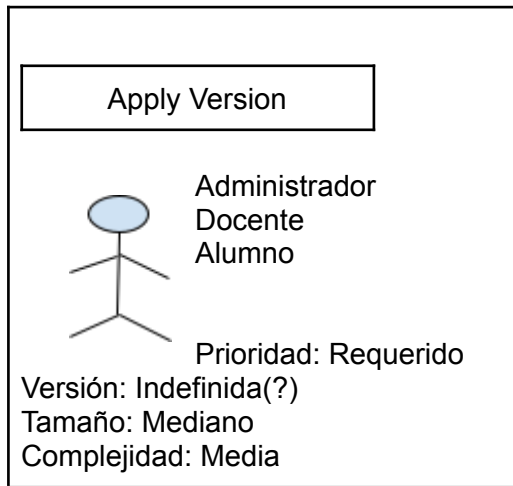


2.Get Versions

Flujo: Obtener
versiones

UML Use Case Diagram for '2.Get Versions': A rectangular box contains the title '2.Get Versions' at the top. Below the title, the flow 'Flujo: Obtener versiones' is listed.

CASO DE USO 3 APPLY VERSION



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	<ul style="list-style-type: none">• El usuario debe estar autenticado.• 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 origen 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	<ul style="list-style-type: none">• El objeto JSON de destino tiene una versión guardada en el sistema de control de versiones.• El historial de versiones del objeto JSON se ha actualizado con la nueva versión.• 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	<ul style="list-style-type: none">• Debe estar autenticado en el sistema.• Debe tener acceso al objeto del cual desea obtener el historial de versiones.• 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	<ul style="list-style-type: none">• El usuario selecciona el objeto para el cual desea obtener el historial de versiones.• No se muestra el icono informativo indicando que hay versiones disponibles.	
Postcondiciones	<ul style="list-style-type: none">• El usuario obtiene la lista de versiones disponibles para el objeto seleccionado.• 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	<ul style="list-style-type: none">• El usuario debe estar autenticado en el sistema de control de versiones.• El usuario debe tener acceso y permisos para editar la fuente a la que se aplicará la versión.• 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	<ul style="list-style-type: none">• El usuario no aplica la versión seleccionada.• El sistema de control de versiones mantiene la versión fuente.	
Postcondiciones	<ul style="list-style-type: none">• La fuente se actualiza con los cambios de la versión especificada por el usuario.• El sistema de control de versiones registra la acción de aplicar la versión en el historial del objeto.• El usuario recibe una confirmación de que la versión se ha aplicado con éxito.	

	Urgencia					
I m p a c t o		1-Baja	2-Menor	3-Moderada	4-Alta	5-Obligatoria
	5-Muy alto	5	10	15	20	25
						CU-1
	4-Alto	4	8	12	16	20
					CU-3	CU-2
	3-Medio	3	6	9	12	15
	2-Bajo	2	4	6	8	10
	1-Muy bajo	1	2	3	4	5

5.MODELO E/R

Diagrama Entidad- Relación

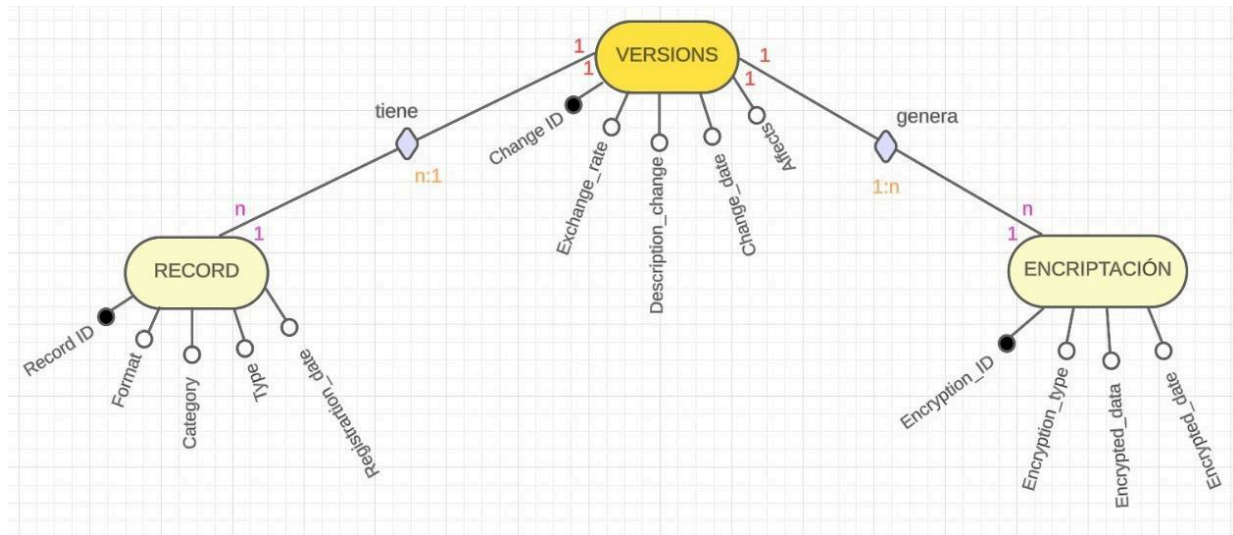
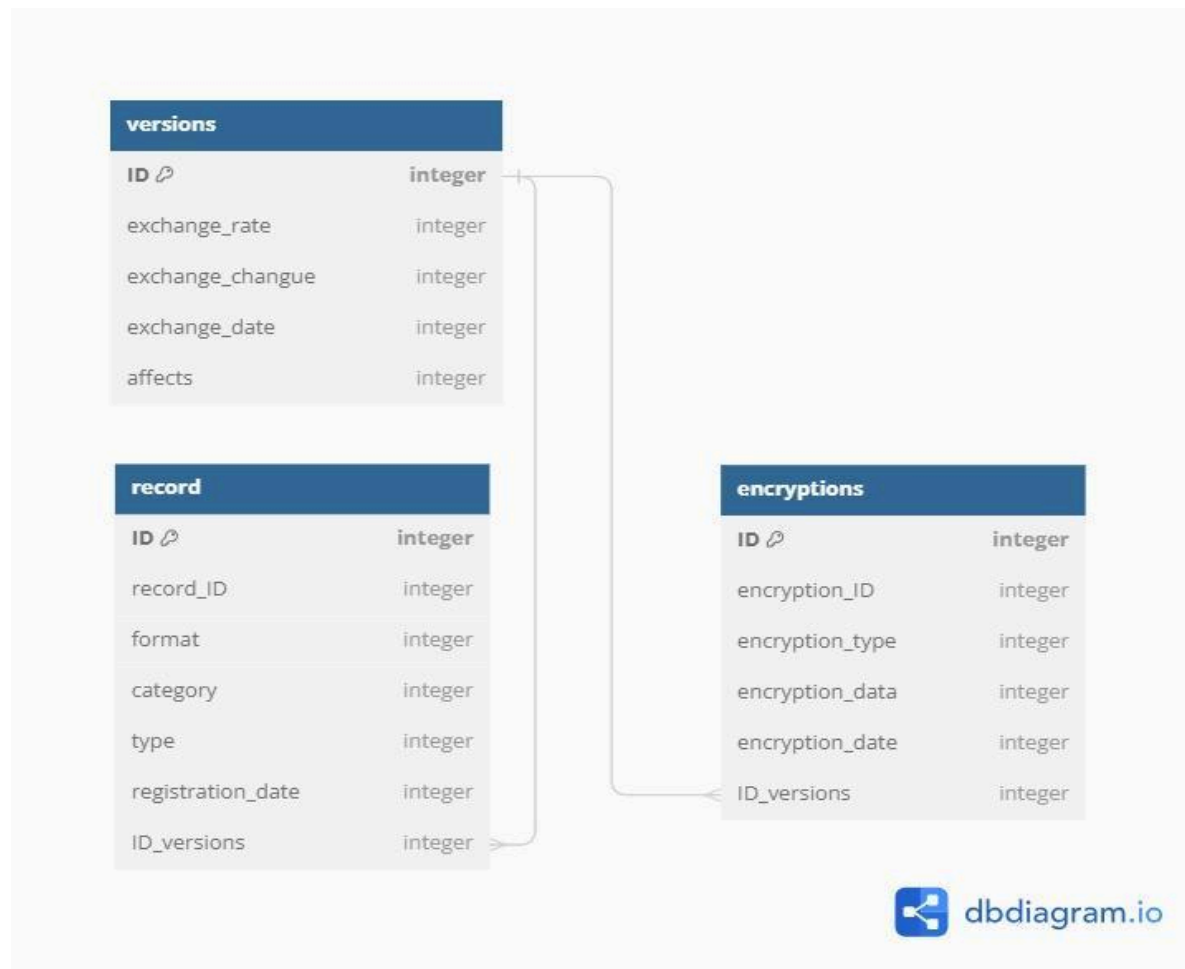


Diagrama relacional



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]  
}
```