DISEÑO Y DESARROLLO DE SOFTWARE EDUCATIVO II

Yurgen Antonio Hoyos Aldana

José Carlos Negrete Hoyos

Johana Patricia Díaz Padilla

Profesor:

Alexander Toscano

UNIVERSIDAD DE CORDOBA

FACULTAD DE EDUCACIÓN Y CIENCIAS HUMANAS

LICENCIATURA EN INFORMÁTICA

MONTERÍA-CÓRDOBA

2023

# Etapa 1 Diseño de la Aplicación y Análisis de Requisitos

**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.

## 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.

## 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 USO

**Título del Caso de Uso 1: Guardar Nueva Versión**

**(saveNewVersion(targetJson) guarda una nueva versión calculando la diferencia entre el objeto de origen y el objeto JSON de destino en un sistema de control de versiones.)**

Actores Principales: Usuario, Sistema de Control de Versiones

Resumen: Este caso de uso describe cómo el usuario puede guardar una nueva versión de un objeto JSON en el sistema de control de versiones, calculando y almacenando la diferencia entre el objeto de origen y el objeto JSON de destino.

Condiciones previas:

El usuario debe estar autenticado en el sistema de control de versiones.

El usuario debe tener permisos para editar el objeto JSON de destino.

Deben existir versiones anteriores del objeto JSON en el sistema de control de versiones.

Flujo Básico:

El usuario inicia sesión en el sistema de control de versiones.

El usuario selecciona el objeto JSON de destino que desea actualizar.

El usuario proporciona el objeto JSON de destino, que es la nueva versión del objeto. El sistema de control de versiones compara el objeto de origen con el objeto JSON de destino para calcular la diferencia.

El sistema de control de versiones almacena la diferencia como parte de la nueva versión. El sistema de control de versiones asigna un número de versión único y una marca de tiempo a la nueva versión.

El sistema de control de versiones actualiza el historial de versiones del objeto JSON con la nueva versión.

El recibe una confirmación de que la nueva versión se ha guardado exitosamente.

Flujo Alternativo:

Si el usuario no tiene permisos para editar el objeto JSON de destino, el sistema de control de versiones muestra un mensaje de error y el proceso se detiene.

Si no existen versiones anteriores del objeto JSON en el sistema de control de versiones, se crea la primera versión con la diferencia entre el objeto de origen y el objeto JSON de destino.

Postcondiciones:

El objeto JSON de destino tiene una nueva versión guardada en el sistema de control de versiones.

El historial de versiones del objeto JSON se ha actualizado con la nueva versión. La diferencia entre la nueva versión y la versión anterior se almacena en el sistema de control de versiones para futuras referencias.

Este caso de uso describe cómo un usuario puede guardar una nueva versión de un objeto JSON en un sistema de control de versiones, permitiendo un seguimiento y gestión eficiente de las diferentes versiones del objeto a lo largo del tiempo.

**Título del Caso de Uso 2: Obtener Historial de Versiones**

**(getHistoryVersions()que obtiene la lista de versiones del historial disponibles en un módulo de control de versiones)**

Actores Principales: Usuario, Sistema de Control de Versiones

Resumen: Este caso de uso describe cómo el usuario puede obtener la lista de versiones del historial disponibles para un objeto en el sistema de control de versiones.

Condiciones previas:

El usuario debe estar autenticado en el sistema de control de versiones.

El usuario debe tener acceso al objeto del cual desea obtener el historial de versiones.

Flujo Básico:

El usuario inicia sesión en el sistema de control de versiones.

El usuario selecciona el objeto para el cual desea obtener el historial de versiones. El sistema de control de versiones recopila y presenta al usuario la lista de versiones disponibles para ese objeto.

El usuario puede ver información adicional para cada versión, como la marca de tiempo, el número de versión y una breve descripción de los cambios realizados en esa versión. El usuario puede seleccionar una versión específica si desea ver más detalles o restaurar esa versión.

Flujo Alternativo:

Si el usuario no tiene acceso al objeto o el objeto no tiene historial de versiones registradas en el sistema, se muestra un mensaje informativo indicando que no hay versiones disponibles.

Postcondiciones:

El usuario obtiene la lista de versiones disponibles para el objeto seleccionado. El sistema de control de versiones presenta la información del historial de versiones, incluyendo detalles como la marca de tiempo y una descripción de los cambios. El usuario puede seleccionar una versión específica para ver detalles adicionales o restaurar esa versión si es necesario.

Este caso de uso permite a los usuarios del sistema de control de versiones acceder de manera efectiva al historial de versiones de un objeto, lo que les permite visualizar y restaurar versiones anteriores según sea necesario para su trabajo o referencia.

**Título del Caso de Uso 3: Obtener Versión Anterior**

**(getPreviousVersion()que permite a un usuario obtener la versión anterior basada en la versión actual en un módulo de control de versiones)**

Actores Principales: Usuario, Sistema de Control de Versiones

Resumen: Este caso de uso describe cómo un usuario puede obtener la versión anterior de un objeto en el sistema de control de versiones basado en la versión actual.

Condiciones previas:

El usuario debe estar autenticado en el sistema de control de versiones.

El usuario debe tener acceso al objeto del cual desea obtener la versión anterior. Debe haber al menos dos versiones registradas del objeto en el sistema.

Flujo Básico:

El usuario inicia sesión en el sistema de control de versiones.

El usuario selecciona el objeto para el cual desea obtener la versión anterior.

El sistema de control de versiones muestra la versión actual del objeto.

El obtener solicita la versión anterior a partir de la versión actual del usuario.

El sistema de control de versiones recupera y muestra la versión anterior del objeto. El usuario puede ver información adicional para la versión anterior, como la marca de tiempo, el número de versión y una breve descripción de los cambios realizados en esa versión.

Flujo Alternativo:

Si el usuario no tiene acceso al objeto o no hay al menos dos versiones registradas del objeto, se muestra un mensaje informativo indicando que no se puede obtener la versión anterior.

Postcondiciones:

El usuario obtiene la versión anterior del objeto basada en la versión actual.

El sistema de control de versiones presenta la información de la versión anterior, incluyendo detalles como la marca de tiempo y una descripción de los cambios realizados en esa versión.

Este caso de uso permite a los usuarios del sistema de control de versiones acceder a la versión anterior de un objeto en función de la versión actual, lo que les facilita la comparación y restauración de versiones anteriores cuando sea necesario para su trabajo o referencia.

**Título del Caso de Uso 4: Obtener Siguiente Versión**

**(getNextVersion(), la cual permite a un usuario obtener la siguiente versión basada en la versión actual en un módulo de control de versiones)**

Actores Principales: Usuario, Sistema de Control de Versiones

Resumen: Este caso de uso describe cómo un usuario puede obtener la siguiente versión de un objeto en el sistema de control de versiones basado en la versión actual.

Condiciones previas:

El usuario debe estar autenticado en el sistema de control de versiones.

El usuario debe tener acceso al objeto del cual desea obtener la versión siguiente. Debe haber al menos dos versiones registradas del objeto en el sistema.

Flujo Básico:

El usuario inicia sesión en el sistema de control de versiones.

El usuario selecciona el objeto para el cual desea obtener la versión siguiente.

El sistema de control de versiones muestra la versión actual del objeto.

El usuario solicitará la siguiente versión a partir de la versión actual.

El sistema de control de versiones recupera y muestra la versión siguiente del objeto. El usuario puede ver información adicional para la versión siguiente, como la marca de tiempo, el número de versión y una breve descripción de los cambios realizados en esa versión.

Flujo Alternativo:

Si el usuario no tiene acceso al objeto o no hay al menos dos versiones registradas del objeto, se muestra un mensaje informativo indicando que no se puede obtener la siguiente versión.

Postcondiciones:

El usuario obtiene la siguiente versión del objeto basada en la versión actual. El sistema de control de versiones presenta la información de la versión siguiente, incluyendo detalles como la marca de tiempo y una descripción de los cambios realizados en esa versión.

Este caso de uso permite a los usuarios del sistema de control de versiones acceder a la siguiente versión de un objeto en función de la versión actual, facilitando la visualización y comparación de versiones posteriores cuando sea necesario para su trabajo o referencia.

**Título del Caso de Uso 5: Aplicar Versión a la Fuente**

**(applyVersionToSource(version), que permite aplicar una versión especificada a la fuente en un módulo de control de versiones)**

Actores Principales: Usuario, Sistema de Control de Versiones

Resumen: Este caso de uso describe cómo un usuario puede aplicar una versión específica de un objeto a la fuente actual en el sistema de control de versiones.

Precondiciones:

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:

El usuario inicia sesión en el sistema de control de versiones.

El usuario selecciona el objeto (fuente) al que desea aplicar una versión específica.

El sistema de control de versiones muestra la lista de versiones disponibles para ese objeto.

El usuario selecciona la versión específica que desea aplicar a la fuente actual.

El sistema de control de versiones aplica los cambios de la versión seleccionada a la fuente actual.

El sistema de control de versiones actualiza la fuente y muestra una confirmación de que la versión se ha aplicado correctamente.

Flujo Alternativo:

Si el usuario no tiene acceso o permisos para editar la fuente, se muestra un mensaje de error y el proceso se detiene.

Si no existen versiones registradas del objeto en el sistema, se muestra un mensaje informativo indicando que no hay versiones disponibles para aplicar.

Postcondiciones:

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.

Este caso de uso permite a los usuarios aplicar versiones específicas a la fuente actual en el sistema de control de versiones, lo que facilita la restauración de versiones anteriores o la aplicación de cambios específicos según sea necesario para su trabajo o proyecto.

**Título del Caso de Uso 6: Obtener Objeto Fuente de Versión Inicial**

**(getInitialVersion(), que permite obtener el objeto fuente de la versión inicial en un módulo de control de versiones)**

Actores Principales: Usuario, Sistema de Control de Versiones

Resumen: Este caso de uso describe cómo un usuario puede obtener el objeto fuente de la versión inicial en el sistema de control de versiones.

Precondiciones:

El usuario debe estar autenticado en el sistema de control de versiones.

Deben existir al menos dos versiones registradas del objeto en el sistema.

Flujo Básico:

El usuario inicia sesión en el sistema de control de versiones.

El usuario selecciona el objeto del cual desea obtener la versión inicial.

El sistema de control de versiones muestra la lista de versiones disponibles para ese objeto.

El usuario selecciona la versión inicial (la primera versión) del objeto.

El sistema de control de versiones recupera y muestra el objeto fuente de la versión inicial. El usuario puede ver el objeto fuente en su estado original sin cambios.

Flujo Alternativo:

Si no existen versiones registradas del objeto en el sistema o si el usuario no tiene acceso al objeto, se muestra un mensaje informativo indicando que no se puede obtener la versión inicial.

Postcondiciones:

El usuario obtiene el objeto fuente en su estado original de la primera versión registrada.

El sistema de control de versiones muestra el objeto fuente sin cambios.

El usuario puede utilizar el objeto fuente en su estado inicial según sea necesario para su trabajo o referencia.

Este caso de uso permite a los usuarios acceder al objeto fuente en su estado original, lo que puede ser útil para comparar con versiones posteriores, restaurar a un estado anterior o revisar la evolución del objeto en el tiempo en el sistema de control de versiones.

**Título del Caso de Uso 7: Obtener Objeto Fuente de Última Versión**

**(getLatestVersion(), que permite obtener el objeto fuente de la última versión en un módulo de control de versiones)**

Actores Principales: Usuario, Sistema de Control de Versiones

Resumen: Este caso de uso describe cómo un usuario puede obtener el objeto fuente de la última versión en el sistema de control de versiones.

Precondiciones:

El usuario debe estar autenticado en el sistema de control de versiones.

Deben existir al menos dos versiones registradas del objeto en el sistema.

Flujo Básico:

El usuario inicia sesión en el sistema de control de versiones.

El usuario selecciona el objeto del cual desea obtener la última versión.

El sistema de control de versiones muestra la lista de versiones disponibles para ese objeto.

El usuario selecciona la última versión del objeto.

El sistema de control de versiones recupera y muestra el objeto fuente de la última versión. El usuario puede ver el objeto fuente en su estado más reciente.

Flujo Alternativo:

Si no existen versiones registradas del objeto en el sistema o si el usuario no tiene acceso al objeto, se muestra un mensaje informativo indicando que no se puede obtener la última versión.

Postcondiciones:

El usuario obtiene el objeto fuente en su estado más reciente, correspondiente a la última versión registrada.

El sistema de control de versiones muestra el objeto fuente en su estado más reciente. El usuario puede utilizar el objeto fuente en su estado actual según sea necesario para su trabajo o referencia.

Este caso de uso permite a los usuarios acceder al objeto fuente en su estado más reciente, lo que puede ser útil para trabajar con la versión actual del objeto, realizar ediciones o seguir su evolución en el sistema de control de versiones.

**Título del Caso de Uso 8: Aplicar Versión Inicial a la Fuente**

**(applyInitialVersion(), que permite aplicar la versión inicial a la fuente en un módulo de control de versiones)**

Actores Principales: Usuario, Sistema de Control de Versiones

Resumen: Este caso de uso describe cómo un usuario puede aplicar la versión inicial (el estado original) a la fuente en el sistema de control de versiones.

Precondiciones:

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 inicial.

Deben existir al menos dos versiones registradas del objeto en el sistema.

Flujo Básico:

El usuario inicia sesión en el sistema de control de versiones.

El usuario selecciona el objeto (fuente) al que desea aplicar la versión inicial.

El sistema de control de versiones muestra la lista de versiones disponibles para ese objeto.

El usuario selecciona la versión inicial (la primera versión) del objeto.

El usuario solicita aplicar la versión inicial a la fuente actual.

El sistema de control de versiones aplica los cambios de la versión inicial a la fuente actual. El sistema de control de versiones actualiza la fuente y muestra una confirmación de que la versión inicial se ha aplicado correctamente.

Flujo Alternativo:

Si el usuario no tiene acceso o permisos para editar la fuente, se muestra un mensaje de error y el proceso se detiene.

Si no existen versiones registradas del objeto en el sistema, se muestra un mensaje informativo indicando que no se puede aplicar la versión inicial.

Postcondiciones:

La fuente se actualiza con el contenido de la versión inicial (el estado original). El sistema de control de versiones registra la acción de aplicar la versión inicial en el historial del objeto.

El usuario recibe una confirmación de que la versión inicial se ha aplicado con éxito. Este caso de uso permite a los usuarios restablecer la fuente a su estado original o a la versión inicial registrada en el sistema de control de versiones, lo que puede ser útil para deshacer cambios no deseados o volver al punto de partida en un proyecto.

**Título del Caso de Uso 9: Aplicar Última Versión a la Fuente**

**(applyLatestVersion(), que permite aplicar la última versión a la fuente en un módulo de control de versiones)**

Actores Principales: Usuario, Sistema de Control de Versiones

Resumen: Este caso de uso describe cómo un usuario puede aplicar la última versión registrada en el sistema de control de versiones a la fuente.

**Precondiciones:**

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 última versión.

Deben existir al menos dos versiones registradas del objeto en el sistema.

**Flujo Básico:**

El usuario inicia sesión en el sistema de control de versiones.

El usuario selecciona el objeto (fuente) al que desea aplicar la última versión.

El sistema de control de versiones muestra la lista de versiones disponibles para ese objeto.

El usuario selecciona la última versión registrada del objeto.

El usuario solicita aplicar la última versión a la fuente actual.

El sistema de control de versiones aplica los cambios de la última versión a la fuente actual. El sistema de control de versiones actualiza la fuente y muestra una confirmación de que la última versión se ha aplicado correctamente.

**Flujo Alternativo:**

Si el usuario no tiene acceso o permisos para editar la fuente, se muestra un mensaje de error y el proceso se detiene.

Si no existen versiones registradas del objeto en el sistema, se muestra un mensaje informativo indicando que no se puede aplicar la última versión.

**Postcondiciones:**

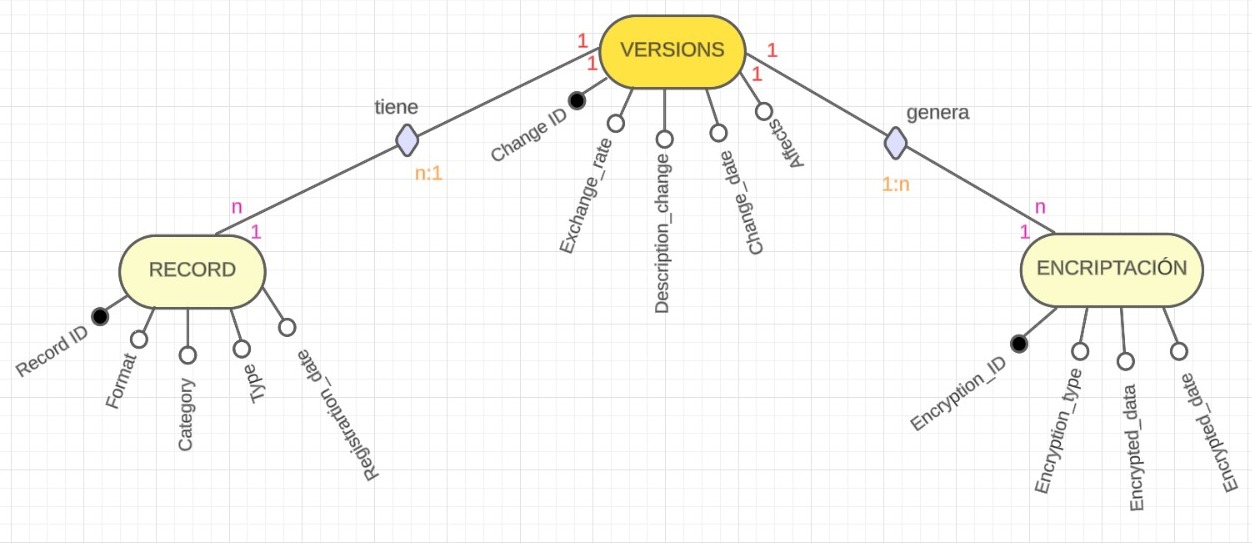
La fuente se actualiza con el contenido de la última versión registrada en el sistema. El sistema de control de versiones registra la acción de aplicar la última versión en el historial del objeto.

El usuario recibe una confirmación de que la última versión se ha aplicado con éxito. Este caso de uso permite a los usuarios actualizar la fuente a su estado más reciente, lo que puede ser útil para trabajar con la versión actual del objeto, aplicar los cambios más recientes o continuar desde la última versión en el sistema de control de versiones.

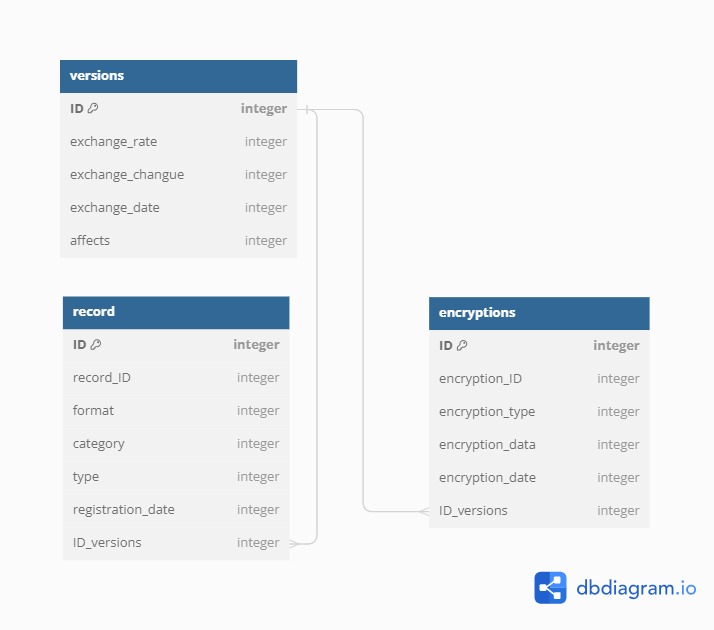
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Urgencia | | | | | |
| Impacto |  | 1-Baja | 2-Menor | 3-Moderada | 4-Alta | 5-Obligatoria |
| 5-Muy alto | 5 | 10 | 15 | 20 | 25 |
|  |  | CU-9 | CU-4 | CU-1 |
| 4-Alto | 4 | 8 | 12 | 16 | 20 |
|  |  | CU-7 | CU-5 |  |
| 3-Medio | 3 | 6 | 9 | 12 | 15 |
|  | CU-3 | CU-6 | CU-8 |  |
| 2-Bajo | 2 | 4 | 6 | 8 | 10 |
|  | CU-2 |  |  |  |
| 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]  } |  |