DBCASE WEB 4.0

DBCASE WEB 4.0

TRABAJO FIN DE GRADO
CURSO 2024-2025

AUTOR
MARÍA BARCENILLA HINCHADO

DIRECTOR
FERNANDO SÁENZ PÉREZ

CALIFICACIÓN: 8′5

GRADO EN INGENIERÍA INFORMÁTICA
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

DEDICATORIA

A mi familia y amigos, por su amor incondicional, apoyo constante y por haberme motivado a conseguir mis metas.

AGRADECIMIENTOS

Quiero expresar mi más sincero agradecimiento a todas las personas que han hecho posible la realización de este Trabajo de Fin de Grado.

En primer lugar, me gustaría agradecer a mi tutor, Fernando, por su orientación y paciencia a lo largo de todo el proceso.

A mis profesores y profesoras de la Facultad de Informática por brindarme su conocimiento y por haberme motivado a lo largo de todo mi recorrido académico.

A mi familia y amigos, por su paciencia, comprensión y por ser mi refugio durante las etapas de estrés y presión. Gracias por hacer que este camino sea más ameno y por estar siempre ahí cuando más os necesitaba.

RESUMEN

DBCASE WEB 4.0

El proyecto DBCASE Web 4.0, es el último entregable de la versión web del proyecto DBCASE. El objetivo del desarrollo de esta aplicación es el diseño de diagramas entidad-relación de bases de datos relacionales y la traducción de este a esquemas relacionales y físicos.

Esta aplicación web tiene como usuarios objetivos los estudiantes de la Universidad Complutense de Madrid, en concreto los estudiantes de la asignatura de Bases de Datos. Es por tanto una herramienta que busca ser empleada de forma didáctica, como material de estudio y apoyo al aprendizaje de esta materia.

El objetivo principal de este proyecto es mejorar y desarrollar nuevas funcionalidades sobre la versión anterior, entregando una aplicación web más acabada, que mejore la experiencia de usuario.

Palabras clave

Base de datos, Modelo Relacional, Modelo Entidad relación, Aplicación Web, Agregación, Deshacer, Rehacer

ABSTRACT

DBCASE WEB 4.0

The DBCASE Web 4.0 project is the last deliverable of the web version of the DBCASE project. The objective of the development of this application is the design of entity-relational diagrams of relational databases and the translation of this to relational and physical schemas.

This web application has as target users the students of the Complutense University of Madrid, specifically the students of the subject of Databases. It is therefore a tool that seeks to be used in a didactic way, as study material and support for the learning of this subject.

The main objective of this project is to improve and develop new features over the previous version, delivering a more finished web application that improves the user experience.

Keywords

Databases, Relational model, Entity-relationship model, Web application, Aggregations, Undo, Redo

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

Capítulo 1 - Introducción

DBCASE Web 3.0 es la última versión de un proyecto cuyo desarrollo se inició en 2007 con el objetivo de ofrecer a los estudiantes una herramienta accesible e intuitiva para comprender el funcionamiento de las bases de datos relacionales y diseñar esquemas de manera sencilla. Desde sus inicios, DBCASE ha evolucionado para adaptarse a las necesidades de los usuarios, proporcionando un entorno que facilita el diseño de diagramas Entidad-Relación.

Esta nueva entrega se enfoca en mejorar la implementación de agregaciones y adaptar la generación de esquemas conforme a estas actualizaciones, asegurando la precisión de las traducciones. Además, se añaden funcionalidades de edición, como las opciones de deshacer (Ctrl + Z) y rehacer (Ctrl + Y), que brindan una mayor flexibilidad y control al usuario. Adicionalmente, se ha trabajado en mejorar la usabilidad general de la aplicación, perfeccionando desarrollos previos y corrigiendo errores para que el uso sea más intuitivo y fluido.

La aplicación, sigue evolucionando como una herramienta educativa robusta y de uso sencillo, diseñada para proporcionar una experiencia de usuario óptima en el aprendizaje de la asignatura bases de datos.

1.1 Motivación

La motivación principal para elegir este proyecto de TFG surge de mi experiencia como estudiante de la asignatura Base de Datos. Durante el curso, noté la falta de una herramienta que facilitara el estudio y la mejor comprensión de las traducciones entre modelos. Esta necesidad me llevó a elegir el proyecto DBCase Web para poder trabajar en una solución que pudiera cubrir las necesidades y problemáticas que yo misma enfrenté. Al haber sido estudiante de la asignatura, cuento con una perspectiva privilegiada para entender las necesidades y expectativas de los usuarios. Esto me sitúa en una posición en la que poder tomar decisiones fundamentadas en cuanto a la usabilidad y la utilidad de la aplicación, priorizando aquellas funciones y actualizaciones que realmente mejoren la experiencia de los estudiantes. Además, me motiva saber que este proyecto tiene el potencial de generar un impacto real y positivo en la vida

académica de otros estudiantes, convirtiéndose en una herramienta práctica que responde a necesidades concretas y aporta un valor tangible.

En resumen, este proyecto no solo nace de una inquietud personal, sino también de mi compromiso por ofrecer una solución efectiva que realmente pueda marcar una diferencia.

1.2 Antecedentes

Es esencial mencionar y entender las versiones anteriores de este proyecto para poder realizar y explicar esta nueva ampliación del proyecto. Se detalla a continuación las versiones previas.

1.2.1 DBDT de sistemas informáticos (2007 - 2008)

Esta es la primera iteración que se hizo de este proyecto. Se desarrolla como una aplicación de escritorio que permite diseñar y comprobar un esquema de base de datos y generar un script SQL [1].

Autores:

- Alberto Milán Gutiérrez
- Miguel Martínez Segura
- Francisco Javier Cáceres González
- Yolanda García Ruiz (Tutora)

1.2.2 DBCASE de sistemas informáticos (2008 - 2009)

Este segundo proyecto sobre la aplicación amplía la herramienta para que sea compatible con los gestores de bases de datos más comunes y añadir nuevas funcionalidades en el diseño de los modelos entidad-relación [2].

Autores:

- Rodrigo Denís Cepeda Mateos
- Cristina Marco de Francisco

- Tello Serrano Gordillo
- Fernando Saénz Pérez (Tutor)

1.2.3 DBCASE 2.0 (2018 – 2019)

El proyecto tenía como objetivo rehabilitar una aplicación para su uso académico en las aulas. Para ello, se actualizó el diseño de la interfaz, mejorando la usabilidad y simplificando el flujo de trabajo, además de rediseñar el panel de códigos y mejorar la visualización del diagrama entidad-relación. También se corrigieron errores de versiones previas, se añadieron nuevas funcionalidades, como un tema oscuro, y se completó la traducción al modelo lógico para facilitar la comprensión del usuario [3].

Autores:

- Miguel Arriba García
- Fernando Saénz Pérez (Tutor)

1.2.4 DBCASE Web 2.0 (2019 - 2020)

Con esta nueva versión, se consigue realizar una aplicación multiplataforma partiendo de la versión de escritorio realizada el año anterior. De esta manera, la aplicación empezó a ser accesible desde cualquier navegador, eliminando la necesidad de instalaciones y dependencia de un software específico [4].

Autores:

- Luis Eduardo Pacuar Cerrón
- Yrving David Conde Cubas
- Fernando Saénz Pérez (Tutor)

1.2.5 DBCASE Web 3.0 (2020 - 2021)

En esta ampliación del proyecto, se amplían las funcionalidades de la aplicación, incorporando nuevas construcciones de diseño conceptual, implementando de participación y cardinalidad de las relaciones y la mejora visual de la herramienta [5].

Autores:

- Roxanne A. Caiafa
- Fernando Saénz Pérez (Tutor)

1.3 Objetivos

El objetivo de esta nueva versión del proyecto es solventar errores de traducción entre los diferentes modelos y ampliar las funcionalidades de la aplicación para mejorar la experiencia del usuario. Estas nuevas funcionalidades se centran en realizar una implementación más compleja de las agregaciones, para ofrecer una mayor flexibilidad y usabilidad en el diseño de bases de datos. Además, se introducirá el soporte para controles de teclado, como las opciones de deshacer (Ctrl + Z) y rehacer (Ctrl + Y), lo que permitirá realizar cambios con mayor facilidad, mejorando considerablemente la experiencia de usuario.

También se trabajará en la mejora de la usabilidad general de la aplicación, para que la interacción sea más fluida y amigable. Por último, se corregirán problemas detectados en los elementos y sus traducciones, con el fin de asegurar que todas las funciones sean coherentes y precisas.

1.4 Plan de trabajo

El proyecto ha sido planificado según unas fases llevadas a cabo durante la totalidad del desarrollo. En paralelo a estas tres fases se ha documentado los avances del proyecto en esta memoria. Explicamos a continuación el detalle de las actividades realizadas en cada una de las tres fases.

1. Investigación

En primer lugar, se ha hecho un trabajo de investigación sobre los proyectos DBCase realizados con anterioridad, con el objetivo de entender el contexto de esta nueva versión. Junto con esta investigación se lleva a cabo la primera toma de contacto con el funcionamiento de la aplicación. Durante este proceso se configura y prepara el entorno para compilar el proyecto y probar el funcionamiento de la aplicación. En concreto es necesario descargar y configurar Maven 4.0 como gestor de

dependencias y constructor de proyectos. También es necesario la configuración de Java (JDK 1.8).

2. Desarrollo

En esta segunda fase se priorizan las tareas y principales objetivos. De esta forma, se decide empezar la implementación de los desarrollos más complejos, la agregación y las opciones de deshacer y rehacer, seguido de las mejoras funcionales y corrección de incidencias.

Esta última parte se refiere a aquellas incidencias detectadas antes del comienzo del proyecto y a aquellas detectadas durante las primeras tomas de contacto o a lo largo de las pruebas de los desarrollos.

Durante esta fase, se han utilizado buscadores y foros [6] [7] [10] para solucionar errores, resolver dudas específicas relacionadas con el código y entender el comportamiento de ciertas tecnologías.

El proyecto y todas las modificaciones realizadas sobre él se encontraran en el siguiente repositorio de GitHub: https://github.com/MariaBarcenilla/DBCaseWeb4.0

3. Pruebas

El proceso de pruebas se ha realizado de forma paralela a lo largo del desarrollo, llevando a cabo pruebas unitarias al finalizar las modificaciones y validando la correcta integración de las nuevas funcionalidades con el resto de la aplicación

Figura 1-1 Diagrama de Gantt de las fases del proyecto

Capítulo 2 - Tecnologías

Para alcanzar los objetivos del proyecto es fundamental entender las tecnologías usadas, los lenguajes de programación y las librerías que se emplean en el desarrollo de este trabajo.

2.1 Frameworks

Un framework es un conjunto de herramientas, bibliotecas y convenciones que facilitan el desarrollo de software al proporcionar una estructura predefinida. Detallamos a continuación los utilizados en este proyecto.

2.1.1 Bootstrap

Bootstrap es un framework front-end que combina CSS y JavaScript para diseñar y estilizar elementos en páginas HTML. Está diseñado para facilitar el desarrollo de aplicaciones y sitios web adaptándose automáticamente al tamaño de pantalla del dispositivo utilizado por el usuario. Es ampliamente utilizado para construir interfaces de usuario atractivas mediante componentes predefinidos y herramientas que simplifican el diseño y la disposición visual.

Este framework se incorporó al proyecto no solo por su sencillez sino también por garantizar una excelente experiencia de usuario.

2.1.2 Spring Boot

Spring Boot es un framework de desarrollo de aplicaciones Java que simplifica y acelera la creación de aplicaciones web y microservicios. Fue diseñado como una extensión del popular Spring Framework, ofreciendo una experiencia de desarrollo más ágil. Su enfoque principal es reducir la complejidad del desarrollo mediante configuraciones automáticas de dependencias permitiendo que los desarrolladores trabajen de forma más eficiente.

Este framework fue introducido en el proyecto al crear la versión web del proyecto de escritorio DBCase.

Se emplea ambos frameworks junto con HTML como parte de la vista de la arquitectura de la aplicación (figura 3-1).

2.2 Lenguajes de Programación

Los lenguajes de programación son herramientas fundamentales para crear software. Cada lenguaje tiene sus propias características, sintaxis y áreas de aplicación. A continuación, se describen algunos de los lenguajes de programación utilizados en el proyecto DBCase.

2.2.1 HTML

HTML (HyperText Markup Language) es el estándar fundamental para definir la estructura básica de una página web y organizar cómo se presenta su contenido. El verdadero potencial de HTML se despliega cuando se combina con otras tecnologías, como CSS para diseñar la apariencia y JavaScript para agregar interactividad, lo que permite construir páginas web dinámicas y completas. HTML es la base de la vista en la arquitectura de la aplicación (figura 3-1).

2.2.2 Java

Java es un lenguaje de programación multiplataforma, orientado a objetos y centrado en la red, ampliamente utilizado para desarrollar una variedad de aplicaciones y sistemas. La versatilidad, confiabilidad y popularidad de este lenguaje, lo convierten en una elección ideal para crear aplicaciones web y de escritorio, entre otras. Se trata de unos de los lenguajes más usados en esta aplicación, a nivel de arquitectura (figura 3-1) tanto el modelo como el controlador están desarrollados en Java.

2.2.3 JavaScript

JavaScript es un lenguaje de programación de secuencias de comandos que permite crear aplicaciones web dinámicas e interactivas, haciendo que las páginas web respondan en tiempo real a las interacciones de los usuarios. Este lenguaje no solo controla los aspectos visuales y funcionales de los sitios web, sino que también es

fundamental para muchas de las experiencias interactivas modernas. Es el motor detrás de la mayoría de las funciones dinámicas de los sitios web y un componente esencial en el desarrollo web actual, lo que lo convierte en una herramienta indispensable para este proyecto. Se emplea este framework junto con HTML como parte de la vista, estructura que forma parte de la arquitectura de la aplicación (figura 3-1).

2.3 Librerías

Las librerías son colecciones de funciones y recursos predefinidos que los desarrolladores pueden utilizar para agilizar el proceso de desarrollo. Se explican a continuación las librerías utilizadas en este proyecto.

2.3.1 jQuery

jQuery es una biblioteca de JavaScript de código abierto diseñada para simplificar y agilizar el desarrollo web. Surge como una colección de recursos que permite implementar funcionalidades avanzadas y efectos visuales sin necesidad de escribir grandes cantidades de código desde cero. Con jQuery, los desarrolladores pueden manipular documentos HTML, manejar eventos, crear animaciones, integrar interacciones AJAX y enriquecer la experiencia del usuario de manera eficiente [11].

2.3.2 Vis js

Vis.js es una biblioteca de JavaScript diseñada para crear visualizaciones de datos interactivas en aplicaciones web. Es conocida por su capacidad para manejar grandes volúmenes de datos dinámicos, permitiendo a los usuarios interactuar directamente con ellos. Vis.js facilita la personalización de la apariencia de las visualizaciones, adaptándolas a las necesidades específicas de cada proyecto. Su sencillez y capacidad para gestionar datos complejos hacen de Vis.js una herramienta fundamental para enriquecer la experiencia del usuario del proyecto y potenciar la presentación de la web [8].

Ambas librerías se emplean en la vista de la arquitectura (figura 3-1).

Capítulo 3 - Contexto

En el desarrollo de software, una arquitectura bien definida y los diagramas de clase juegan un papel crucial en la organización y representación de la aplicación. La arquitectura establece las bases estructurales, definiendo cómo se organizan y relacionan los componentes principales del proyecto. Por otro lado, los diagramas de clase proporcionan una representación clara y detallada de las entidades principales, sus atributos, métodos y relaciones. En este apartado se explicarán ambos aspectos con el objetivo de facilitar la comprensión del proyecto realizado.

3.1 Arquitectura

Este apartado ofrece una breve introducción a la arquitectura del sistema para facilitar la comprensión de los cambios realizados.

La aplicación está diseñada con una estructura multicapa, lo que proporciona ventajas clave como la modularidad y la adaptabilidad. Estas características se evidencian en la transición de una aplicación de escritorio a una aplicación web, donde se mantuvo la lógica principal y se implementó una nueva interfaz.

Para llevar a cabo esta arquitectura, se aplicaron los siguientes patrones de diseño:

- Modelo-Vista-Controlador (MVC).
- Patrón Transferencia (Transfer).
- Patrón Objeto de Acceso a Datos (DAO).

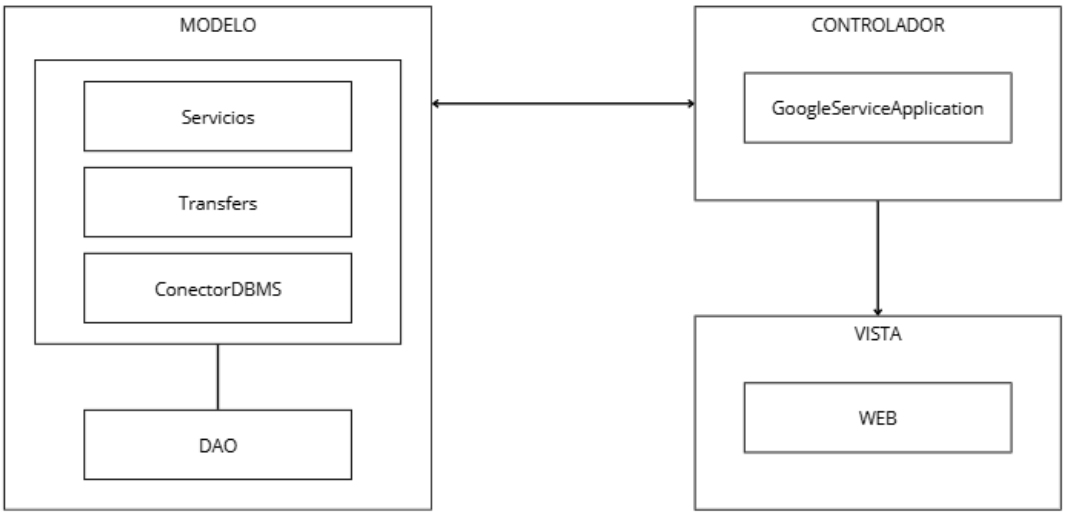

Figura 3-1 Diagrama de la arquitectura de la aplicación

En esta estructura, el Modelo y el Controlador se comunican entre sí, mientras que la Vista recibe los datos del Controlador para presentarlos al usuario.

3.2 Diagramas de clase

Los diagramas de clase son una herramienta que sirve para comprender la estructura y arquitectura del sistema desarrollado. Estos diagramas permiten representar de manera clara y concisa las clases principales de la aplicación, sus atributos, métodos y las relaciones entre ellas, facilitando la comprensión del diseño del software.

3.2.1 Servicios

Contienen la lógica de la aplicación, coordinando tanto las operaciones entre los distintos componentes como los flujos de información.

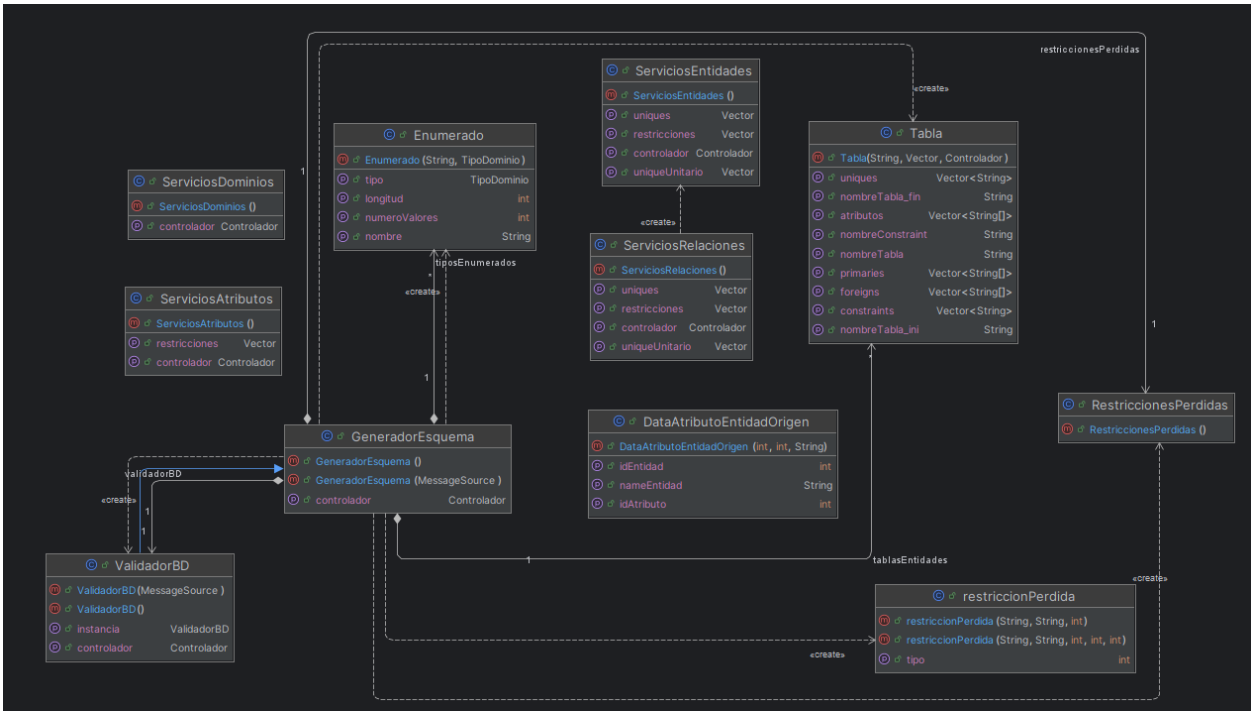

Figura 3-2 Diagrama de clase de Servicios (1/3)

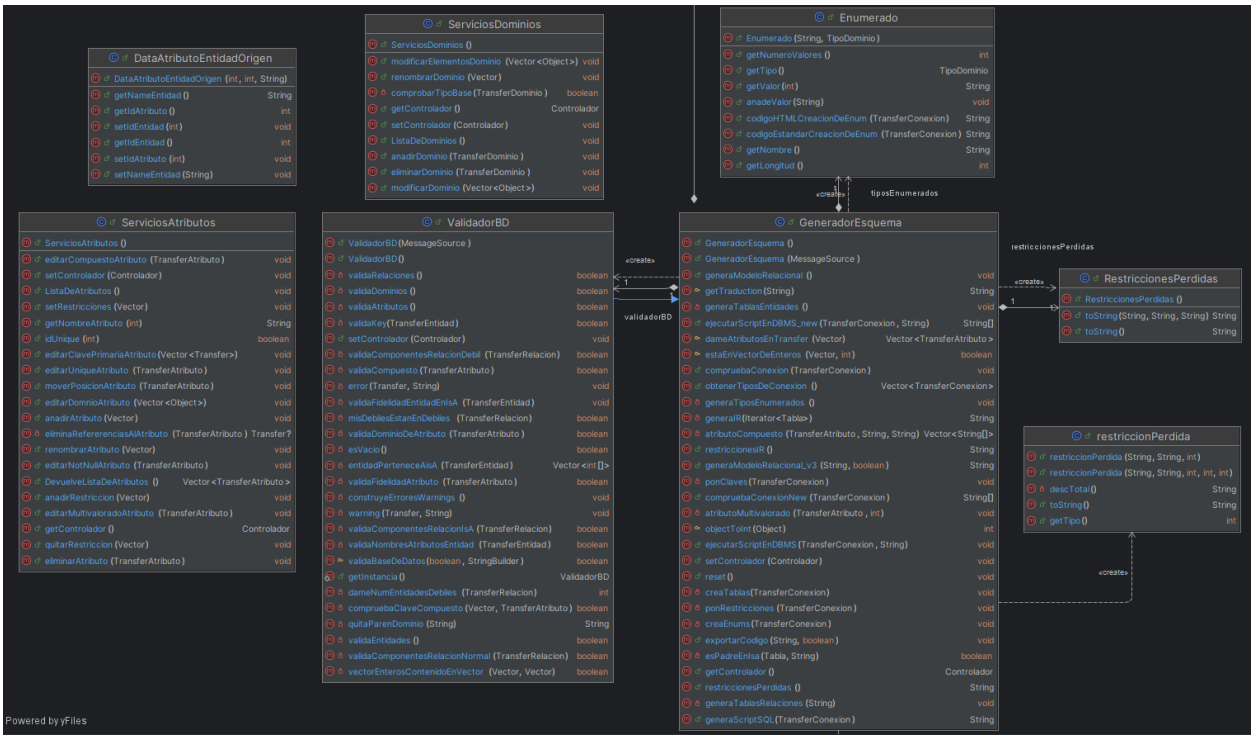

Figura 3-3 Diagrama de clase de Servicios (2/3)

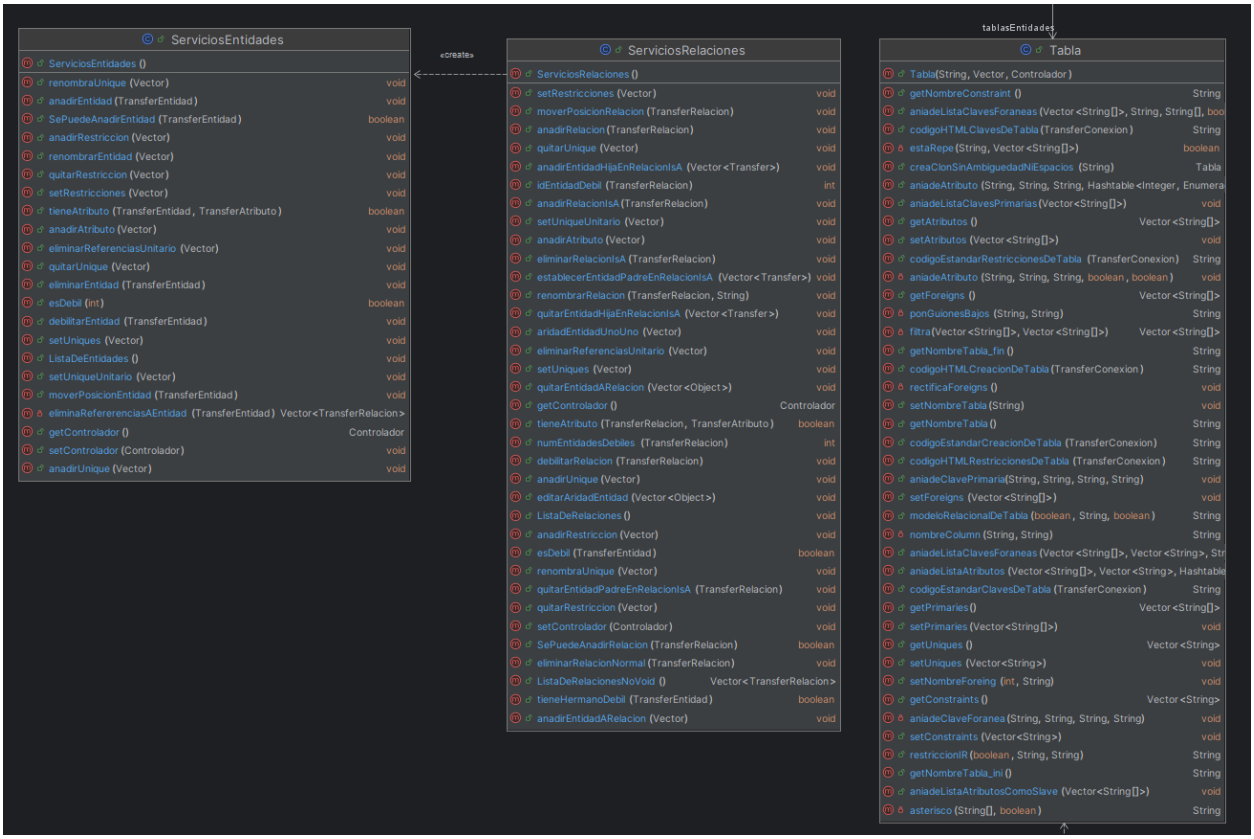

Figura 3-4 Diagrama de clase de Servicios (3/3)

3.2.2 Transfers

Representan los objetos de transferencia de datos, estructuras utilizadas para transportar información entre las capas de la aplicación sin exponer la lógica del mod

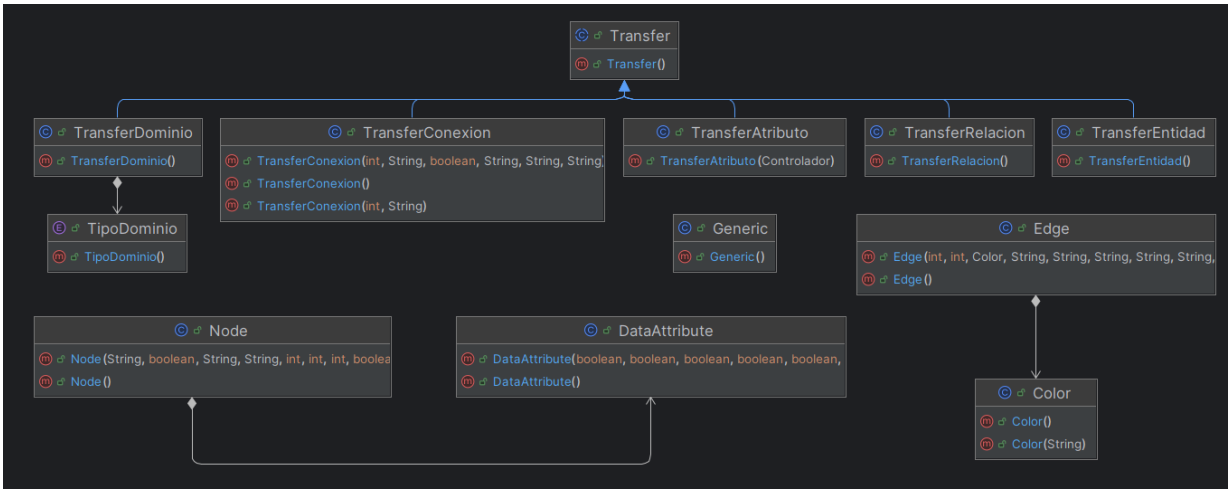

Figura 3-5 Diagrama de clase de Transfer (1/5)

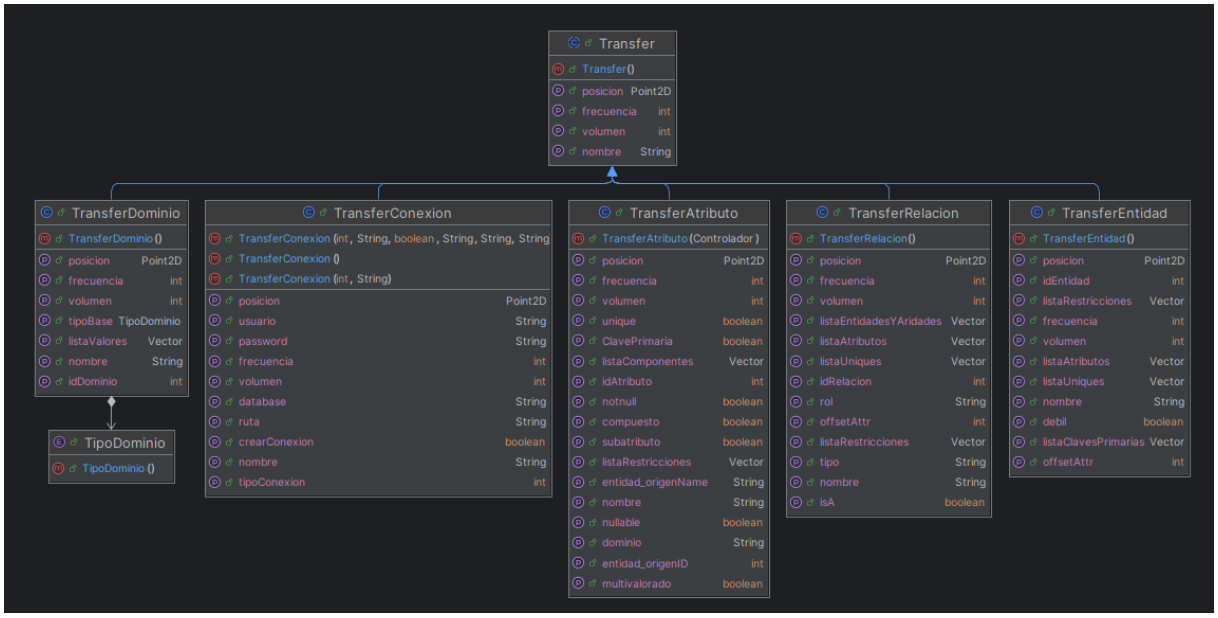

Figura 3-6 Diagrama de clase de Transfer (2/5)

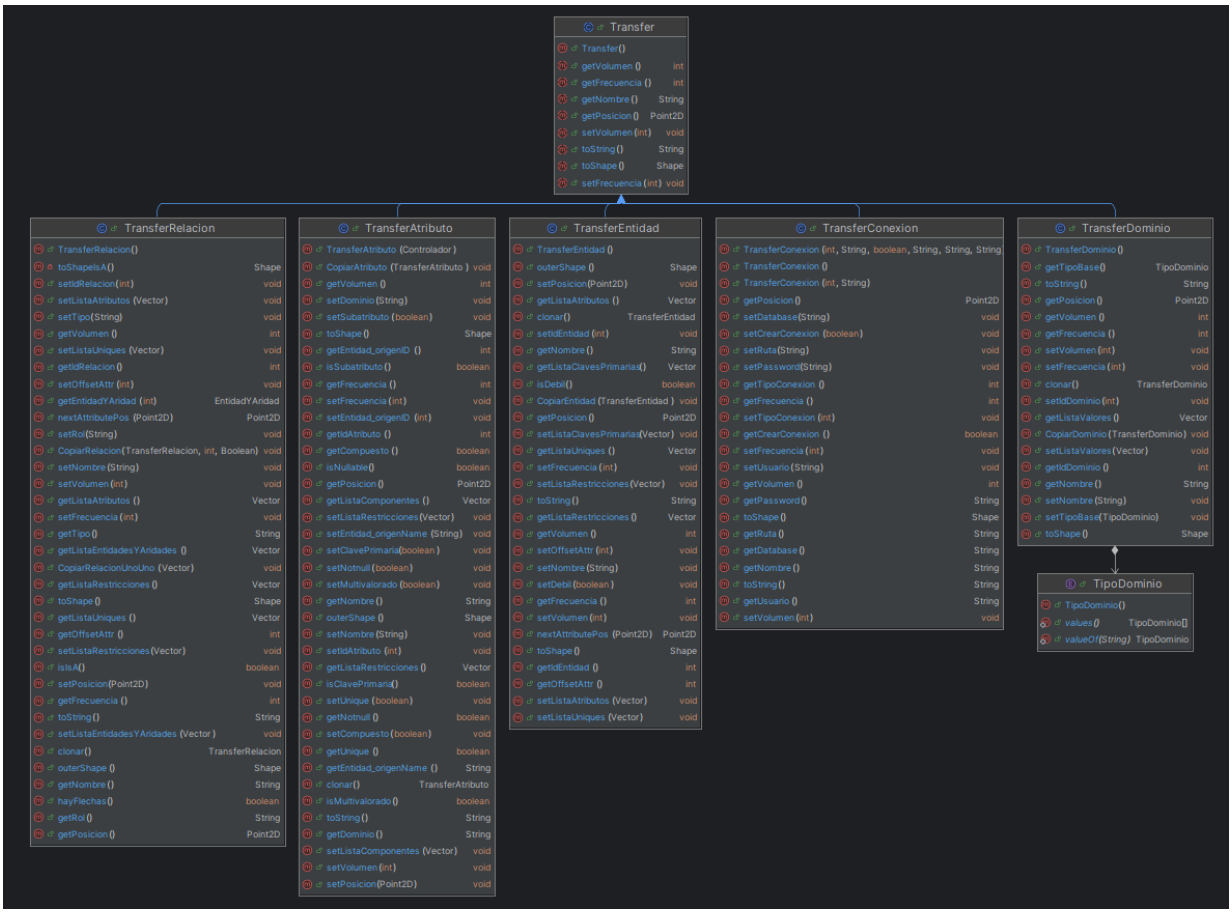

Figura 3-7 Diagrama de clase de Transfer (3/5)

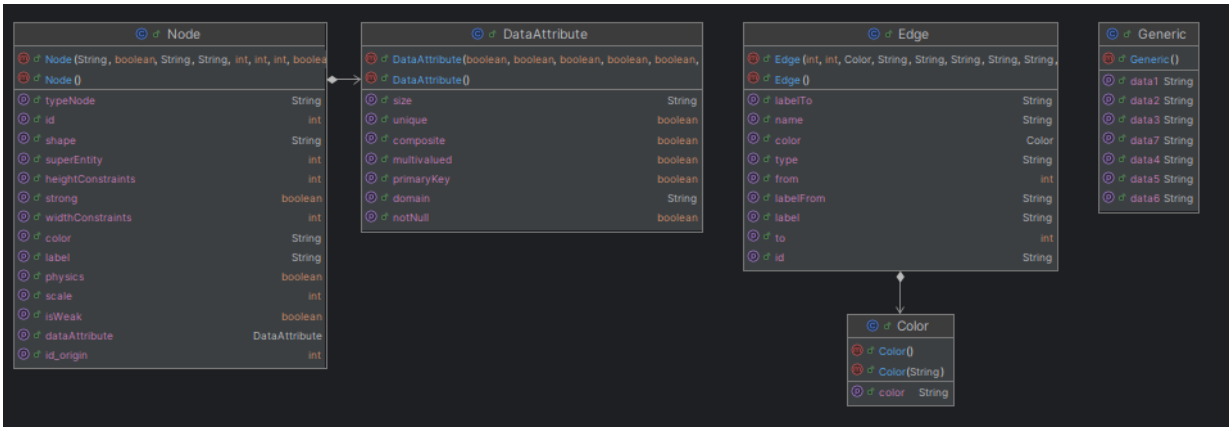

Figura 3-8 Diagrama de clase de Transfer (4/5)

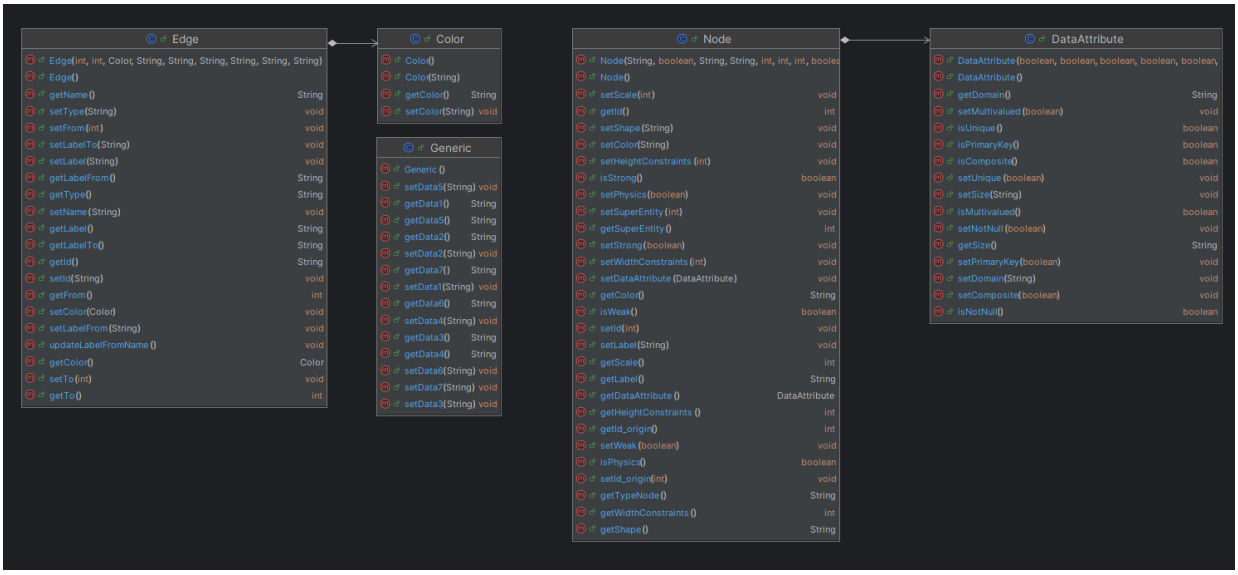

Figura 3-9 Diagrama de clase de Transfer (5/5)

3.2.3 ConectorDBMS

Funciona como un intermediario entre la lógica de la aplicación y los diferentes conectores a las bases de datos.

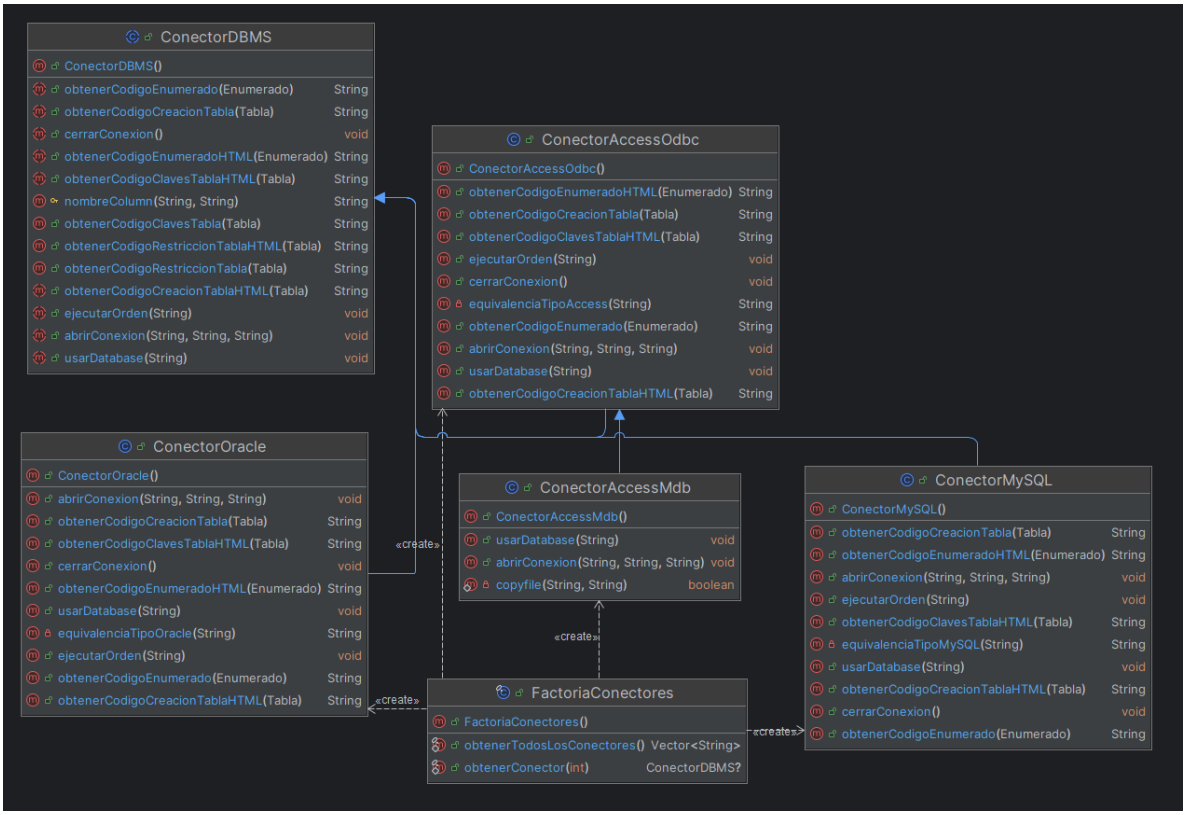

Figura 3-10 Diagrama de clase de ConectorDBMS

3.2.4 Persistencia

El paquete de persistencia se encarga de la interacción directa con la base de datos, gestionando cómo se almacenan, recuperan y actualizan los datos en el sistema.

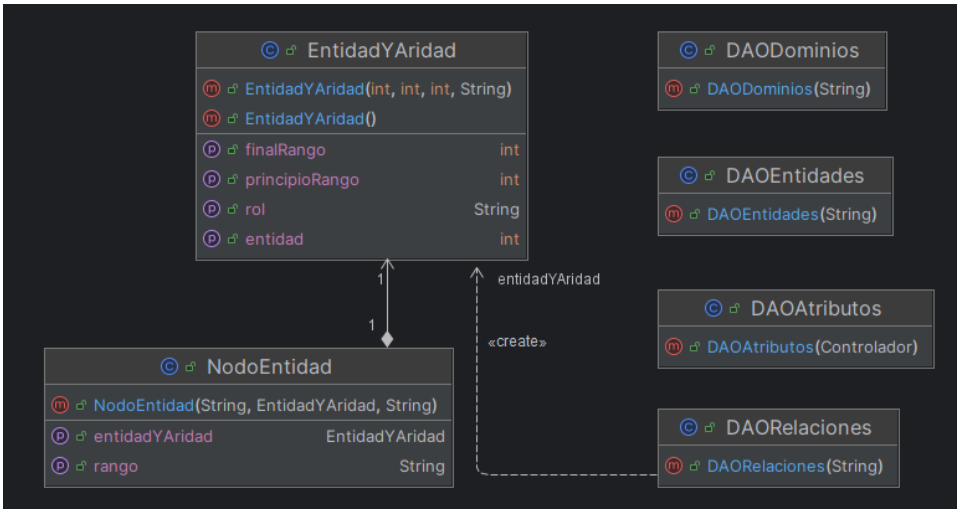

Figura 3-11 Diagrama de clase de Persistencia (1/2)

Figura 3-12 Diagrama de clase de Persistencia (2/2)

3.2.5 Controlador

El paquete de controlador actúa como intermediario entre el usuario y la aplicación, gestionando de manera eficiente las solicitudes entrantes y generando las respuestas correspondientes. Su principal función es procesar las peticiones de los usuarios asegurando la validación de los datos recibidos para garantizar su integridad. Además, se encarga de invocar los servicios del modelo, que contienen la lógica de la aplicación, para ejecutar las operaciones solicitadas y devolver al usuario los resultados adecuados.

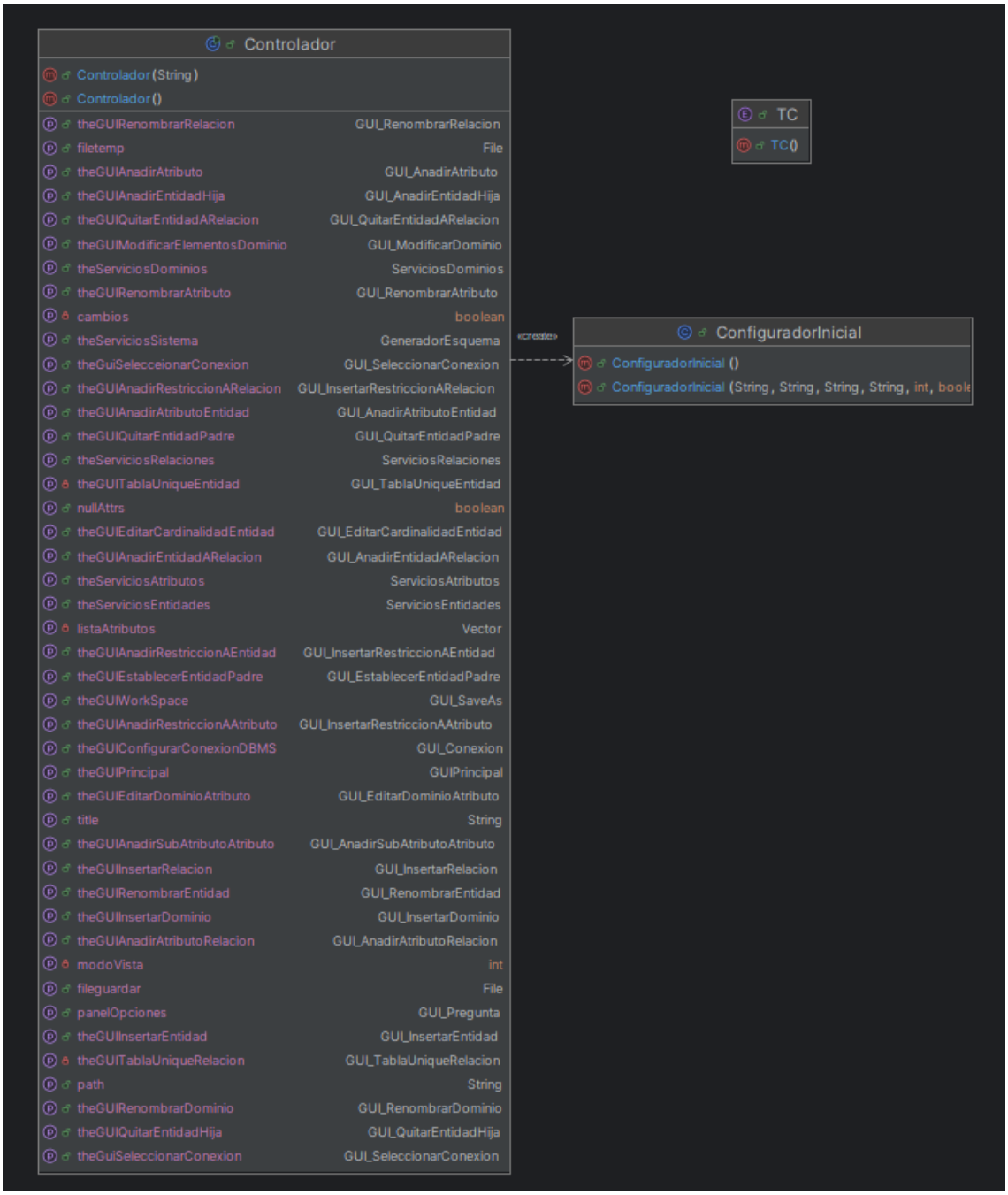

Figura 3-13 Diagrama de clase de Controlador

Capítulo 4 - Desarrollo

El desarrollo del proyecto DBCaseWeb4.0, se ha dividido en cuatro grupos. En primer lugar, uno de los principales objetivos, la adaptación de las agregaciones. Seguido de la implementación de los comandos deshacer y rehacer. Y por último las mejoras funcionales y la resolución de incidencias encontradas durante el desarrollo y pruebas de este proyecto.

4.1 Agregación

Las agregaciones se representaban mediante un nodo de la biblioteca vis en formato de imagen. La única opción posible para incluir un elemento en una agregación consistía en añadir todos los elementos creados en el canvas, sin posibilidad de dejar algún elemento no seleccionado fuera de la estructura de la agregación.

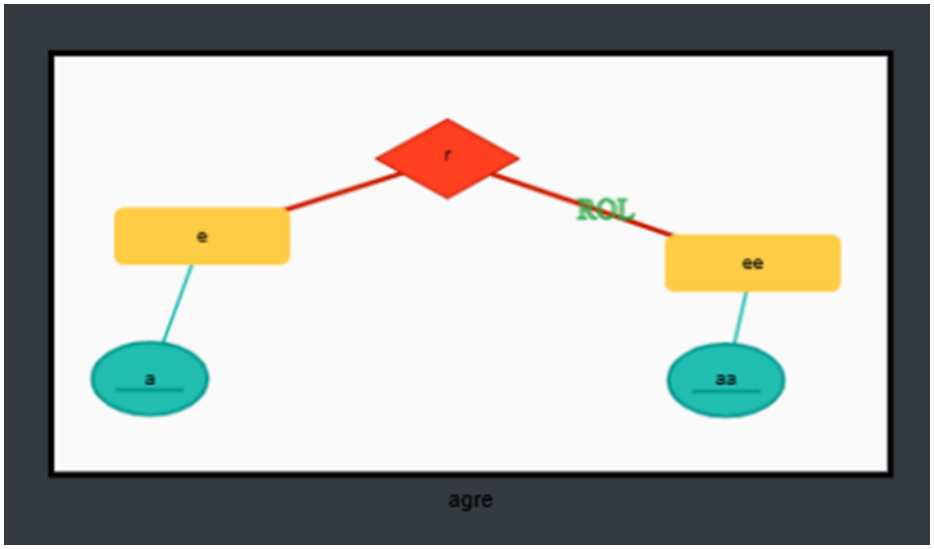

Figura 4-1 Formato de agregación anterior

Una vez creada, la única funcionalidad desarrollada era la opción de eliminar la agregación, con o sin los elementos incluidos en ella.

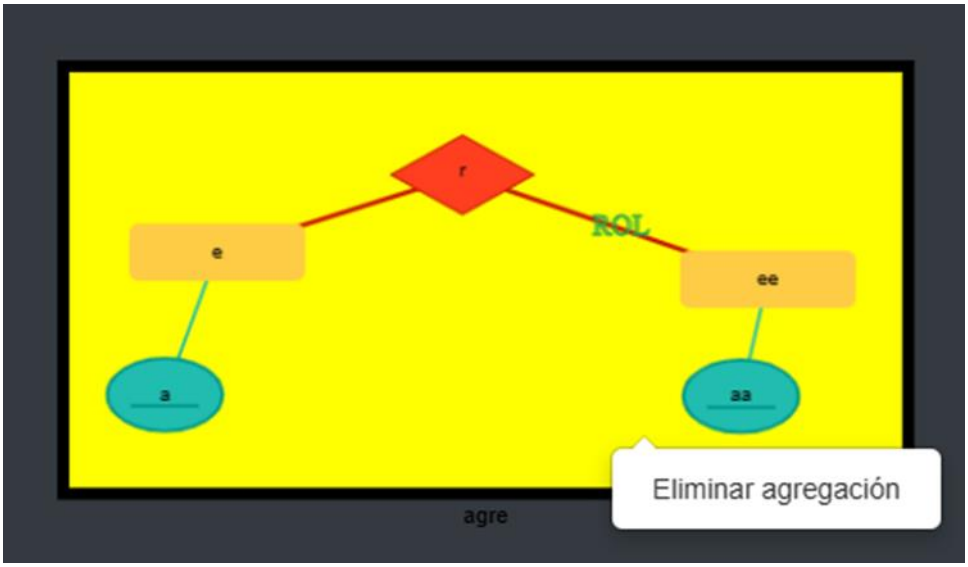

Figura 4-2 Menú de opciones de la agregación anterior

Como se puede apreciar en las figuras 4-1 y 4-2, otro de los problemas de esta implementación es la etiqueta del nombre, ya que no se controla el color de la fuente según el tema seleccionado, sino que se crea siempre en color negro. Esto supone un problema a la hora de utilizar el tema oscuro, ya que resulta muy difícil leer el texto.

Una vez entendido el desarrollo previo de este proyecto, podemos avanzar con la mejora de esta implementación. Las modificaciones realizadas se dividen en dos grupos, los desarrollos visuales y los funcionales.

4.1.1 Desarrollo visual

Los cambios visuales tratan aquellas implementaciones relacionadas con cómo se muestran las agregaciones.

Apariencia de las agregaciones

En primer lugar, se deshace el desarrollo empleado en el proyecto anterior, de forma que, al crear una agregación, no se añada un nodo de tipo imagen. En lugar de esa forma de nodo, se elige de entre las opciones de la librería vis, el nodo tipo box. Con el fin de distinguir ambos elementos, además de permitir que los elementos que

pertenecen a la agregación sean perfectamente visibles, se plantea hacer la agregación transparente, manteniendo el borde del mismo color que las entidades.

Etiqueta nombre

El color de la fuente del nombre de las agregaciones se modifica en la librería vis, según el color del tema seleccionado. En caso del tema claro, la letra será negra y con el tema oscuro, el nombre será blanco. Además, para que esta relación color-nombre siempre se mantenga, se crea un evento en js, que se dispare cuando se seleccione una de las opciones de tema. De esta forma se controla dinámicamente el color de fuente que debe de tener este elemento.

Tras eliminar el desarrollo anterior, el nombre de la agregación aparece situado en el centro del elemento. Esto entorpece tanto la vista del contenido de la agregación como la lectura del propio nombre. Para resolver esta situación colocamos la etiqueta fuera de la agregación, en el borde inferior.

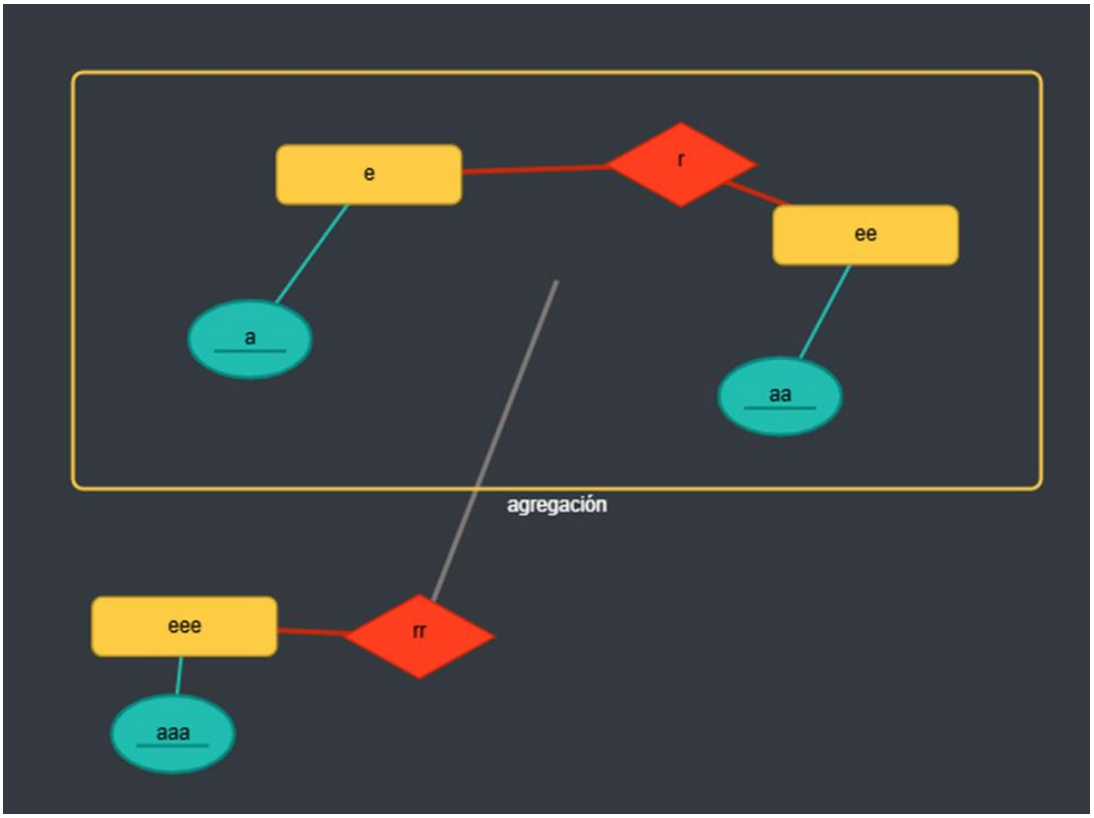

Figura 4-3 Aspecto final de las agregaciones

Aristas conectadas dentro de la agregación

En adición a esto, durante las pruebas del nuevo desarrollo se detecta un problema que presenta esta forma de representación. Las aristas se unen a los elementos desde su punto medio. Al ser la mayoría de los elementos opacos, no se aprecia, pero al ser la agregación de color transparente, se ven las aristas y "ensucia" el diagrama. Esta problemática nos obligó a cambiar de nuevo el color de este elemento. La forma de representación escogida hasta el momento es visualmente muy clara ya que es evidente que la agregación contiene a otros elementos. Es por ello por lo que se decide igualar el color de la agregación con el del fondo. Manteniendo eso sí, el borde de color amarillo como previamente.

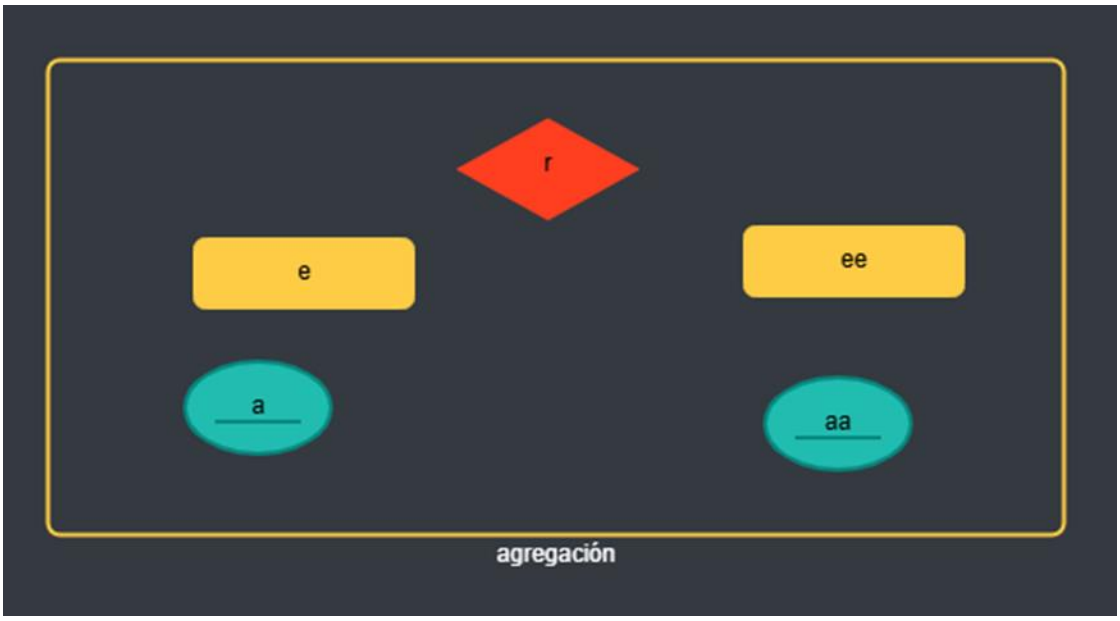

Figura 4-4 Agregación con fondo opaco

Como se aprecia en la imagen, no se muestran las aristas que conectan los diferentes elementos de la agregación. Esto se debe al renderizado de los elementos en la librería vis. Las aristas siempre se dibujan por debajo de los nodos, por cuestiones estéticas principalmente. Al modificar el color de fondo de la agregación, las aristas no se muestran correctamente. Precisamente por este motivo se decide volver a la solución anterior, quedando la red representada como se muestra en la figura 4-3.

Color de las aristas

Finalmente se ha decidido mostrar los elementos como en la figura 4-3. La solución a la que se ha llegado para mitigar el conflicto visual que presentan las aristas conectadas a las agregaciones, es la de mostrar dichas aristas con un color que permita diferenciarlas del resto. Se ha elegido para ello un gris oscuro, de forma que no desentone con la aplicación, pero que no deje de ser identificable.

4.1.2 Desarrollo funcional

Las modificaciones funcionales se refieren a las implementaciones realizadas en relación al funcionamiento de las agregaciones.

Identificación de agregaciones y sus elementos

Identificar a nivel de código qué elemento de bases de datos representa cada nodo era sencillo hasta ahora, ya que cada uno tenía una forma. En el nuevo desarrollo, las agregaciones y las entidades se representan mediante la forma "box" o caja. Esto supone un problema a la hora de identificar una agregación. Para resolverlo, se crea una nueva propiedad en los nodos, "isSuperEntity", que permitirá identificarlas fácilmente.

Por otro lado, los elementos que forman parte de una agregación se identifican porque se añaden a una estructura de nodo adicional creada para ello. Siempre que se quiera consultar si un elemento forma parte de una agregación es necesario hacer una consulta sobre esta estructura. Con la intención de simplificar dicha consulta, creamos otra propiedad que identifique si un nodo pertenece a una agregación.

Orden de renderizado de los elementos

El primer problema de la implementación funcional de las agregaciones lo provocaba el orden de renderizado. Este orden venía dado por el orden de creación de los elementos. Debido a ello, al crear los nodos y posteriormente la agregación, esta última quedaba dibujada por delante del resto de nodos. Este comportamiento no

permitía seleccionar los nodos creados antes y, por tanto, no se podía interactuar con ellos si el nodo de la agregación se encontraba situado sobre ellos.

Se averigua que el orden de representación de los nodos depende del identificador de los elementos. Aquellos con un valor menor se dibujan antes que los de mayor valor. Hasta entonces el identificador se asignaba aleatoriamente pero en orden ascendente de creación, dibujándose primero el primer elemento creado.

Para poder controlar de forma efectiva este funcionamiento, se crea un contador de elementos, que se emplea para asignar el identificador a cada nodo y otro contador de agregaciones que se usa para asignar identificadores a estos otros componentes.

Llevar a cabo este desarrollo pasó por dos etapas. Una primera en la que se dio una solución aparentemente efectiva y que posteriormente se modificó de forma que el desarrollo fuera mucho más eficiente y efectivo. Se explican a continuación ambas soluciones.

Para poder forzar el renderizado de la agregación y que se muestre por detrás del resto de elementos en vez de por delante, se le debe asignar el id más pequeño de la red, el 0. En el momento de creación de cada uno de los elementos se les asigna como id, el valor del contador de nodos. Al añadir una agregación a la red, se tienen que haber creado otros nodos previamente y, por lo tanto, el id 0 ya estará asignado a uno de ellos. Como no pueden existir dos elementos con el mismo identificador, se tiene que modificar ese valor. Para ello, y para mantener el orden de creación del resto de elementos, se aumenta en una unidad el número de identificación de todos los nodos. Este es un proceso que se debe realizar de manera inversa, es decir, que el id del primer elemento que se modifica, será el del último nodo agregado. Así se libera el id usado para asignarlo a otro y que no pueda haber pérdida de datos ni identificadores duplicados al realizar esta operación. Tras esta modificación, se crea la agregación con id 0.

Durante este recorrido de todos los nodos de la red, se aprovecha para modificar la propiedad superEntity de los elementos que formen parte de la agregación.

Los cambios adicionales de esta modificación repercuten únicamente en el conjunto de aristas que forman la red. Estos elementos tienen las propiedades form y to, que guardan el id de los nodos que conectan. Como estos identificadores se modifican en el conjunto de nodos, también deben hacerlo en el conjunto de aristas. Para ello se recorre la lista de aristas aumentando en una unidad el valor de dichos campos.

De esta manera se mantiene la coherencia de la red con respecto a cómo estaba antes de crear la agregación.

Esta es la primera solución planteada para la resolución del renderizado. Tras analizarlo con más detenimiento, se decidió modificar el desarrollo. El motivo principal fue el elevado coste que provocaba recorrer todos los nodos y aristas de la red modificando los ids. Para poder evitarlo, se crea un segundo contador. Se implementa un contador de agregaciones, que se inicializa a 0 y se modifica el anterior contador de nodos comunes inicializado a 1000. Quedan por tanto reservados 1000 ids para las agregaciones.

Con esta nueva solución, resulta mucho más sencillo controlar el renderizado, además de tratarse de un desarrollo mucho más sencillo y eficiente.

Cálculo de coordenadas y dimensiones de una agregación

Para calcular la ubicación de la agregación se recorren los nodos que pertenecen a ella en busca las coordenadas mínimas y máximas del eje X y del eje Y. Las coordenadas x e y de los elementos representan el punto medio de cada nodo de la red. Sumando y posteriormente dividiendo entre 2 el máximo y mínimo de cada eje, se obtienen las coordenadas del centro de la agregación.

Con la librería vis, al mover los nodos, las variables X e Y no se actualizan. Esto supone un problema ya que al mover los nodos de la red y crear posteriormente una agregación, la posición de dicho elemento se calcula en base a valores obsoletos.

Para solucionar esto, se emplea el evento dragEnd, que se activa al finalizar el movimiento de un nodo, actualizando sus nuevas coordenadas.

Para calcular las dimensiones del elemento, se aprovecha el cálculo anterior. Se utilizan los puntos máximo y mínimo de cada eje y se calcula su diferencia. De esta forma

se calculan las distancias entre los nodos más extremos de cada eje. Con las distancias de los ejes se obtienen la altura y el ancho aproximados de la agregación.

Los valores de estos cálculos se aumentan para dejar algo de margen entre los bordes de los elementos situados en los extremos y los bordes de la propia agregación. Además, el cálculo de distancias se realiza con los puntos medios de los elementos, no con las coordenadas de sus extremos, por lo que, en cualquier caso, habría que aumentar el valor de las distancias calculadas.

Redimensionamiento de las agregaciones

La posibilidad de mover los elementos que componen una agregación es una de las características a mejorar en el desarrollo de estos elementos en relación con proyectos anteriores.

El reto que esta mejora ha supuesto es la redimensión de la agregación al moverse uno de sus elementos. Para ello, hacemos uso de dos desarrollos explicados y realizados anteriormente.

En primer lugar, se modifica el funcionamiento en JavaScript del listener dragEnd. En caso de que el nodo movido es parte de una agregación, se vuelven a calcular sus dimensiones como se explica en el punto anterior. Al haber guardado las coordenadas de los nodos movidos, el recálculo de las dimensiones de la agregación se realiza con los datos actualizados.

Movimiento conjunto de la agregación y sus componentes

Otro punto esencial por desarrollar es el de mover la agregación a través del canvas y que los nodos contenidos en ella se muevan de manera conjunta.

Se emplea para este desarrollo un event listener que se active al mover un elemento. En caso de que ese elemento sea una agregación, se marcan como seleccionados todos los nodos que pertenecen a dicha agregación.

De esta manera, al mover la agregación, se moverán todos sus elementos al mismo tiempo.

Control en la creación de agregaciones

El primer criterio utilizado a la hora de elegir qué elementos forman parte de la agregación es sencillo. Serían parte de la agregación aquellos elementos que fueran seleccionados. Tras un análisis más detallado, esta forma de creación de agregaciones se modifica con el objetivo de construir una solución más sencilla y coherente. De esta forma solo se muestra la opción que permite crear una agregación en el menú de las relaciones, opción que antes también se mostraba en las entidades. Son por tanto las relaciones seleccionadas por el usuario las que se incluyen en las agregaciones.

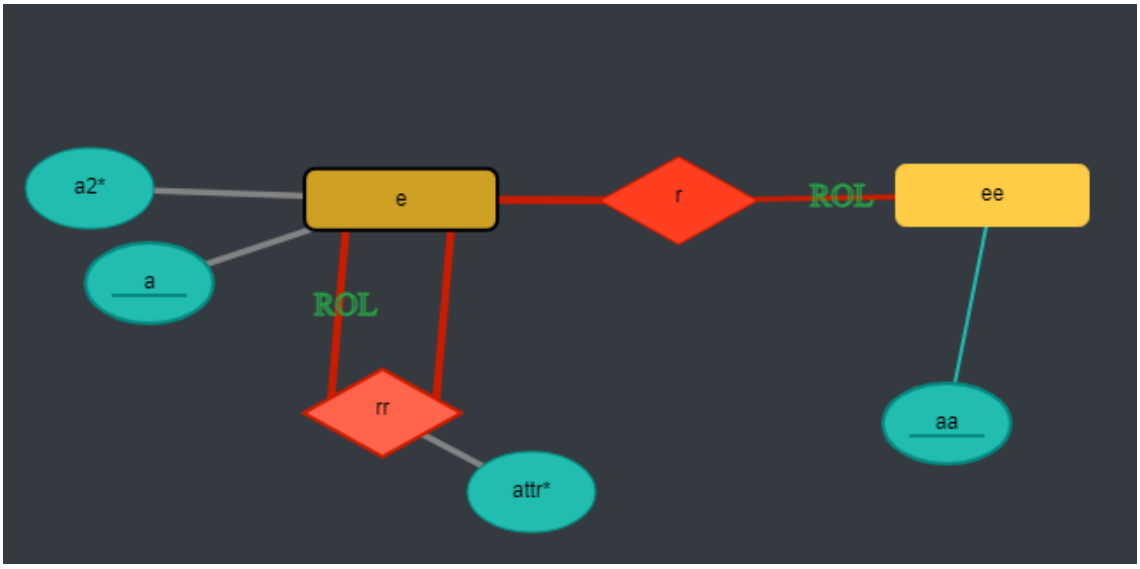

Figura 4-5 Esquema sin agregación

Adicionalmente, se añade un control para ciertos elementos. Para las relaciones seleccionadas, se incluirán en la agregación aquellas entidades que estén conectadas a ellas. Por ejemplo, en la figura 4-5 se aprecia el esquema con dos elementos seleccionados, la entidad *e* y la relación *rr*. Si únicamente se selecciona y añade la relación r a una agregación, la entidad e también será añadida.

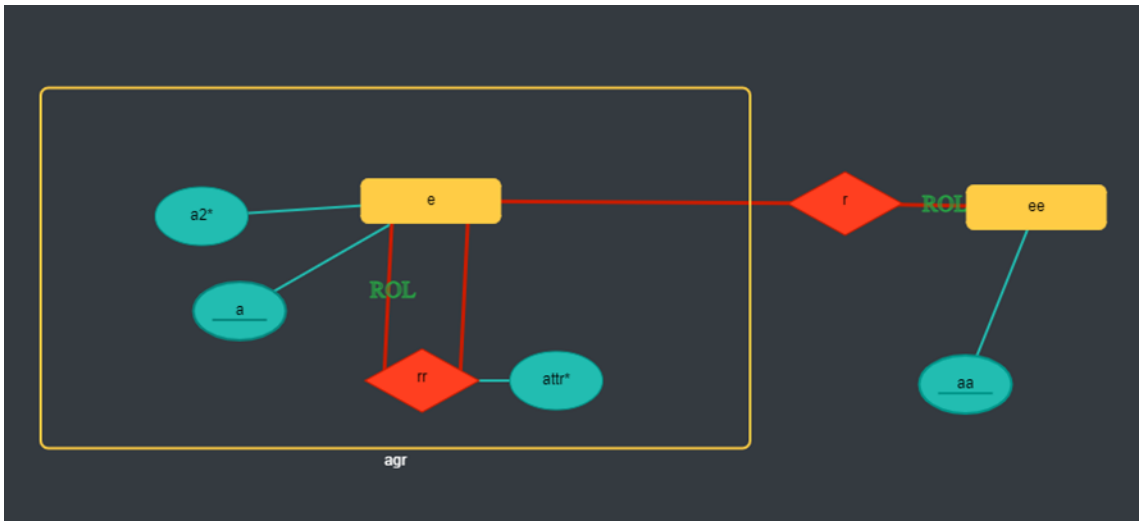

Figura 4-6 Esquema con agregación

En el caso de las relaciones y entidades todos sus atributos también formarán parte de la agregación. En el caso de los atributos compuestos, también se añadirán a la agregación todos sus subatributos, siempre y cuando el atributo y la entidad a la que pertenezcan, cumplan con los criterios citados anteriormente. Como ejemplo ilustrativo empleamos el caso anterior. Para formar una agregación con la entidad e y la relación rr de forma correcta, sus atributos también deben formar parte de ella, quedando como se aprecia en la segunda imagen (figura 4-6).

Una vez creada, la única manera para añadir un elemento a la agregación es crear un atributo sobre alguno de los elementos ya contenidos o añadir una entidad a una relación de la agregación. En cualquier otro caso se deberá eliminar la agregación y volver a crearla.

Eliminación de agregaciones

Según el funcionamiento de la aplicación original, al eliminar la agregación se da a elegir si eliminar únicamente la agregación o eliminar también sus elementos.

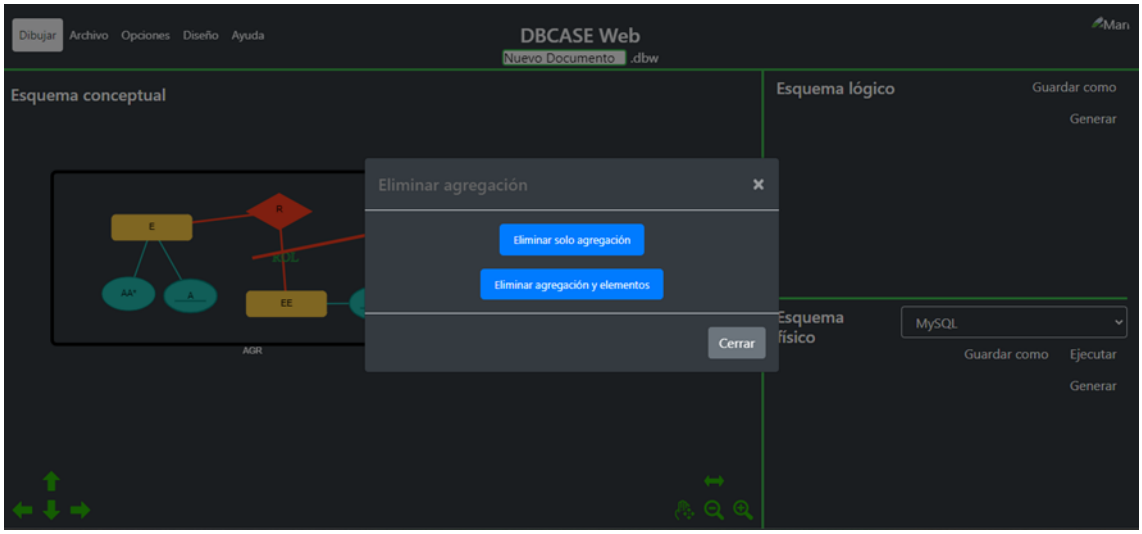

Figura 4-7 Modal de eliminación de agregaciones

Si se selecciona la primera opción, el desarrollo no cambia. En cambio, si se selecciona la segunda, debemos actualizar la propiedad "superEntity" de todos los elementos de la agregación.

Opciones de modificación de la agregación

Hasta ahora, las opciones que se ofrecían en el menú desplegado al seleccionar la agregación eran limitadas. Únicamente se permitía eliminar la agregación, dando las opciones de eliminar junto a la agregación los elementos que pertenecen a ella.

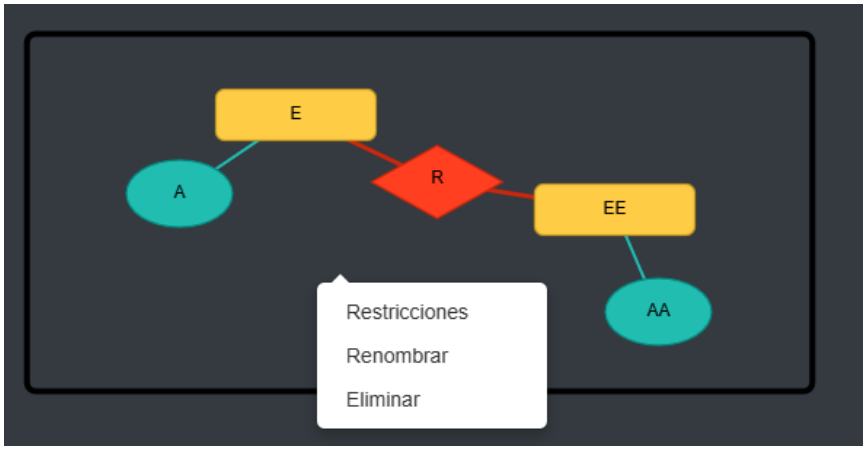

Figura 4-8 Nuevo menú de opciones de las agregaciones

Además de esta opción se han añadido la opción de renombrar la agregación y de añadir restricciones de la misma manera que se venía haciendo con las entidades (figura 4-8).

Creación de varias agregaciones

Como última modificación sobre estos componentes, se ha decidido permitir que existan varias agregaciones. El motivo por el que originalmente no se permitía esta casuística es debido a la rareza de las bases de datos con estas características en el contenido de la asignatura de bases de datos. Al tratarse de una aplicación para los alumnos de esta asignatura esto tenía sentido, pero se ha considerado una buena opción a implementar con el objetivo de no limitar las opciones de los estudiantes a la hora de diseñar otras bases de datos.

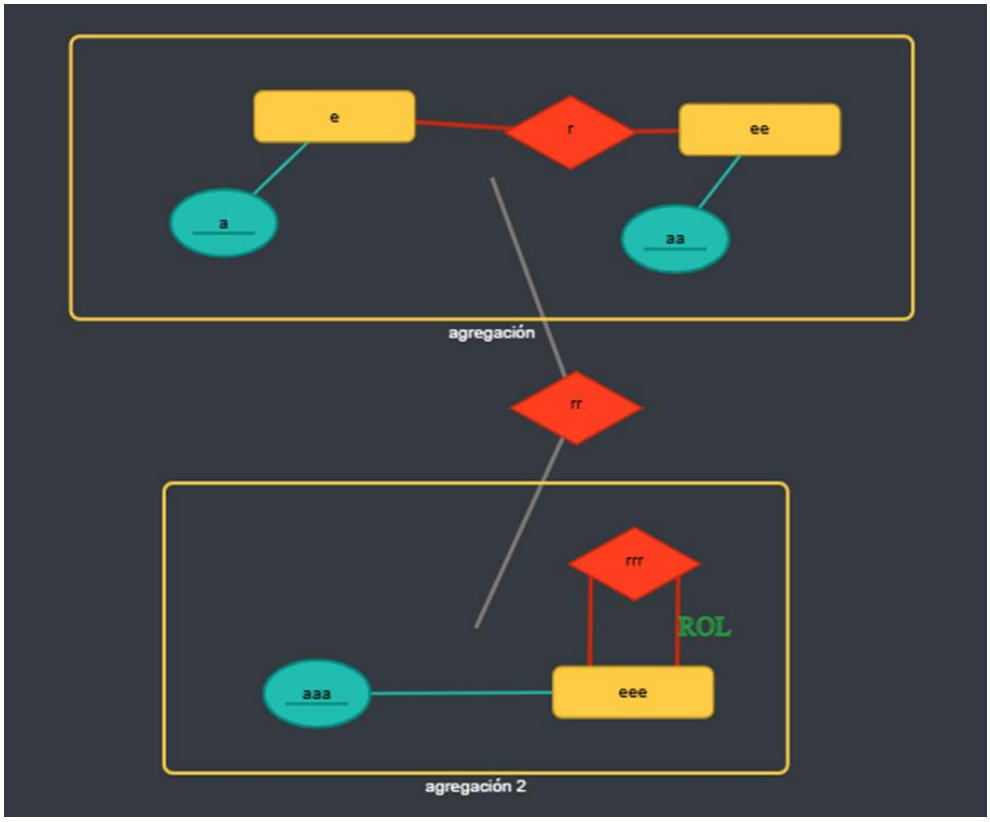

Figura 4-9 Esquema con 2 agregaciones

En primer lugar, el cambio que se ha realizado es el de adaptar los desarrollos realizados previamente. El mayor impacto de esta modificación gira entorno a la variable booleana "superEntity". Esta propiedad de los nodos que informaba si un elemento pertenecía a una agregación, es ahora un identificador de la agregación asociada al elemento. Es decir, las variables que empleamos servirán como clave para relacionar una agregación con los elementos que la componen. En el caso de las que no pertenezcan a una agregación, guardarán un valor negativo. En la figura 4-9 se aprecia el resultado de esta modificación.

4.2 Deshacer y rehacer

Las funcionalidades de deshacer y rehacer cambios son uno de los principales objetivos del proyecto, ya que se trata de acciones altamente utilizadas en cualquier herramienta y su implementación aporta a este desarrollo una mayor usabilidad.

4.2.1 Deshacer (CTRL + Z)

Para realizar la implementación, se deben guardar los cambios realizados en el canvas en una estructura de datos. Se emplea para ello un array donde guardar el historial de acciones en el que se almacenarán el tipo de acción llevada a cabo y un body con la información del nodo o arista, previa al cambio. Esta estructura en la que se guarda la información funciona como una pila, de forma que, al deshacer, se saca del array la última acción agregada.

Para la correcta ejecución de esta funcionalidad, es necesario emplear un event listener que se active cuando se pulse las teclas alguna tecla del teclado y ejecute la función de deshacer al pulsar la combinación de 'Ctrl' y 'z', ya sea en mayúscula o minúscula.

4.2.2 Rehacer (CTRL + Y)

Para la realización de este desarrollo, se replica la estructura de datos creada para el funcionamiento de ctrl+z. Se utiliza de nuevo una pila para rehacer los cambios que se hayan eliminado con la ejecución del comando deshacer. Para ello, se almacenan las acciones guardadas en el buffer actionHistory para que posteriormente

se puedan recrear. Sin embargo, a diferencia del comando deshacer, en este caso se ejecutará la acción guardada en vez de la inversa, de esta forma podremos rehacer los cambios deshechos al pulsar ctrl+z.

Es esencial mencionar que se debe tener en cuenta que, al deshacer una acción y después ejecutar otra nueva acción, el contenido almacenado en el buffer de rehacer se elimina.

En este caso, actualizaremos con datos el buffer, cada vez que se ejecute ctrl+z, ya que estas son las únicas acciones que se podrán rehacer.

4.2.3 Botones deshacer y rehacer

Tras la implementación de los comandos, resulta coherente con el proyecto y con las aplicaciones comunes de diseño, que se cuente también con botones con esta misma funcionalidad.

Para ello se utilizan iconos de Bootstrap [9] en forma de flechas que apuntan a la izquierda y a la derecha y que sean intuitivas para los usuarios. La localización de estos botones es también muy importante. En la mayoría de las aplicaciones se suelen situar en la esquina superior izquierda y por ello, en este proyecto también se colocan en esa ubicación. Se muestran las flechas de color verde al igual que el del resto de iconos del canvas. Además, se configura su sombreado al pasar por encima, al igual que sucede con los botones de los esquemas lógico y físico. Como una mejora adicional y para que se localicen más fácilmente, se coloca un borde verde en torno a los iconos de forma que sean más intuitivos y se entienda que son botones.

4.2.4 Mensajes informativos al deshacer y rehacer

La implementación de las opciones deshacer y rehacer en el canvas mejoran significativamente la experiencia de usuario de la aplicación. Para terminar de realizar este desarrollo se decide incluir mensajes informativos cuando no hay acciones que deshacer o rehacer. Se emplea para ello una alerta en js que se activa cada vez que se intenta deshacer o rehacer una acción, ya sea por teclado o botón.

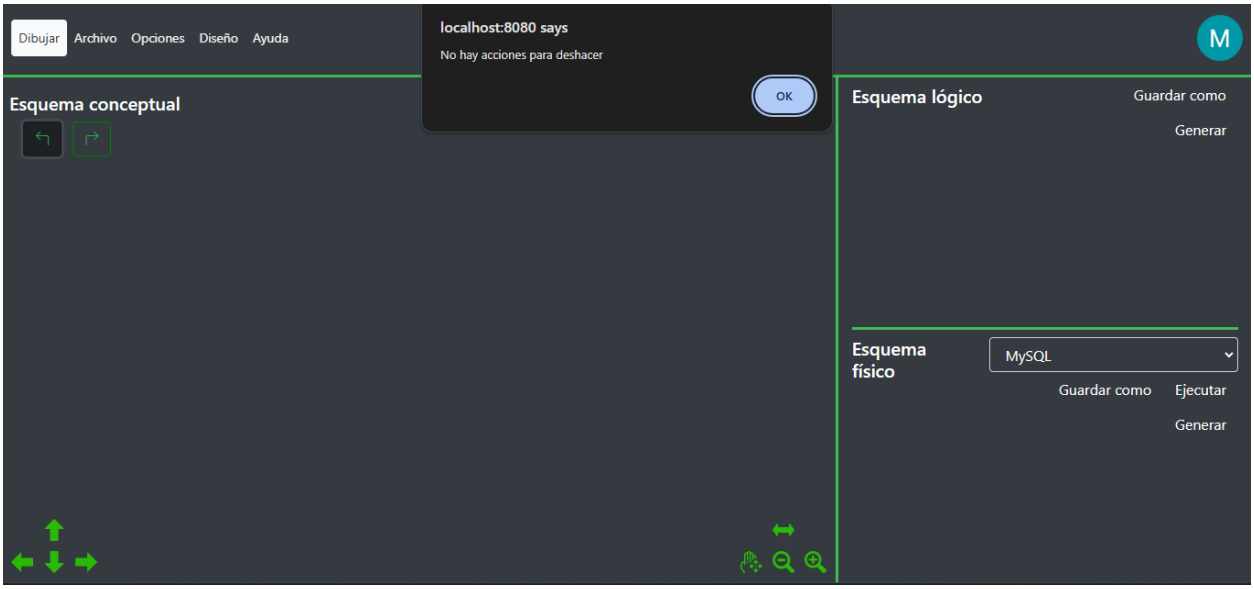

Figura 4-10 Mensaje sin diagrama indicando que no hay acciones que deshacer

En la imagen 4-10 se muestra un canvas nuevo y vacío y el mensaje mostrado cuando no hay acciones que deshacer. Del mismo modo, en la imagen 4-11 se muestra la notificación mostrada cuando no hay acciones que rehacer.

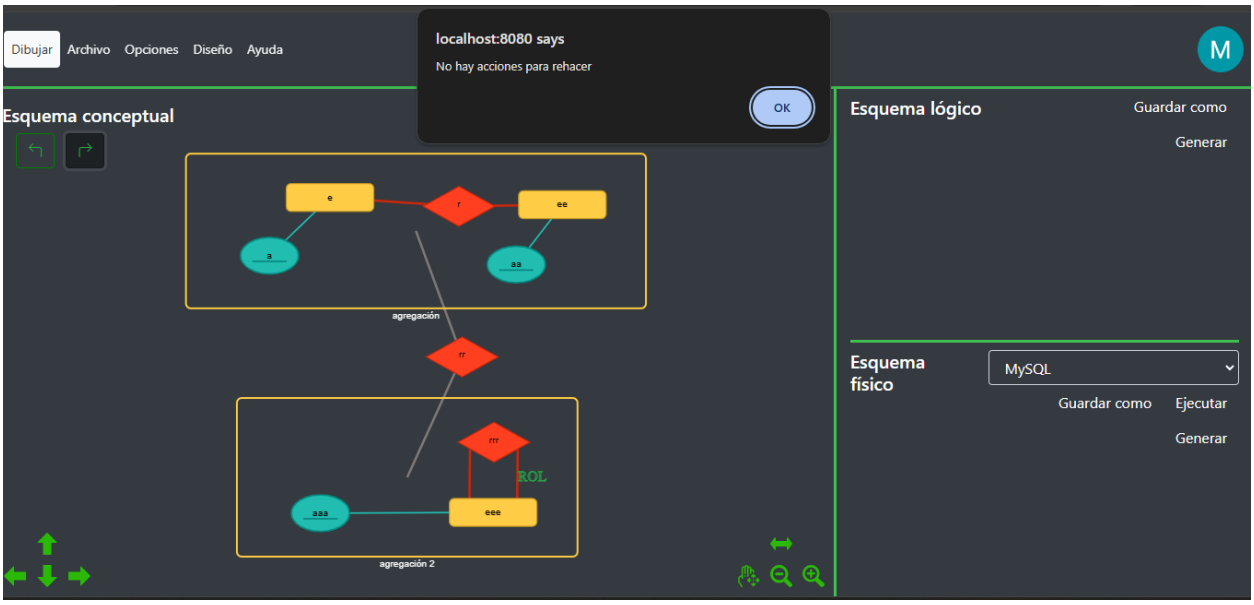

Figura 4-11 Mensaje con diagrama indicando que no hay acciones que rehacer

4.3 Mejoras funcionales

En este apartado se explican las implementaciones realizadas con el fin de mejorar el funcionamiento de la aplicación y enriquecer la experiencia de usuario.

Visión de elementos seleccionados

Con el objetivo de mejorar la visualización de los elementos seleccionados, se modifican los colores. En el proyecto anterior, se modificaron los colores de fondo de los nodos cuando estos son seleccionados.

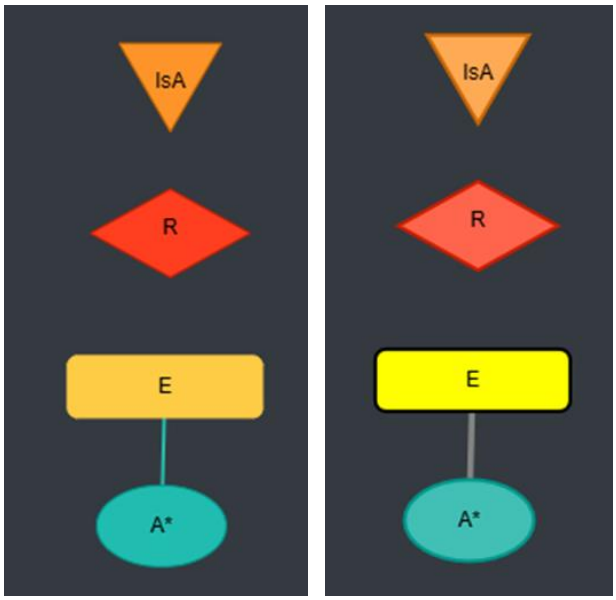

Figura 4-12 Vista modo oscuro de elementos sin seleccionar (izq) y seleccionados (dch) en el proyecto anterior

Como se aprecia en las imágenes 4-12 y 4-13, en algunos casos los colores son muy llamativos y en otros más sutiles. En este proyecto se ha querido homogeneizar esta situación, de forma que la elección de tonalidades sea más coherente.

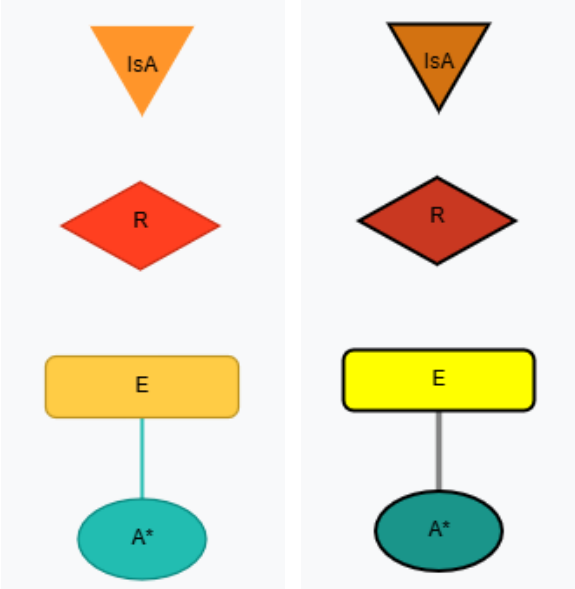

Figura 4-13 Vista modo claro de elementos sin seleccionar (izq) y seleccionados (dch) en el proyecto anterior

Con este propósito, se decide oscurecer el color de fondo de los elementos y engrosar sutilmente el borde de las figuras de forma que se detecte de manera inmediata e inequívoca qué elementos están seleccionados y cuáles no.

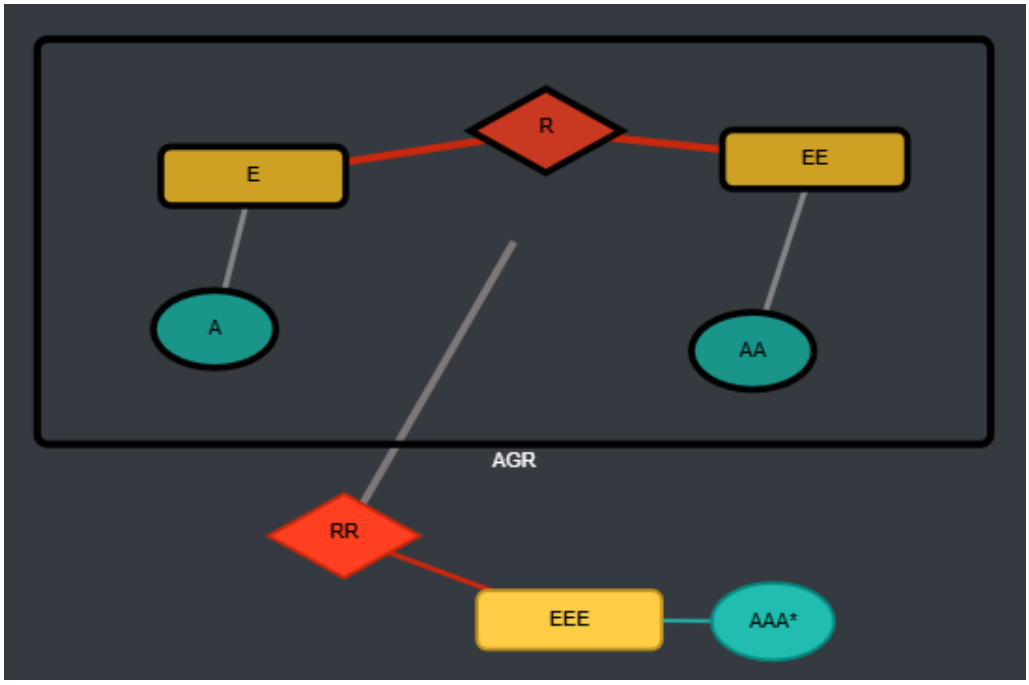

Figura 4-14 Vista en modo oscuro de los elementos seleccionados

Ejemplo de esta implementación es la figura 4-14, donde se aprecia que los elementos de la agregación están seleccionados y el resto no lo están. También incluimos un esquema con tema claro para ilustrar la visualización de este desarrollo en ambos casos (Figura 4.15).

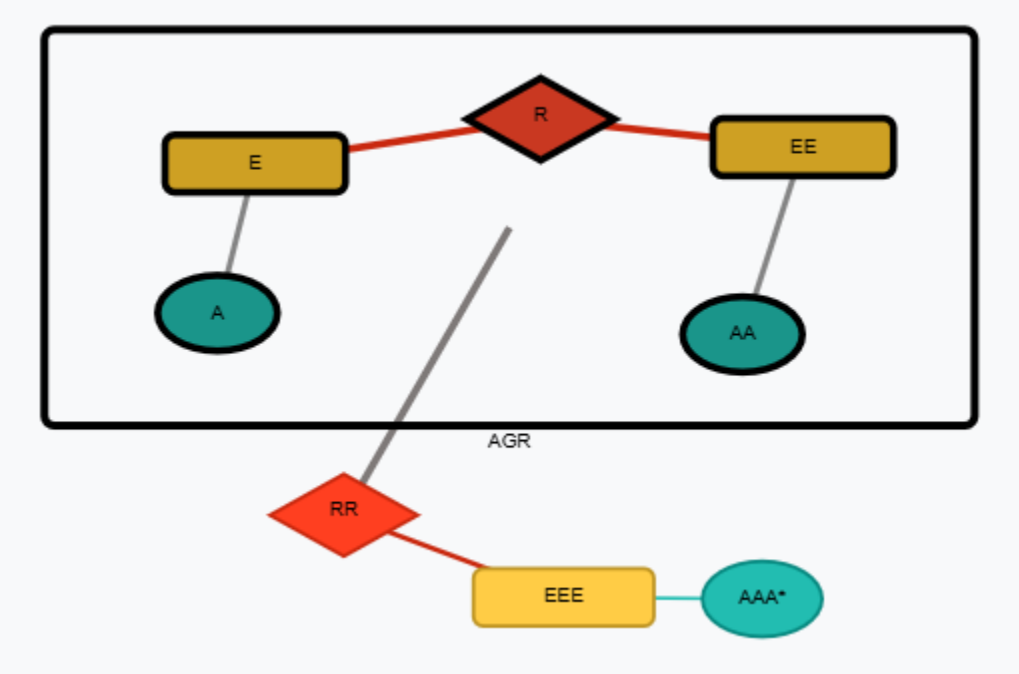

Figura 4-15 Vista en modo claro de los elementos seleccionados

Mejora de textos en los menús de opciones

La intención de este cambio es el de aportar valor a la aplicación, algo que pasa por la mejora de la experiencia de usuario. En los diferentes menús de opciones, se observan algunos textos que no describen de la mejor manera la acción que se lleva a cabo pulsando ese botón. Por ejemplo, en las relaciones, la opción "Editar Relación" (figura 4-16), solo permite al usuario cambiar el nombre de esta. Modificando el texto por "Renombrar" se comprende mejor la funcionalidad real de esta opción. La misma modificación, también se ha hecho sobre los menús de las agregaciones.

En las relaciones también se decide cambiar el nombre de la opción "Editar la cardinalidad o rol de una relación" por "Modificar" (imagen 4-19). Con este cambio se consigue un menú más sencillo con texto más breves.

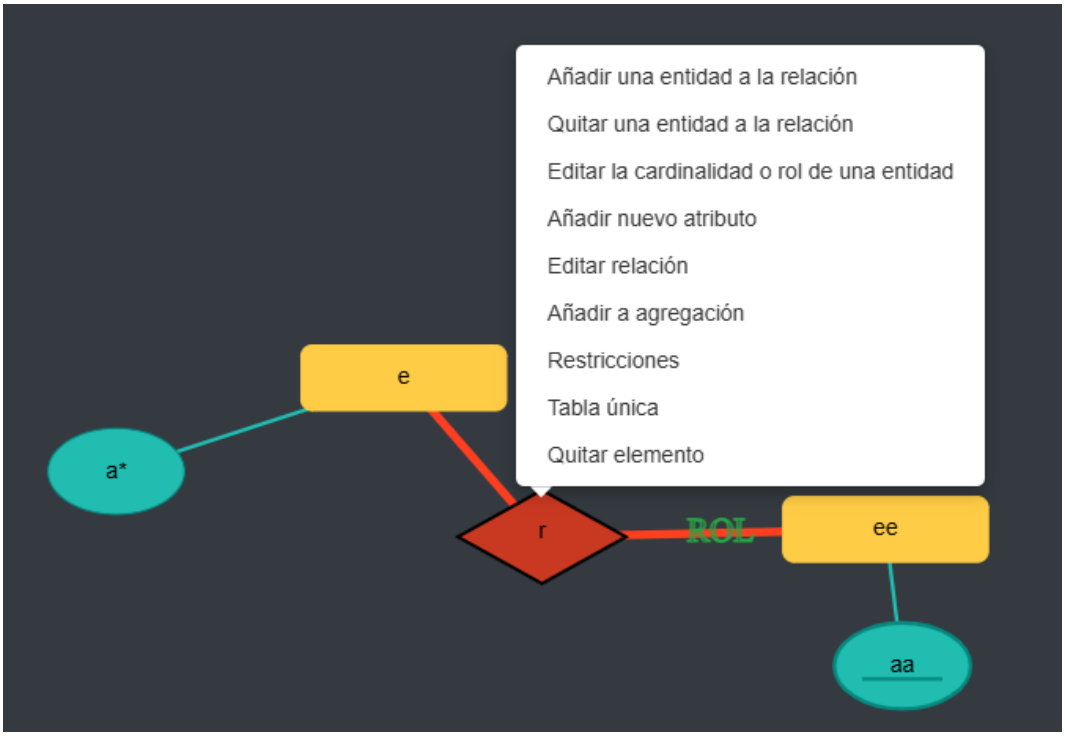

Figura 4-16 Antiguo menú de opciones de una relación

Al igual que con la opción de editar, sucede con la opción "Quitar elemento", que aparece en los menús de todos los elementos, no solo en el de las relaciones. Teniendo en cuenta que esta opción elimina el nodo seleccionado parece más razonable que el texto que la describe sea "Eliminar". Este cambio se refleja en todos los menús de los elementos que existen.

Un nombre similar se encuentra en el menú de opciones de las generalizaciones. En este caso, como se refiere a quitar una relación de una entidad padre o hija, el texto si refleja la función que desempeña y no se modifica.

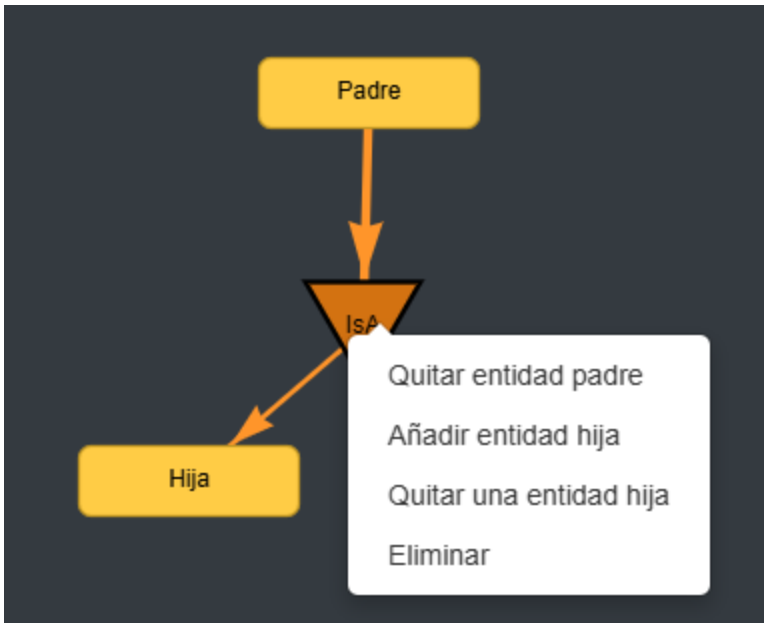

Figura 4-17 Menú de generalización con padre e hijos

Otra de las modificaciones estudiadas sobre este menú es modificar el texto de la opción "Establecer entidad padre", de forma que sea más homogéneo y coherente con el del botón que añade entidades hijas.

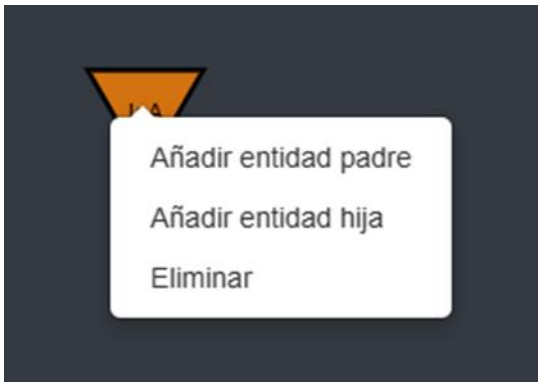

Figura 4-18 Menú de generalización sin padre ni hijos

Como se puede apreciar en las figuras anteriores, las opciones mostradas en el menú de las agregaciones se han modificado de forma que varían según las condiciones del elemento. De esta forma, si la generalización no tiene entidad padre ni hija, las opciones para quitarlas no se mostrarán. Adicionalmente, cuando se añada una

entidad padre, la opción que permite añadirla se oculta ya que únicamente puede haber una.

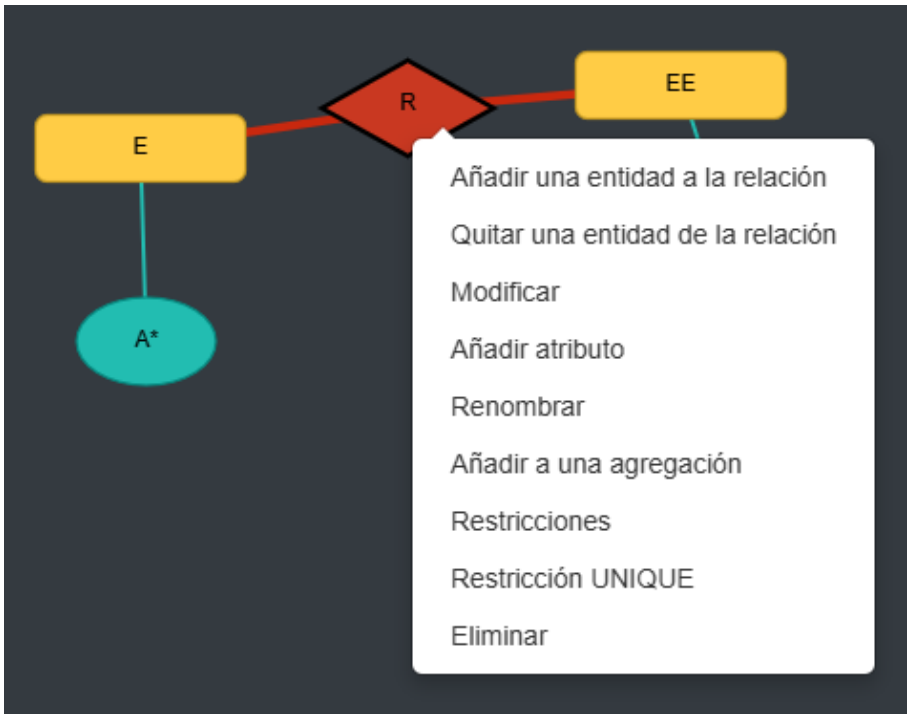

Figura 4-19 Nuevo menú de opciones de una relación

De la misma forma que las relaciones, en los menús de opciones de las entidades y atributos, se han cambiado los textos "Editar Entidad" y "Editar Atributo" por "Modificar". Esto es debido a que en estos casos además del nombre se especifican otras características de los elementos como definir si una entidad es débil o si un atributo es compuesto o multivalorado.

En cuanto a la opción de "Añadir a agregación" se modifica para todos los elementos, quedando como "Añadir a una agregación", siendo este nuevo texto mejor construido y más coherente con el resto de las opciones de los menús.

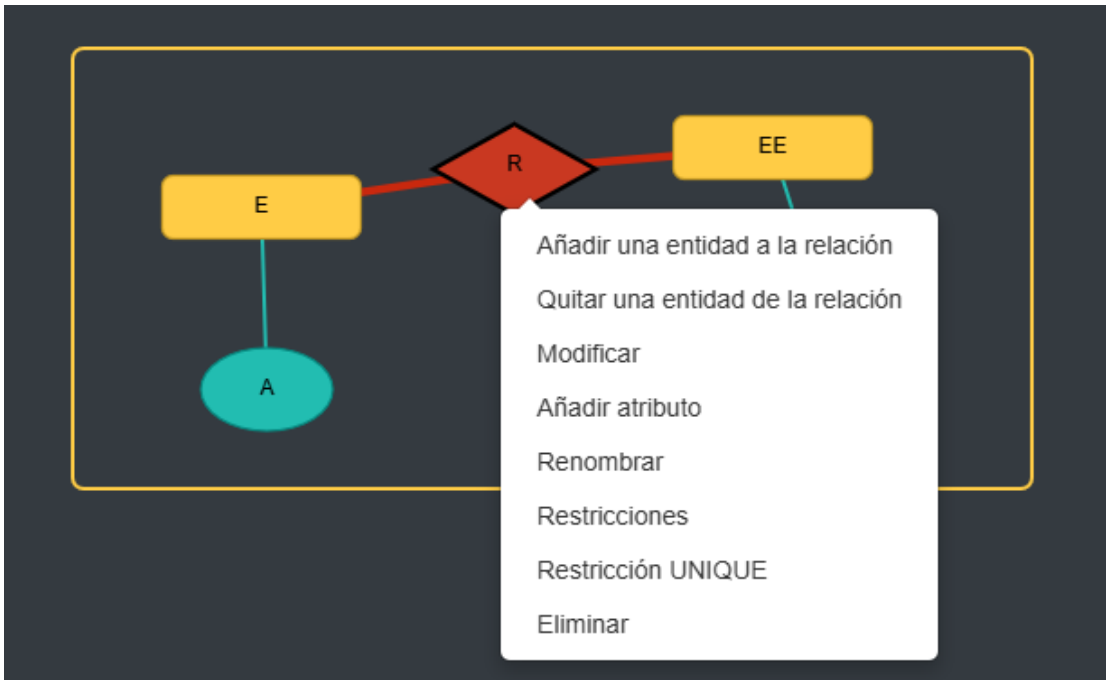

Figura 4-20 Menú de opciones de una relación dentro de una agregación

Adicionalmente, esta última opción, se oculta cuando un elemento ya forma parte de una agregación (imagen 4-20). Vuelve a hacerse visible en cuanto la agregación a la que pertenece se elimina. Sucede de la misma manera en el resto de los elementos que se incluyan en una agregación.

También se cambia el texto de *Tabla Única* por *Restricción UNIQUE* tanto en la opción en español como en inglés (figura 4-21).

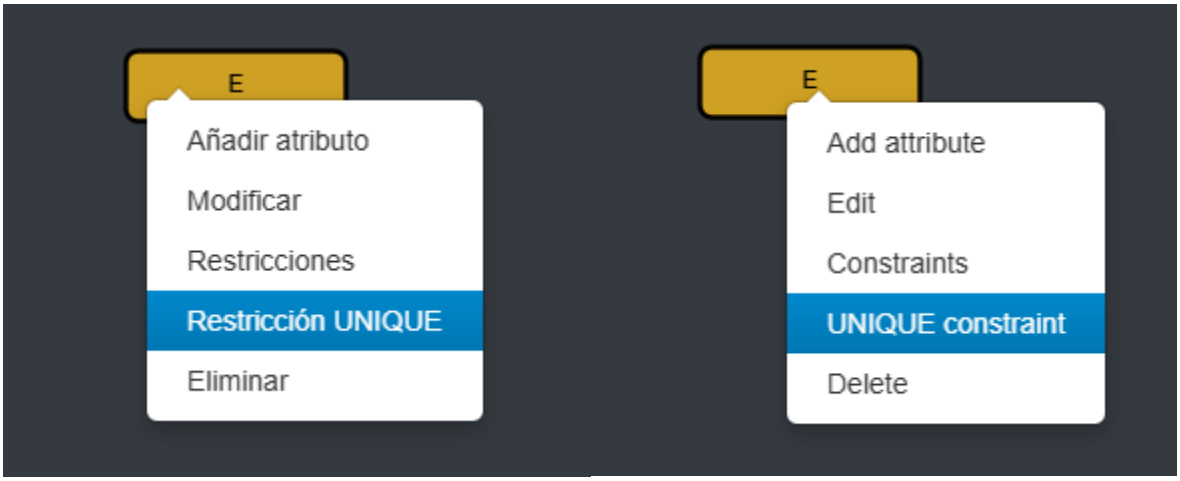

Figura 4-21 Reemplazo de opción Tabla única en español (izq) e inglés (dcha)

Mejora de los textos en los modales

En el modal de adición de una entidad a una relación, uno de los campos a rellenar tiene el nombre repetido, ROL. La solución es sencilla, cambiamos el nombre por uno más adecuado, quedando finalmente como se muestra en la figura 4-22.

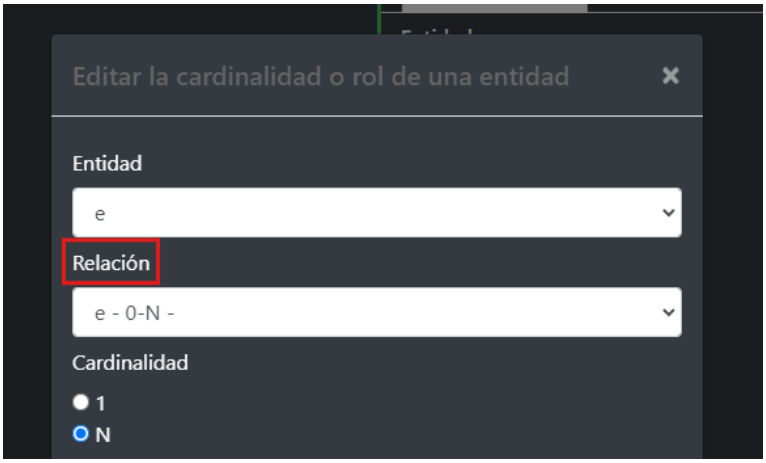

Figura 4-22 Modificación modal editar relaciones

Mejora del modal para añadir entidades a una relación

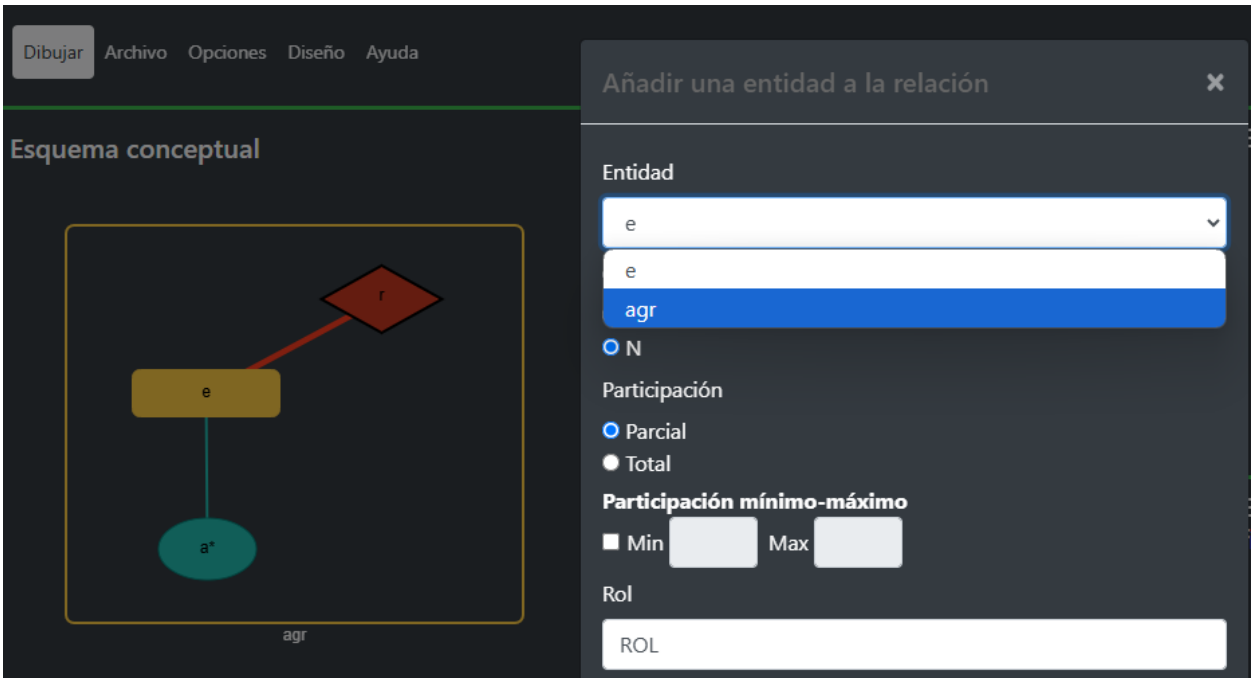

Figura 4-23 Antiguo modal "Añadir entidad a relación"

Durante las pruebas de integración de la agregación, se detecta que, en el listado de entidades posibles para añadir a una relación, se incluye la propia agregación. Para el correcto funcionamiento de los desarrollos, la agregación se trata como una entidad a nivel de código y por ello, se muestra en este listado. En algunos casos es correcto, pero si la relación a la que se quiere añadir la entidad forma parte de una agregación, ésta no debe mostrarse en el listado. Para ello se ha modificado la lógica del modal de forma que en estos casos concretos no aparezca la agregación en la lista, como se muestra en la figura 4-24

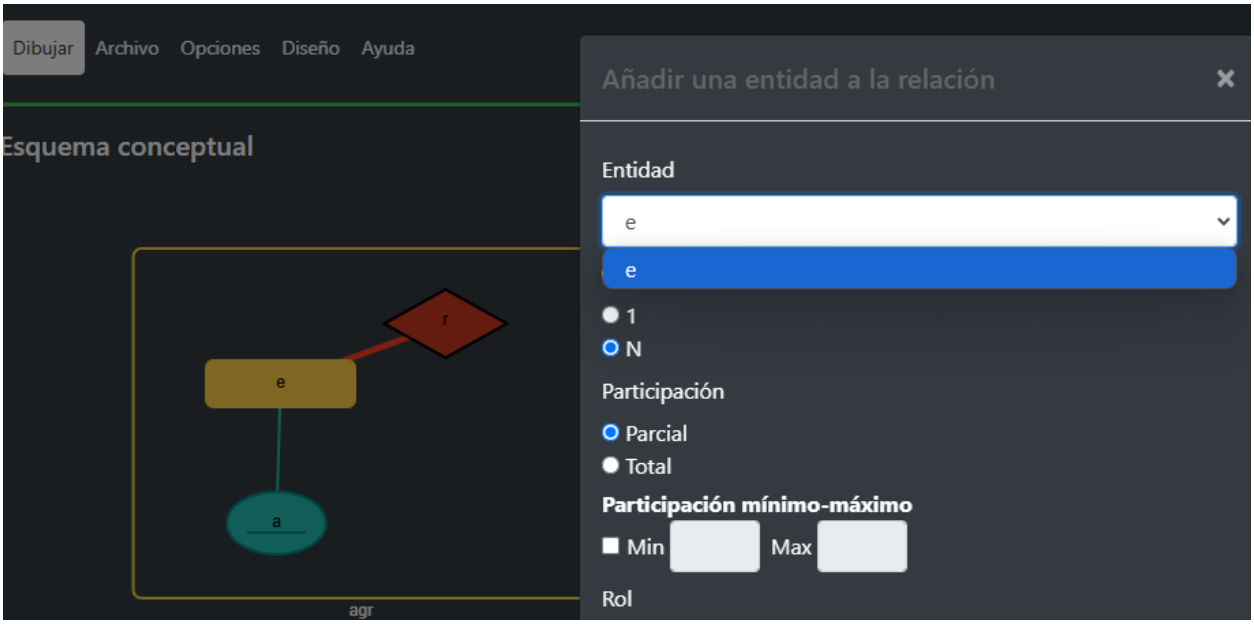

Figura 4-24 Modal "Añadir entidad a relación" modificado

Listado de entidades a modificar

En el modal de edición de relaciones, se muestra un desplegable que permite al usuario elegir la entidad sobre la que se quiere modificar la o las relaciones. Como ejemplo ilustrativo usamos el caso de la figura 4-25. La relación rr está enlazada con las entidades ee y eee. Sin embargo, en el desplegable se muestran todas las entidades del esquema.

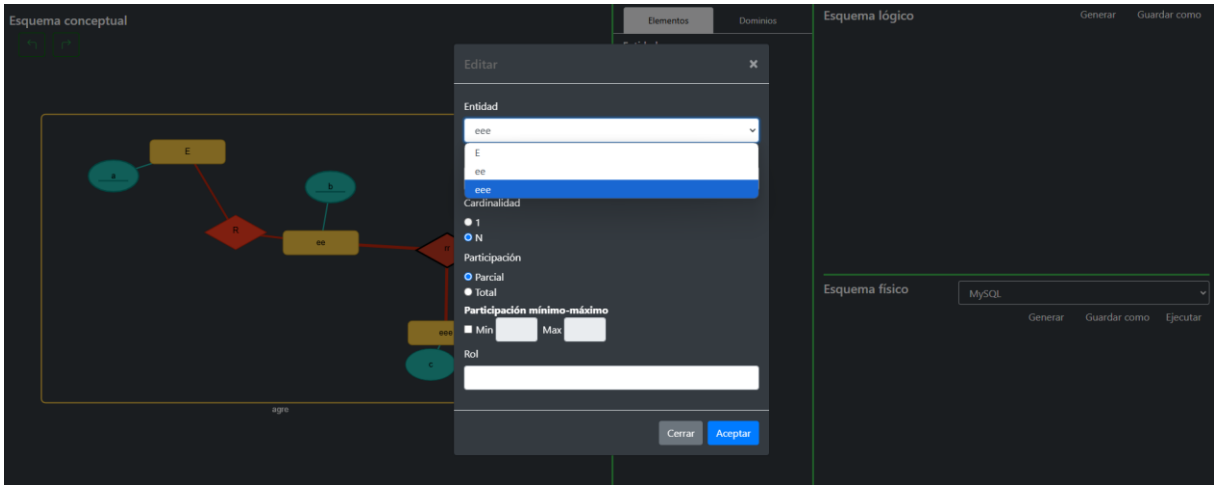

Figura 4-25 Listado de entidades a editar sin modificar

Con el objetivo de mejorar la experiencia del usuario en lo referente a este modal, se modifica el listado que aparece en el desplegable. Se muestran las entidades con conexiones existentes (figura 4-26).

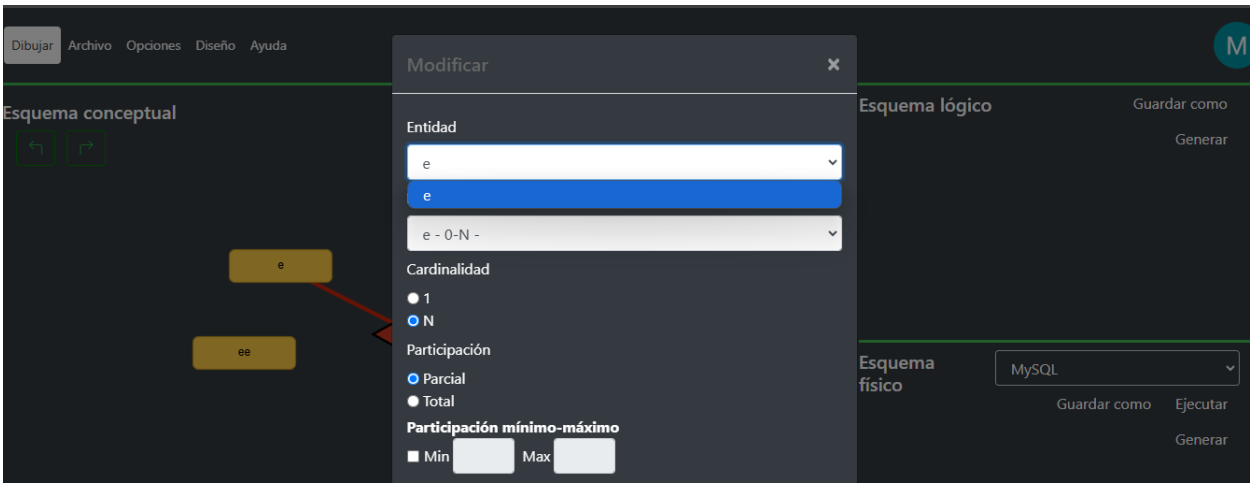

Figura 4-26 Listado de entidades a editar modificada

Adicionalmente, se controla mediante un evento el listado mostrado en el segundo desplegable. Este otro listado es el de uniones existentes relacionados con las diferentes entidades del esquema. El nuevo desarrollo restringe esta lista a las uniones que existan entre la entidad seleccionada en el desplegable superior y la relación.

Control de nombres al añadir relación de entidad débil

En casi todos los modales que añaden elementos, se hace una comprobación al escribir el nombre, de forma que se muestra un mensaje de error si ya existe un componente con el mismo nombre asignado. Como se aprecia en la figura 4-27, en el caso de las relaciones creadas junto a una entidad débil, esto no sucede.

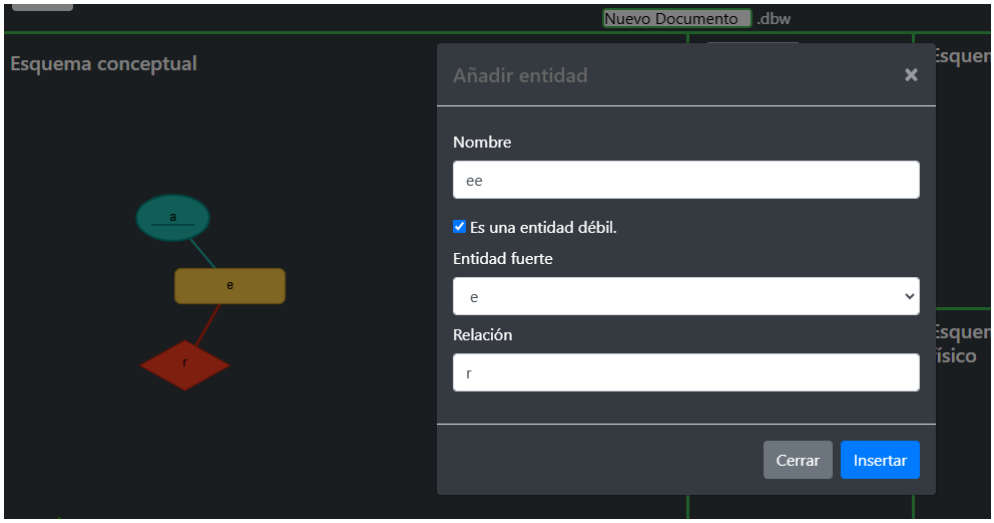

Figura 4-27 Modal añadir entidad débil sin comprobación por nombre

Para solucionarlo, se modifica el HTML del modal, de forma que se consulten los nombres del resto de relaciones creadas y se muestre el mensaje de error en caso de que ya exista (figura 4-28).

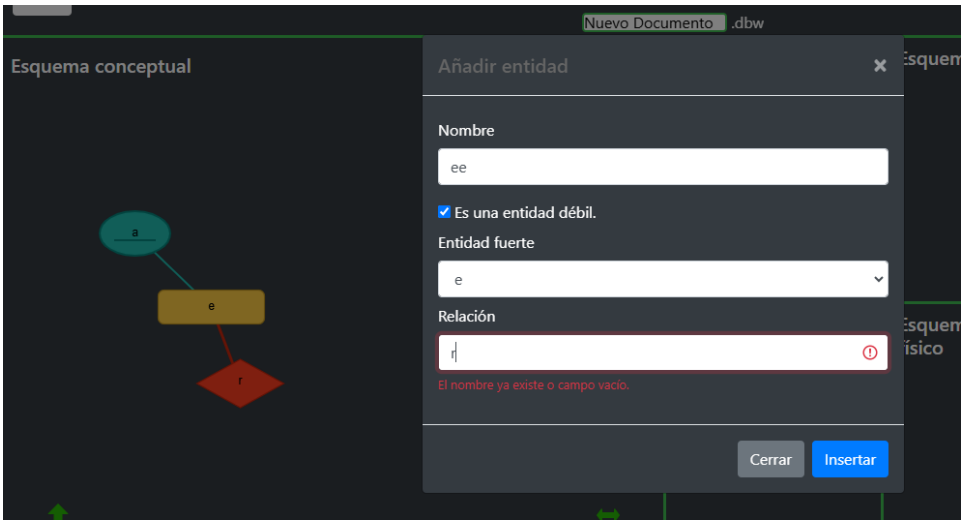

Figura 4-28 Modal añadir entidad débil con comprobación por nombre

4.4 Resolución de incidencias

En este apartado se explican los desarrollos necesarios para solventar las incidencias detectadas en la aplicación.

Duplicidad de atributos en esquemas lógico y físico

Se ha detectado que las entidades con más de 2 atributos no traducen correctamente el esquema. En la figura 4-29 se muestra este problema. El esquema, que cuenta con 3 atributos, duplica uno de ellos en los esquemas lógico y físico. Se detecta que sucede con todos los atributos salvo los añadidos el primer y último lugar.

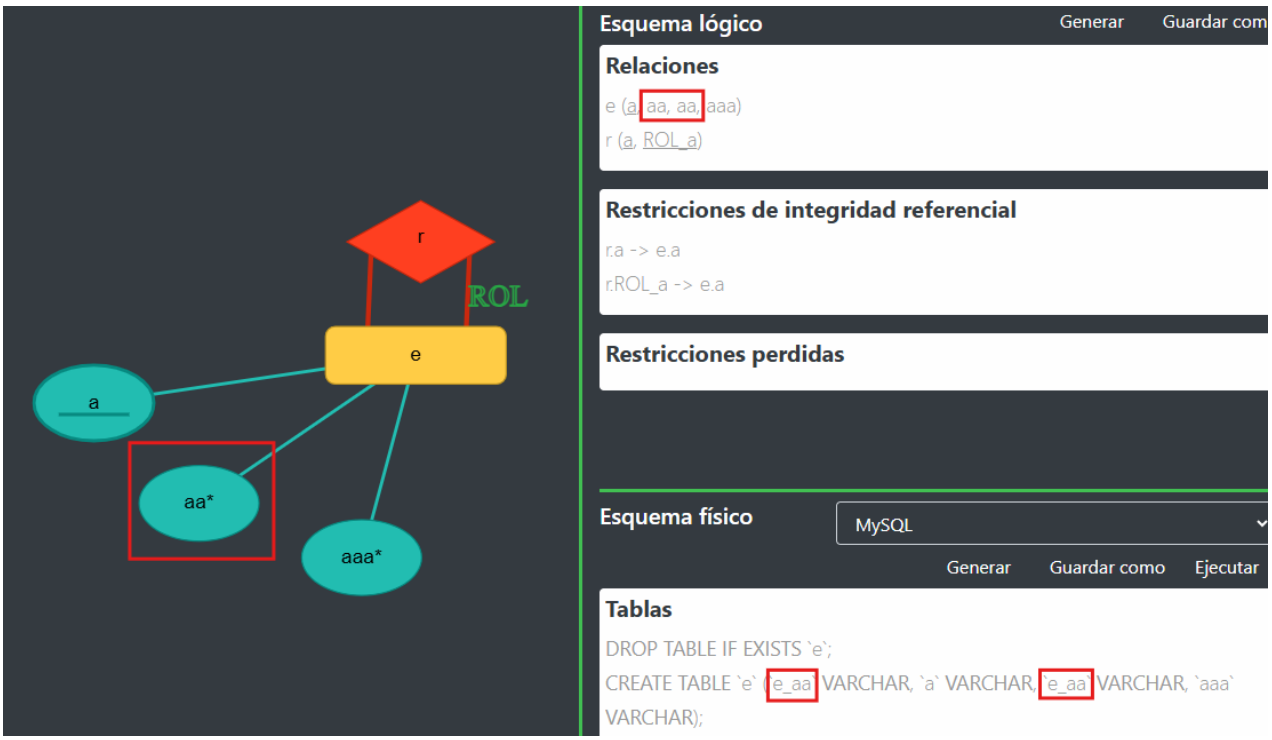

Figura 4-29 Ejemplo de duplicidad de atributos en esquema lógico y físico

La causa de esta incidencia es debida a un fallo de control en la modificación de los atributos en los DAOs de las entidades y las relaciones. Al solucionarlo no solo se resuelve la incidencia detectada, también permite la correcta incorporación de atributos en relaciones. Esto último, es un desarrollo que no estaba incluido hasta ahora.

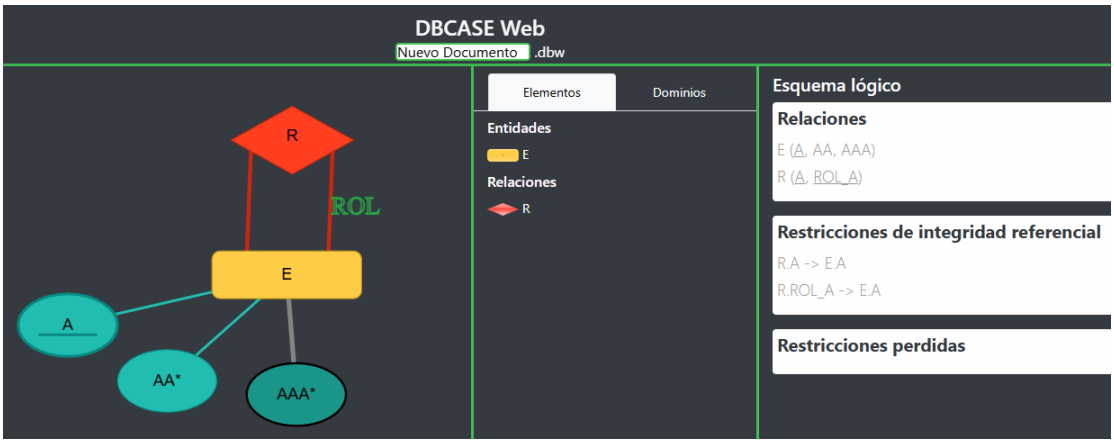

Figura 4-30 Ejemplo de resolución de duplicidad de atributos en esquema lógico y físico

Se puede ver la correcta implementación de la resolución en la figura 4-30.

Error al crear agregaciones después de eliminar otra

Al eliminar una agregación pulsando el botón del teclado suprimir no se ejecuta la función creada para ello. Sin esta ejecución, no se actualiza el listado de nodos y aristas que pertenecen a la agregación, pero que al eliminarla dejan de hacerlo. Esto supondría un problema si tras la eliminación de una agregación se intenta crear otra que contenga cualquiera de los elementos que perteneciera a la anterior, ya que se intentaría añadir un elemento a un listado en el que ya existe. Para resolver este problema aparentemente tan sencillo, debemos crear un listener sobre el teclado para que se ejecute la función de eliminación de agregaciones al pulsar el botón supr. Elegimos esta función que únicamente elimina la agregación frente a la otra que elimina sus elementos también al ser la opción que conserva más elementos de la agregación.

Problemas de permanencia en variables del canvas

Al cambiar de idioma o el tema en las opciones superiores del canvas, provoca errores en el funcionamiento de varios desarrollos realizados. Esta incidencia es debida a que, al cambiar estas opciones, el canvas se carga de nuevo, inicializando de nuevo las variables creadas para las implementaciones del proyecto.

Para solucionar este problema, se debe almacenar de manera persistente. En estos casos, únicamente se necesita conservar el valor de las variables durante la sesión

del navegador. Guardar estos datos como variables de sesión, permite almacenarlos temporalmente, siendo eliminados automáticamente cuando el navegador o la pestaña que los contiene se cierra.

Etiqueta de rol redundante

Se detecta una incidencia con la etiqueta de rol en el formulario. Al generar los valores que se informan en el formulario, se muestra por defecto la etiqueta 'ROL' al crear relaciones no recursivas.

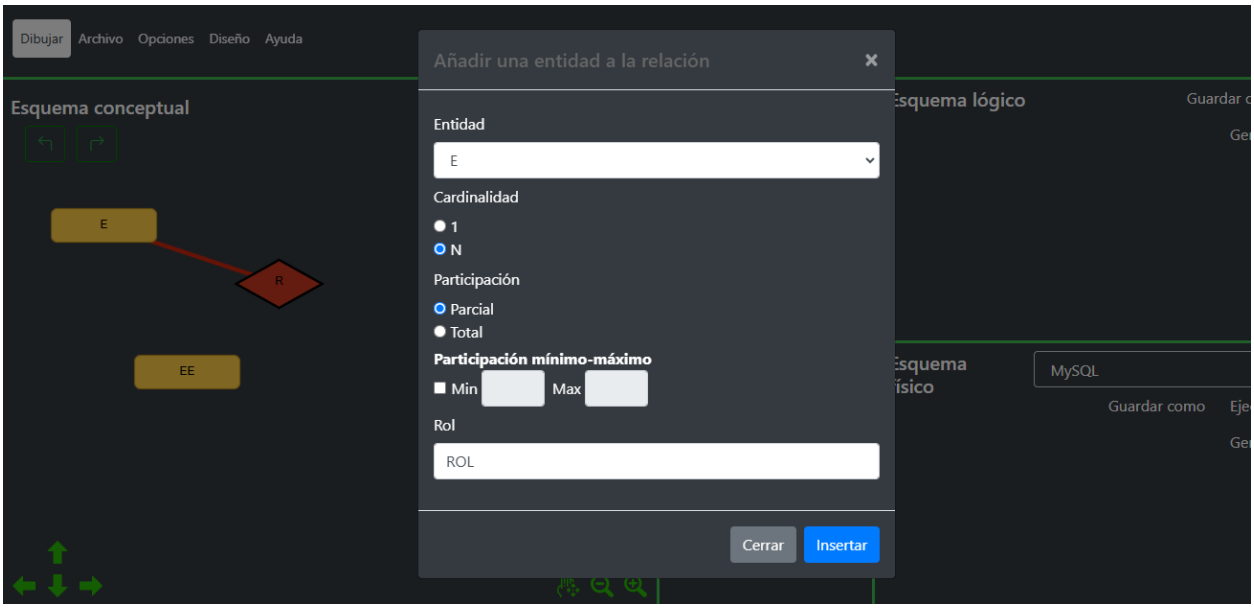

Figura 4-31 Formulario de generación de una relación recursiva con etiqueta ROL correcta

Es correcto que por defecto se haga para las recursivas, pero no es necesario que una relación no recursiva cuente con una etiqueta de rol. Como ejemplo ilustrativo empleamos las figuras 4-31 y 4-32, que muestran el formulario por defecto en una relación recursiva y en otra no recursiva, respectivamente.

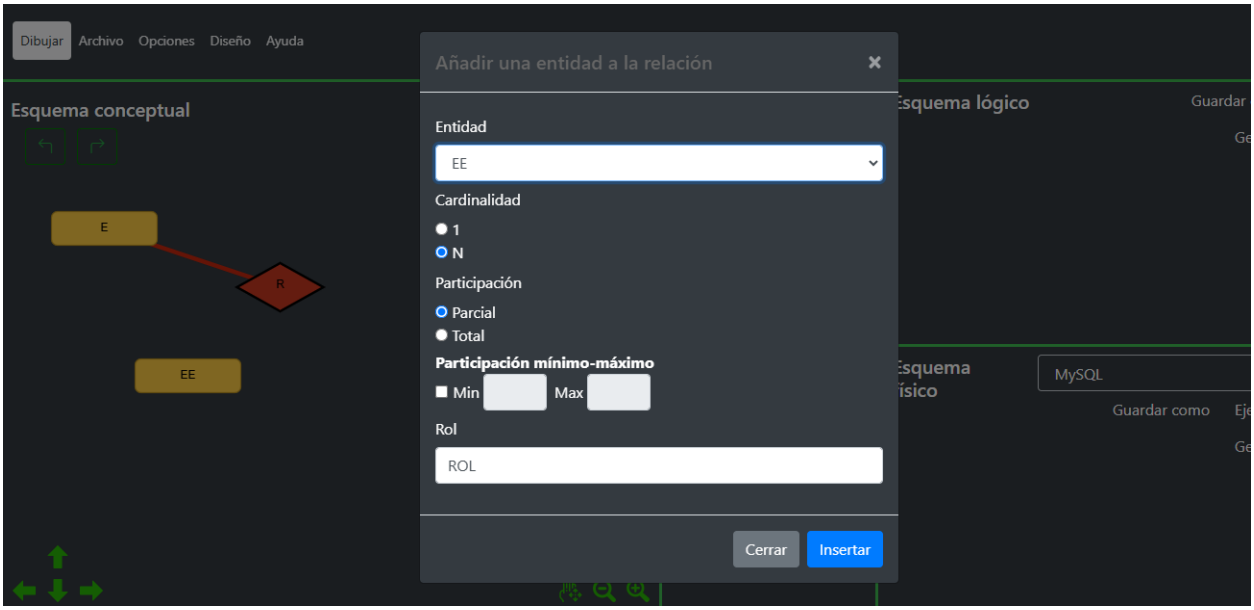

Figura 4-32 Formulario de generación de una relación no recursiva con etiqueta ROL incorrecta

Adicionalmente, si se borra la etiqueta se muestra un mensaje de error como se observa en la figura 4-33, que impide añadir la entidad a la relación. En el caso ejemplificado en la figura 4-31, sería correcto no permitirlo ya que es obligatorio para las relaciones recursivas. En el otro caso se puede usar una etiqueta, pero no es necesario y por ello se debe permitir insertar el formulario.

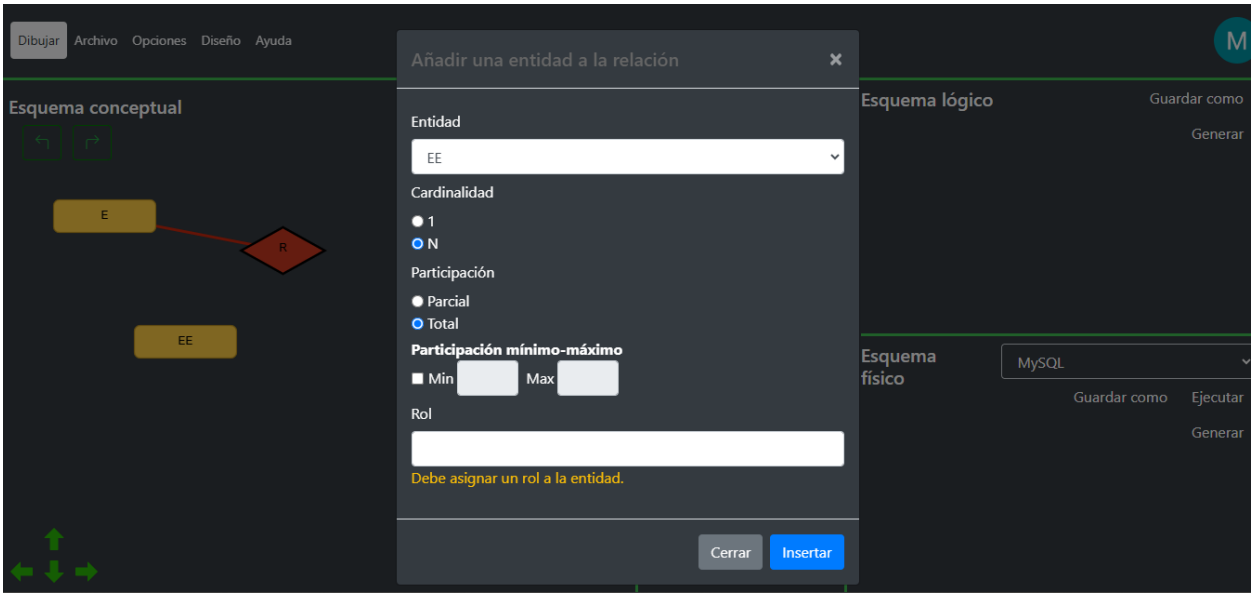

Figura 4-33 Funcionamiento incorrecto del mensaje de error mostrado cuando falta el rol

Mensaje de error mostrado incorrectamente

En el formulario usado para añadir una entidad a una relación se muestra el mensaje de error mostrado en la figura 4-33. Este mensaje se muestra siempre que el cuadro de texto de rol deba tener una etiqueta, pero esté vacío.

En la figura 4-33 se aprecia el funcionamiento equivocado de esta lógica. Aparece el mensaje de error sobre la relación a crear entre R y la entidad EE. Estos elementos carecen de relaciones previas, por lo que la etiqueta de rol no es necesaria y el mensaje de error no debería mostrarse.

Para solventarlo, mejoramos el evento creado para este modal, de forma que el texto de error se limpie si se cambia de entidad a modificar, asegurando que solo se muestra cuando sea necesario.

Adicionalmente se crea otro evento que se dispara cuando se escribe o elimina el texto de la caja de texto rol. Esto hace que el texto de error y la habilitación o deshabilitación el botón de envío del formulario sean mucho más dinámicos. Antes de este desarrollo era necesario clicar fuera de la caja de texto tras escribir o borrar sobre ella para que se modificaran.

Corrección de erratas

Se detectan algunos casos de malas traducciones o de erratas en los textos. Modificamos los textos en los modales y las traducciones. Detallamos a continuación los diferentes cambios.

En la figura 4-34 se aprecia la corrección de la errata “Entitys” por la correcta traducción del plural de entidades.

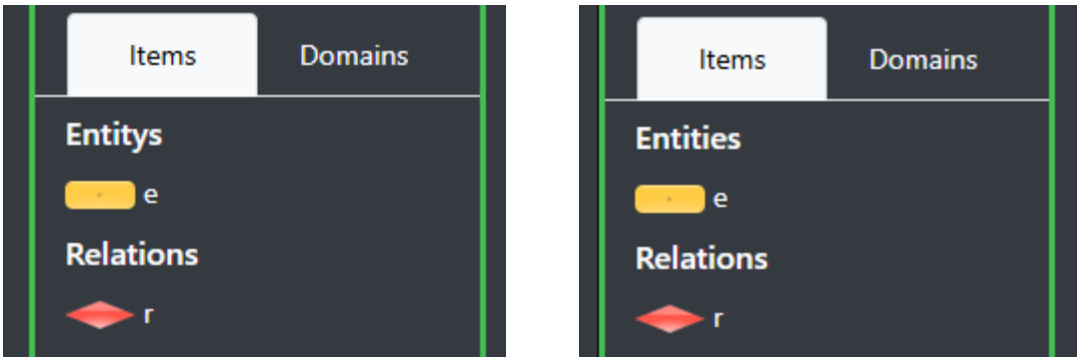

Figura 4-34 Modificación errata "Entitys" por "Entities"

Se corrige también la errata "Contrainst" por "Constraints" (figura 4-35). Además, este texto se muestra en inglés cuando el idioma seleccionado es español.

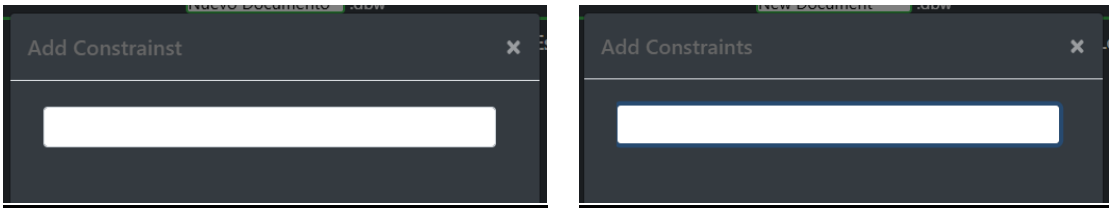

Figura 4-35 Modificación errata "Contrainst" por "Constraints"

Este error se debe a que, en el modal HTML, no se usa el mensaje de propiedades para mostrar el texto. Se implementa correctamente la opción en español y se modifica el modal para que emplee la variable según el idioma (figura 4-36).

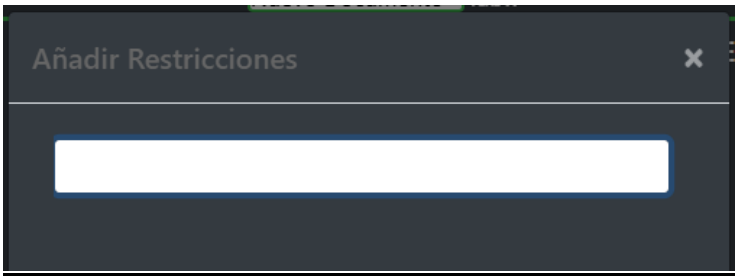

Figura 4-36 Modal para añadir restricciones en español

Problema de Bloqueo de Teclas en Modales

Algunos modales de la aplicación cuentan con una caja de texto. Al escribir en ella y usar las flechas para mover el cursor, se mueven los elementos del canvas. Esto es

un error debido a la captura de estas teclas para la funcionalidad de los botones del canvas (figura 37). Esta caja de texto controla con un evento las teclas permitidas en el mismo. Entre las opciones no se encuentra ni el botón de suprimir ni las flechas del teclado. Añadiendo estas opciones se corrige este comportamiento.

Figura 4-37 Flechas del canvas

Capítulo 5 - Conclusiones y trabajo futuro

Este proyecto parte de una base previa muy trabajada, y una de las principales preocupaciones al iniciarlo fue comprender cómo estaba planteada la aplicación y entregar un trabajo a la altura.

El desarrollo de las funcionalidades pendientes o mejorables, planteadas al inicio del proyecto, me ha permitido profundizar en el aprendizaje sobre las aplicaciones web. En particular, destacaría el uso de JavaScript y jQuery, tecnologías que apenas había aplicado durante mis años de carrera y que han sido fundamentales en el desarrollo de este proyecto.

La aplicación entregada ahora ofrece una solución eficiente, intuitiva y bien adaptada a las necesidades planteadas, demostrando el éxito de la implementación de las mejoras propuestas como objetivos del trabajo. Junto con los desarrollos previos, podemos afirmar que se ha construido una herramienta ideal para la enseñanza de bases de datos, proporcionando una experiencia de usuario de calidad.

5.1 Trabajo futuro

Se han identificado posibles desarrollos que representan oportunidades para mejorar o añadir nuevas funcionalidades en futuras iteraciones de este proyecto. A continuación, se detallan algunas de estas propuestas:

- **Restricción UNIQUE**: Mejorar el modelo físico para que se cree automáticamente la restricción UNIQUE en la tabla durante la generación del modelo, evitando la necesidad de agregarla posteriormente con ALTER TABLE.
- **Representación de aristas**: Mejorar la visualización de las aristas que conectan una agregación con elementos externos, buscando una representación gráfica más clara e intuitiva.
- **Creación de esquema físico:** Generar esquema físico a partir del esquema relacional y no desde el esquema entidad-relación.
- **Participación total**: Al añadir o modificar una relación para que sea total, se crean dos aristas con el mismo id. Esta lógica debe ser modificada

Introduction

DBCASE Web 3.0 is the latest version of a project whose development began in 2007 with the aim of providing students with an accessible and intuitive tool to understand how relational databases work and to design schemas in a simple way. Since its inception, DBCASE has evolved to adapt to the needs of users, providing an environment that facilitates the design of Entity-Relationship diagrams.

In this new release, the focus is on improving the implementation of aggregations and adapting the generation of schemas according to these updates, ensuring the accuracy of translations. In addition, editing functionalities have been added, such as the undo (Ctrl + Z) and redo (Ctrl + Y) options, which provide greater flexibility and control to the user. In addition, work has been done to improve the general usability of the application, perfecting previous developments and fixing errors to make its use more intuitive and fluid.

The application continues to evolve as a robust and easy-to-use educational tool, designed to provide an optimal user experience in learning the subject of databases.

Motivation

The main motivation for choosing this TFG project arises from my experience as a student of the subject Database. During the course, I noticed the lack of a tool that would facilitate the study and better understanding of translations between models. This need led me to choose the DBCase Web project to work on a solution that could cover the needs and problems that I myself faced. Having been a student of the subject, I have a privileged perspective to understand the needs and expectations of the users. This puts me in a position to make informed decisions regarding the usability and usefulness of the application, prioritizing those features and updates that will truly enhance the students' experience. Furthermore, it motivates me to know that this project has the potential to generate a real and positive impact on the academic lives of other students, becoming a practical tool that responds to concrete needs and provides tangible value.

In short, this project is not only born out of a personal concern, but also out of my commitment to offer an effective solution that can really make a difference.

Background

It is essential to mention and understand the previous versions of this project to carry out and explain this new extension of the project. The previous versions are detailed below.

DBDT de sistemas informáticos (2007 - 2008)

This is the first iteration of this project. It is developed as a desktop application that allows to design and test a database schema and generate a SQL script [1].

Authors:

- Alberto Milán Gutiérrez
- Miguel Martínez Segura
- Francisco Javier Cáceres González
- Yolanda García Ruiz (Tutora)

DBCASE de sistemas informáticos (2008 - 2009)

This second project on the application extends the tool to be compatible with the most common database managers and adds new functionalities in the design of entity-relationship models [2].

Authors:

- Rodrigo Denís Cepeda Mateos
- Cristina Marco de Francisco
- Tello Serrano Gordillo
- Fernando Saénz Pérez (Tutor)

DBCASE 2.0 (2018 – 2019)

The project aimed to rehabilitate an application for academic use in classrooms. To this end, the interface design was updated, improving usability and simplifying the workflow, as well as redesigning the code panel and improving the visualization of the entity-relationship diagram. We also corrected bugs from previous versions, added new functionalities, such as a dark theme, and completed the translation to the logical model to facilitate user comprehension [3].

Authors:

- Miguel Arriba García
- Fernando Saénz Pérez (Tutor)

DBCASE Web 2.0 (2019 - 2020)

With this new version, a multiplatform application was achieved, starting from the desktop version made the previous year. In this way, the application started to be accessible from any browser, eliminating the need for installations and dependence on specific software [4].

Authors:

- Luis Eduardo Pacuar Cerrón
- Yrving David Conde Cubas
- Fernando Saénz Pérez (Tutor)

DBCASE Web 3.0 (2020 - 2021)

In this extension of the project, the functionalities of the application are expanded, incorporating new conceptual design constructions, implementing participation and cardinality of relations and visual improvement of the tool [5].

Authors:

- Roxanne A. Caiafa
- Fernando Saénz Pérez (Tutor)

Objectives

The objective of this new version of the project is to solve translation errors between the different models and to extend the functionalities of the application to improve the user experience. These new functionalities focus on making a more complex implementation of aggregations, to offer greater flexibility and usability in database design. In addition, support for keyboard controls, such as undo (Ctrl + Z) and redo (Ctrl + Y), will be introduced, allowing changes to be made more easily, greatly improving the user experience.

Work will also be done to improve the general usability of the application, to make interaction more fluid and user-friendly. Finally, problems detected in the elements and their translations will be corrected to ensure that all functions are consistent and accurate.

Work plan

The project has been planned according to a series of phases carried out during the entire development. In addition to these three phases, the progress of the project has been documented in this report. We explain below the details of the activities carried out in each of the three phases.

1. Research

Firstly, research has been done on previous DBCase projects, in order to understand the context of this new version. Along with this research, the first contact with the operation of the application is carried out. During this process the environment is configured and prepared to compile the project and test the operation of the application. It is necessary to download and configure Maven 4.0 as dependency manager and project builder. It is also necessary to configure Java (JDK 1.8).

2. Development

In this second phase the tasks and main objectives are prioritized. Thus, it is decided to start the implementation of the most complex developments, the

aggregation and the undo and redo options, followed by the functional improvements and bug fixes.

This last part refers to those issues detected before the start of the project and those detected during the first contact or during the testing of the developments.

During this phase, search engines and forums [6] [7] [10] have been used to solve bugs, resolve specific doubts related to the code and understand the behavior of certain technologies.

The project and all modifications made to it can be found in the following GitHub repository: https://github.com/MariaBarcenilla/DBCaseWeb4.0

3. Testing

The testing process has been carried out in parallel throughout the development, carrying out unit tests at the end of the modifications and validating the correct integration of the new functionalities with the rest of the application

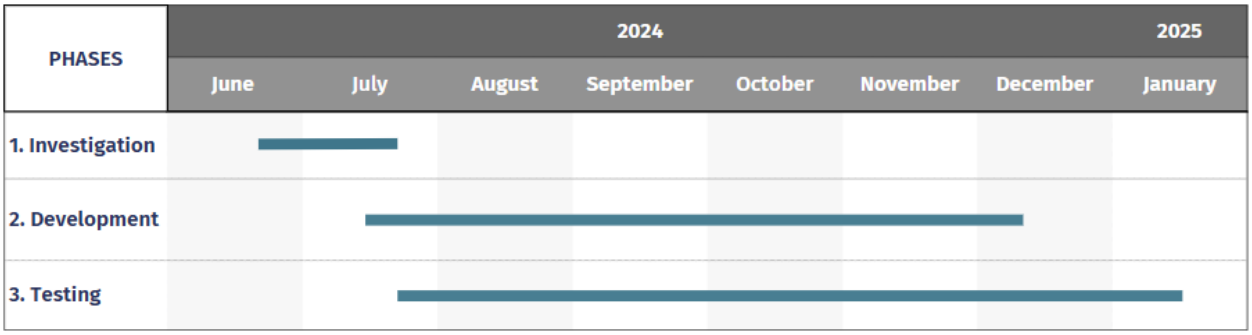

Figura 5-1 Gantt chart of the work plan phases

Conclusions and future work

This project starts from a very well worked previous base, and one of the main concerns at the beginning of the project was to understand how the application was planned and deliver a work up to par.

The development of the pending or improvable functionalities, raised at the beginning of the project, has allowed me to deepen my learning about web applications. **I** would highlight the use of JavaScript and jQuery, technologies that I had hardly applied during my career and that have been fundamental in the development of this project.

The application delivered now offers an efficient, intuitive and well adapted solution to the needs raised, demonstrating the success of the implementation of the improvements proposed as objectives of the work. Together with the previous developments, we can affirm that an ideal tool for teaching databases has been built, providing quality user experience.

Future Work

Potential developments have been identified that represent opportunities to improve or add new functionality in future iterations of this project. Some of these proposals are detailed below:

- **UNIQUE constraint**: Improve the physical model so that the UNIQUE constraint is automatically created in the table during model generation, avoiding the need to add it later with ALTER TABLE.
- **Edge representation**: Improve the visualization of the edges that connect an aggregation with external elements, looking for a clearer and more intuitive graphical representation.
- **Generation of physical schema**: Generate physical schema from the relational schema and not from the entity-relationship schema
- **Total participation generation**: When adding or modifying a relationship to be total, two edges with the same id are created. The logic behind it should be fix.

BIBLIOGRAFÍA

[1] Universidad Complutense de Madrid, «Database desing tool» [En línea]. Available: https://hdl.handle.net/20.500.14352/54309. [Último acceso: 05 11 2024].

[2] Universidad Complutense de Madrid, «DBCASE: una herramienta para el diseño de bases de datos» [En línea]. Available: https://hdl.handle.net/20.500.14352/54450. [Último acceso: 05 11 2024].

[3] Universidad Complutense de Madrid, «DBCASE 2.0» [En línea]. Available: https://hdl.handle.net/20.500.14352/15371. [Último acceso: 05 11 2024].

[4] Universidad Complutense de Madrid, «DBCASE Web "Una herramienta para el diseño de Bases de Datos » [En línea]. Available: https://hdl.handle.net/20.500.14352/10286. [Último acceso: 05 11 2024].

[5] Universidad Complutense de Madrid, «DBCASE Web 3.0 "Una herramienta para el diseño de Bases de Datos"» [En línea]. Available: https://hdl.handle.net/20.500.14352/10605. [Último acceso: 05 01 2025].

[6] W3Schools, «Tutoriales y recursos de desarrollo web,» [En línea]. Available: https://www.w3schools.com/. [Último acceso: 03 01 2025].

[7] GeeksforGeeks, «Resources for programming and development, » [En línea]. Available: https://www.geeksforgeeks.org/. [Último acceso: 17 01 2025].

[8] Vis.js, «Vis Network Documentation,» [En línea]. Available: https://visjs.github.io/vis-network/docs/network/. [Último acceso: 29 12 2024].

[9] Bootstrap Icons, «Icon library and usage guide,» [En línea]. Available: https://icons.getbootstrap.com/. [Último acceso: 17 12 2024].

[10] Stack Overflow, «Programming Questions and Answers Community,» [En línea]. Available: https://stackoverflow.com/. [Último acceso: 29 12 2024].

[11] jQuery API Documentation, «Interactive library documentation,» [En línea]. Available: https://api.jquery.com/. [Último acceso: 27 12 2024].