Universidad Zaragoza
1542

Trabajo Fin de Grado

Actualización y ampliación de la herramienta DEReditor

Autor

David Utrilla Villa

Directores

Piedad Garrido Picazo

Francisco J. Martínez Domínguez

Escuela Universitaria Politécnica de Teruel

Septiembre de 2021

TABLA DE CONTENIDO

1. ÍNDICE DE ILUSTRACIONES

2. RESUMEN

DEReditor_v2 es una aplicación de escritorio que se ha desarrollado como trabajo fin grado en los estudios de Ingeniería Informática en el ámbito de las BBDD.

Este proyecto surge como evolución de una aplicación desarrollada anteriormente, y que pretende actualizarla para obtener una herramienta moderna y adecuada a los requerimientos didácticos.

DEReditor_v2 sigue el patrón de diseño Modelo-Vista-Controlador. Esta característica estructural del código interno fue planteada como uno de los cambios que debían darse en la nueva aplicación.

La interfaz de usuario es moderna, clara, atractiva a la vista y tiene un alto nivel de usabilidad, mejorando notablemente la apariencia que se mostraba en un principio.

El desarrollo de diagramas usando la aplicación se hace relativamente sencillo, pues la propia aplicación tutoriza el trabajo, guiando al usuario hacia un orden de implementación correcto.

Se ha decidido escribir el código en inglés siguiendo los consejos para una programación con un carácter más profesional.

La aplicación permite guardar los proyectos como ficheros “. erd”, los diagramas como archivos “.jpg”, y los códigos tanto lógico como SQL en formato PDF.

Palabras clave: BBDD, Modelo-Vista-Controlador, SQL, PDF

Abstract

DEReditor_v2 is a desktop application that has been developed as a final degree project in Computer Engineering studies in the field of BBDD.

This project arises as an evolution of a previously developed application, and aims to update or obtain a modern and an adequate tool which fits the current didactic requirements.

DEReditor_v2 follows the Model-View-Controller design pattern. This structural characteristic of the internal code was considered as one of the changes to be made in the new application.

The user interface is modern, clear, visually appealing and has a high level of usability, significantly improving the former appearance.

The development of diagrams using the application is relatively easy, as the application itself guides the user towards the correct order of implementation.

The code has been written in English, making it more professional character.

The application allows saving the projects as ".erd" files, the diagrams as ".jpg" files, and both logic and SQL codes in PDF format.

Keywords: BBDD, Model-View-Controller, SQL, PDF

3. INTRODUCCIÓN

En la época actual, comúnmente denominada “la era de la explosión de los sistemas de información”, se hace inevitable pensar en las bases de datos como una de las herramientas de almacenaje y ordenación más antiguas y utilizadas. El volumen de datos que rige el mundo actual es tal, que una buena organización de los mismos permitirá su correcto uso, tanto en eficiencia como en corrección.

Los sistemas informáticos, presentes en prácticamente cualquier proceso de la vida actual, facilitan la tarea de ordenar y almacenar la incipiente cantidad de datos que cada día se generan.

El implacable paso del tiempo propicia en cualquiera de los elementos de nuestro entorno, tanto como si cambia y se adapta, como si permanece y se minimiza, una evolución o un retroceso condenatorio. Acentuar el cambio o acentuar la quietud son formas diferentes de evolucionar, y como es lógico con resultados muy distintos. En ocasiones evolucionar es crecer o menguar, mejorar o empeorar, envejecer o rejuvenecer. Cuando hablamos de tecnología todos estos cambios son aplicables a su inexistente inmovilidad.

En un mundo en constante evolución, donde el paso del tiempo acelera al propio tiempo, la tecnología sigue a pies juntillas este rápido cambio. Pocos campos de nuestra vida experimentan con tanta rapidez la evolución como el mundo tecnológico.

Este tren de alta velocidad al cual va subida la tecnología, provoca que ésta nazca, crezca y muera en periodos de tiempo relativamente pequeños. La obsolescencia de hardware o de software está a la orden del día. Las actualizaciones de los sistemas son sin duda la herramienta que permite alargar la vida de los sistemas y rescatarlos de una muerte prematura. Proyectos como el que a continuación se va a exponer son necesarios para mantener vivos hilos de desarrollo que surgieron años atrás.

En esta vorágine de constante avance y evolución, hemos de considerar tan importante el surgir de ideas nuevas, como la adaptación de ideas pasadas que el tiempo se ha encargado de envejecer.

Así, la actualización de nuestros equipos, tanto hardware como software, es imprescindible para no apearse del vagón de la evolución que arrastra el mundo tecnológico.

El documento que a continuación se redacta, describe el desarrollo de la actualización de un sistema anterior, con el fin de adaptarlo a un tiempo moderno que permita su utilización eliminando los inconvenientes surgidos por el paso del tiempo.

DEReditor_v2 es la segunda parte de un proyecto anterior denominado DEReditor[1], desarrollado por Manuel Coll Villalta en el año 2005, que al igual que el caso que nos ocupa, sirvió como proyecto final de sus estudios de informática en la escuela.

[1] http://dereditor.sourceforge.net

4. OBJETIVOS

DEReditor_v2 es una aplicación de escritorio diseñada para la elaboración de diagramas entidad relación en el desarrollo de bases de datos.

Esta aplicación es la modernización del anterior programa denominado DEReditor. En esta nueva versión se pretende dar una estructura interna del código siguiendo el patrón de programación Modelo, Vista y Controlador. El fin de esta modificación es dotar al código de un orden y legibilidad que anteriormente no tenía, de tal manera que su entendimiento sea mucho más sencillo. Así, además se aplicarán conocimientos y criterios de programación adquiridos durante los años de estudio.

Por otra parte, se buscará dar a la aplicación un carácter visual mucho más vanguardista y sofisticado que el que tenía su predecesor. Se atenderá a la elegancia de la parte visual y a la manejabilidad de la interfaz.

La finalidad de esta aplicación es facilitar el diseño de diagramas entidad relación a cualquier diseñador de bases de datos, pero se considera que está especialmente dirigida al alumnado de la escuela, sirviendo de apoyo para la realización de ejercicios en el aprendizaje del diseño de bases de datos. La usabilidad de la aplicación y su facilidad de uso, serán pilares fundamentales en el diseño de esta herramienta.

A continuación, se enumerarán las funcionalidades que ofrece la aplicación, explicando las utilidades de cada una de ellas y el valor que ofrecen al usuario.

1. **Administración de diagramas entidad-relación**
    a. Creación – El usuario debe ser capaz de crear un diagrama entidad-relación desde cero. Será capaz de introducir cualquier elemento válido en un diagrama entidad-relación. La creación de cada uno de estos elementos generará su correspondiente visualización en el panel de dibujo, de tal manera que éste podrá ser colocado en el lugar que el usuario crea conveniente. La posición de aparición de cada una de las representaciones será un lugar prefijado por el sistema. Posteriormente el usuario desplazará la figura donde corresponda.
    La creación de cada uno de los elementos debe tener un orden lógico, orden que el propio sistema se encarga de recordar al usuario para evitar generación de diagramas erróneos. Así, por ejemplo, el sistema no permitirá la creación de una relación binaria si previamente no se han incluido dos entidades en el panel. Estas restricciones que impone la aplicación van dirigidas a guiar al usuario, el cual se presupone que puede ser relativamente inexperto.
    b. Gestión – La aplicación permitirá almacenar y recuperar los diagramas diseñados en cualquier unidad de disco sin límite de tamaño. Además, el sistema realizará el filtrado de los archivos alojados en el directorio para mostrar únicamente aquellos que correspondan a los diagramas almacenados. Se ha decidido que los archivos que almacenen los diagramas entidad-relación generados por esta herramienta, irán caracterizados por la extensión ". erd". Esto supone un cambio respecto de los archivos que se generaban con la versión anterior, cuyos ficheros se marcaban con la extensión ". der". El cambio de extensión se ha decidido porque determinados archivos correspondientes a certificados digitales ya van

acompañados de dicha extensión, lo cual podía generar algún tipo de confusión y conflicto.

c. Conversión de documentos – La aplicación es capaz de generar tres documentos diferentes, uno gráfico y dos de tipo texto. El documento de tipo gráfico es el propio diagrama entidad-relación, el cual podrá ser exportado y almacenado como fichero ".jpg". Los dos documentos de tipo texto se corresponden en primer lugar con el modelo lógico, y en segundo lugar con el código SQL necesario para generar la base de datos diseñada. Estos documentos podrán ser convertidos y almacenados como ficheros ".pdf".
d. Impresión – Esta funcionalidad no se ha implementado como tal, pues se ha considerado que los programas que manejan los archivos convertidos ya realizan dicha función y no sería necesario implementarlo de nuevo.

2. Asistencia en el desarrollo de diagramas

a. Desarrollo guiado – Ya se ha apuntado anteriormente que la aplicación espera un orden lógico en la creación de diagramas.

   En este sentido señalaremos algunas de las directrices que la propia aplicación marca en el momento del desarrollo.

   - La interfaz de usuario cuenta con una ventana de avisos que indican instantáneamente el error que se está cometiendo al ejecutar alguna acción que en ese momento no corresponde.
   - La aplicación no permite el uso nombres equivocados para los elementos, por ejemplo, la existencia de dos entidades con el mismo nombre.
   - No se permite la creación de entidades sin clave primaria obligatoria.
   - No se podrá crear una relación u objeto agregado si no existen las entidades que se han de relacionar.
   - No se podrán añadir atributos sin una preselección del elemento al que van asociados.
   - El sistema no permite añadir un atributo a un elemento que no admite atributos. Por ejemplo, no se puede añadir un atributo a otro atributo.
   - La aplicación no permitirá relacionar dos elementos no relacionables, por ejemplo, no se podrá crear una relación entre un atributo y na entidad.
   - La aplicación no permitirá la creación de objetos agregados entre elementos no agregables. No se podrán agregar una entidad y una relación.

b. Edición de elementos – Durante el desarrollo del programa la aplicación permite la edición del nombre de cada elemento, lógicamente habiendo sido creado anteriormente y previa selección del mismo.
c. Borrado de elementos – El borrado de elementos es coherente con su creación de tal manera que la eliminación por ejemplo de una entidad conllevará la eliminación automática de todos los atributos que tenga ligados. El borrado de los elementos se hará de forma individualizada y previa selección del elemento a borrar.
d. Borrado total del diagrama – En el momento que el usuario lo desee puede hacer un borrado total del diagrama. Para evitar borrados totales accidentales del diagrama, el programa se asegurará de que el usuario da el consentimiento afirmativo para borrar definitivamente todo el diagrama.

3. Generación de código

a. Automático e instantáneo – A partir del diagrama diseñado, la aplicación es capaz de desarrollar dos documentos de texto. Estos documentos se desarrollarán de forma transparente para el usuario. El usuario no escribe el código, el código se genera.

b. Diseño lógico – El primero de los documentos estará compuesto del pseudocódigo correspondiente al modelo lógico del diagrama entidad-relación. Dicho código sigue la nomenclatura general indicada para el diseño de este modelo, tanto en diferentes bibliografías dedicadas, como en el material didáctico expuesto en la Escuela. El diseño lógico generado por DEReditor_v2, pretende optimizar la posterior creación de tablas. Con esto se evita que todos los elementos del sistema vayan a tener su tabla correspondiente.

   El documento del modelo lógico es un listado compuesto por una descripción formateada en al cual se incluirán los nombres de las tablas obtenidas, lo campos necesarios que componen esas tablas, el dominio de los datos que rellenarán esos campos, la indicación de las claves primarias que identifican los elementos representados, las claves ajenas que intervendrán en cada tabla y finalmente las restricciones que cada campo tendrá.

c. Código SQL – El segundo de los documentos recoge el código SQL necesario para generar la base datos en cualquiera de los múltiples gestores existentes en el mercado. El código SQL se obtiene a partir de la generación del modelo lógico anterior, con lo cual, la aplicación avisará al usuario si no se realizan las acciones en el orden conveniente. Primero se diseña un diagrama, después se genera el modelo lógico, y por último se obtiene el código SQL correspondiente.

d. Visible y dinámico – La generación de los códigos explicados se puede ejecutar en cualquier momento del desarrollo, es decir no será necesario haber terminado el diagrama para poder observar cómo se van generando estos documentos. Como es lógico un diagrama no concluido generará códigos erróneos o inacabados.

e. Exportables y almacenables – Cuando el usuario lo crea conveniente se podrá convertir estos documentos de código en ficheros de texto. Los ficheros que se crean a partir de los textos generados se almacenarán de forma persistente en las unidades de almacenamiento que el propio usuario decida. Se ha diseñado la aplicación para que los documentos de modelo lógico y de código SQL sean almacenados como ficheros ".pdf". Para ello el sistema despliega los diálogos de almacenamiento pertinentes, permitiendo al usuario elegir el lugar en el cual quiere alojar estos archivos. Una vez guardados estos documentos, estos pueden ser configurados e imprimidos a gusto del usuario.

5. REQUISITOS

A continuación, se realizará una enumeración detallada de todos los requisitos especificados para la creación de la aplicación del proyecto. Esta especificación se desarrolla siguiendo las directrices marcadas por la norma IEEE "*Guide to Software Requirements Specifications*" [1].

1. **Introducción**
   1.1. Propósito

   El propósito del siguiente apartado es definir cuáles son los requisitos necesarios que debe cumplir DEReditor_v2, tanto en prestaciones, rendimientos y calidad, para obtener un resultado final satisfactorio. Los requisitos que se describirán serán en gran parte los que su día fueron especificados para el producto original. A estos se añadirán nuevos requerimientos, se eliminarán algunos de ellos y se modificarán otros, de tal manera que se obtenga un resultado con valor propio que introduzca mejoras e innovaciones y elimine posibles imperfecciones anteriores.

   1.2. Ámbito

   DEReditor v2 es una aplicación de diseño de diagramas en dos dimensiones, capaz de generar pseudocódigo correspondiente al diseño lógico de bases de datos, y código SQL para la creación final de las tablas que componen dicha base de datos.

   La interfaz de usuario y los diálogos necesarios durante su ejecución están internacionalizados a dos idiomas, inglés y español.

   Tanto los símbolos utilizados en el diagrama entidad-relación, como el pseudocódigo utilizado en la confección del diseño lógico, siguen la forma general vista en diferentes bibliografías o en las clases de las asignaturas de bases de datos impartidas durante el Grado en Ingeniería Informática.

   1.3. Definiciones, acrónimos y abreviaturas.

   Diagrama – En el contexto de esta aplicación, diagrama es un conjunto de elementos visualmente reconocibles cuyo compendio correctamente estructurado y organizado representará la forma final de la base de datos al que corresponde.

   Elemento – Cada uno de los objetos que formarán parte del diagrama. Cada elemento tiene unas características distintas en función del objeto al que representa, bien sea una Entidad, una Relación o un Atributo, por ejemplo.

   Unión – Cada una de las líneas que aparecen en la creación del diagrama y que sirven para interconectar cada uno de los elementos que conforman el gráfico. El diseño de la aplicación no permite la manipulación de las uniones, y su inclusión, borrado y posicionamiento es automático. La razón de este comportamiento es que una unión siempre va ligada a dos elementos, los cuales sí se pueden manejar por parte del usuario.

   Restricción – Cada una de las condiciones lógicas que un determinado elemento debe cumplir según las especificaciones del problema. El cumplimiento de las restricciones garantizará el funcionamiento deseado de la base de datos y por esto se consideran pilares fundamentales del diseño.

   UI – Del inglés "User Interface" cuya traducción es Interfaz de Usuario.

   Interfaz principal – Es la pantalla más importante con la que el usuario va a interaccionar. En esta pantalla principal se integran todos los botones e iconos que el diseñador necesita para crear su diagrama entidad-relación. Además, esta pantalla alberga el panel que servirá de lugar de trabajo donde se representará el diagrama que se está creando.

Panel de diálogo – Son pantallas secundarias que aparecen automáticamente a la vista del usuario y cuya finalidad es informar sobre alguna determinada acción, recoger algún dato que deberá introducir el usuario. La aceptación o cancelación de aquello que propone el panel de diálogo implica la desaparición visual del mismo.

Acceso rápido – Se llama acceso rápido a determinados botones o iconos situados en la interfaz de usuario, cuyo acceso y accionamiento no requiere de ninguna otra acción previa como puede ser desplegar un menú anterior.

2. Descripción general

2.1. Perspectiva del producto

Al igual que su antecesor, DEReditor_v2 cumple con los requerimientos generales del sistema. Así, podremos decir que el producto presentado es un sistema multiplataforma en el que no se utilizarán llamadas al sistema que condicionen su uso a un sistema operativo u otro.

El modo de ejecución es de interpretación, de tal manera que las instrucciones ejecutadas son interpretadas a lenguaje máquina, adecuado a las características del entorno operativo en el que se ejecuta la aplicación.

El producto es portable e independiente de cualquier sistema hardware conectado al ordenador.

2.2. Funciones del producto

2.2.1. Selección de idioma de la interfaz. La herramienta está internacionalizada al inglés y al español.

2.2.2. Creación y manipulación de diagramas

- 2.2.2.1. Crear un nuevo diagrama en blanco
- 2.2.2.2. Poner título al nuevo programa en blanco
- 2.2.2.3. Introducir elementos en el diagrama en blanco
- 2.2.2.4. Eliminar elementos del diagrama
- 2.2.2.5. Editar elementos del diagrama
- 2.2.2.6. Borrar todo el diagrama
- 2.2.2.7. Desplazar elementos dentro del diagrama
- 2.2.2.8. Guardar el diagrama en un fichero ".erd"
- 2.2.2.9. Abrir un diagrama desde fichero
- 2.2.2.10. Exportar y guardar el diagrama como fichero ".jpg"
- 2.2.2.11. Modificar un diagrama abierto desde fichero

2.2.3. Creación y manipulación de elementos

- 2.2.3.1. Entidades fuertes
- 2.2.3.2. Entidades débiles
- 2.2.3.3. Entidades especializadas
- 2.2.3.4. Atributos clave primaria
- 2.2.3.5. Atributos sin restricción
- 2.2.3.6. Atributos derivados
- 2.2.3.7. Atributos no nulos
- 2.2.3.8. Atributos multi-valuados
- 2.2.3.9. Relaciones binarias
- 2.2.3.10. Relaciones ternarias
- 2.2.3.11. Relaciones reflexivas

2.2.3.12. Objetos agregados
2.2.3.13. Especializaciones parcial-disjunta
2.2.3.14. Especializaciones parcial-solapada
2.2.3.15. Especializaciones total-disjunta
2.2.3.16. Especializaciones total-solapada
2.2.3.17. Uniones automatizadas

2.2.4. Generación de códigos
2.2.4.1. Pseudocódigo del diseño lógico
2.2.4.2. Código SQL
2.2.4.3. Exportación y guardado del pseudocódigo como fichero ".pdf"
2.2.4.4. Exportación y guardado del código SQL como fichero ".pdf"

2.3. Características del usuario

Los usuarios del sistema pueden ser muy variopintos, partiendo de alumnos inexpertos, nuevos en el ámbito de las bases de datos, o pueden ser experimentados programadores necesitados de esta herramienta para el desarrollo de su trabajo. No obstante, la idea con la que se ha diseñado esta segunda versión de DEReditor, es la de ir dirigida muy especialmente a los alumnos que están inmersos en el aprendizaje del desarrollo y funcionamiento de las bases de datos. La manipulación del programa orienta al usuario con el fin de hacer más sencillo la creación de un diseño final correcto, si bien es cierto que los diagramas no se hacen solos, y se necesita de unos mínimos conocimientos para finalizar con un resultado satisfactorio.

Lo cierto es que, en ambos casos, tanto si el usuario es inexperto como si es un experimentado conocedor de las bases de datos, la persona que se ponga a los mandos del programa se encontrará una aplicación amistosa, cómoda, sencilla, eficiente y muy válida para realizar diseños con apariencia moderna y profesional. Igualmente podrán obtener códigos cuyos usos pueden ir desde el mero estudio, hasta la generación de bases de datos plenamente funcionales.

2.4. Obligaciones generales

En el momento del comienzo del proyecto, se plantearon varios requisitos mínimos o aspectos fundamentales que se debían cumplir para que el proyecto tuviese un por qué necesario para su realización, es decir las razones por las cuales se debía desarrollar esta aplicación tal y como se ha hecho.

En primer lugar, se debía mejorar notablemente la interfaz gráfica de usuario. Ésta debía ser transformada en una interfaz con una apariencia fresca, moderna y atractiva para el alumno. A su vez esta interfaz debía proporcionar una usabilidad muy alta, proporcionando una facilidad de uso muy ágil, y eliminando posibles dudas durante el desarrollo de los proyectos.

En segundo lugar, la aplicación debía eliminar los bugs detectados de la versión anterior, especialmente críticos los respectivos al almacenamiento persistente y posterior recuperación de diagramas desde fichero.

Además, la aplicación deberá evolucionar en la generación de tablas, haciendo de forma eficiente tanto el número que se crean de estas como en la forma de crearlas. Es decir, no todos los elementos que conforman el diagrama van a tener su tabla asociada. La eficiencia en este caso radica en que es sistema sea capaz de decidir de forma correcta qué elementos han de tener su tabla y cuáles no, e incluir en los lugares que

corresponda las referencias adecuadas de identificación de cada uno de los elementos que intervengan en cada acción.

Por otro lado, a nivel interno, la codificación de la aplicación debía ser estructurada de manera correcta siguiendo en este caso el patrón Modelo, Vista, Controlador. En la versión anterior no se siguió dicha estructura y el orden de la estructura de ficheros que conforman el proyecto se ve comprometido.

3. Requisitos específicos

3.1. Requisitos funcionales

3.1.1. Características de un elemento

Todo elemento que se introduce en el diagrama pertenece a una superclase denominada **Elemento**. De esta superclase heredan las subclases que definen cada uno de diferentes componentes del diagrama. Así, podremos decir que la superclase Elemento homogeneiza las siguientes subclases, Entidad, Atributo, Relación, Agregación, Especialización y Unión.

Todos los elementos, sean de la subclase que sean, serán capaces de generar una vista propia y característica de cada uno de ellos. Dicha vista será la cualidad visible para el usuario dentro del diagrama. La generación de cada una de las vistas es transparente para el usuario y su aparición es automática.

Elemento, para su definición se deberá indicar un nombre, una posición instantánea, una posición inicial, un booleano sobre la admisión de atributos, un booleano sobre la admisión de relaciones y un booleano sobre la admisión de agregaciones. Cabe señalar que, aunque la posición nos recuerda más a características de la vista, en el caso que nos ocupa es necesario guardar estos valores como recuerdo, principalmente cuando se recupera un diagrama desde fichero y debemos saber dónde se deben situar las vistas que se generan. Si no se guardasen estas posiciones obtendríamos diagramas completos, pero apelotonados en una posición única, amasijos de figuras que habría que recolocar cada vez que se recargase un diagrama desde fichero. La superclase Elemento contendrá una lista de los atributos asociados a dicho elemento.

Entidad, subclase que a su vez está dividida en varios tipos, Entidad fuerte, Entidad débil y Entidad especializada. Estos elementos guardarán, si las hubiera, los tipos de herencia que contuviesen, la entidad fuerte a la que están ligadas en caso de ser débiles y la entidad padre de las colgasen en caso de ser entidades especializadas.

Atributo, subclase que necesita de un dominio y de una serie de restricciones posibles, que pueden ser clave primaria, valor no nulo, derivado o multi-valuado. A efectos del programa, el dominio sólo tendrá utilidad cuando se generen el diseño lógico o el código SQL. El dominio no tiene ninguna repercusión en el diagrama.

Cuando se crea una entidad, en la que sea obligado llevar asociada una clave primaria, ambos elementos, entidad y atributo clave primaria serán creados a la vez, y no tendrán ningún sentido uno sin el otro.

Ilustración 1: vista entidad y diferentes tipos de atributos

Relación, subclase dividida en tres posibles relaciones, binaria, binaria reflexiva y terciaria. Las relaciones guardarán una referencia de los elementos que relacionan y de la cardinalidad con que lo hacen.

Ilustración 2: vista relación con cardinalidades

Agregación, subclase de un único tipo que almacenará una referencia a los elementos que agrega y a la cardinalidad con que lo hace.

Ilustración 3: vista agregación con cardinalidades

Especialización, subclase dividida en cuatro tipos, PARCIAL-SOLAPADA, PARCIAL-DISJUNTA, TOTAL-SOLAPADA Y TOTAL-DISJUNTA. Este elemento almacenará una referencia a la entidad padre a la cual se asocia la especialización.

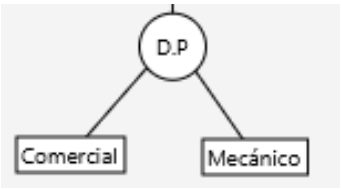


Ilustración 4:vista especialización PARCIAL-DISJUNTA

Unión, es el elemento que liga al resto de elementos del diagrama. Cada unión se definirá según el tipo de elementos que deba conectar. Serán diferentes uniones aquellas que contengan un indicador de multi-valor, que indique una restricción de existencia o que unan una relación con una entidad débil. Toda unión guarda una referencia de los elementos que une. Las uniones no serán manipulables por el usuario.

3.1.2. Creación de elementos

Siguiendo con uno de los requisitos generales más importantes, la creación de los elementos que se deben integrar en el diagrama debe ser lo más sencilla e intuitiva posible. Se intentarán evitar los diálogos ambiguos o la posibilidad de acciones al azar por parte de usuario. Durante la creación del elemento se configurarán todas las características que lo definan. Una vez se haya configurado el elemento a introducir en el diagrama, el usuario podrá aceptar o desechar dicho elemento.

3.1.3. Diagrama de trabajo

Será una superficie plana y vacía inicialmente, capacitada para albergar visualmente las representaciones de los elementos que van a conformar el diagrama. El diagrama permitirá que en su extensión los elementos se introduzcan, se eliminen, se modifiquen y se desplacen.

3.1.4. Creación de un diagrama de trabajo nuevo

Al comenzar la sesión de trabajo en el programa, el usuario tendrá la opción de crear un diagrama de trabajo nuevo. En cualquier momento del desarrollo, el usuario podrá decidir crear un nuevo diagrama de trabajo. En tal caso, la aplicación deberá advertir al usuario de que el diagrama actual que se encuentra en pleno progreso se perderá si no se hace un guardado persistente del mismo. Cuando esto ocurra el usuario podrá aceptar la operación o cancelarla.

3.1.5. Desplazamiento de los elementos por el diagrama

Los elementos introducidos en el diagrama podrán ser desplazados por él con la finalidad de hacer una distribución coherente de los componentes del mismo. El arrastre de cada elemento se hará manteniendo la selección con el puntero del ratón sobre dicho elemento y moviendo hasta soltar en el lugar deseado. Las uniones serán elementos no movibles por sí mismas, sino que seguirán de forma automática el movimiento de los elementos que unen.

3.1.6. Selección de los elementos del diagrama

Cada uno de los elementos que conforman el diagrama será seleccionable mediate un clic de ratón. Todos los elementos salvo las líneas de unión que ya hemos dicho que no son manipulables por el usuario.

3.1.7. Edición de los elementos del diagrama

La edición de los elementos del diagrama se limitará únicamente al cambio de nombre de cada elemento. El programa actualizará automáticamente el tamaño de su representación gráfica. El usuario podrá aceptar o cancelar la operación de edición antes de cambiar nada.

3.1.8. Eliminación de elementos del diagrama

Se podrán eliminar elementos del diagrama de forma individual o total. La eliminación total del diagrama deberá ser autorizada de nuevo por el usuario. La eliminación de elementos debe seguir la coherencia de creación, es decir si se elimina una entidad, automáticamente con ella desaparecerán todos sus atributos y uniones que la conectaban. La decisión de la eliminación total de elementos de un programa deberá poder ser aceptada o rechazada después del aviso por parte de la aplicación a través del panel de diálogo pertinente.

3.1.9. Guardado persistente del trabajo

En cualquier momento del desarrollo de un diagrama, el usuario tendrá la posibilidad de almacenarlo de forma persistente, garantizando su integridad en una posterior recarga. El usuario tendrá la posibilidad de nombrar al fichero donde se guarde el trabajo a fin de reconocerlo en su posterior apertura. De forma

automática el sistema incluye la extensión ".erd" en el nombre del fichero almacenado.

3.1.10. Apertura de un diagrama de trabajo desde fichero

Al inicio de una sesión o durante el transcurso de ésta, el usuario podrá decidir abrir un diagrama desde fichero. El fichero que se intente abrir deberá ser compatible con la aplicación. En el caso de abrir un diagrama desde fichero teniendo un diagrama en desarrollo, la aplicación deberá avisar al usuario de la pérdida del trabajo actual. En tal caso el usuario podrá aceptar o cancelar la operación.

3.1.11. Exportación del diagrama como imagen

En el momento que usuario lo decida, el programa permitirá convertir el diagrama en desarrollo en una imagen de tipo ".jpg". Dicha imagen se guardará de forma persistente con el nombre que el usuario elija. Esta posibilidad está dirigida a la posterior manipulación del diagrama para poder ser incrustado en otros documentos o para ser impreso en papel, por ejemplo.

3.1.12. Generación de código para el diseño lógico

A partir del diagrama que se está desarrollando, el usuario tendrá la posibilidad generar el pseudocódigo correspondiente al diseño lógico del diagrama. El diseño lógico se generará de forma instantánea en cualquier momento de trabajo. Este diseño se mostrará en un panel creado para tal efecto. Diagramas incompletos generarán códigos erróneos.

3.1.13. Diseño lógico

La estructura y forma del código generado será entendible por cualquiera que tenga unos mínimos conocimientos de bases de datos, es decir su forma y estructura seguirá los estándares vistos en la bibliografía consultada o en la materia vista en clase para la asignatura de "Bases de Datos I" impartida por la profesora docente Piedad Garrido Picazo.

3.1.14. Exportación del diseño lógico a archivo de texto

En cualquier momento del desarrollo, el usuario podrá decidir exportar el diagrama a un archivo ".pdf". Este archivo se guardará de forma persistente con el nombre que el usuario elija.

3.1.15. Generación de código SQL

En cualquier momento del desarrollo, el usuario podrá generar el código SQL correspondiente al diagrama que se está desarrollando. Para generar un código SQL válido se deberá seguir un orden de actuación correcto, pues las instrucciones SQL se generan a partir del diseño lógico previo.

3.1.16. Código SQL

La finalidad de generar el código SQL correspondiente al diagrama, es que este código deberá cumplir con la estructura y sintaxis necesaria para poder ser introducido en cualquiera de los gestores de bases de datos existentes en el mercado, y generar la base de datos tal y como se ha diseñado en el diagrama. Es por esto que el código SQL estará compuesto por las instrucciones y comandos que ofrece el estándar SQL.

6. ESTADO DEL ARTE

Analizar el estado del arte en determinados campos, en ocasiones se puede hacer largo y tedioso, pues poner límites a lo que vas a analizar o comparar puede ser complicado e ocasiones.

Sondeando el mercado y un poco más a la aplicación docente de la herramienta vamos a ver con cierto detalle varias de las herramientas que también fueron concebidas para el ámbito de la enseñanza.

6.1 DBDAP

Es una herramienta pensada para el ámbito educativo, que pretende que el alumno tenga un apoyo en su uso, y le ayude a desarrollar problemas de modelado. Incorpora soporte para la definición de esquemas en estrella para almacenes de datos y diferentes notaciones gráficas para el diseño conceptual, como el diagrama E/R y el lenguaje UML.

Funcionalidades de la herramienta

Esta herramienta dispone de un entorno gráfico para la creación de diagramas entidad-relación con un sistema de detección de errores del diseño. Permite incluir información sobre dependencias funcionales y dependencias multivaluadas que existan entre los datos.

Para el diseño de los diagramas entidad-relación esta herramienta distingue entre modelos básicos y modelos extendidos. Respecto al primero maneja elementos como entidades fuertes y débiles, atributos identificadores y cualificadores, sus dominios, relaciones y sus tipos. En el modelo extendido se pueden mencionar tipos de atributos, cardinalidades, interrelaciones exclusivas, generalización/especialización y agregaciones.

También, aunque no es muy usual diseñar las bases de datos utilizando el lenguaje UML, la herramienta también permite realizar esquemas conceptuales mediante este sistema.

Este programa permite configurar la notación a utilizar en el diagrama entidad-relación.

La herramienta también está diseñada para hacer la normalización de los esquemas relacionales desarrollados.

La obtención del diseño lógico a partir del diagrama entidad-relación es automático. Ofrece ver las transformaciones paso a paso con el fin de proporcionar al alumnado un mayor entendimiento sobre el proceso.

La herramienta añade automáticamente restricciones para garantizar que se cumplen las propiedades deseadas de los datos.

Para poder llevar a cabo la normalización el sistema detecta automáticamente dependencias funcionales y multivaluadas.

Se puede realizar una traducción del diseño lógico a código SQL. Este proceso genera un archivo con instrucciones ejecutables desde los distintos Sistemas de Gestión de Bases de Datos.

La herramienta permite la definición de esquemas lógicos en estrella y su transformación a SQL.

El sistema permite guardar en soporte físico los documentos generados, tanto en imagen como en texto.

Ventajas y desventajas DBDAp vs DEReditor_v2

DBDAp es una aplicación concebida con la misma idea inicial de ayudar al alumnado en sus desarrollos de los diagramas entidad-relación.

Vamos a considerar una ventaja de esta aplicación que en algunos aspectos esta herramienta es más completa que DEReditor_v2. Por ejemplo:

- La herramienta permite generar los diseños en UML.
- Se añaden algunas restricciones de diseño de forma automática.
- Permite notaciones alternativas en el modelo E/R
- Tiene utilidades aplicables en almacenes de datos
- Realiza la normalización de esquemas relacionales
- Permite la definición de esquemas en estrella

Bien es cierto que esta herramienta tiene más funcionalidades que DEReditor_v2, pero entendamos que DEReditor_v2 está pensado desde su inicio como una herramienta de desarrollo de diagramas entidad-relación, su diseño lógico y el código SQL correspondiente.

Así pues, vamos a considerar que no será una ventaja todo aquello que contradiga el principio básico y fundamental que se ha querido imprimir a la aplicación, se busca la simplicidad lo máximo posible. Por tanto:

- Funcionalidades no útiles en la esencia del proyecto pueden confundir al usuario.
- La interfaz de desarrollo no abre segundas ventanas durante el proceso de creación del diagrama, con lo cual expone de forma mucho más sencilla las opciones de diseño.
- El diseño de la interfaz de usuario de DEReditor_v2 es mucho más llamativa y actual.

6.2 DIA

El programa “DIA” sirve para crear y editar gráficos que se está utilizando de manera docente como diagramas entidad-relación. El problema que surge con este software es que no es una herramienta específica para el caso que nos ocupa y por tanto sus prestaciones son muy limitadas al respecto.

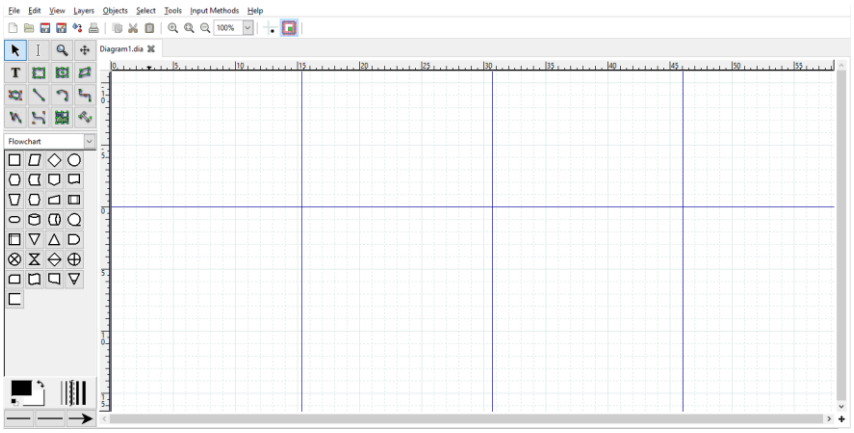


Ilustración 5: entorno de trabajo DIA

Es cierto que el entorno de trabajo que nos ofrece esta aplicación es muy intuitivo y usable. Sigue la misma filosofía que se ha aplicado en DEReditor_v2. Sin embargo, al no ser una herramienta específicamente pensada para bases de datos no se podrán obtener los resultados más satisfactorios.

Ventajas y desventajas DIA vs DEReditor_v2

Algunas ventajas señalables pueden ser:

- La creación de un diagrama puede ser sencilla pues los elementos se tienen a la vista.
- Cada elemento es altamente editable, tamaño, color, grosor de línea…
- Permite la impresión de los diagramas desde la propia aplicación.

Pero realmente estas ventajas que podemos indicar pierden toda su validez cuando enumeramos las desventajas que surgen:

- Con DEReditor_v2 se crean elementos de un diagrama, no dibujos. Es decir, un usuario creará una Entidad y su cabeza pensará en un objeto Entidad no en un dibujo.
- Con DIA cualquier diagrama puede estar bien erróneamente porque no se rige por las reglas de bases de datos para realizar los diagramas entidad-relación. Errores en el diagrama pueden pasar desapercibidos.
- El diagrama de DEReditor es un entidad-relación específicamente. DIA crea diagramas genéricos.
- DEReditor_v2 genera a partir del diagrama entidad-relación, un diseño lógico y código SQL. DIA no puede hacer eso pues no es un programa específico de bases de datos.

6.3 Visual Paradigm Online

Esta herramienta se encuentra en la red. Genera los diagramas online. Visitando esta dirección se puede ver qué es lo que se puede conseguir con esta herramienta.

Visual Paradigm[2] es una herramienta que permite construir múltiples tipos de diagramas, incluidos los diagramas entidad relación.

Ventajas y desventajas de Visual Paradigm Online vs DEReditor_v2

Enumeremos algunas posibles ventajas:

- Es un programa que genera múltiples diagramas no sólo entidad-relación.
- Provee de plantillas al usuario que puede utilizar para hacer sus diseños.

Pero sinceramente no se ve en este programa la exclusividad que buscamos. Queremos un programa de aplicación a bases de datos. Que genere diagramas entidad-relación, diseños lógicos y código SQL.

Por tanto, las desventajas que observamos van por este camino:

[2] https://online.visual-paradigm.com/drive/#diagramlist:proj=0&new=ERDiagram

- Si ofrecer la posibilidad de crear diferentes diagramas es loable, en este caso no aporta nada más que caos para encontrar la opción que deseamos.
- La notación de los diagramas que se crean no es la notación que hemos estudiado ni la que esperamos utilizar en la parte práctica de la asignatura. No se debe liar al alumno.
- Esta aplicación no genera código, ni diseño lógico si SQL.
- No es nada intuitivo crear un diagrama entidad-relación con este programa.

6.4 DEREDITOR

DEReditor es una aplicación que se desarrolló como proyecto final de un alumno en el año 2005. Cuando se planteó la posibilidad de realizar un nuevo proyecto que fuese la actualización del proyecto de 2005, no se pensó en lo finalmente ha ocurrido. DEReditor_v2 es un proyecto nuevo, con apariencia nueva y códigos nuevos. Por tanto, DEReditor se a considerar una herramienta más que vamos a comparar con DEReditor_v2.

DEReditor es una herramienta para el apoyo en la creación de diagramas entidad-relación, capaz de generar el diseño lógico correspondiente y el posterior código SQL.

Ventajas y desventajas DEReditor vs DEReditor_v2

Algunas características del antiguo DEReditor no se han implementado en la segunda versión. Han sido decisiones de diseño, que por diferentes razones no se ha creído necesario implementar. Algunas de estas características o funcionalidades que ahora no aparecen en el nuevo programa pueden considerarse desventajas respecto de la aplicación anterior. Por esto, las enumeraremos como inconvenientes o pérdida de prestaciones respecto de DEReditor:

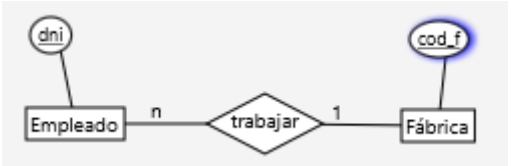
- En primer lugar, DEReditor genera los archivos con extensión ".der". DEReditor_v2 lo hace con extensión "erd". La extensión ".der" la tiene aplicada, y así lo reconocen los sistemas, archivos de certificación de seguridad, lo cual genera conflictos con la aplicación y confusiones en el usuario.
- DEReditor permite una edición de los elementos generados más al detalle que DEReditor_v2.
- DEReditor permite la impresión del diagrama en papel. DEReditor_v2 exporta sus documentos como ficheros ".jpg" y "pdf". La impresión se podrá realizar después desde los programas adecuados que abran esos ficheros.
- DEReditor permite cargar un diagrama dentro de otro ya existente. En DEReditor_v2 no se ha planteado esa posibilidad.

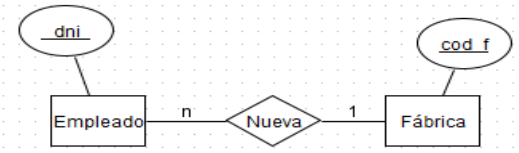
Al desarrollarse DEReditor_v2 con la idea de mejorar la primera versión, las ventajas en este caso son más claras, pues en algunos casos DEReditor_v2 repara determinados bugs de DEReditor:

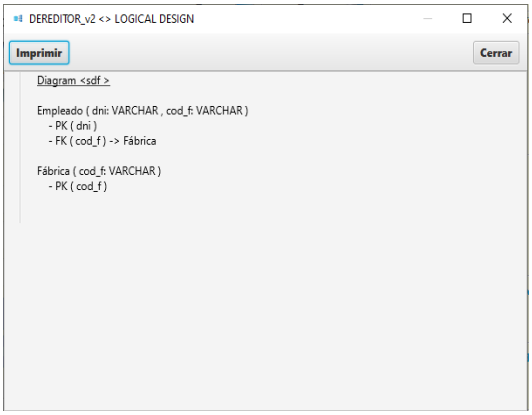
- DEReditor_v2 realiza una gestión correcta de los archivos generados, de tal manera que hasta la fecha no ha ocurrido ninguna pérdida de información por ningún tipo de excepción, o por la imposibilidad de recarga de ningún diagrama desde archivo.
- La creación de diagramas usando DEReditor es mucho más complicada en el sentido que los elementos que se van creando abren nuevas ventanas donde la introducción de datos no es tan intuitiva.

- La interfaz gráfica de la primera versión se ve obsoleta, nada llamativa o vanguardista. Esto es debido a que en el primer caso la interfaz gráfica se realizó utilizando “Java Swing”, mientras que en la versión actual se ha utilizado “JavaFX” [2] con el apoyo de “Scene Builder” [3].
- La generación del diseño lógico por parte de la primera versión, no optimiza el número de tablas que van a formar parte de la base de datos, sino que todos los elementos que aparecen en el diagrama generarán su tabla correspondiente. Por ejemplo:

  Dada la siguiente estructura creada en las








```
-- Inicio de pseudocodigo --

/* Entidad Empleado: */
Empleado (dni:dom_dni )
     Clave Primaria { dni }


/* Entidad Fábrica: */
Fábrica (cod_f:dom_cod_f )
     Clave Primaria { cod_f }


/* Relacion trabajar: */
trabajar (dni:dom_dni, cod_f:dom_cod_f )
     Clave Primaria { cod_f }
     Valor No Nulo { dni }
     Clave Ajena { dni } hace referencia a Empleado
     Clave Ajena { cod_f } hace referencia a Fábrica


-- Fin de pseudocodigo --
```

  DEReditor genera tres tablas, Empleado, Fábrica y trabajar, mientras que DEReditor_v2 realiza la inclusión de las claves ajenas en las tablas adecuadas para para no crear las que surgen de las relaciones.

- Durante el desarrollo de los diagramas, cuando utilizamos DEReditor el programa permite al usuario cometer determinados errores, como puede ser, añadir varias claves primarias a una entidad. DEReditor_v2 pretende guiar al usuario hacia un correcto diagrama y evita que se realicen acciones erróneas.
- En ocasiones DEReditor, entra en bucles de error (se cuelga), no dejando muchas alternativas al usuario para salir esta situación sin realizar ninguna medida drástica, como reiniciar, borrar todo o empezar de nuevo. Hasta la fecha se ha dado ninguna situación similar con DEReditor_v2.
- Algunas modificaciones del diagrama no generan los cambios adecuados en el diseño lógico.

7. ANÁLISIS Y DISEÑO DE LA INTERFAZ

La interfaz de usuario es una parte importantísima de este proyecto. La mejora sustancial de la interfaz de usuario es una de las partes principales del porqué de este trabajo.

A continuación, vamos a mostrar la evolución de cómo se pensó que debía ser la apariencia visual, siguiendo una evolución de los primeros diseños hasta el resultado final.

Para mostrar estos diseños usaremos la herramienta "Balsamiq" [4] con la obtendremos los "Mock Up" de la interfaz.

Primeras ideas

Al principio se planteó una pantalla con un gran panel de trabajo dominando el espacio. Por encima en primer lugar una típica barra de herramientas, y entre la barra de herramientas y el panel una serie de desplegables que permitiesen la creación de los elementos del diagrama.

Esta disposición se desechó porque mientras se manipulaban los desplegables cubrían de forma incómoda el diagrama.

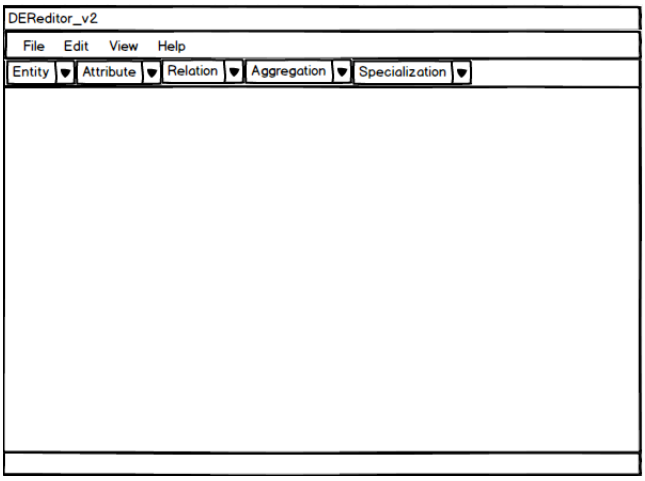


Ilustración 6: primer acercamiento al diseño de la interfaz de usuario

Segundo paso

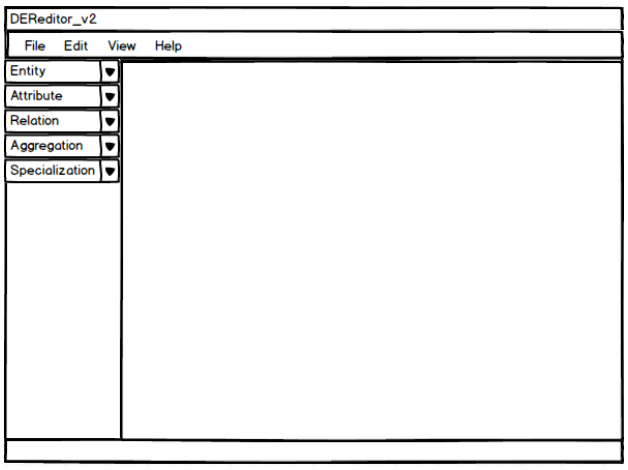


Ilustración 7: segundo diseño de la interfaz

Con el segundo diseño se conseguía evitar la molestia de tapar el panel mientras se usan otros controles. Pero en este diseño se echaban de menos botones, o mejor aún iconos de acceso rápido para hacer determinadas acciones sin necesidad de buscar dentro menús.

Tercer paso

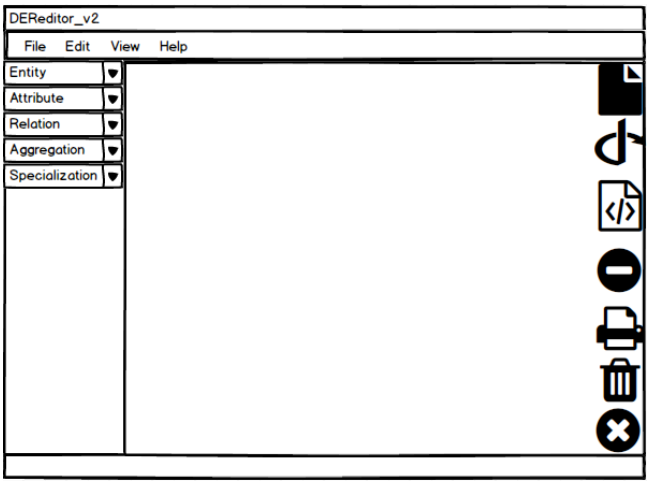


Ilustración 8: tercer diseño de la interfaz de usuario

Se incorporan seis iconos que lanzan un evento al clic de ratón. Se sitúan a la derecha del panel y siempre están visibles. Crea un diagrama nuevo, abrir una desde fichero, editar un elemento, borrar un elemento o el diagrama entero, guardar el diagrama como ".jpg" o salir de la aplicación son las funcionalidades implementadas en barra vertical de iconos.

Cuarto paso

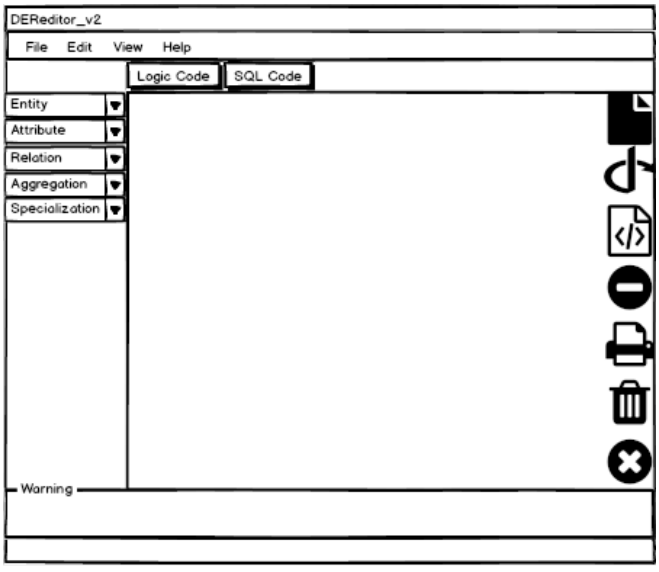


Ilustración 9:cuarto diseño de la interfaz de usuario

Finalmente se decide colocar dos controles más en la parte superior del panel con los que poder generar los códigos del diseño lógico y SQL, de forma rápida e instantánea.

Para hacer la labor de tutorización en el diseño, se añade en la parte inferior del panel un espacio en el irán apareciendo los avisos que en determinados momentos sean necesarios.

8. ARQUITECTURA Y FUNCIONAMIENTO

DEReditor_v2 es un programa pensado para actualizar la versión inicial DEReditor. Para el desarrollo del código de programación se ha optado por la utilización de la tecnología "*JavaFX*". La estructuración del proyecto en desarrollo que ofrece es tecnología, se hacía muy adecuada para conseguir parte de los objetivos marcados para la aplicación. Además, "*JavaFX*" cuenta con la compatibilidad de un programa aliado, que evoluciona la creación de interfaces gráficas haciendo que el antiguo "*Swing*" de "*Java*" se quede obsoleto. Hablamos de "*Scene Builder*". Así, aprovechando que "*JavaFX*" parte de una estructura de proyectos Modelo, Vista y Controlador, y que "*Scene Builder*" proporciona una eficiencia muy avanzada en la creación de interfaces de usuario, tenemos ante nosotros las herramientas ideales para realizar el desarrollo del proyecto.

A diferencia de la primera versión de DEReditor, en esta ocasión se ha decidido escribir el código de programación en inglés. De igual manera, se ha decidido ir más allá e internacionalizar la aplicación al español y al inglés. En principio es una aplicación orientada a ser utilizada en la universidad de Zaragoza, pero se ha pensado que toda aplicación que se precie debe por lo menos ofrecer su vista en el idioma local y en inglés. Este idioma será seleccionable por el usuario al inicio de cada sesión de trabajo.

8.1 ARQUITECTURA DE LA APLICACIÓN

DEReditor_v2 sigue el patrón de diseño MVC. En busca de la mayor simplicidad posible, se ha pretendido diseñar una aplicación que desarrolle prácticamente toda su actividad utilizando una única interfaz principal de usuario, apoyada para determinadas acciones por sencillos diálogos de confirmación o solicitud de algún dato aislado.

La estructura de los proyectos desarrollados mediante la tecnología "*JavaFX*", van guiados hacia una configuración siguiendo el patrón modelo-vista-controlador. Las vistas se ligan automáticamente con un controlador que maneja sus eventos. Por lo tanto, al tener una UI principal, también se contará con un controlador principal que maneja las acciones de esta interfaz.

La interfaz principal desencadena la gran mayoría de los eventos necesarios en la aplicación, lo cual hace que se produzca un crecimiento excesivo en el tamaño del controlador. A la vista de la evolución de este crecimiento, se tomó la decisión de crear sub-controladores manejados por el controlador principal para delegar acciones y hacer del control principal una clase más manejable.

Cada uno de los sub-controladores se encargará de gestionar los eventos que compartan una determinada temática. De esta manera por ejemplo se tendrá un sub-controlador para los eventos de entidad, un sub-controlador para los eventos de relación, etc.

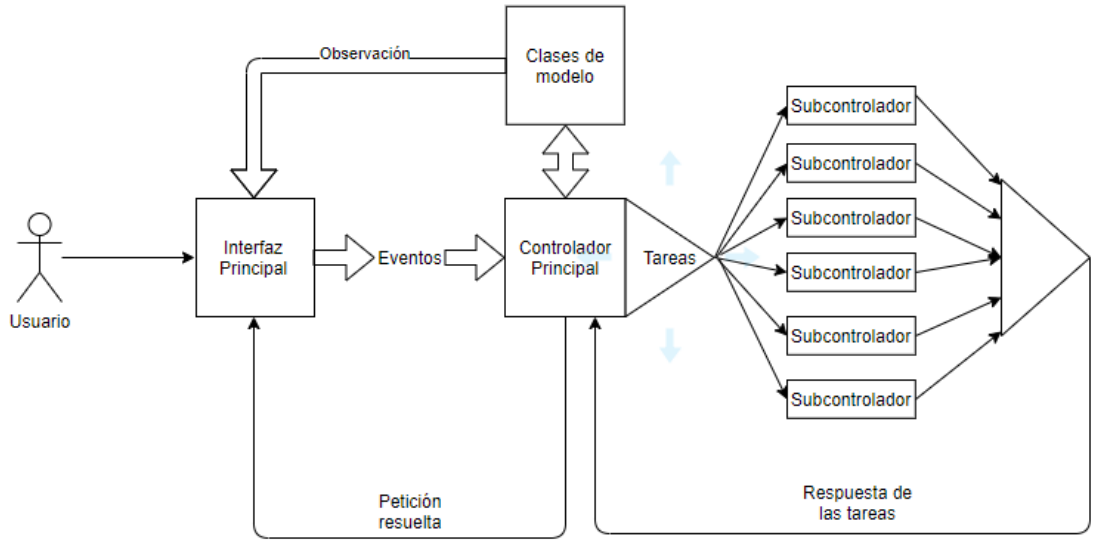


Ilustración 10: modelo general de la arquitectura de la aplicación

Se cuenta con un total de nueve sub-controladores. Cada uno de ellos tiene asignada una serie de tareas correspondientes a los elementos que se manipulan, bien sean de creación de elementos o bien para dar respuesta a determinados eventos. Así, los nueve sub-controladores son:

- EntityCtrl
- AttributeCtrl
- RelationCtrl
- AggregationCtrl
- SpecializationCtrl
- UnionCtrl
- EditionCtrl
- MenuBarCtrl
- QuickAccessCtrl

Cuando se genera un evento en la vista, este requiere de una respuesta por parte del controlador principal. Este controlador delegará en el sub-controlador correspondiente la tarea requerida, obteniendo la respuesta adecuada y siendo, si fuese preciso, devuelta a la vista o enviada al modelo.

8.2 VISTA

8.2.1 Interfaces de usuario

La vista de la aplicación es una parte fundamental de la aplicación. Se ha puesto mucho interés en crear una interfaz amigable, agradable al usuario y muy intuitiva. Su desarrollo se ha llevado a cabo utilizando el programa "*Scene Builder*", lo cual facilita en gran medida el diseño de interfaces modernas y llamativas.

Pantalla de bienvenida

Para empezar, la herramienta da la bienvenida al usuario y le invita a seleccionar el idioma que desea encontrarse en la interfaz de trabajo.

Ilustración 11: pantalla de bienvenida de la aplicación

Tras la selección del idioma y el accionamiento del botón de entrada, la pantalla desaparece para dejar su espacio a la interfaz principal de usuario. La internacionalización de la aplicación es una de las mejoras introducidas en esta segunda versión de DEReditor.

Pantalla principal

Es la pantalla donde se va a desarrollar la gran mayoría de las acciones. Esta interfaz está mayoritariamente ocupada por el panel de trabajo. Alrededor de éste, y de forma muy accesible, se encuentran los botones e iconos se servirán para la construcción del diagrama.

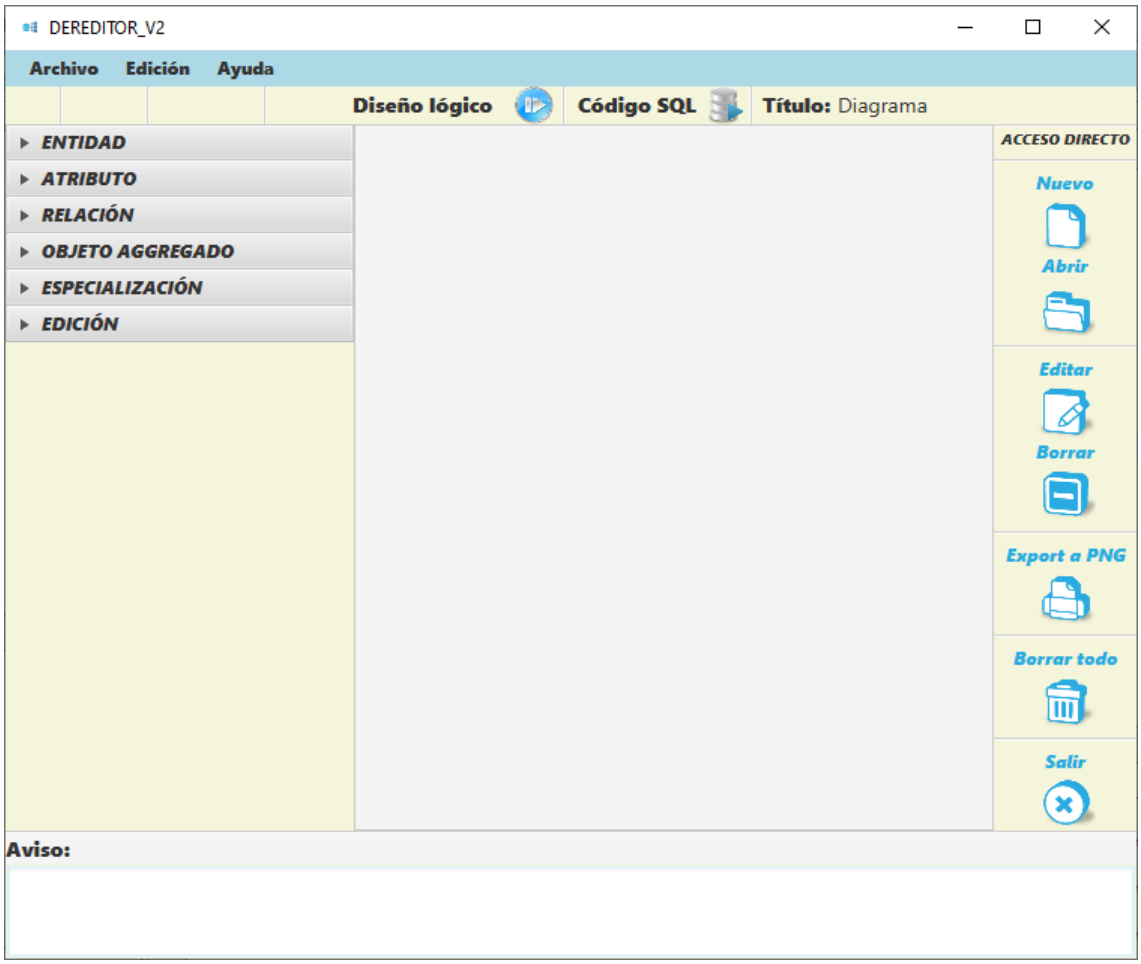


Ilustración 12: pantalla principal

Como puede observarse, esta interfaz ofrece al usuario una paleta de herramientas visibles y de fácil acceso, haciendo que el usuario no deba especular sobre dónde puede encontrar una u otra opción.

Se ha intentado en todo momento no perder la perspectiva de a qué tipo de usuarios va dirigida la aplicación, es decir se presuponen usuarios en estado de aprendizaje con conocimientos reducidos en la materia. De ahí que se busque la máxima simplicidad y tutorización en la creación de diagramas con esta herramienta. El usuario debe invertir su tiempo en entender lo que está haciendo y no en entender la herramienta con la que está trabajando.

A primera vista se pueden observar tres bloques principales de botones o iconos de acción.

En el bloque de la izquierda se pueden leer los nombres de todos aquellos elementos que pueden formar parte de un diagrama entidad-relación. Cada uno de estos es un menú desplegable que abre la usuario todas las acciones y opciones que puede seleccionar cuando crea un elemento.

Ilustración 13: menús desplegables para cada elemento

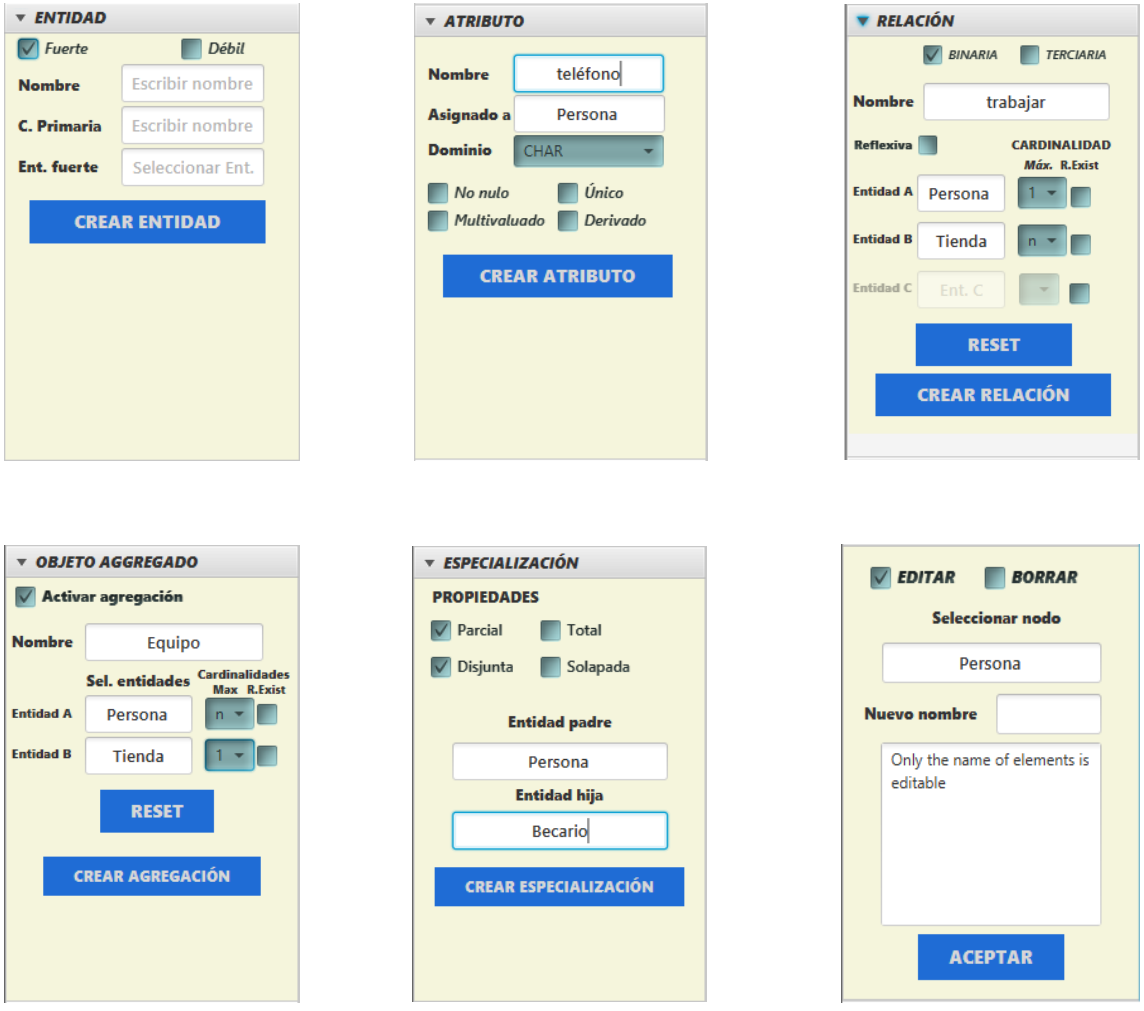


Ilustración 14: paletas de creación de elementos

El bloque de la derecha es un conjunto de iconos que admiten un clic de ratón para generar un evento. Estos iconos lanzan acciones como crear un nuevo diagrama, abrir desde fichero un diagrama ya existente, editar un elemento, borrar un elemento o un diagrama completo, exportar el diagrama a un fichero del tipo ".jpg" y salir de la aplicación.

Ilustración 15: iconos de acceso rápido

En la parte de arriba del panel podemos encontrar dos iconos que al accionarlos generan los códigos correspondientes al modelo lógico y al código SQL respectivamente. Al accionar estos iconos, una nueva pantalla se superpone a la vista principal mostrando los documentos generados.

Ilustración 16: iconos de generación de códigos lógico y SQL

En la parte superior de la pantalla se integra la clásica barra de herramientas en la cual se despliegan diferentes opciones.

Ilustración 17: barra de herramientas

La interfaz de usuario principal integra en una sola pantalla prácticamente todas las funcionalidades básicas necesarias para desarrollar un diagrama desde cero, o para cargarlo si éste se encontrase alojado en una unidad de almacenamiento persistente.

Cada elemento que se quiere introducir al diagrama se hace de forma inequívoca, pues la propia aplicación va permitiendo o no realizar las acciones que en cada momento sean adecuadas.

Aunque si bien es cierto que prácticamente toda la actividad del programa se desencadena desde esta interfaz principal, determinadas acciones deben ir apoyadas por algún tipo de diálogo emergente y automático. Su apariencia y uso siguen la misma dinámica de simplicidad aplicada para toda la aplicación. Estos diálogos a los que nos referimos servirán por ejemplo para confirmar acciones críticas, como el borrado total del diagrama, para solicitar el nombre del fichero con el que se quiere guardar un diagrama, o para seleccionar el idioma en el cual se encuentre escrita la interfaz de usuario.

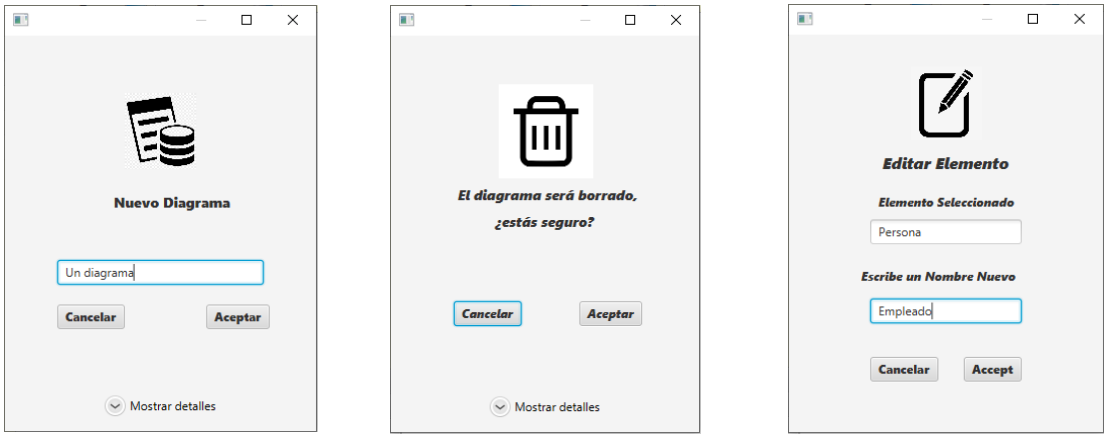


Ilustración 18: vista de los diálogos de la aplicación

Es evidente la claridad y simplicidad de las solicitudes realizadas por los diálogos de apoyo.

Es de destacar también la no necesidad de búsqueda por parte del usuario de estos diálogos, sino que es la aplicación la que en el momento oportuno lanzará su aparición para poder realizar la acción solicitada. Tras seleccionar la opción elegida por el usuario estos diálogos desaparecen automáticamente.

Por otro lado, esta vista principal genera dos nuevas sub-vistas correspondientes a los códigos generados, diseño lógico y código SQL. Esta vista de los códigos aparece automáticamente al solicitar estas acciones con los botones correspondientes.

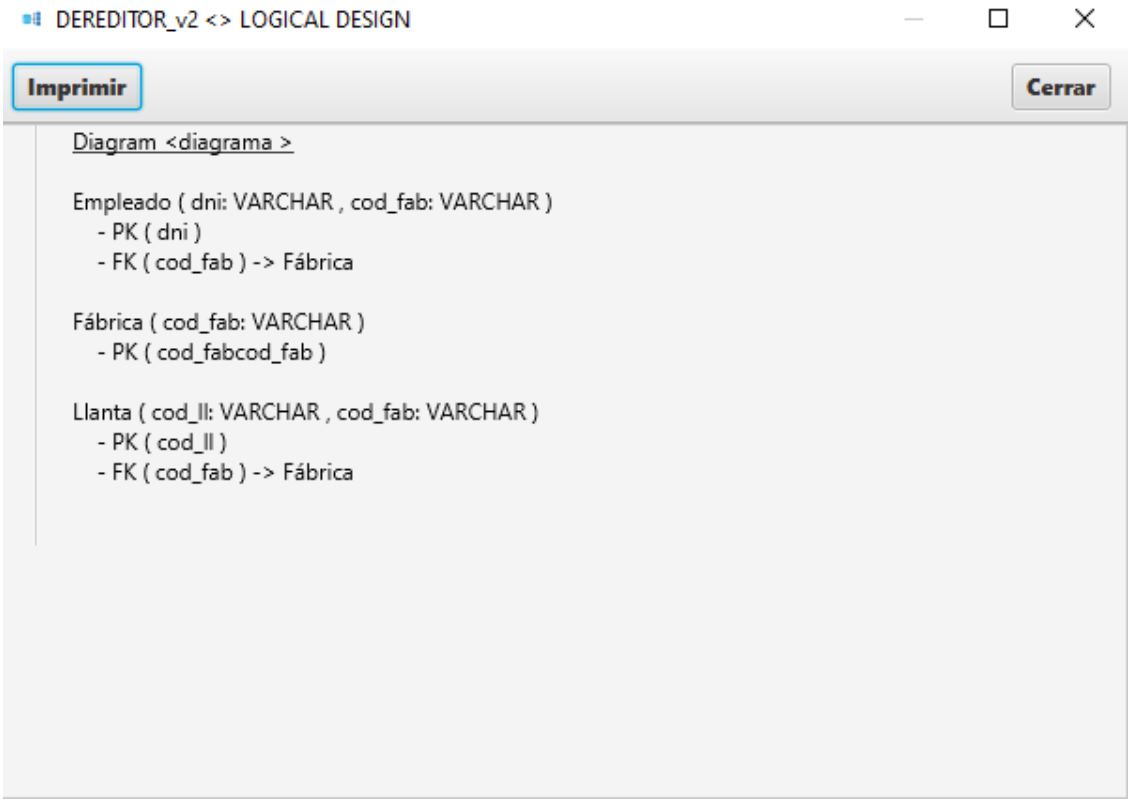


Ilustración 19: vista código lógico generado

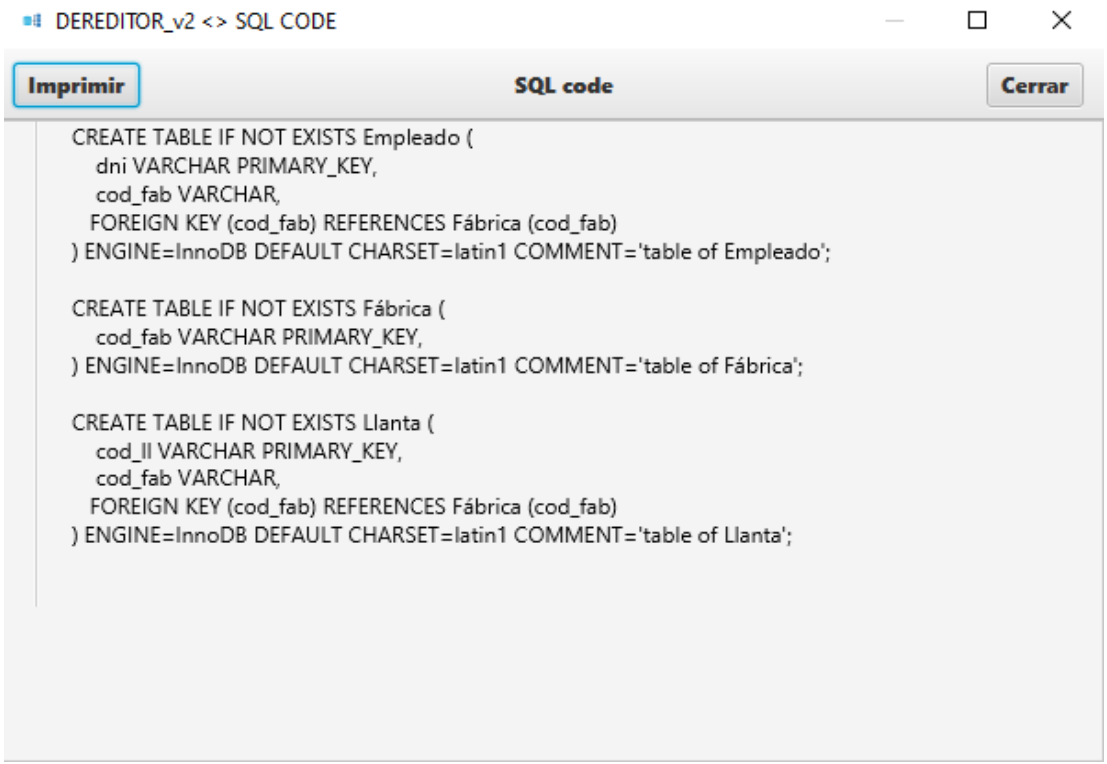


Ilustración 20: vista código SQL generado

Por último, en la parte inferior de la pantalla se encuentra un panel informativo en el cual irán apareciendo, según se vayan dando las acciones, los mensajes informativos o de advertencia que guiarán al usuario para diseñar un diagrama correctamente.

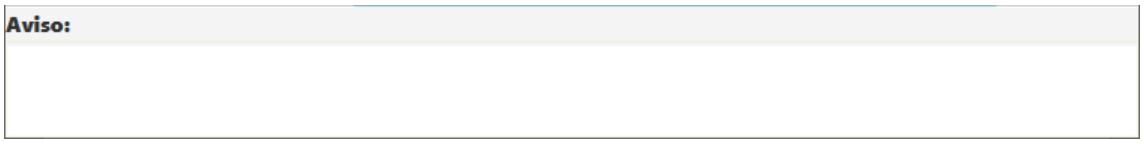


Ilustración 21: panel informativo

Cada uno de los elementos de los elementos que forman una interfaz, botones, campos de texto, etiquetas, etc…, son manipulables desde el controlador correspondiente a esa interfaz. Para ello, cuando se crea la interfaz con el programa "*Scene Builder*" se indica cuál debe ser el controlador a la que se asocia. Una vez ligados interfaz y controlador, cada elemento se referencia sin más que declarar en el controlador el tipo de elemento y su identificador precedidos de la etiqueta @FXML. De esta manera escribiendo el código adecuado se conseguirá el comportamiento deseado de cada uno de estos elementos.

Los archivos correspondientes a interfaces de usuario son definidos mediante la extensión ".fxml". Y por convenio, los nombres de los controladores deben estar formados por el nombre asignado a la interfaz que controlan añadiéndole la palabra "*Controller*" al final, por ejemplo: una vista podría ser "mainView.fxml", y el controlador correspondiente se deberá llamar "mainViewController.java".

8.2.2 Representación de los elementos

Cada uno de los elementos que va a conformar un diagrama entidad-relación, será una vista correspondiente a cada uno de los objetos que se encuentran en el paquete Modelo. De esta manera, lo que un usuario genera cuando interactúa con la aplicación son objetos del modelo, que la aplicación automáticamente se encarga de interpretar para darles una representación gráfica utilizable dentro del diagrama.

La representación gráfica de cada uno de los elementos del diagrama sigue los estándares vistos en los apuntes de la asignatura de BBDD utilizados para impartir la asignatura durante el transcurso de la carrera universitaria [5][6]:

- El objeto Entidad está representado por un rectángulo que contiene le nombre de la entidad.
- El objeto atributo es una elipse que contiene en su interior el nombre del elemento.
- La Relación se representa mediante un rombo con su nombre en el interior.
- Las Agregaciones se representan con un rombo inscrito en un rectángulo con el nombre en el interior.
- Los indicadores de herencia entre entidades son círculos que indicarán el tipo de herencia existente entre entidades.
- Las uniones entre elementos son líneas rectas que conectan cada elemento.

Así, el siguiente gráfico muestra una representación a modo de ejemplo de cada uno de los elementos descritos.

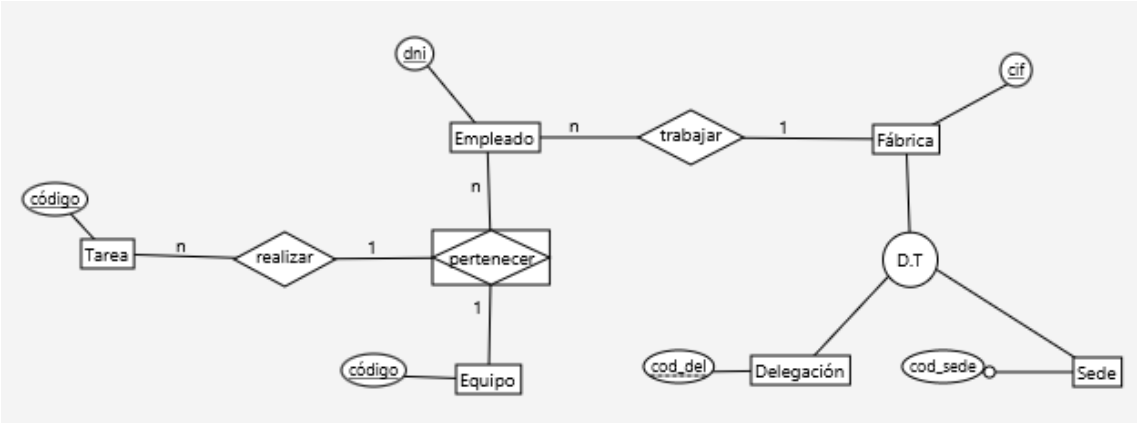


Ilustración 22: Ejemplo vista de elementos de diagrama

8.3 CONTROLADOR

En una aplicación que sigue el patrón de diseño modelo-vista-controlador, decir que una parte es más importante que otra no tiene ningún sentido. Pero al igual que se ha hablado de una interfaz de usuario principal, también hemos de hablar de un controlador principal ligado a esta interfaz.

En este controlador se encuentran referenciados todos los elementos que aparecen en la interfaz, y que nos va a permitir controlar su funcionamiento y recoger todos los eventos que estos generen para darles la respuesta adecuada.

Para que el controlador actúe sobre los elementos de la vista que controla, este controlador es indicado en el diseño de la vista, y además cada elemento es definido en él precedido de la etiqueta @FXML. Mediante esta etiqueta se pueden referenciar tanto elementos como acciones que generan estos elementos.

En la vista cada elemento lleva asociado un identificador único, el cual será utilizado en el controlador para supervisar su funcionamiento.

8.3.1 Estructura del controlador

En la tecnología que se está usando el controlador tiene algunos elementos característicos que ayudan a estructurar el código de forma ordenada.

En el caso que nos ocupa el código sigue, de forma general, la siguiente estructura:

- En primer lugar, se realizan las importaciones necesarias de paquetes y clases que se vayan a usar.
- A continuación, se declaran e identifican los elementos de la vista que van a ser controlados, tanto en su modelaje visual, como en los eventos que genera su accionamiento. Para generar el enlace entre los elementos de la vista y las declaraciones del controlador se debe usar una etiqueta @FXML en su declaración. Además, el nombre identificador debe ser el mismo en el controlador y en la vista.
- Método "***initialize***". Este método es ejecutado en primer lugar, cuando se carga el controlador. Debe usarse para inicializar aquello que requiera de esta acción. En esta aplicación este método realiza la configuración general de los elementos de la interfaz de usuario, es decir se fija un estado inicial que permita una actividad correcta al iniciar una sesión de trabajo.

- Una vez configurada la interfaz de usuario, se asignan los eventos a los que se debe reaccionar cuando se interactúa con el panel de dibujo.
- Por último, se desarrollan las respuestas a los eventos que se generan desde la vista.

La interfaz principal soporta la mayoría de los eventos que se lanzan en el desarrollo de un diagrama. Por esta razón el controlador principal que gestiona esta interfaz es una clase que contiene una gran cantidad de líneas de código. Para hacer más manejable el módulo controlador, esta clase delega funciones a otros diez sub-controladores manejados por el controlador principal. De esta manera, el flujo de datos e instrucciones entre vista y control presenta la siguiente estructura:

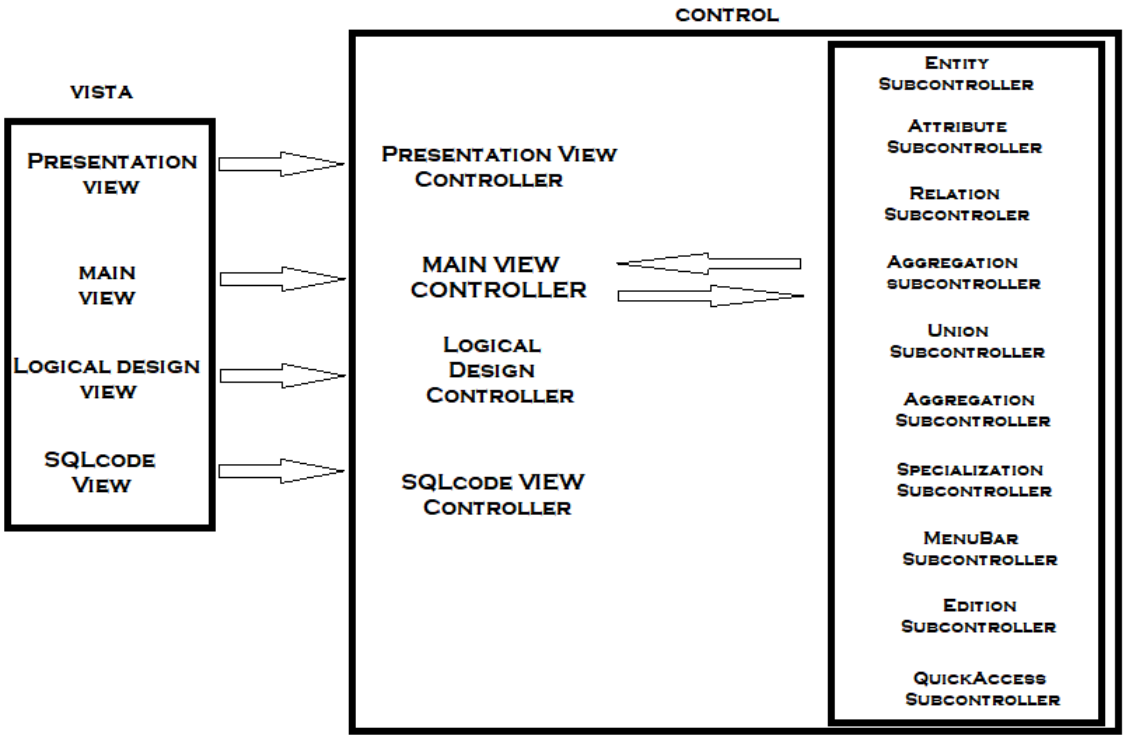


Ilustración 23:estructura flujo vista-control

Cada vista es controlada por su controlador correspondiente, que como podemos ver sigue la convención de nombres ya mencionada. En el caso de la vista principal (MainView), el controlador principal (MainViewController), delega sus acciones a controladores especializados que descargan de obligaciones a este controlador principal.

8.3.2 Método “initialize”

Como es lógico en este documento no vamos a entrar a explicar todas las clases y métodos que componen este proyecto. No obstante, sí que en determinadas ocasiones se debe hacer un inciso y pararse un momento a explicar o a dar una visión general pero puntualizada sobre alguna parte concreta de este código. Es el caso de un método muy importante que aparece automáticamente al crear un controlador en JavaFX. Estamos hablando del método **“initialize”.**

Cuando el programa carga un determinado controlador, pues se requiere su actuación para alguna tarea, el método “initialize” es ejecutado automáticamente y, en primer lugar.

Este método de inicialización debe estar codificado de tal manera que prepare al sistema para comenzar a usarse en las condiciones que la aplicación en sí misma necesite.

Cada controlador incluye su propio método de inicialización, el cual en algunos casos no es necesario utilizar. Esto es debido a que es posible que en el momento de su lanzamiento no sea necesaria una inicialización específica del sistema.

En el caso que nos ocupa nos ceñiremos al método de inicialización del controlador principal. Recordemos que el controlador principal controla el comportamiento de la interfaz principal de usuario. También se ha puntualizado que el control principal se encuentra delegado en parte a otros sub-controladores que aligeran de trabajo al módulo principal.

La pantalla principal de usuario no presenta siempre el mismo estado, y en determinados momentos algunas funcionalidades no se encuentran disponibles o están deshabilitadas. Pues bien, en el momento inicial en el que arranca la aplicación es necesario situar esta interfaz principal en un estado convenientemente configurado, para que el usuario pueda comenzar a trabajar adecuadamente.

Así pues, el método “**initialize**” de nuestro controlador principal va a hacer lo siguiente:

- En primer lugar, se instancian los sub-controladores que apoyarán las tareas de la interfaz principal.
- A continuación, se establece un estado inicial de activación o desactivación de cada una de las partes que conforman la interfaz gráfica, esto es botones, etiquetas, iconos, etc…
- Por último, se añaden eventos de ratón a elementos como el diagrama o se inicializan algunas variables.

Una vez que el controlador ha sido cargado y el método “inicialize” ejecutado, no se volverán a inicializar mediante este método las características del sistema.

8.3.3 PrinterFX

La herramienta que este proyecto implementa, básicamente sirve para generar tres documentos distintos, obviamente con su lógica correspondiente, pero finalmente tras finalizar una sesión de trabajo, si el proceso de desarrollo ha sido satisfactorio, el usuario terminará obteniendo un diagrama entidad-relación, un documento con el diseño lógico del diagrama y finalmente otro documento que contendrá el código SQL [6] necesario para implementar la base de datos representada por el diagrama.

Parece lógico y necesario que estos tres documentos que genera la aplicación se puedan exportar o guardar de forma persistente a algún tipo de formato de fichero. De esta tarea se encargan tres controladores distintos, el controlador principal para manipular el diagrama

entidad-relación, el controlador de diseño lógico para manipular el documento de del diseño lógico y finalmente el controlador de código SQL para manipular el documento de código SQL.

El diagrama entidad-relación se ha decidido que, como gráfico o imagen que es, es muy adecuado que sea almacenado en un formato de archivo ".jpg".

Por otro lado, tanto el documento del modelo lógico como el de código SQL, se ha decidido que exista la posibilidad de convertir estos documentos en archivos "pdf".

Para que los controladores puedan dar respuesta a la petición de guardado de los documentos descritos, se ha implementado una clase denominada "PrinterFX", la cual tras ser instanciada permite que utilizando los métodos implementados se puedan realizar las exportaciones solicitadas.

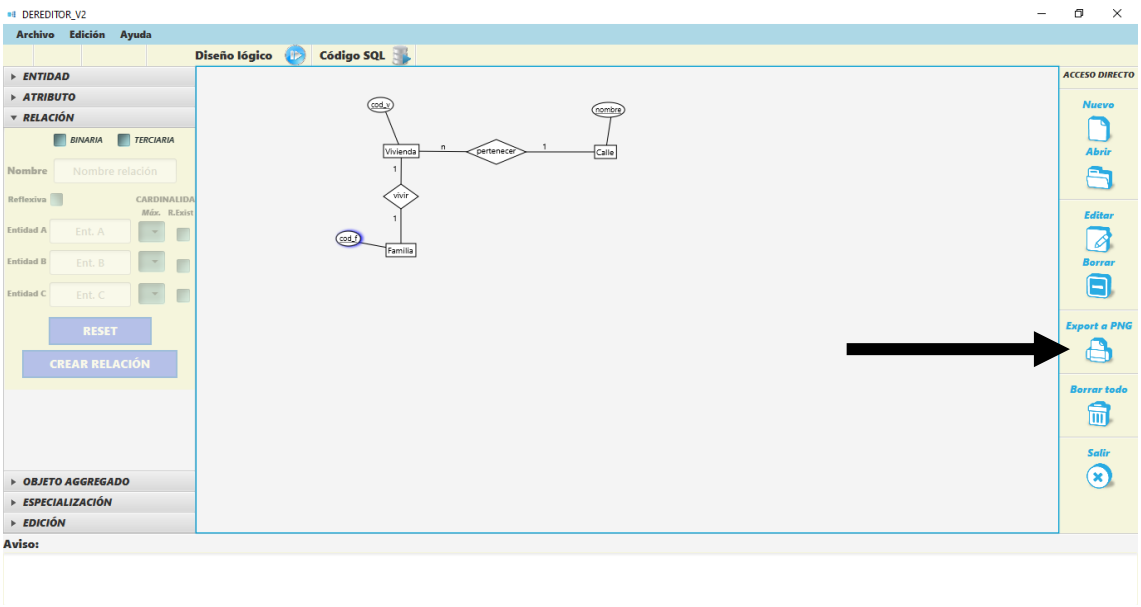


Ilustración 24:vista de diagrama para exportar a .jpg

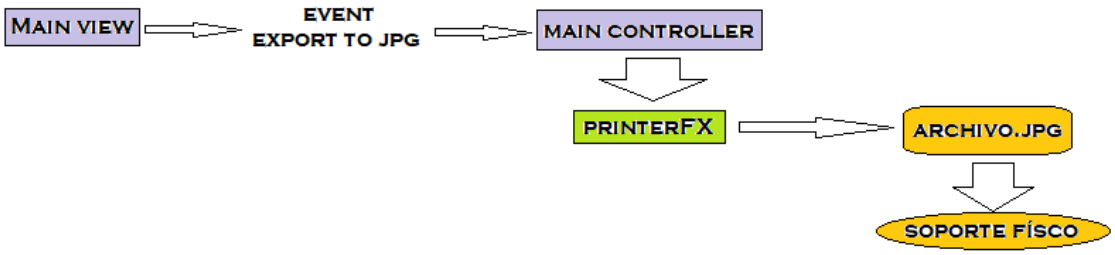


Ilustración 25: flujo de instrucciones en evento de exportación a .jpg

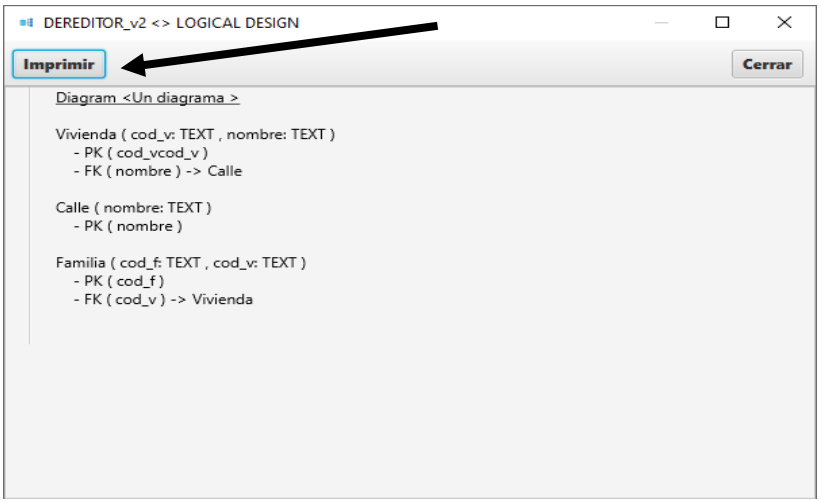


Ilustración 26:diseño lógico preparado para imprimir como .pdf

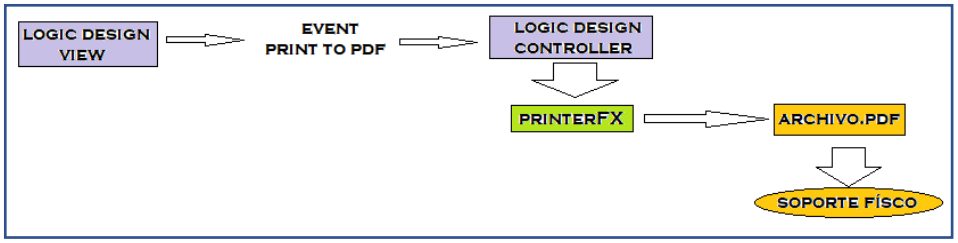


Ilustración 27: flujo de instrucciones de evento de impresión de diseño lógico a.pdf

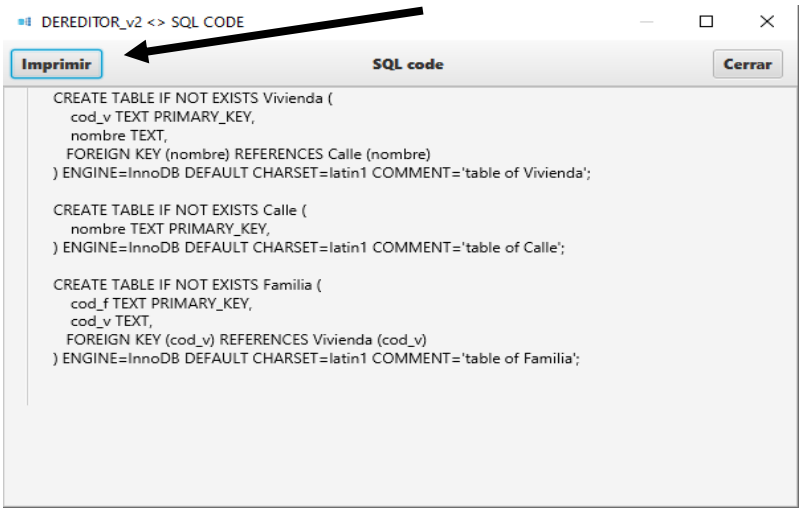


Ilustración 28: código SQL preparado para imprimir como .pdf

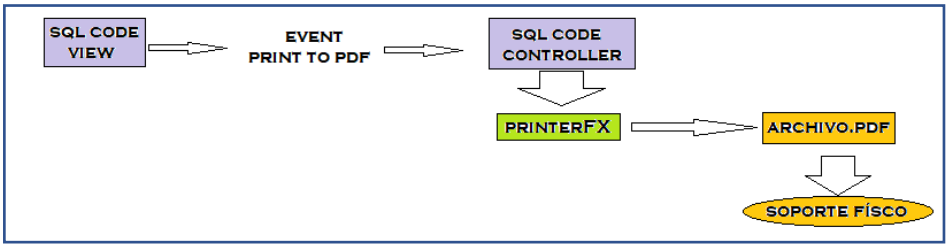


Ilustración 29: flujo de instrucciones de evento de impresión de código SQL a.pdf

8.4 MODELO

El modelo de esta aplicación contiene principalmente las clases que abstraen cada uno de los elementos que conforman los diagramas entidad relación. También dentro del paquete modelo encontraremos algunas clases que ayudarán a estructurar los datos o la información que debemos guardar.

Así pues, dentro del paquete modelo se encontrarán las clases Entidad, Relación, Atributo, Especialización, Agregación o Unión, principalmente. Todas ellas son clases hijas de una Superclase que es denominada Elemento. El uso de la herencia propia de la programación orientada a objetos, permite hacer una manipulación general de estos objetos, dándole a todos ellos tratamiento de Elemento.

Podemos entender que la creación de un diagrama entidad-relación consistirá en la instanciación de diferentes objetos de tipo Elemento, que serán ordenados en un contenedor de Elementos, y cuya representación gráfica se verá reflejada en el panel de trabajo dedicado a tal uso.

Si un usuario se dispone a crear una "Entidad", el sistema se encargará de instanciar un objeto de la clase Entidad la cual hereda las características que tiene la superclase Elemento. Esta estructura de datos, es decir la clase Entidad, subclase de la clase Elemento, es la que contiene los datos que posteriormente se almacenarán de forma persistente como archivos ". der".

Cuando se crean los objetos que modelan los elementos que forman un diagrama entidad-relación, automáticamente y de forma transparente para el usuario el sistema se encarga de generar su representación visual.

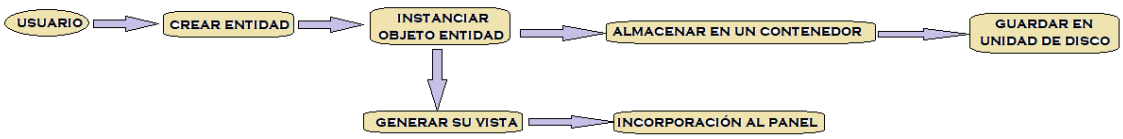


Ilustración 30: estructuración de los datos al crear un objeto de panel

8.4.1 Clases correspondientes al paquete modelo

Cada uno de los elementos que encontramos en un diagrama entidad-relación, tiene de forma interna una estructura de datos que lo modeliza, en este caso su objeto correspondiente. Entonces dentro del paquete modelo encontraremos las siguientes clases:

- Elemento: se podría decir que es la clase principal. Es la superclase que las demás clases van a extender. De esta manera se podrá generalizar el tratamiento de la colección de elementos, particularizando las acciones realizadas sobre los objetos según sean de una subclase u otra.
- Entidad: clase que modeliza una entidad. Entiéndase ésta como el concepto estudiado en el ámbito de las bases de datos.
- Atributo: clase que modeliza un atributo. Será un atributo cada una de las cualidades que caracterizan a las entidades.
- Relación: clase que modeliza una relación. Representación de la forma de relacionarse que tienen los elementos del diagrama.

- Agregación: clase que modeliza los objetos agregados. Un objeto agregado representa la simbiosis entre dos entidades que se relacionan, para formar un nuevo objeto resultante que se relacione con terceros elementos diferentes.
- Especialización: clase que modeliza la especialización entre entidades. Las especializaciones serán de cuatro tipos, parcial-disjunta, parcial-solapada, total-disjunta y total-solapada. La especialización implementa la herencia en las bases de datos desarrolladas.
- Unión: esta clase modeliza la unión existente entre dos elementos del diagrama. Estas uniones podrán ser fuertes, de una línea, o débiles, de línea doble.

9. PRINCIPIOS DE USABILIDAD

Para testear el nivel de usabilidad de la interfaz de nuestra aplicación, vamos a hacer uso de los conocidos “Diez principios de usabilidad de Nielsen” [7], valorando el cumplimiento de cada uno de ellos.

1. Visibilidad del estado del sistema: en este punto destacaremos que la aplicación está diseñada para que toda la actividad se desarrolle en una única interfaz principal, apoyada si se da el caso por reducidos cuadros de diálogo que indican claramente para qué han sido desplegados. Consideraremos por tanto que en este sentido el usuario no tendrá dudas de lo que está haciendo en cada momento.
2. Relación entre el sistema y el mundo real: basándonos en todo momento en el principio aplicado para el desarrollo de la interfaz, es decir máxima sencillez en todas las acciones, tanto el lenguaje utilizado como el momento en que se utiliza es perfectamente entendible y cercano al usuario.
3. Control y libertad del usuario: todas las acciones que conlleven un arrepentimiento por parte del usuario tienen la posibilidad de ser canceladas previamente a su aceptación, haciendo que la pantalla vuelva a su imagen anterior.
4. Consistencia y estándares: todas las opciones que el usuario puede elegir vienen indicadas, bien en lenguaje natural o con un icono estandarizado. La estructura general de la interfaz sigue una disposición similar a las herramientas gráficas afianzadas en el mercado.
5. Prevención de errores: siguiendo el carácter didáctico de la aplicación, ésta en las ocasiones pertinentes desactiva determinadas acciones para evitar errores en su manejo.
6. Reconocimiento antes que recuerdo: la aplicación está provista de un panel informativo que advierte al usuario sobre el proceder de algunas de las acciones que pretenda.
7. Flexibilidad y eficiencia de uso: en este caso las acciones se realizan utilizando los botones habilitados para tal uso. No se han implementado por ejemplo tajos de teclado.
8. Estética y diseño minimalista: sencillez, colores básicos, claridad e información justa y adecuada, hacen a la interfaz amistosa y nada ambigua.
9. Ayudar a los usuarios a reconocer, diagnosticar y recuperarse de errores: el panel informativo de ayuda lanza en momento los avisos o guías para que un usuario pueda modificar su acción en el caso de ser errónea.
10. Ayuda y documentación: es este caso no se ha incorporado una ayuda específica consultable, salvo los ya mencionados mensajes guía que aparecen cuando alguna acción no es totalmente correcta.

A la vista de los resultados explicados punto por punto, se podría decir que la aplicación tiene una interfaz con un nivel de usabilidad alto, pues de los diez puntos considerados por Nielsen ocho se cumplen perfectamente.

10. PRINCIPALES ESCOLLOS Y PROBLEMAS

Es inevitable que en un trabajo con cierta magnitud aparezcan determinados problemas o situaciones que se hacen más complicados de resolver.

Uno de los problemas que había que resolver respecto de DEReditor era que conseguir que los archivos guardados fuesen posteriormente recuperados manteniendo la integridad total del diagrama. En este punto la dificultad surgió porque la recuperación de los diagramas en principio era correcta, pero tras manipular los elementos en el diagrama estos tomaban posiciones erróneas que daban al traste con la configuración correcta. Tras horas y horas de estudio, pruebas y trabajo, se llegó a la solución de incorporar a cada elemento dos variables, “x“ e “y”, que guardasen su posición instantánea en cada momento. Una vez cargado el diagrama desde fichero se hace una lectura de estas variables para situar correctamente cada elemento.

Por otro lado, el tener una interfaz principal que lanza la gran mayoría de los eventos conllevó un crecimiento en demasía del controlador principal de esta interfaz. Para aligerar este controlador, tanto en líneas de código como en acciones a realizar, se optó por instanciar controladores no ligados a ninguna interfaz en particular, sino que eran “súbditos” del propio controlador principal y resolvían la acciones que se les requiere. Estos sub-controladores se encargan de determinados eventos según el bloque al que están designados. Por ejemplo, encontraremos un sub-controlador para la gestión de las entidades, otro para las relaciones, etc…

Se le puede llamar escollo, problema, o simplemente poca agilidad con el idioma. Al comenzar el proyecto se decidió escribir el código en inglés. Según voces expertas la codificación es más profesional si se realiza en inglés. Por esto nos lanzamos a ello, lo cual ha restado algo de agilidad en el desarrollo, pues se debía consultar con relativa frecuencia el correspondiente traductor. No obstante, según avanzó el proyecto estos problemas de traducción se fueron solventando por razones obvias.

11. TECNOLOGÍAS UTILIZADAS

En este apartado vamos a describir de forma general las tecnologías utilizadas para el desarrollo del proyecto.

Programación: Java JDK 1.8 [8]

Lenguaje de programación interpretado orientado a objetos con una muy alta estandarización a escala global. Es un lenguaje muy extendido en la comunidad de programación y actualmente ocupa la tercera posición de los lenguajes más usado por detrás de C y Python.

Plataforma JavaFX

JavaFX es una plataforma que amplía las prestaciones de las aplicaciones creadas en Java, pues se encuentra preparada para generar presentaciones y experiencias visuales muy atractivas. Esto se consigue dando la capacidad a los desarrolladores de integrar gráficos vectoriales, animación, sonido y activos web de video en aplicaciones cuyo resultado final es muy llamativo y sofisticado.

Se encuentra disponible para Windows, Linux y Mac OS X, y su descarga es gratuita.

Entorno de programación: Apache NetBeans IDE 11.1

NetBeans es un entorno de desarrollo integrado para todo tipo de aplicaciones Java (Java SE (incluido JavaFX), Java ME, web, EJB y aplicaciones móviles.

Aparte de Java también tiene extensiones para desarrollar C, PHP, C++, HTML5 y JavaScript.

Proporciona un entorno de trabajo cómodo, con un editor de código avanzado, asistencia en la compilación y ejecución.

Las prestaciones de este entorno pasan por ejemplo en hacer el resaltado de la sintaxis, marcar errores, tabulación automática, concreción automática de nombres de funciones, etc.

Asistente visual: Scene Builder

Scene Builder es un generador de escenas para JavaFX, que permite a los desarrolladores crear con cierta facilidad interfaces gráficas sin necesidad de codificar. Su filosofía de trabajo es la de arrastrar los elementos dentro de la interfaz para ir configurando el diseño de ésta.

Los diseños realizados con Scene Builder se combinan con hojas de estilo para darles la apariencia deseada, y con el código Java necesario para dotar de funcionalidad a cada uno de los elementos que conforman la escena.

Scene Builder se puede usar con cualquier IDE de Java, pero está especialmente diseñado para combinarse con NetBeans.

12. POSIBLES MEJORAS Y EXPECTATIVAS FUTURAS

En el párrafo de conclusiones se explicará con más detalle, pero DEReditor_v2 ha pasado de ser una actualización de un programa anterior, a un programa nuevo con todo lo que ello conlleva.

Por esta razón, un programa nuevo tiene todas las posibilidades de ser ampliado y mejorado en el futuro. Recordemos que esta aplicación fue concebida pensando en que se convirtiese en una herramienta didáctica, cuya máxima fuera la sencillez en su uso. Creo, y así debería ser, que si en un futuro la aplicación es ampliada o mejorada no se debería perder el carácter de sencillez que se le ha querido imprimir.

Quizás si en algún momento se pretendiese mejorar esta aplicación se podría intentar conseguir un carácter editable de los elementos más detallista.

Como ya se expuso en el test de usabilidad, se podrían incluir atajos y procedimientos al programa para conseguir una mayor agilidad en los desarrollos.

Igualmente, es posible que una guía consultable sobre el uso del programa o sobre el desarrollo de diagramas entidad-relación, sería otro factor de mejora importante.

Dentro de la propia funcionalidad de la herramienta, quizás se podría ir un paso más allá, de tal manera que no sólo se quedase en generar un paquete de instrucciones SQL, sino que directamente se crease la base de datos diseñada utilizando algún SGBD.

En cuanto a las expectativas que se esperan de esta aplicación, diremos que la idea que acompañó desde un principio al desarrollo de DEReditor_v2, es que fuese una herramienta que tuviese el nivel suficiente para poder ser utilizada en la Escuela como parte de las asignaturas de BBDD. Durante estos últimos años ha sido DEReditor el apoyo que ha tenido las asignaturas mencionadas.

Cuando se propuso el desarrollo de DEReditor_v2, se planteó un objetivo de mejora de aquellos aspectos en los fallaba la versión anterior. Por tanto, si los requisitos de mejora se han cumplido, tenemos la expectativa de que finalmente DEReditor_v2 sea una herramienta útil en la Escuela pueda ser utilizada en la formación de los nuevos alumnos. No obstante, será la experiencia y si su uso los que determinen si finalmente se ha conseguido una herramienta adecuada para esta función.

13. CONCLUSIÓN

Datos, datos y más datos. Estamos en un mundo en el cada día hay más y más datos. Se crean, se destruyen, se envían se guardan. Son miles de millones los que cada día es necesario almacenar y gestionar. Estructuras como las bases de datos se hacen imprescindibles en esta labor. Toda la cantidad de información que el mundo genera debe estar ordenada y clasificada, si no se haría inmanejable, cada día más.

Computadoras, grandes o pequeñas, con más o menos capacidades, se convierten en los grandes aliados del hombre para dar un manejo eficiente de esos datos. La automatización de procesos, de almacenaje o de búsqueda, es posible gracias a la ordenación que los ordenadores consiguen utilizando las bases de datos.

Así, se hace imprescindible que todo aquel que adquiera conocimientos o realice estudios sobre informática, deba de alguna manera obtener unas nociones básicas sobre bases de datos. En Ingería Informática dos son las asignaturas dedicadas a tal fin y una tercera a Almacenes y Minería de Datos.

Por esta razón es justificable que una herramienta como DEReditor_v2 se encuentre disponible tanto para el profesorado como para el alumnado.

Y podríamos pensar que ya existen otras herramientas suficientemente buenas y avanzadas para desempeñar las funciones que se requieren en el aprendizaje del desarrollo de las bases de datos, pero hemos de tener en cuenta que, habiéndolas, ninguna tendrá la simplicidad con la que se ha diseñado DEReditor_v2.

En los procesos de aprendizaje un alumno debe centrarse en aprender la materia que se le está enseñando para su formación. Si se está aprendiendo bases de datos el alumno debe invertir su tiempo en aprender bases de datos.

DEReditor_v2 ha intentado ser desde el primer momento en que fue imaginada, ser una herramienta de ayuda, simple y amistosa, sin grandes alardes, pero que cuando fuese utilizada por cualquier usuario proporcionase resultados satisfactorios, sin la necesidad de mucho esfuerzo por entender la propia herramienta.

El ser una aplicación creada por un alumno y pensada y dedicada para los alumnos, da un valor especial al resultado, pues con ello se ha conseguido lo que se planteó el primer día, que cualquiera con unos mínimos conocimientos sobre bases de datos, es capaz de realizar con muy poquito esfuerzo un diagrama entidad-relación correspondiente a una base de datos y obtener sus códigos posteriores.

El proceso de desarrollo ha sido diferente al que en su día se pensó. En principio este proyecto se planteó como una adaptación o una actualización del proyecto DEReditor del año 2005. La conversación entre la directora del proyecto y el desarrollador fue inicialmente encaminada a que la realización de DEReditor_v2 consistiría en realizar una reestructuración de las clases que conformaban el proyecto DEReditor, a fin de conseguir una configuración que cumpliese con el patrón modelo-vista-controlador, y se subsanasen los problemas de funcionamiento encontrados en la versión anterior. A partir de ahí, habría que invertir el tiempo en crear una vista atractiva, usable y de vanguardia, que limpiase la cara de una interfaz que se veía pasada en el tiempo y poco atractiva para el usuario. Pues bien, nada más lejos de la realidad. Una vez elegidas las herramientas de trabajo con las que íbamos a llevar a cabo el desarrollo de la

aplicación y, tras hacer un estudio del proyecto anterior, se llegó a la conclusión de que DEReditor_v2 iba a ser un proyecto totalmente nuevo, que cumpliría con todo lo planteado en la propuesta oficial, pero que finalmente no ha reutilizado ni una sola línea de código de la primera versión.

Es por esto que en realidad DEReditor_v2 es una primera versión de una aplicación, que deja la puerta abierta a una segunda versión mejorada para proyectos futuros.

14. LICENCIA SOFTWARE Y LICENCIA DOCUMENTAL

El software al cual se refiere este documento se encuentra protegido bajo licencia GPL, anexada dentro del proyecto como archivo “License”.

El código fuente al cual se refiere esta documentación y el archivo ejecutable de la aplicación se encontrarán disponibles en el repositorio GitLab de la universidad.

15. BIBLIOGRAFÍA

- [1] Guía para la Especificación de Requisitos según el estándar de IEEE 830. IEEE Std. 830-1998, 22 de octubre de 2008
- [2] Tokio, D., formativos, P., informáticas, R., datos, A. y Net, T., 2021. *¿Qué es JavaFX y para qué se utiliza? - Escuela Tokio*. [en línea] Tokio School. Disponible en: <https://www.tokioschool.com/noticias/que-es-javafx-usos/> [Consultado el 9 de septiembre de 2021].
- [3] Docs.oracle.com. 2021. *Uso de Scene Builder JavaFX con IDE de Java: Uso de Scene Builder con IDE de NetBeans | Tutoriales y documentación de JavaFX 2*. [en línea] Disponible en: <https://docs.oracle.com/javafx/scenebuilder/1/use_java_ides/sb-with-nb.htm> [Consultado el 9 de septiembre de 2021]
- [4] Balsamiq.com. 2021. *Balsamiq. Rapid, Effective and Fun Wireframing Software | Balsamiq*. [online] Available at: <https://balsamiq.com/> [Accessed 10 September 2021].
- [5] Garrido, P., *Bases de datos II: Diseño lógico*.
- [6] Garrido, P., *Bases de datos II: Diseño conceptual*
- [7] Medium. 2021. *Los 10 mandamientos (principios heurísticos) de Jakob Nielsen*. [online] Available at: <https://eugeniacasabona.medium.com/los-10-mandamientos-principios-heur%C3%ADsticos-de-jakob-nielsen-ed4ad54468a9> [Accessed 10 September 2021].
- [8]Docs.oracle.com. 2021. *Java Platform SE 7*. [en línea] Disponible en: <https://docs.oracle.com/javase/7/docs/api/> [Consultado el 9 de septiembre de 2021].