Trabajo de fin de grado del

Grado en Ingeniería del Software

Curso 2019/2020

Facultad de Informática

Universidad Complutense de Madrid

DBCASE Web

"Una herramienta para el diseño de Bases de Datos."

"A tool for the design of Databases."

Autores *Luis Eduardo Paucar Cerrón*

Yrving David Conde Cubas

Profesor director *Fernando Sáenz Pérez*

Agradecimientos.

A nuestro director de proyecto, Fernando Sáenz Pérez por su confianza desde el inicio del desarrollo de este proyecto, que en todo momento estuvo disponible para ayudarnos y aconsejarnos en todo lo relacionado a esta aplicación, incluso con todas las dificultades que han surgido en este curso académico tan difícil. Sin duda sin su ayuda esto hubiera sido más difícil de poder conseguir.

A todos los profesores con los que hemos contado a lo largo de estos años en la universidad, sin los conocimientos adquiridos gracias a ellos el desarrollo de este proyecto no hubiera sido realizado con éxito.

Resumen

DBCASE Web es una nueva versión del proyecto DBCASE 2.0 el cual ofrece una manera fácil e intuitiva de diseñar diagramas entidad-relación y poder construir bases de datos relacionales. La finalidad principal de este proyecto fue convertir la aplicación de escritorio a una aplicación web compatible con cualquier dispositivo cuyo único requisito para su acceso sea un navegador web, evitando así problemas de versiones no compatibles o dependencia de software específico para su ejecución.

DBCASE Web permite a un usuario autenticado con su cuenta de la UCM realizar todo el proceso, desde el diseño de una base de datos relacional pasando por todas sus fases, lo cual permitirá que puedan experimentar, adquirir y asentar conocimientos sobre las Bases de Datos.

La aplicación ha sido dotada de más abstracciones de asociación que permiten generar diagramas más complejos y robustos los cuales son convertidos a un modelo lógico y modelo físico a partir del diagrama diseñado por el usuario, así como también ejecutar directamente el código generado en un SGBD (Sistema de Gestión de Bases de Datos).

Palabras Clave

Bases de datos | Java | Modelo Entidad Relación | Modelo Relacional | SQL | JavaScript | Spring Boot | Maven | JQuery | Bootstrap

Abstract

DBCASE Web is a new version of the project DBCASE 2.0, which offers an easy and intuitive way to design entity-relationship diagrams and to build relational databases.

The main purpose of this project was to convert the desktop application to a web application compatible with any device whose only requirement for access is a web browser, this way the user avoids any problems of incompatible versions or dependency on specific software for correct performance.

DBCASE Web allows the user to authenticate with his UCM account and implement the entire process, from the design of a relational database till completing all phases, the user will be able to experiment, acquire and establish knowledge about databases.

The application has been equipped with more association abstractions that allow generating more complex and robust diagrams which are converted to a logical model and physical model from the diagram designed by the user, as well as directly executing the generated code in a DBMS (Database Management System).

Keywords

Database | Java | Entity–Relationship Model | Relational Model | SQL | JavaScript | Spring Boot | Maven | JQuery | Bootstrap

1. Introducción

DBCASE Web continúa la implementación del proyecto DBCASE cuyo desarrollo empezó en el año 2007 implementando nuevas funcionalidades y nuevos cambios en su interfaz de usuario con el objetivo de facilitar el uso y acceso a la aplicación.

La aplicación ha sido desarrollada para poder dar una herramienta a los alumnos de la facultad que les permita entender mejor el funcionamiento de las bases de datos relacionales, sin la necesidad de que se conozca ningún lenguaje de programación, útil para usuarios que estén empezando con la materia.

Pero también útil para usuarios con amplios conocimientos que le permita construir una base de datos relacional de manera fácil y rápida.

La aplicación desarrollada en lenguaje de programación Java cuenta para esta nueva versión con una estructura basada en el framework Spring Boot el cual permite entre otras cosas poder obtener aplicaciones web con configuraciones nada complejas. También se ha añadido el gestor de repositorios Maven el cual permite poder controlar mejor las dependencias existentes en el proyecto así como añadir fácilmente las nuevas librerías que se han añadido.

1.1. Antecedentes

Como ya contamos, la aplicación DBCASE Web precede a varias versiones realizadas por compañeros en años pasados. Los detallamos a continuación.

(2007 - 2008) DBDT de sistemas informáticos

Autores:

- Alberto Milán Gutierrez
- Miguel Martinez Segura
- Francisco Javier Cáceres González
- Yolanda Gárcia Ruiz (Tutor)

(2008 - 2009) BDCASE de sistemas informáticos

Autores:

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

(2018-2019) DBCASE 2.0 (2018/2019)

Autores:

- Miguel Arriba García
- Fernando Sáenz Pérez (Tutor)

La última versión realizada en el año anterior contaba con varias mejoras respecto a las versiones anteriores sin embargo seguía siendo una aplicación de escritorio, con todas las ventajas y desventajas que ello implicaba.

1.2. Objetivos

Buscamos con la nueva actualización del proyecto DBCASE, añadir la capacidad de ser multiplataforma lo que permitirá desde cualquier navegador el acceso a la aplicación, evitando de esta manera la instalación de una aplicación externa y condicionando al usuario de un sistema operativo concreto.

Teniendo en cuenta que nuestro proyecto se extiende de la versión DBCASE 2.0 realizada como una aplicación de escritorio, nuestra prioridad fue maximizar la reutilización del código respecto de los proyectos predecesores.

Uno de los puntos más importantes en esta actualización consiste en añadir las agregaciones lo que permite ampliar las limitaciones del usuario en el entorno de trabajo.

1.3. Planificación del proyecto

Este proyecto se inició a mediados de Noviembre de 2019, tras la realización de reuniones entre compañeros y profesor para planificar el inicio del proyecto y decidir las herramientas que utilizariamos para el desarrollo.

Antes de comenzar con el desarrollo de las tareas discutimos la tecnología a utilizar, ya que teníamos total libertad para la elección. Teníamos diferencias respecto al lenguaje a utilizar ya que aunque el equipo había utilizado Spring en proyectos anteriores, no éramos expertos en ello. Tras diversos debates se redujeron las opciones a dos: PHP o Java.

PHP: El equipo tenía más experiencia realizando proyectos en este lenguaje, aunque la utilización de PHP, tenía una gran desventaja la poca reutilizacion de codigo respecto a utilizar Java.

Java: Elegimos este lenguaje ya que podíamos conseguir la reutilización casi total del núcleo de la aplicación. A la conclusión que llegamos fue que podíamos crear un nuevo controlador que consiga interactuar entre los servicios del proyecto predecesor y la nueva interfaz web.

Creamos un listado de historias de usuario sobre todo lo que debía tener la nueva aplicación, la cual íbamos modificando en función de los comentarios y correcciones del profesor.

Cada historia de usuario fue dividida en tareas pequeñas de tal forma que cada usuario pueda hacer cualquier tarea de forma relativamente rápida y fácil.

Para esto utilizamos la herramienta Trello, la cual nos permite planificar proyectos sabiendo qué hace cada uno de nosotros y darle prioridad a alguna tarea necesaria para desarrollar otras. Las tareas más prioritarias estaban puestas arriba del todo el cada listado.

DBCASE Web

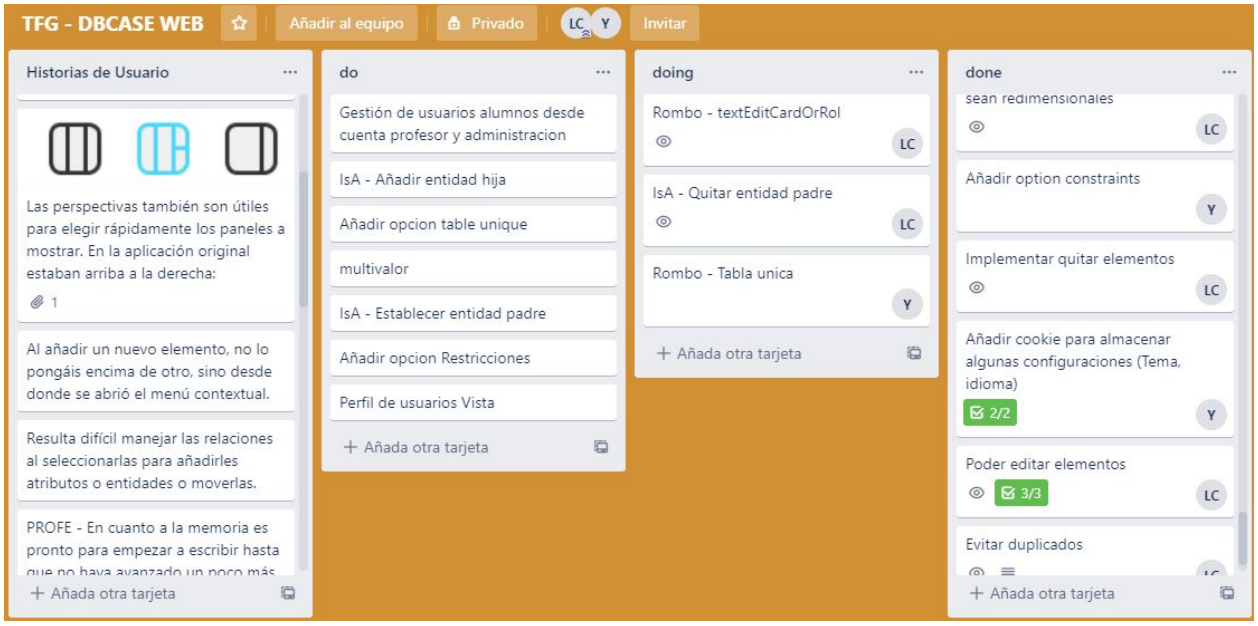

Figura 1.1 Lista con tareas que se ha creado en panel de Trello.

En Trello creamos 4 listados:

- Historias de usuarios: Añadimos todas las funcionalidades que debe tener la aplicación y también son correcciones directas por parte del profesor.
- Do: Añadimos las tareas una vez divididas las historias de usuario, estas no tienen asignación pero sí que estaban ordenadas en función de su prioridad.
- Doing: Son las tareas que se están haciendo e indica la persona que la está realizando.

Todas esta planificación se ha mantenido durante todo el desarrollo del proyecto, el ritmo de avance de cada tarea iba en función del tiempo que dedicamos a ello.

1.4 Plan de trabajo

En esta sección se explican las fases que han tenido lugar durante todo el proyecto que se ha desarrollado.

1. Investigación y estudio: Esta primera fase se llevó a cabo al inicio del proyecto y se realizó sobre el trabajo que íbamos a realizar: Investigación sobre frameworks disponibles. La investigación inicial se basó en la posibilidad de utilizar un framework web que nos facilitara el trabajo a la hora de desarrollar la aplicación. Una vez encontrado un framework y librerías que creímos eran útiles para el proyecto se estudió en profundidad el código ya creado anteriormente, el uso de las nuevas librerías y también del framework a utilizar.

2. Desarrollo: La segunda fase, la de desarrollo de la aplicación, se comenzó una vez que la fase de investigación estuvo suficientemente avanzada sabiendo que la investigación sería constante durante todo el proyecto. En este momento se inició la implementación de la aplicación: La cual en comunicación constante entre profesor y miembros del equipo se pudo desarrollar con las herramientas previstas con anterioridad.

3. Pruebas: En esta tercera fase, por el tipo de proyecto que se ha realizado podríamos decir que tanto la fase de desarrollo y pruebas se realizaron de forma paralela.

Cada módulo creado se probaba constantemente, tanto por cada miembro como por el profesor enviando revisiones cada cierto tiempo. Se desarrolló, codificó, probó y rediseño la aplicación en numerosas ocasiones hasta lograr el diseño definitivo.

4. Resultados: En la última fase se procedió al análisis de todos los resultados obtenidos durante las pruebas realizadas a lo largo del proyecto y a la extracción de conclusiones a partir de dichos resultados. Según los resultados obtenidos, se volvieron a hacer cambios en el código de la fase anterior y a realizar nuevas pruebas que pudiesen mejorar los resultados y asegurar su corrección.

2. Tecnologías

Como ya hemos especificado, uno de los objetivo principales de este proyecto ha sido añadir a la funcionalidad existente una interfaz web que nos permita evitar problemas de incompatibilidades de versiones en Java y poder acceder a la aplicación a través de un navegador web.

Después de investigar las distintas formas de realizar esta tarea, hemos optado por utilizar algunas librerías específicas que nos permiten realizar distintas acciones como la creación de los elementos del modelo entidad relación, poder crear ficheros PDF, añadir los menús emergentes o para el diseño de la web, entre otros.

2.1 Aplicación web

Las aplicaciones web nos permiten acceder a diferentes tipos de recursos de Internet utilizando un navegador bien sea a través de internet (lo habitual) o bien a través de una red local. A través del navegador se puede acceder a toda la funcionalidad y tener cualquier tipo de solución como:

- Aplicaciones para gestiones internas y completa de una entidad (facturación, stock, clientes, usuarios, socios, contabilidad, fichar, gestión de personal…),
- Herramientas de trabajo diversas para entidades (intranets, gestión documental, trabajo en red, herramientas compartidas accesibles por múltiples usuarios, accesos diferenciados…)
- Servicios a usuarios (gestión de incidencias, accesos a contenidos diferenciados por permisos, gestión de espacios…)
- Herramientas de comunicación digital (mailings, boletines digitales, comunicaciones personalizadas a clientes o usuarios…)
- Herramientas de control de calidad de la empresa.
- Herramientas web (tiendas virtuales personalizadas, web complejas con multitud de elementos y accesos, repositorios y buscadores, gestión de ventas online…)
- Otros tipos de servicios (cualquier tipo de aplicación gestión de inmuebles, comunidades de propietarios, turismo, mapas, formación, colegios, tiendas… y otra muchas que están por crear)

Todo esto tiene grandes ventajas respecto a una aplicación de escritorio:

- No es necesario un entorno específico o ningún tipo de instalación, ya que se accede a través de un navegador.
- No es necesario que el dispositivo desde el que se accede tenga mucha potencia ya que el peso no lo soporta el equipo, sino el servidor donde está alojada.
- La aplicación puede estar en la nube, con lo que sería accesible para cualquier ordenador con acceso a internet o estar en una intranet.

- Es muy adaptable, visualmente intuitiva y muy fácil de actualizar.

Por ventajas como estas, actualmente se opta más por realizar una aplicación web que por una aplicación de escritorio como se hacía hace unos años ya que tiene muchas ventajas y cubren igualmente las necesidades.

2.2 Lenguajes

A continuación indicaremos cada lenguaje utilizado para construir la aplicación web:

HTML5 (HyperText Markup Language)

Es un lenguaje de marcado que se utiliza para el desarrollo de páginas web.

HTML se encarga de desarrollar una descripción sobre los contenidos que aparecen como textos y sobre su estructura, complementando dicho texto con diversos objetos (como fotografías, animaciones, etc).

Es un lenguaje simple y general que sirve para definir otros lenguajes que tienen que ver con el formato de los documentos. El texto en él se crea a partir de etiquetas, también llamadas tags, que permiten interconectar diversos conceptos y formatos.

Para su escritura se crean etiquetas que aparecen especificadas a través de corchetes o paréntesis angulares: < y >. Entre sus componentes, los elementos dan forma a la estructura esencial del lenguaje, ya que tienen dos propiedades (el contenido en sí mismo y sus atributos).

HTML permite añadir scripts, los cuales brindan instrucciones específicas a los navegadores que se encargan de procesar el lenguaje. Entre los scripts que pueden agregarse, los más conocidos y utilizados son JavaScript y PHP.

La versión HTML5 permite Incorpora etiquetas (canvas 2D y 3D, audio, vídeo) con códecs para mostrar los contenidos multimedia los cuales han sido utilizados para realizar este proyecto.

CSS3 (Cascading Style Sheets) (“Hojas de estilo en cascada”) es un lenguaje que se utiliza para estilizar los elementos(colores, formas, margenes, etc...) de un lenguaje de marcado como HTML a uno o varios documentos de forma masiva.

Se le denomina estilos en cascada porque se aplican de arriba a abajo (siguiendo un patrón denominado herencia) y en el caso de existir ambigüedad, se siguen una serie de normas para resolverla.

La idea de CSS es la de utilizar el concepto de separación de presentación y contenido, intentando que los documentos HTML incluyan sólo información y datos, relativos al significado de la información a transmitir (el contenido), y todos los aspectos relacionados con el estilo se encuentren en un documento CSS independiente (la presentación).

Ventajas:

- Si necesitamos hacer modificaciones visuales lo hacemos en un solo lugar y no tenemos que editar todos los documentos HTML en cuestión por separado.

- Se reduce la duplicación de estilos en diferentes lugares, por lo que es más fácil organizar y hacer cambios. Además, al final la información a transmitir es considerablemente menor (las páginas se descargan más rápido).
- Es más fácil crear versiones diferentes de presentación para otros tipos de dispositivos: tablets, smartphones o dispositivos móviles, etc...

JavaScript

Es un lenguaje de programación interpretado. Se utiliza principalmente del lado del cliente en páginas web HTML para realizar operaciones y únicamente en el marco de la aplicación cliente, sin acceso a funciones del servidor. Actualmente es ampliamente utilizado para enviar y recibir información del servidor junto con ayuda de otras tecnologías como Ajax. JavaScript se interpreta en el agente de usuario al mismo tiempo que las sentencias se van descargando junto con el código HTML. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico. Actualmente también es usado en muchos entornos fuera del navegador, tal como Node.js, Apache CouchDB y Adobe Acrobat. Se basa en prototipos, multiparadigma, de un solo hilo, dinámico, con soporte para programación orientada a objetos, imperativa y declarativa. El estándar para JavaScript es ECMAScript. A partir del 2012, todos los navegadores modernos soportan completamente ECMAScript 5.1. Los navegadores viejos soportan al menos ECMAScript 3. Después de la publicación en 2015 de ECMAScript 6 los estándares ECMAScript están en ciclos de lanzamiento anuales.

Para desarrollar el proyecto DBCASE Web tuvimos que buscar distintas librerías que nos facilitarán el desarrollo tanto de los distintos gráficos como de todas las herramientas necesarias para poder diseñar y manipular un modelo E/R con todas las opciones existentes, a continuación describiremos cada una de ellas.

2.3 Librerías

Vis Js

Es una librería Javascript que nos permite crear redes con nodos y aristas. Permite añadir distintas formas de nodos, estilos, colores, tamaños, imágenes y formas personalizadas.

La visualización de la red funciona en cualquier navegador ya que utiliza un transcompilador llamado Babel que permite convertir el código Javascript en una versión de JavaScript compatible con versiones anteriores que pueden ejecutar los motores JavaScript más antiguos. Vis JS tiene doble licencia en:

- Apache 2.0 License.
- The MIT License

Actualmente contribuyen 90 personas en su desarrollo el cual empezó a realizarse desde 2013.

Como dijimos anteriormente esta librería nos permite poder realizar muchos de los gráficos en los diseños que hemos utilizado, sin embargo no todos, por lo que tuvimos que añadir

funcionalidades(subrayados, dobles líneas, movimiento de elementos, entre otros.) y modificaciones sobre las figuras de los nodos ya existentes.

JQuery

Es una librería de JavaScript rápida, compilada en un solo fichero que nos permite recorrer y manipular documentos HTML, manejar eventos, animación, Ajax entre otros, haciendo su uso simple y compatible con una multitud de navegadores. Es una librería con una gran comunidad de usuarios, por lo que está actualizándose constantemente, existe una extensa documentación, es compatible y convive correctamente con otras librerías Javascript.

jsPDF

Es una librería Javascript que nos permite generar archivos PDF a partir de código HTML. Esta librería nos ayudará a generar los documentos PDF de los diagramas que se diseñe en la aplicación, tiene una licencia MIT y su desarrollo empezó en el año 2010.

Bootstrap 4

Es una librería CSS y JavaScript. Permite a los desarrolladores y diseñadores crear o diseñar rápidamente sitios web totalmente adaptativos. Evita escribir mucho código CSS con diseños ya predefinidos en la librería.

Algunas de las ventajas principales son las siguientes.

- Sistema de cuadrículas receptiva predefinidas: Bootstrap viene con un sistema de cuadrícula predefinido, el cual ya se adapta correctamente a las pantallas desde la que se acceda a la web
- Componentes: Bootstrap viene con diferentes diseños de componentes (formularios, alertas, símbolos, botones, listas, carruseles, barras de navegación, entre otros.) que se agrega a la web mediante etiquetas y clases HTML.
- JavaScript: Bootstrap es compatible y utiliza JQuery permitiendo a los desarrolladores utilizar complementos personalizados. JQuery brinda soluciones para ventanas emergentes modales, transiciones, carruseles de imágenes, entre otros..
- Documentación: Bootstrap cuenta con una gran cantidad de documentación muy clara para el desarrollador. Cada fragmento de código se describe y explica con detalle explícito en su sitio web. Cada explicación incluye ejemplos con una implementación básica, lo que simplifica el proceso para cada desarrollador.
- Comunidad: Bootstrap es un proyecto de código abierto, cuenta con una gran comunidad de diseñadores y desarrolladores.

3. Diseño

Debido a que el apartado más importante de este proyecto es convertir una aplicación de escritorio en una aplicación web, la parte del diseño ha sido cambiada totalmente, manteniendo la disposición de los elementos en algunos aspectos, pero también añadiendo otros elementos o sectores para poder ser una aplicación totalmente manejable para el usuario. A continuación se describen con detalles los principales cambios y mejoras que se han implementado con respecto a la versión de escritorio.

3.1 Portada o página de login

Una de las nuevas funcionalidades fue poder autenticarse para poder acceder a la aplicación web por lo que hemos diseñado una portada que permita acceder al sistema e iniciar sesión. Al ser una portada se ha intentado hacer lo más simple con los tres logos principales de la aplicación, logo de la UCM, logo de nuestra aplicación y logo de la Facultad de informática.

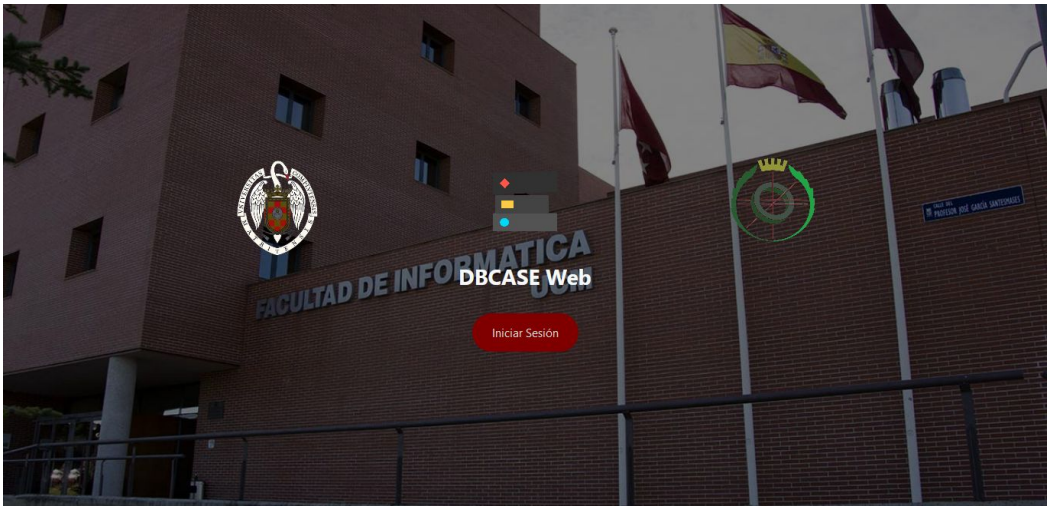

Figura 3.1 Página de bienvenida a la aplicación web.

3.2 Página principal

Una vez se ha iniciado sesión se puede ver el diseño principal que tiene la aplicación web. Se ha intentado dar un aspecto moderno para lo cual se ha utilizado Bootstrap en todos los elementos y secciones de la página. Como perspectiva por defecto se puede ver la división de la página en cuatro paneles(esquema conceptual, elementos y dominios, esquema lógico y esquema físico), en la parte superior izquierda un menú para la configuración del diseño, y en la parte superior derecha un apartado del perfil de usuario.

Utilizar Bootstrap nos permite que la aplicación web se adapte fácilmente a cualquier pantalla haciendo que cada capa se adapte al dispositivo desde el que se accede a la web, tanto un portátil, tablet o dispositivo móvil.

Para todos los tamaños de texto definidos en la aplicación se han utilizado los tamaños predefinidos en Bootstrap ya que cuenta con variedad de clases que nos hacen modificar su tamaño fácilmente, el tipo de letra es “Roboto” ya que es una fuente moderna y fácilmente accesible, incluye diferentes grosores fino, ligero, regular, medio, negrita y negra con estilos oblicuos correspondientes, está entre las más usadas y permite apreciar con claridad todo el contenido de la web.

Los colores de los elementos, paneles, botones y menús dependen del tema que haya elegido en la aplicación.

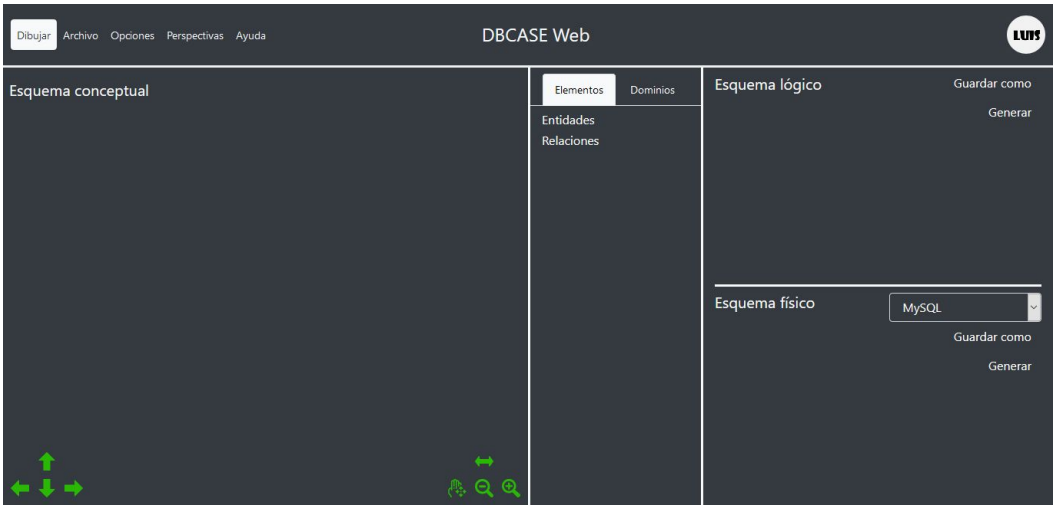

Figura 3.2 Panel de desarrollo para realizar los diagramas.

3.3 Menú

El menú ha sido diseñado igual que toda la aplicación con Bootstrap, se han utilizado listas desplegables para poder realizar el efecto de un menú.

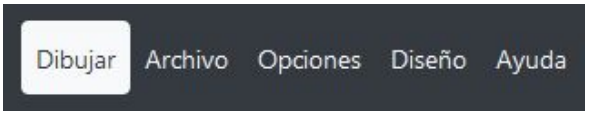

Figura 3.3 Menú disponible del panel de desarrollo.

El menú de la aplicación web cuenta con los siguientes apartados:

- Dibujar: Es un botón que está acoplado al menú para poder abrir el panel lateral de añadir elementos, Se explicará su funcionamiento de forma detallada más adelante.
- Archivo: Permite gestionar los cambios que se realizan sobre el sistema: A continuación se indica cada subapartado:
    - Nuevo: Permite poder empezar otro proyecto desde cero, pidiendo confirmación si existe un diagrama en el panel, de tal forma que se pueda realizar un diseño otra vez.

- Abrir: Despliega un cuadro de diálogo en el que se puede abrir un archivo .dbw, para poder continuar con un proyecto que se ha realizado. Este apartado será explicado detalladamente en el apartado importación de archivos.
- Guardar: Lanza un evento el cual guarda el diagrama diseñado, permitiendo descargar un fichero .dbw con el diagrama realizado. Este punto será explicado de forma detallada en el apartado "Generación y descarga de archivos .dbw".
- Exportar como PDF: Permite poder descargar un fichero PDF con el diagrama realizado en el panel de diseño. Este punto será explicado de forma detallada en el apartado "Generación y descarga de archivos PDF".
- Opciones: Estas funcionalidades ya existían anteriormente en la aplicación de escritorio, sin embargo, al ser una aplicación web se ha realizado todo el diseño otra vez con lenguaje de marcado HTML y hojas de estilos CSS:
    - Temas: Para facilitar la creación de cada tema se ha utilizado la librería Bootstrap, se ha utilizado variables de sesión para que la elección del usuario persista durante toda la sesión. Este apartado se explica con más detalle en el apartado Temas.
    - Lenguaje: Al adaptar todo a web, actualizamos todo el proyecto a un proyecto con el framework de Spring Boot, el cual cuenta con un plugin que facilita la gestión de idiomas en aplicaciones web, Este punto será explicado de forma detallada en el apartado Implementación de los lenguajes.
- Diseño: Se han mantenido las perspectivas existentes, y a su vez se añadieron nuevos diseños para facilitar la visibilidad del usuario. Este punto será explicado de forma detallada en el apartado Perspectivas.
- Ayuda: Se ha diseñado para la aplicación web el mismo contenido, añadiendo un cuadro modal de la librería Bootstrap la forma de mostrar los datos. Permite mostrar información de los autores y versiones anteriores de la aplicación.

3.4. Autenticación

Al implementar la autenticación en el panel principal se añadió un desplegable con la imagen del email del usuario y una opción para poder finalizar la sesión.

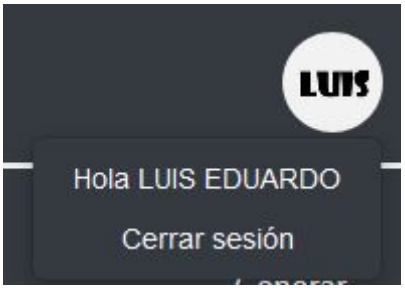

Figura 3.4 Menú con el inicio de sesión del usuarios y su perfil.

Detalles de la implementación de la autenticación

Spring Boot cuenta con una librería “Spring Security OAuth 2.0 Client” el cual ha sido instalado mediante el repositorio Maven y se ha configurado con el servicio “Google OAuth2 Sign-In” para poder autenticarse con la cuenta de la UCM.

Spring Security OAuth 2.0 Client implementa la autenticación mediante el servicio de autorización de OAuth 2.0, este permite a una aplicación web (DBCASE Web) obtener acceso limitado a un servicio HTTP en nombre del propietario de un recurso (Google).

3.5. Disposición de los paneles

Se ha creado la interfaz para la aplicación web manteniendo casi todo el mismo diseño principal de la aplicación de escritorio, ya que consideramos que es una GUI clara, sencilla e intuitiva con las mismas funcionalidades existentes y también las que se han añadido.

La interfaz principal muestra en una sola página el diagrama entidad-relación y la generación de código de forma que ahora se puede ver de un vistazo el diagrama y el código generado.

El panel de generación de código está dividido en dos paneles distintos: Esquema lógico y esquema físico. De esta manera quedan ambos lenguajes en paneles independientes.

Cada uno de estos paneles han sido tratados de forma independiente cada uno para poder crear las distintas perspectivas disponibles los cuales serán explicados con detalle más adelante.

Se ha quitado de la vista principal el panel con la barra lateral para añadir elementos, ya que en una aplicación web optamos con los efectos de JQuery que se deslice desde la izquierda del panel solo cuando se quiera dibujar sobre el diagrama entidad-relación.

3.6. Temas

Se han mantenido los dos temas que ya existían en la aplicación de escritorio, ya que se consideraba importante (los temas permiten reducir la fatiga visual y hacer más cómodo el trabajo en entornos con poca luz), pero al haber cambiado a la aplicación web con Bootstrap se ha implementado un diseño principal, tanto con el tema oscuro (por defecto), tan usado en las interfaces de usuario modernas, como el tema claro.

Detalles de la implementación de los temas

Todo el diseño de los temas se ha desarrollado con el framework CSS Bootstrap mediante clases en cada sección y elemento. La configuración de los temas se almacenan en variables de sesión para que la configuración persista en el sistema.

Spring Boot permite almacenar datos en variables que se asocian a un identificador generado por Spring Boot Starter Oauth2, estas variables se almacenan en la sesión y son destruidas cuando la sesión ha finalizado.

Los colores principales de la aplicación (tanto por paneles, botones, fondo, letras, etc) son colores predefinidos de Bootstrap.

Las clases text-dark y bg-dark para el tema oscuro y text-light y bg-light para el tema ligero o claro permiten modificar todos los colores de la aplicación.

La modificación del tema se realiza desde el menú de opciones de la página principal de la aplicación.

3.7. Barra lateral para añadir elementos

Figura 3.5 Barra lateral con los elementos a añadir disponibles.

Se ha diseñado el panel con las mismas características de la aplicación de escritorio. Sin embargo ya no está permanentemente sobre la aplicación, se ha añadido un botón el cual con un evento JQuery permite desplazar para aparecer o desaparecer el panel según lo necesite el usuario.

Permite crear elementos nuevos de forma intuitiva, pero también permite crear haciendo clic derecho sobre el panel del diagrama.

A continuación se detallan las principales características de este panel:

- Permite al usuario añadir cuatro tipos de elementos: entidades, relaciones, relaciones ISA y atributos.

- Al añadir entidad y añadir relación muestran un cuadro modal de Bootstrap en el que se permite insertar el nombre y las características del elemento.
- El botón ISA, al no necesitar nombre ni configuración, inserta directamente la relación en el diagrama.
- El botón atributo permite seleccionar mediante un menú desplegable a qué entidad o relación pertenece el atributo, también se le pide al usuario el nombre y las características del atributo.
- Los elementos añadidos desde este panel se ponen de manera aleatoria sobre la parte visible del panel. Para realizar esto generamos espacios aleatorios sobre la parte visible del diagrama con funciones de la librería de vis.js.

Cambios en el panel de diseño

El panel de diseño del diagrama entidad-relación ha sufrido cambios ya que al haber realizado el cambio a aplicación web, se ha añadido más opciones para poder desplazarse por el diagrama de tal forma que la experiencia del usuario sobre la aplicación sea exitosa. Se ha decidido quitar alguna opción de visualización, ya sea porque se ha sustituido por otra más intuitiva o por que se ha realizado de otra forma. Muchas de las opciones ya estaban implementadas en la librería Vis Js, aunque también hemos tenido que implementar funcionalidad para poder cumplir con las especificaciones indicadas por el profesor.

A continuación se describe los cambios y también la funcionalidad que se ha añadido:

- La aplicación de escritorio contaba con una vista del panel de entidad-relación en miniatura cuya funcionalidad permitía movernos por todo el panel arrastrando el ratón sobre este.

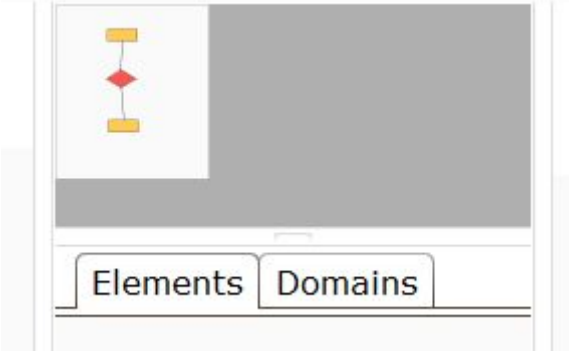

Figura 3.6 Antiguo panel para poder visualizar el panel del diagrama.

- Se ha añadido un botón  para centrar el diagrama sobre la vista del panel. Esto es una parte de la funcionalidad del panel en miniatura, ya que permite con un click poder observar todo el diagrama que se haya diseñado en la parte visible del panel. Para realizar esta acción se tiene en cuenta gracias a la librería Vis Js las coordenadas que existen de cada elemento que se haya diseñado, con estos datos en una función se obtiene el tamaño

total del diagrama, con una función focus a la que se envía como parámetro el tamaño total se obtiene el diagrama sobre la parte visible del panel.

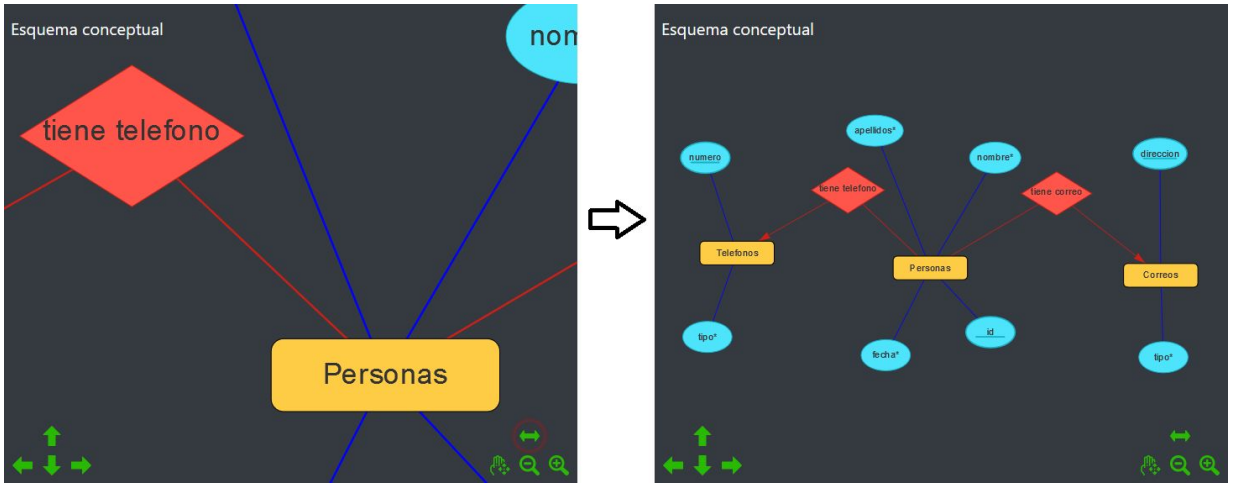

Figura 3.7 A la izquierda diagrama con vista acercada, a la derecha diagrama con vista alejada. Nueva funcionalidad.

- El panel de diseño permite poder seleccionar varios elementos o poder desplazarse por el diagrama, para poder realizar ambas acciones se tiene que arrastrar el ratón, se ha añadido un botón  que si esta como en la imagen permite arrastrar el diagrama(esta asi por defecto), y si esta como en la siguiente imagen  con un fondo púrpura permite realizar la selección múltiple. Estas funcionalidades también sustituyen al panel en miniatura.
- La aplicación web también permite realizar la selección múltiple manteniendo pulsada la tecla “Ctrl”.

  Para realizar ambas funcionalidades obtenemos las posiciones de las coordenadas desde que se ha pulsado el click derecho hasta que se ha quitado el click, con esto si se quiere arrastrar sobre el diagrama, se realiza el cálculo del centro de cada coordenada y con la función focus de la librería vis.js se mueve el diagrama. Si lo que se quiere es la selección múltiple, con las coordenadas obtenidas se calculan cuales son los elementos que se encuentran dentro de esas coordenadas, estos elementos son almacenados en un array, y con otra función de la librería vis.js seleccionamos cada elemento.

- Se han añadido botones de flecha  , alternativos al desplazamiento sobre el diagrama arrastrando el ratón. Creímos que era importante dar múltiples opciones de desplazamiento sobre el panel de diseño.

- Se han añadido también los botones de zoom  para poder alejar o acercar la visualización del panel, pudiendo alejar para tener una vista global de todo lo diseñado, o acercarnos para centrar la vista del panel en alguna parte específica del diagrama. Esta funcionalidad también se puede realizar con la rueda de desplazamiento del ratón.

- La aplicación web también tiene implementado el submenú para poder añadir elementos al Panel de diseño al igual que la barra lateral para añadir elementos dando más usabilidad al usuario. Todos los submenús dentro del panel se utilizan con el botón derecho del ratón, el cual cambia en función de donde se hace clic derecho. Si se hace clic derecho sobre una parte libre del panel aparecerán las opciones para la creación de elementos, si se hace clic derecho sobre un elemento aparecerán las opciones de un elemento.

Se ha añadido también la opción para poder generar los elementos de tipo agregación, el cual aparece en el submenú de los elementos tipo relación o tipo entidad.

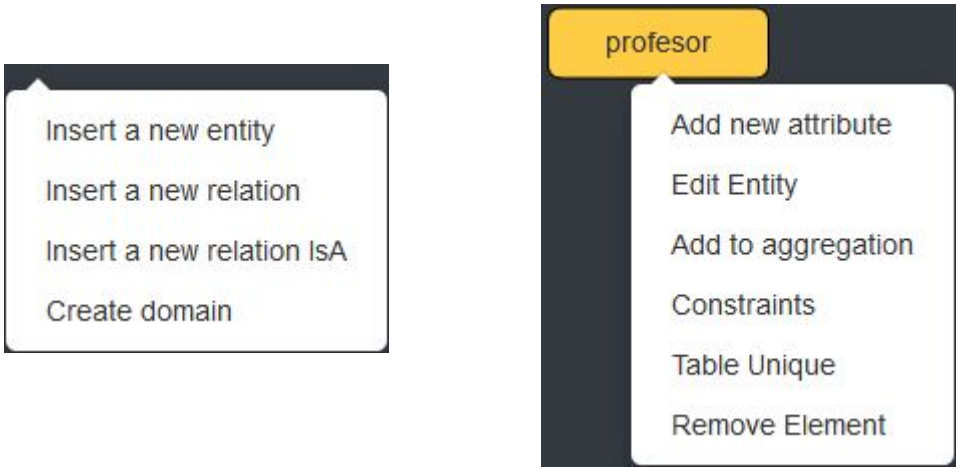

Figura 3.8 Menú con botón derecho sobre el panel a la izquierda y sobre un elemento a la derecha.

3.8. Panel de información

Se ha creado la interfaz para la aplicación web manteniendo el diseño principal de la aplicación de escritorio.

El panel de información muestra un esquema en forma de árbol del modelo entidad relación existente en el panel de diseño, y en él se puede ver de forma esquemática todos los elementos y cómo están relacionados. Además,este panel muestra la información en el caso de que sea una entidad de agregación, todos estos son actualizados constantemente con un evento Ajax que se ejecuta siempre que se realiza algún cambio sobre el panel de el diagrama entidad-relación.

El contenido de cada elemento se visualiza en el panel de la siguiente forma:

Figura 3.9 Nuevo diseño del panel central Elementos.

- **Entidades**: muestra todas las entidades del esquema y sus atributos. Muestra el dominio del atributo al lado del nombre y representa gráficamente de forma correcta los subatributos de atributos compuestos (como ramas de los mismos).
- **Relaciones**: muestra todas las relaciones del esquema y para cada una de ellas muestra sus atributos y las entidades que relaciona. Para cada una de las entidades muestra la cardinalidad al lado del nombre.

Para las relaciones ISA muestra la entidad padre como la primera entidad de la relación, y las entidades hijas con un icono que las diferencia como tal.

Para la renderización del árbol se ha utilizado listas HTML (<li>), con eventos JQuery para minimizar su información cuando no sea necesaria y clases Bootstrap para mantener el mismo diseño en toda la web.

3.9. Panel de dominios

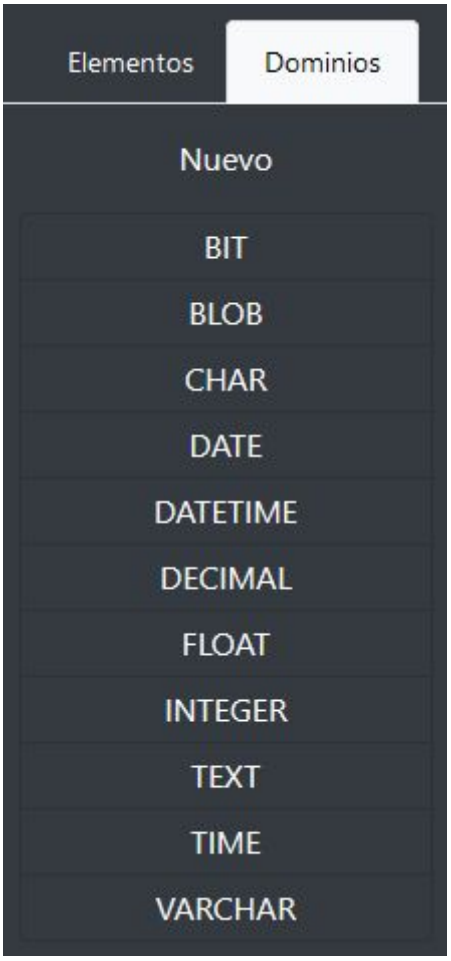

Figura 3.10 Nuevo diseño del panel central Dominios.

Se ha creado la interfaz para la aplicación web con Bootstrap manteniendo el diseño principal de la aplicación de escritorio ya que lo consideramos intuitivo.

Este panel de dominios informa al usuario de las opciones disponibles que tiene para asociar con los atributos creados, también permite crear nuevos tipos de dominio que el usuario considere necesarios.

Detalles de la implementación del panel de dominios

Pese a que el funcionamiento es el mismo, todo este proceso se realiza en el front de la aplicación, por lo que se ha vuelto a desarrollar para el entorno web.

Se ha creado una clase en Javascript que permite crear automáticamente los dominios por defecto en cada nueva sesión, añadir nuevos dominios y devolver en una lista HTML con clases de Bootstrap para pintarlos en el panel correspondiente.

Con JQuery se detecta si en los dominios han existido cambios, si es el caso hace la llamada para actualizar el panel.

3.10. Paneles de textos

Figura 3.11 Nuevo diseño del panel de esquema lógico y físico con el resultado de un diagrama.

Los códigos de los textos generados en el esquema lógico y esquema físico son generados en HTML, por lo que se ha realizado la adaptación del diseño con Bootstrap ya que nos ofrece diferentes diseños adaptables para cualquier dispositivo en los dos temas existentes en el sistema.

3.11. Perspectivas

La versión anterior de la aplicación contaba con tres perspectivas, se han mantenido esas tres y creado todo el diseño para la vista web, se ha añadido tres perspectivas más para poder dar al usuario la opción de acceder a cada panel del sistema.

Bootstrap 4 presenta un sistema de cuadrículas que se escala en 12 columnas, esto con JQuery ha facilitado la creación de cada perspectiva. Tal que si era necesario dividir la pantalla en dos, sobre una etiqueta div con clase row, y dos etiquetas hijas div con clase col-md-6 se dividía en dos la vista.

A continuación se describe cada perspectiva existente en la aplicación web:

Figura 3.12 Perspectivas disponibles en la nueva aplicación web.

- **Perspectiva 1**: Es la perspectiva por defecto y permite visualizar todos los paneles del sistema. Este diseño ya existe en la aplicación de escritorio pero al ser una aplicación web se ha diseñado todo desde cero. Permite visualizar:

  Esquema conceptual / Elementos-Dominios / Esquema logico /Esquema fisico

- **Perspectiva 2**: Esta perspectiva permite visualizar principalmente el esquema donde se dibuja el diagrama entidad-relación y también los códigos que son generados con el diagrama. Permite visualizar:

  Esquema conceptual / Esquema logico /Esquema fisico

- **Perspectiva 3**: Esta perspectiva es útil para diseñar el diagrama entidad-relación, ya que también se visualizan los elementos y dominios existentes. Permite visualizar:

  Esquema conceptual / Elementos-Dominios

- **Perspectiva 4**: Esta perspectiva ha sido diseñada para la relación entre el diagrama entidad-relación y también el esquema lógico. Permite visualizar:

  Esquema conceptual / Esquema lógico

- **Perspectiva 5**: Esta perspectiva permite visualizar la relación entre el diagrama entidad-relación y también el esquema lógico. Permite visualizar:

  Esquema lógico / Esquema fisico

- **Perspectiva 6**: Esta perspectiva es útil para la relación entre lo generado en el esquema físico y el esquema lógico y también observar los elementos existentes. Permite visualizar:

  Elementos-Dominios / Esquema conceptual / Esquema lógico

3.12. Cuadros de diálogo

Se ha mantenido el diseño de cada cuadro de diálogo pero se ha tenido que diseñar para la aplicación web, la librería usada Boostrap, nos aporta un diagrama unos cuadros de diálogo predefinidos, los cuales son adaptables a cualquier plataforma. Tiene un diseño limpio e ideal para nuestra aplicación.

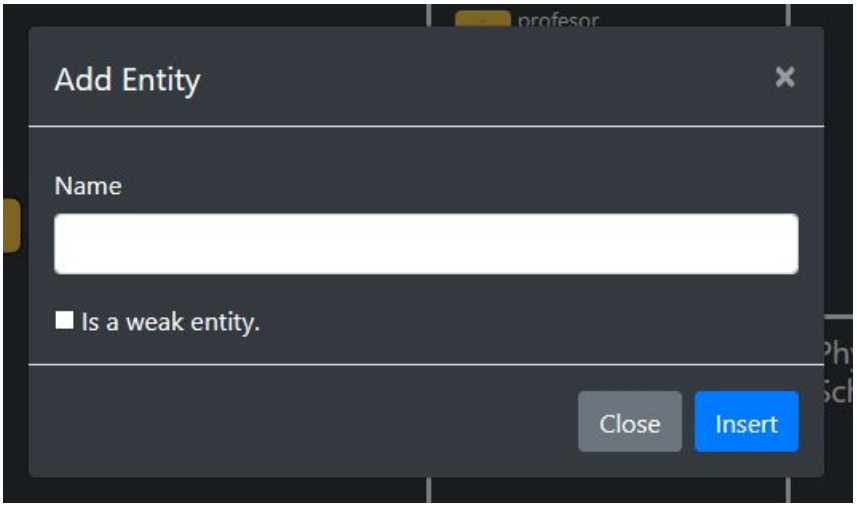

Figura 3.13 Diseño de un cuadro de diálogo en la aplicación

Detalles de la implementación del cuadro de diálogo

Se ha diseñado una función que recibe como parámetro un array asociativo (con el título, inputs, labels y botones existentes del modal) el cual devuelve un código HTML con un formulario interno envueltos en el cuadro de diálogo. Cada funcionalidad que necesite el cuadro de diálogo llama a la función enviando el array asociativo el cual se muestra en la pantalla. Al pulsar enviar o confirmar, se trata el formulario enviado y se observa el campo oculto “typeAction” la cual redirige a ejecutar la opción que se necesite con los datos del formulario.

4. Funcionalidad

Los cambios principales de la aplicación han sido principalmente:

- La adaptación del diseño.
- Nuevas funcionalidades para dotar de más propósitos a la aplicación

Sin embargo también se han realizado algunos cambios como:

- Corregir errores en la traducción del esquema lógico y físico.
- Optimizar algoritmos de traducción.

A continuación detallamos la implementación de cada apartado.

4.1. Autenticación

Una funcionalidad que se implementó en este proyecto es la posibilidad de autenticarse para poder iniciar sesión con una cuenta, Spring boot y OAuth 2.0 permite mediante su repositorio configurar un inicio de sesión ya sea con una cuenta de Google, GitHub, Facebook o Gmail.

En nuestro caso configuramos con los servicios de Google ya que son los servicios que utiliza la universidad para las cuentas de los estudiantes.

Para esto desde la consola de desarrolladores de Google creamos un proyecto y configuramos una identificador de cliente para aplicaciones web, la cual nos devuelve una ID de cliente.

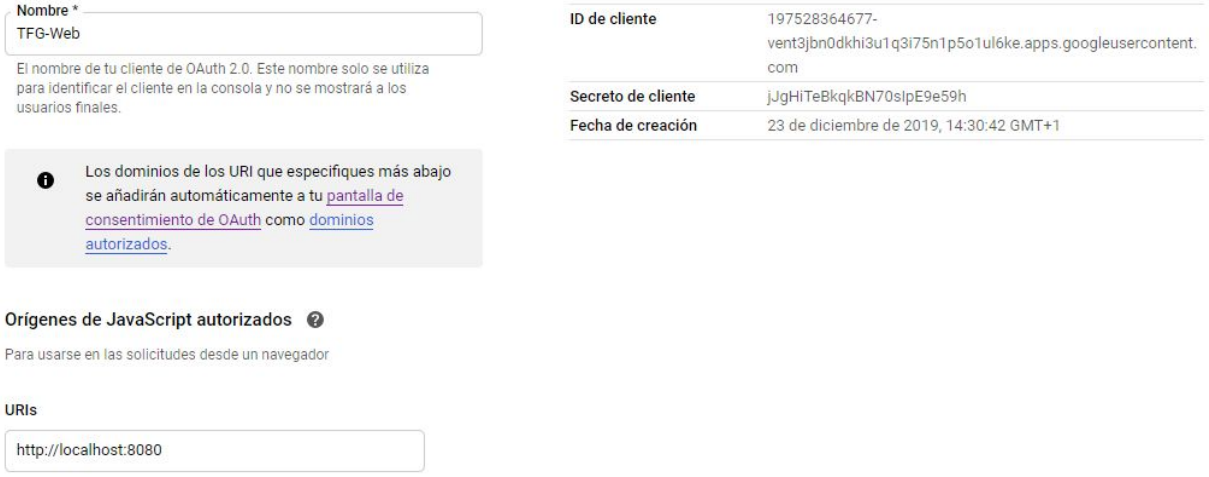

Figura 4.1 Configuración API de Google para la autenticación.

En la aplicación web instalamos la dependencia de Spring Boot spring-security-oauth2 el cual nos añade un fichero en el que se configura los parámetros de acceso al servicio.

```
security.oauth2.client.clientId = 197528364677-vent3jbn0dkhi3u1q3i75n1p5o1ul6ke.apps.googleusercontent.com
security.oauth2.client.clientSecret = jJgHiTeBkqkBN70sIpE9e59h
security.oauth2.client.accessTokenUri  =  https://www.googleapis.com/oauth2/v3/token
security.oauth2.client.userAuthorizationUri  =  https://accounts.google.com/o/oauth2/auth
security.oauth2.client.tokenName = oauth_token
security.oauth2.client.authenticationScheme = query
security.oauth2.client.clientAuthenticationScheme = form
security.oauth2.client.scope = profile email

security.oauth2.resource.userInfoUri  =  https://www.googleapis.com/userinfo/v2/me
security.oauth2.resource.preferTokenInfo = false
```

Figura 4.2 Configuración del proyecto para la autenticación con el servicio de Google

Una vez configurado el servicio, se crea un controlador en el que se configura la página que actuará como inicio de sesión, los recursos a los que se debe acceder sin iniciar sesión, la URL cierre de sesión, las cookies a almacenar, entre otros.

```
public class WebSecurityConfiguration extends WebSecurityConfigurerAdapter {
   @Override
   protected void configure(HttpSecurity http) throws Exception {
       http
       .csrf()
       .disable()
       .authorizeRequests()
       .antMatchers("/index","/", "/**/*.png", "/**/*.css", "/**/*.js", "/**/*.ttf", "/**/*.jpg")
       .permitAll()
       .anyRequest()
       .authenticated()
       .and()
       .formLogin()
       .loginPage("/login")
       .loginProcessingUrl("/perform_login1")
       .defaultSuccessUrl("/homepage.html", true)
       .and()
       .logout().logoutRequestMatcher(new AntPathRequestMatcher("/logout"))
       .logoutSuccessUrl("/index").deleteCookies("JSESSIONID")
       .invalidateHttpSession(true);
   }
}
```

Figura 4.3 Configuración sobre la sesión de los usuarios.

4.2. Exportar archivos

La aplicación permite poder exportar archivos del esquema conceptual, esquema físico y esquema lógico, sin embargo este proceso anteriormente se realizaba con librería no compatibles con aplicaciones web, por lo que nosotros optamos por realizar todo este proceso con una librería JsPDF que permite generar ficheros mediante código Javascript del String que se indique.

1) var exportValue = JSON.stringify(all_nodes, undefined, 2);
2) var blob = new Blob([exportValue], {type: "application/json"});
3) saveAs(blob, "nameFile.dbw");

En el ejemplo indicado en la línea 1 se almacena en una variable tipo JSON “exportValue” en forma de String, esta variable a su vez se guarda en otra variable tipo Blob y la librería JsPDF

nos provee una función saveAs que nos permite generar el fichero “nameFile.dbw” con el contenido de la variable “blob”.

A continuación describiremos el contenido de cada fichero que utilizamos en la aplicación:

- **Exportar esquema conceptual**: Para realizar el fichero del esquema conceptual, como indicamos anteriormente todos los datos se almacenan en 4 arrays, estos cuatro arrays se han unido posteriormente en un solo objeto el cual lo convertimos en variable tipo JSON el cual es almacenado en la variable tipo blob explicado anteriormente. Este fichero se descarga con una extensión tipo “dbw” iniciales de DBCASE Web.

```
var all_nodes = {
        nodesA: nodesB,
        edgesA: edgesB,
        nodes_superA: nodes_superB,
        edges_superA: edges_superB
};
// pretty print node data
var exportValue = JSON.stringify(all_nodes, undefined, 2);
```

Figura 4.4 Array con la variable que almacena todos los nodos y aristas.

- **Exportar esquema lógico**: Para realizar el fichero, mediante JQuery se obtiene todo el contenido del panel del esquema lógico y mediante funciones Javascript quitamos todos los elementos html, el cual una vez en texto puro se almacena en la variable tipo Blob.

```
var str = $("#testResult").html();
var res = str.replace(/<\/p>/g, "\r\n");
res = res.replace(/<p class="h5 text-dark font-weight-bold">/g, "");
res = res.replace(/<p class="text-dark">/g, "");
res = res.replace(/<div class="pl-1 pt-1 pr-1 alert alert-warning">/g, "#");
res = res.replace(/<div class="pl-1 pt-1 pr-1 alert alert-danger">/g, "#");
res = res.replace(/<div class="pl-1 pt-1 pr-1 alert alert-light">/g, "#");
res = res.replace(/<\/div>/g, "");
res = res.replace(/<u>/g, "");
res = res.replace(/<\/u>/g, "");
res = res.replace(/<p>/g, "");
res = res.replace(/&gt;/g, ">");
res = "#"+$("#textGeneratedBy").text()+"\r\n"+res;
var blob = new Blob([res], {type: "text/plain;charset=utf-8"});
```

Figura 4.5 Código que reemplaza las etiquetas HTML para poder generar el fichero del esquema lógico.

- **Exportar esquema físico**: Para realizar el fichero, al igual que con el esquema lógico mediante JQuery se obtiene todo el contenido del panel del esquema físico y mediante funciones Javascript quitamos todos los elementos html, el cual una vez en texto puro se almacena en la variable tipo Blob.

```
var str = $("#resultSPhysicalSchema").html();
var res = str.replace(/<\/p>/g, "\r\n");
res = res.replace(/<p class="h5 text-dark font-weight-bold">/g, "");
res = res.replace(/<p class="text-dark">/g, "");
res = res.replace(/<div class="pl-1 pt-1 pr-1 alert alert-warning">/g, "#");
res = res.replace(/<div class="pl-1 pt-1 pr-1 alert alert-danger">/g, "#");
res = res.replace(/<div class="pl-1 pt-1 pr-1 alert alert-light">/g, "#");
res = res.replace(/<\/div>/g, "");
res = res.replace(/<u>/g, "");
res = res.replace(/<\/u>/g, "");
res = res.replace(/<p>/g, "");
res = res.replace(/<strong>/g, "");
res = res.replace(/<\/strong>/g, "");
res = res.replace(/\*/g, "");
res = "#"+$("#textGeneratedBy").text()+"\r\n"+res;
var blob = new Blob([res], {type: "text/plain;charset=utf-8"});
```

Figura 4.6 Código que reemplaza las etiquetas HTML para generar el fichero del esquema físico.

4.3. Importar archivos:

Para poder importar los ficheros, en el cuadro de diálogo, después de arrastrar un fichero o hacer clic sobre la zona para cargar un archivo.

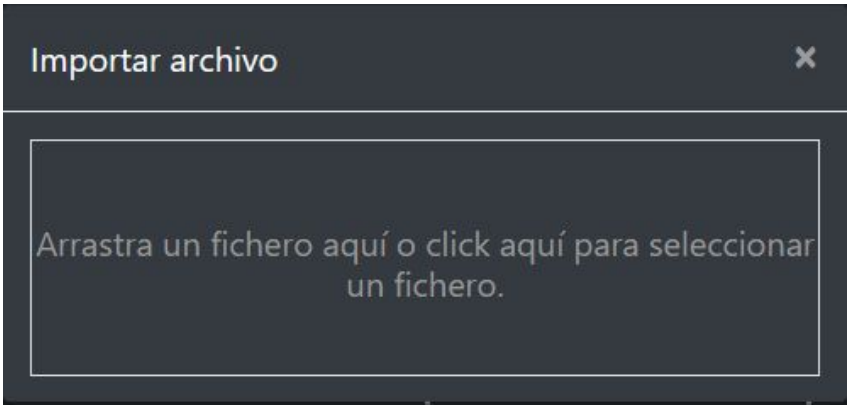

Figura 4.7 Diseño del cuadro de diálogo para importar archivos.

Con la librería jsPDF se lee el contenido del fichero, esto devuelve una variable de tipo String la cual se convierte en una variable tipo JSON, la cual genera una estructura con 4 objetos, la de nodes, nodes_super(elementos de la agregación si existiera), edges y edges_super(aristas de la agregación si existiera).

Cada una de estas variables son añadidas a las variables globales de los nodos y las aristas como se muestra en la imagen con lo que los nodos son mostrados en pantalla.

```
function importNetwork(type, value=null) {
    nodes.clear();
    edges.clear();
    nodes_super.clear();
    edges_super.clear();
    if(type != "session"){
        var inputValue = value;
    }else{
        var inputValue = sessionStorage.getItem('codeSave');
    }
    var inputData = JSON.parse(inputValue);
    for(var i = 0;i<inputData.nodesA.length;i++){
        nodes.add(inputData.nodesA[i]);
    }
    for(var i = 0;i<inputData.edgesA.length;i++){
        edges.add(inputData.edgesA[i]);
    }
    for(var i = 0;i<inputData.nodes_superA.length;i++){
        nodes_super.add(inputData.nodes_superA[i]);
    }
    for(var i = 0;i<inputData.edges_superA.length;i++){
        edges_super.add(inputData.edges_superA[i]);
    }
    updateTableElements();
}
```

Figura 4.8 Función que importa un fichero, convirtiendo el contenido en variable tipo JSON.

4.4. Esquema conceptual

El diseño del esquema conceptual se ha implementado completamente ya que antes se realizaba con librerías Java diseñadas para aplicaciones de escritorio, no compatibles o no diseñadas para aplicaciones web.

Después de sondear e investigar un poco cual sería la librería apropiada para implementarlo, nos decidimos por utilizar una llamada Vis Js, esta librería realizada en Javascript nos permite diseñar todos las figuras que necesitábamos, excepto el elemento "IsA", por lo que creando una clase heredada pudimos crear esa figura.

También se han tenido que realizar modificaciones para añadir dobles bordes a elementos que así lo requerían en función de su configuración.

La generación de los grafos para el diseño se almacenan en 4 arrays, dos array de nodos, en uno de ellos se almacenan los nodos que forman parte de la agregación y en el otro array se almacenan las aristas que no son partes de la agregación.

En otros 2 array se almacenan las aristas, en uno de ellos se almacenan las aristas que forman parte de la agregación y en el otro array se almacenan las aristas que no son partes de la agregación.

Cada nodo y cada arista es un array asociativo los cuales se utilizan para poder realizar el diseño de cada diagrama según las opciones que se tenga añadida.

Cada uno de ellos va asociados a un identificador, el identificador de los nodos es un número que incrementa desde 0 y las aristas tienen un código hexadecimal para poder identificar cada arista.

Elementos: Cada elemento tiene la estructura más abajo mostrada, donde:

- heightConstraint: Tamaño con el que se crea el elemento
- id: Identificador único de cada elemento
- isWeak: Solo para elementos tipo entidad, cuya entidad es débil.
- Label: Nombre que va asociado a cada nodo.
- physics: Gestiona el movimiento que cada nodo respecto a otros nodos.
- scale: El tamaño de los nodos se escalará de acuerdo con la propiedades en este objeto
- Shape: Define la forma del nodo. Los elementos pueden ser de distintos tipos como un elipse, círculo, base de datos, texto. También existe un tipo "custom" que es el que nos permite diseñar uno personalizado.
- Super_entity: Indica los elementos que forman parte de los elementos de agregación.
- widthConstraint: Se especifica una anchura mínima y máxima.
- x: Coordenada x de la posición sobre el panel del elemento.

- y: Coordenada y de la posición sobre el panel del elemento.

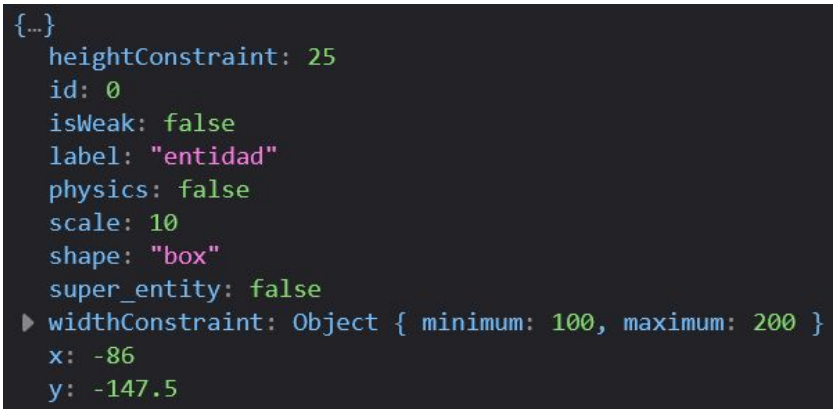

Figura 4.9 Atributos que se almacenan por cada nodo del diagrama.

Aristas: Cada arista tiene la estructura más abajo mostrada, donde:

- from: Identificador del nodo de origen de la arista.
- id: Identificador único de la arista.
- label: Nombre de la etiqueta en la que mostramos la cardinalidad mínima, cardinalidad máxima y el rol en el caso que tenga.
- name: Rol de la arista.
- participation: Variable booleana que es verdadera si la arista tiene una participación.
- participationFrom: Valor de la cardinalidad mínima en el caso de que tenga, este valor se usa para dibujar la arista con doble borde o no.
- participationTo: Valor de la cardinalidad máxima en el caso de que tenga, este valor se usa para dibujar la arista con doble borde o no.
- smooth: Define si la arista será rígida o tendrá movimiento independiente.
- state: Opción que se utiliza para generar aristas asociados doblemente a cada nodo.
- to: Identificador del nodo del final de la arista.

Figura 4.10 Atributos que se almacenan por cada arista del diagrama.

Todos los elementos y las aristas se almacenan en arrays, estos arrays son convertidos en una variable formato JSON. Mediante AJAX de Jquery se envía al servidor, donde se realizará la traducción al esquema físico y lógico lo cual explicaremos en el siguiente apartado de traducciones de código.

4.5. Traducciones de código

Conclusiones previas respecto a la versión anterior

Después de revisar y entender el funcionamiento de la parte lógica del proyecto predecesor DBCASE 2.0, se detectaron varios puntos a tener en cuenta:

1. El proyecto anterior realizaba todas estas comprobaciones en el controlador, por lo que en cada uno de los pasos para la creación y/o modificación de elementos del diagrama se consultaba con los métodos correspondientes y el controlador devuelve esta información a la vista mediante objetos de la biblioteca Javax. Nosotros decidimos que estas comprobaciones las realizaríamos en la vista de manera que no sobrecargamos el controlador con comprobaciones básicas, referentes a los nombres repetidos de elementos, diagramas sin elementos, crear atributos sin tener ninguna entidad previamente creada.

```
192 function eventAddEventRecipient(){
193     $( "#recipient-name" ).on( "blur keyup", function(){
194           var nameValue = $( "#recipient-name" ).val();
195           var tipoAdd = $( "#tipoAdd" ).val();
196           if(!existElementName(nameValue, tipoAdd)){
197           $('#insertModal').prop('disabled', false);
198           $( "#recipient-name" ).removeClass("is-invalid");
199           }else{
200               $('#insertModal').prop('disabled', true);
201               $( "#recipient-name" ).addClass("is-invalid");
202           }
203     });
204 }
205
```

Figura 4.11 Comprobación de nombre de elemento repetido init.js.

```
41      for (Iterator it = lista.iterator(); it.hasNext(); ){
42          TransferEntidad elem_te = (TransferEntidad)it.next();
43          if (elem_te.getNombre().toLowerCase().equals(te.getNombre().toLowerCase())){
44              controlador.mensajeDesde_SE(TC.SE_InsertarEntidad_ERROR_NombreDeEntidadYaExiste,te);
45              return;
46          }
47      }
```

Figura 4.12 Comprobación en versión anterior ServiciosEntidades.java

2. El resultado obtenido por cada diagrama en el controlador se obtiene en formato HTML, el cual es enviado a la vista mediante la petición Ajax que se hizo para realizar la

traducción de código. Una vez en la vista esta es insertada en su panel correspondiente según la traducción que se haya realizada.

Este es el funcionamiento tanto del esquema lógico como físico.

```
<div class='pl-1 pt-1 pr-1 alert alert-light'>
    <p class='h5 text-dark font-weight-bold'>Relaciones</p>
    <p>profesores (<u>DNI</u>, cedula*)</p>
    <p>alumnos (<u>DNI</u>, acceso*)</p>
    <p>personas (<u>DNI</u>, domicilio*, telefono*, nombre*)</p>
    <p></p>
</div>
<div class='pl-1 pt-1 pr-1 alert alert-light'>
    <p class='h5 text-dark font-weight-bold'>Restricciones de integridad referencial</p>
    <p>profesores.personas_DNI -> personas.DNI</p>
    <p>alumnos.personas_DNI -> personas.DNI</p>
    <p></p>
</div>
<div class='pl-1 pt-1 pr-1 alert alert-light'>
    <p class='h5 text-dark font-weight-bold'>Restricciones perdidas</p>
    <p></p>
</div>
```

Figura 4.13 Información enviada a la vista desde el controlador con formato HTML.

.

Figura 4.14 Diseño de la información generada por el diagrama.

3. Como se indicó en el punto 1, en la versión anterior el acceso al controlador se realizaba en cada una de las modificaciones y/o inserciones de elementos en la aplicación. Esto en

una versión web era inviable ya que si manteníamos esta lógica la aplicación no podría ser utilizada concurrentemente. Por tanto decidimos construir todos los objetos del lado del cliente utilizando Javascript. Cuando el usuario desee generar el esquema lógico o físico se envía al controlador un objeto construido mediante Javascript con toda la información necesaria para dicha ejecución, de la misma manera para el botón ejecutar que se encarga de ejecutar las queries en una base de datos insertando las credenciales. Los detalles de estos objetos se aclaran en este mismo apartado.

```
235                 myObj["data1"] = JSON.stringify(resultNodes);
236                 myObj["data2"] = JSON.stringify(edgesData);
237
238                 //tomamos la data de la entidad relacion de alto nivel
239                 myObj["data3"] = JSON.stringify(nodesSuper);
240                 myObj["data4"] = JSON.stringify(edgesSuperData);
241                 var json = JSON.stringify(myObj);
242
243                 $.ajax({
244                     type: 'POST',
245                     url: '/generateData',
246                     data: json,
247                     contentType: "application/json",
248                     success: function (data) {
```

Figura 4.15 Envío de la información al servidor para realizar las traducciones.

Implementación

Front end

Como se indicó en el apartado “Conclusiones previas respecto a la versión anterior”, se crea un objeto con cuatro campos en el caso de generar un Esquema Lógico (/generateData) y un objeto de seis para un Esquema físico (/generateDataScriptSQL) paso a detallar cada uno de ellos :

- data1- Información referente a las entidades, relaciones y atributos que no estén añadidas en ninguna agregación. No guarda las relaciones entre ellas solo guarda la información detallada de cada uno de ellos.
- data2- Información que nos indicará qué elementos están relacionados entre ellos, como puede ser un atributo con una entidad, una relación con una entidad o una relación IS-A con una entidad. Como se verá más adelante la relación IS-A se diferencia de una relación de cardinalidad, ya que se tratan de manera distinta en el controlador.
- data3- Las entidades, relaciones y atributos que están añadidas en una agregación se verán reflejadas en este subobjeto. Debemos diferenciar data1 y data3 para que en el controlador en el caso de existir una agregación se comporte de una manera distinta.
- data4- Información referente a las relaciones dentro de las agregaciones, es el símil del data2 pero para los elementos dentro de la agregación.
- data5- Etiqueta que indica el lenguaje seleccionado en el esquema físico.
- data6- Índice de lenguaje seleccionado en el esquema físico.

```
myObj["data1"] = JSON.stringify(resultNodes);
myObj["data2"] = JSON.stringify(edgesData);

//tomamos la data de la entidad relacion de alto nivel
myObj["data3"] = JSON.stringify(nodesSuper);
myObj["data4"] = JSON.stringify(edgesSuperData);
var json = JSON.stringify(myObj);
```

Figura 4.16 Objeto creado para esquema lógico (init.js).

```
myObj["data1"] = JSON.stringify(resultNodes);
myObj["data2"] = JSON.stringify(edgesData);
myObj["data3"] = JSON.stringify(nodesSuper);
myObj["data4"] = JSON.stringify(edgesSuperData);
myObj["data5"] = $("#selectLenguage option:selected").text();
myObj["data6"] = $("#selectLenguage option:selected").index();

var json = JSON.stringify(myObj);
```

Figura 4.17 Objeto creado para esquema físico (init.js).

Tenemos dos llamadas más al servidor desde el frontend, La primera de ellas es Test el cual envía toda la información solicitada al usuario en el formulario mostrado. Esta opción comprueba que la conexión con las credenciales insertadas es posible. El método encargado de hacer esta comprobación es checkConnection como se observa en la llamada ajax.

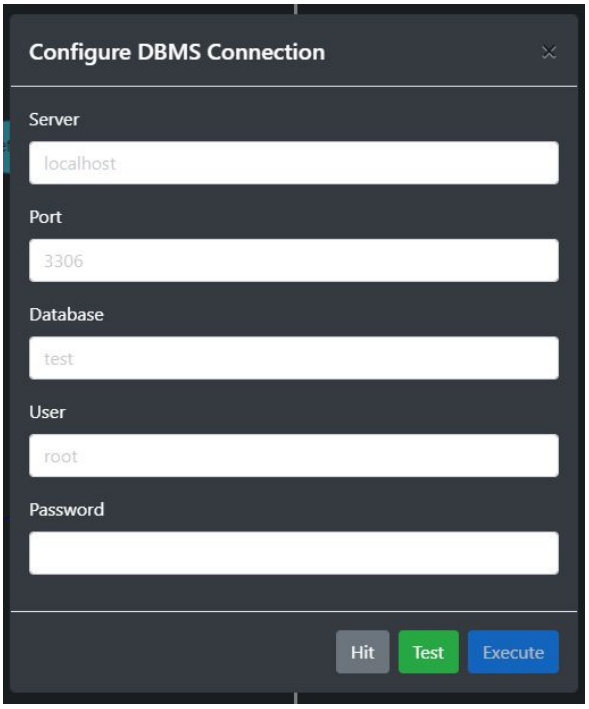

Figura 4.18 Diseño del cuadro de diálogo para realizar una conexión a una BBDD.

```
// tomamos la informacion de la conexión ingr
myObj["data1"] = $("#serverInput").val();
myObj["data2"] = $("#portInput").val();
myObj["data3"] = $("#databaseInput").val();
myObj["data4"] = $("#usernameInput").val();
myObj["data5"] = $("#passInput").val();
myObj["data6"] = $("#selectLenguage").val();
var json = JSON.stringify(myObj);
```

Figura 4.19 Almacenamiento de los datos en una variable JSON del formulario de conexión.

La última llamada incorporada en esta versión es la encargada de insertar el diagrama físico al destino. Un punto importante en esta funcionalidad es que se enviará el esquema físico mostrado en la vista nuevamente al controlador, como se observa en la siguiente imagen se añade esta información en un subobjeto llamado resultSPhysicalSchema.

```
myObj["data1"] = $("#serverInput").val();
myObj["data2"] = $("#portInput").val();
myObj["data3"] = $("#databaseInput").val();
myObj["data4"] = $("#usernameInput").val();
myObj["data5"] = $("#passInput").val();
myObj["data6"] = $("#selectLenguage").val();
myObj["data7"] = $("#resultSPhysicalSchema").html();
var json = JSON.stringify(myObj);
```

Figura 4.20 Almacenamiento de los datos con el esquema físico, para importar a la BBDD.

Backend

Se añadieron las siguientes clases en el paquete modelo, las cuales nos facilitaran el parseo de los objetos enviados desde la vista mediante Ajax:

- Node.java: esta clase se utiliza para el parseo de los elementos que llegan desde la vista, tanto para data1 como para data3. Esta clase tiene un atributo de tipo DataAttribute que será distinto a null si el elemento que se está parseando es un atributo, si fuera una entidad o relación el valor de este atributo sería null.
  Dependiendo del shape podemos detectar de qué tipo se trata, una entidad, atributo, relación is-a o relación de cardinalidad.
  Otro punto importante de esta clase es el atributo id_origin. Este atributo está creado para indicarnos el identificador de la entidad origen de un atributo si se encontrara dentro de una agregación. Ya que como veremos en la descripción del método /generateData, en el caso de que se trate de una agregación, esta se trata como una entidad normal para poder utilizar el campo id_origin y obtener la clave primaria, en el siguiente diagrama se muestra el funcionamiento de esta campo.

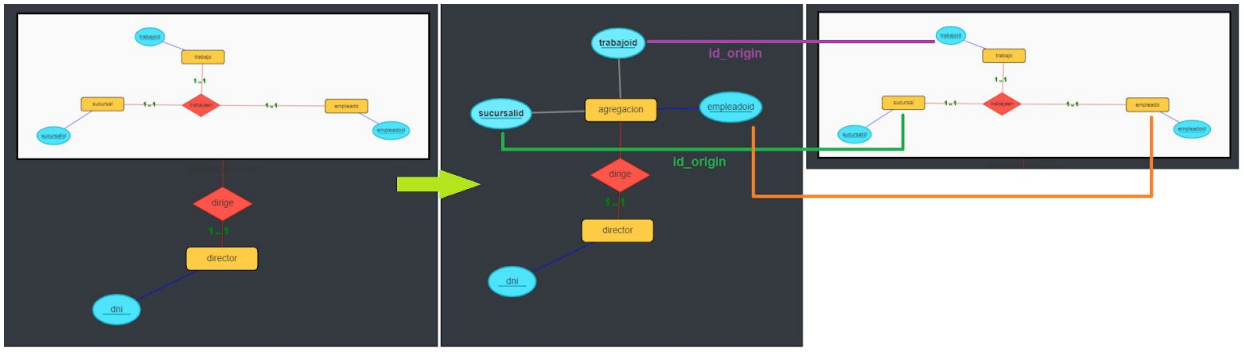

Figura 4.21 Flujo de traducción de un diagrama normal a uno con entidad de agregación.

- Edge.java: nos indica las relaciones entre elementos, esta clase tiene en los atributos from y to los ids de los elementos relacionados y en LabelFrom y LabelTo las cardinalidades si existieran de las relaciones.
- DataAttribute.java: información relacionada con los atributos tamaño, si se trata de una clave primaria, si es compuesta, si puede ser null, única y/o multivaluada.
- Generic.java: Esta clase consta de 7 atributos tipo String, esto la hace ser tan versátil por lo que la utilizamos como parámetro en los requestmapping. A continuación mediante la biblioteca Gson parseamos nuevamente el String alojado en alguno de los atributos de la clase Generic permitiéndonos el acceso a la información de una manera sencilla y limpia.

```
public String generateData(@RequestBody Generic r, HttpServletRequest req, HttpServletResponse resp)

    Gson gson = new Gson();
    //Node model = gson.fromJson(r.getData1(),Node.class);

    Type tipoNode = new TypeToken<List<Node>>(){}.getType();
    Type tipoEdge = new TypeToken<List<Edge>>(){}.getType();


    List<Node> nodes = new ArrayList<>();
    List<Edge> edges = new ArrayList<>();

    List<Node> nodesAltoNivel = gson.fromJson(r.getData3(), tipoNode);
    List<Edge> edgesAltoNivel = gson.fromJson(r.getData4(), tipoEdge);
```

Figura 4.22 Ejemplo de análisis (parsing) de datos.

- Funcionamiento de clases TransferEntidad.java, TransferRelacion.java, TransferAtributo.java: Estas clases tienen el objetivo de ser los objetos encargados de contener y transportar los datos entre capas de la aplicación, cada una relacionada respectivamente con su símil en el diagrama generado por el usuario .

Como se indica en el apartado frontend se ha creado un nuevo controlador (GoogleserviceApplication.java) que consta de cuatro funciones encargadas de gestionar la información enviada desde la vista mediante Ajax.

- Genera Esquema Lógico - /generateData
  Esta es la funcionalidad más compleja dentro de la aplicación, por tanto la descripción se realizará de forma descendiente entre las dependencias de funciones.
  Los atributos que se utilizan dentro del método son:
    - List<Node> nodes- elementos externos de la agregación.
    - List<Node> nodesAltoNivel- elementos internos de la agregación.
    - List<Edge> edges- relaciones entre elementos externos de la agregación.
    - List<Edge> edgesAltoNivel- relaciones entre elementos internos de la agregación.

Lo primero que realizamos es la carga de un HashMap con información de los elementos incluidos en la agregación, el cual tiene como clave el identificador del atributo y como valor el identificador del atributo, identificador de la entidad y por último el nombre de la entidad origen.

Este objeto se utilizará para que seguidamente a la ejecución del diagrama podamos generar una respuesta con sus respectivas entidades de origen, en la imagen del primer punto del apartado backend se puede ver gráficamente.

Procedemos a la creación de una entidad basada en la agregación, tomamos las claves primarias de esta agregación y creamos una entidad con ellas, además enlazamos la relación con la entidad creada.

Al necesitar tanto los elementos externos, como internos de la agregación si existiera. Procederemos a ejecutar el método generateEsquema(...), la cual tiene las siguientes tareas:

- Crear los transfers de entidad, atributos, relación y relación IS_A basados en los nodos que llegan por parámetros, este método comprueba el shape de cada uno de ellos y dependiendo de ello crea un objeto con las características necesarias para que el generador de esquemas puede interpretarlo correctamente.

- Enlazar los transfers basados en los edges que llegan por parámetro, teniendo en cuenta los dos extremos de una relación entre elementos, dependiendo del tipo de estos transfers se procederá a realizar el llamado al método que se ajuste a estas características. Ya que no se trata de la misma manera una relación entre una entidad y un atributo o una relación entre una entidad padre y un IS-A.
- Este método es el encargado de devolver un String con formato HTML, el cual se pintará en la parte del cliente mediante JQuery.

A continuación se procederá a la ejecución del método generaModeloRelacional_v3, al haber simulado casi completamente el funcionamiento de la versión anterior, en este punto la aplicación utiliza prácticamente todos los métodos de su predecesor, exceptuando algunas comprobaciones respecto a las agregaciones. Como podemos observar en la siguiente imagen condicionamos este caso para tomar la entidad origen, que es la que realmente necesitamos de no ser así se tomaría la entidad incorrecta.

```
for (int q=0; q<previasPrimarias.size(); q++){
    String[] clave = new String[5];
    clave[3]="0";
    clave[4]=eya.getPrincipioRango()==0?"0":"1";
    if (!eya.getRol().trim().equals("")) clave[0] = eya.getRol() + "_" + previasPrimarias.get(q)[0];
    else clave[0] = previasPrimarias.get(q)[0];
    clave[1] = previasPrimarias.get(q)[1];
    clave[2] = previasPrimarias.get(q)[2];
    primarias.add(clave);

    if(previasPrimarias.get(q)[2].equals("agregacion"))
        primariasEntidades.add(previasPrimarias.get(q)[3]);
    else
        primariasEntidades.add(previasPrimarias.get(q)[2]);

    referenciadas[q] = previasPrimarias.get(q)[0];
}
```

Figura 4.23 Condición para tomar la entidad origen en el caso de que sea una agregación.

Las restricciones que se deben comprobar en el controlador se modificaron cambiandoles el tipo de retorno para que sea tipo String de esta manera podemos lograr que esta información se muestre al usuario, manteniendo el flujo como el proyecto anterior.

- Genera Esquema Físico - /generateDataScriptSQL

Como se indicó en el punto anterior el flujo de este método es igual que /generateData. Con la diferencia que utiliza un método extra antes de finalizar su ejecución el cual basado en las tablas

creadas en el generador de esquemas, procede a crear las consultas teniendo en cuenta el transfer de conexión enviado por parámetro.

- Comprobar conexión al DBMS- /checkConnection
  Recibe como parámetro un tipo generic, en el cual tenemos todas las credenciales necesarias para realizar el intento de conexión de esta manera testeamos que el enlace con el DBMS es correcto.
  En el atributo data6 del parámetro r obtenemos el tipo, esto nos permite poder diferenciar el tipo de conexión que debemos testear , esto es importante tener en cuenta ya que dependiendo de esta conexión se concatena un encabezado diferente a URL de acceso.
  A continuación de la creación del transfer conexión procedemos a la llamada del método compruebaConexionNew(), la cual es una actualización de una versión anterior encargada de esta tarea, la ejecución de este método nos devuelve un array con dos posiciones las cuales nos indica, el éxito o fracaso del intento. Dependiendo de la respuesta recibida en la vista se activará o se bloqueará el botón Execute para proceder a la ejecución de las consultas en la ubicación testeada.

```
public String[] compruebaConexionNew(TransferConexion tc){
    System.out.println("Datos de conexion a la base de datos");
    System.out.println("-----------------------------------");
    System.out.println("DBMS: " + tc.getRuta() + "(" + tc.getTipoConexion() + ")");
    System.out.println("Usuario: " + tc.getUsuario());
    System.out.println("Intentando conectar...");
    ConectorDBMS conector = FactoriaConectores.obtenerConector(tc.getTipoConexion());
    try {
        conector.abrirConexion(tc.getRuta(), tc.getUsuario(), tc.getPassword());
        conector.cerrarConexion();
    } catch (SQLException e) {
        return new String[] {this.msgSrc.getMessage("textosId.error", null, this.loc)+".\n" +
            this.msgSrc.getMessage("textosId.noDBConexion", null, this.loc)+" \n" +
            this.msgSrc.getMessage("textosId.reason", null, this.loc)+": \n" + e.getMessage(),"0"};
    }
    return new String[] {this.msgSrc.getMessage("textosId.okScriptExecut", null, this.loc),"1"};
}
```

Figura 4.24 Retornos de comprobación de conexión con DBMS.

```
$.ajax({
    type: 'POST',
    url: '/checkConnection',
    data: json,
    contentType: "application/json",
    success: function (data) {
        var dataResponse = JSON.parse(data);
        if(dataResponse.data2 == true){
            alert(dataResponse.data1);
            $("#executeScriptSQL").removeClass("disabled");
        }
        else{
            alert(dataResponse.data1);
            $("#executeScriptSQL").addClass("disabled");
        }


    },
    error: function (xhr, ajaxOptions, thrownError) {
        console.log(xhr.status);
        console.log(xhr.responseText);
        console.log(thrownError);
    }
```

Figura 4.25 Código JQuery que indica que se haya realizado una conexión correcta o no.

- Añade consultas al DBMS - /executeQueries
  Realiza el mismo proceso que se describe en el punto 3, excepto por 2 métodos modificados para que sea posible acoplarlo al sistema web. La primera es getText(String sql), permite limpiar el String que llega desde la vista con la información de las consultas. La funcionalidad de este método nos permite quitar los títulos, espacios en blanco, comas, saltos de línea. El segundo cambio se dio en el método ejecutarScriptDBMS_new, esta en la encarga de la creación de las tablas en el DBMS. Las modificaciones en este método fueron respecto al parámetro de retorno ya que como en el anterior punto era tipo void pensado para un uso local de la aplicación.

5. Código

5.1. Refactorización

Cuando nos encontramos con este proyecto vimos que respecto a versiones anteriores se había optimizado mucho la utilización de dependencias y se había optimizado el código. Ya que se ha eliminado código redundante y se procedió añadir patrones para una mejor estructuración.

Sin embargo, con las tecnologías que existen actualmente, consideramos que era necesario realizar otra refactorización, la cual nos permita poder realizar mejor la implementación de la aplicación web.

- El proyecto estaba realizado en Java, es decir no utilizaba ningún framework que nos permita utilizar módulos o servicios que hagan más fácil el desarrollo y el despliegue de la aplicación.
- No utilizaba ningún repositorio para poder tratar las dependencias, en el caso de que se añada alguna librería se tendría que buscar el JAR y añadirla manualmente, haciendo que la aplicación ocupará más espacio, y su distribución sea más pesada.

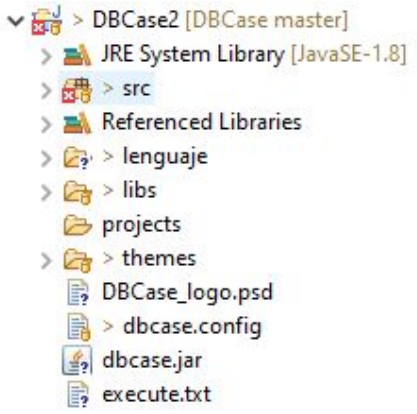

Figura 5.1 Estructura inicial del proyecto .

Lo primero que hicimos fue crear un proyecto desde cero con el framework Spring Boot que está orientado a aplicaciones web con Java creando aplicaciones rápidamente realizando pocas configuraciones previas.

- Añadimos el repositorio Maven, este repositorio utiliza un fichero pom.xml el cual contiene información sobre el proyecto y los detalles de configuración utilizados por Maven para construir el proyecto. Contiene valores predeterminados para la mayoría de los proyectos como los directorios de compilación, directorio de origen, directorio de origen de prueba, entre otros.
- A este fichero se añade unas estructuras XML la cual es interpretada por el repositorio Maven y descarga automáticamente cada dependencia.

```
<dependency>
    <groupId>org.webjars.bowergithub.eligrey</groupId>
    <artifactId>filesaver.js</artifactId>
    <version>683f689326</version>
</dependency>

<!-- https://mvnrepository.com/artifact/org.springframework.securi
<dependency>
    <groupId>org.springframework.security.oauth.boot</groupId>
    <artifactId>spring-security-oauth2-autoconfigure</artifactId>
    <version>2.2.2.RELEASE</version>
</dependency>
```

Figura 5.2 Fichero pom.xml que controla las dependencias del proyecto actual.

- Todas las dependencias que se utilizaban anteriormente de forma manual han sido añadidas a Maven para poder utilizar de forma activa el repositorio y a su vez se han añadido las demás librerías que hemos usado para la aplicación web como JQuery o Bootstrap 4.
- Se ha eliminado la carpeta Theme, ya que al ser aplicación web el funcionamiento ha cambiado.
- Se ha eliminado la carpeta Projects, ya que no almacenamos los ficheros generados por la aplicación.
- La carpeta Language se ha eliminado, ya que el funcionamiento ha cambiado.
- Una vez esto, se copió todo el código y se solucionaron algunos problemas de incompatibilidades lo cual hizo que todo funcionara correctamente con el framework.

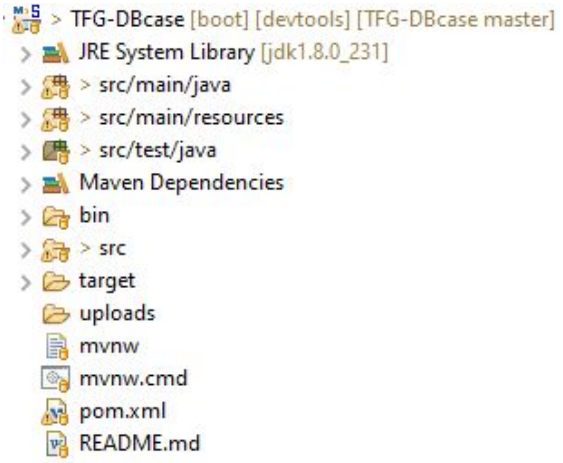

Figura 5.3 Estructura final como proyecto Maven y con el framework Spring Boot.

La nueva estructura contaba con las siguientes carpetas, describimos las más importantes:

- main/java: Donde se almacenaba toda la lógica, clases Java.
- main/resources: Se almacenan los elementos estáticos(plantillas html, css, ficheros de configuración, elementos multimedia, entre otros).
- Maven dependencies: Se almacenan todas las dependencias configuradas en el archivo pom.xml.

- target: Se almacenan los ficheros que se generan cada vez que se compila la aplicación.
- uploads: Se almacenan los ficheros que son subidos a la aplicación.

5.2. Repositorio

Para poder implementar la aplicación, y poder tener un control sobre cada versión del código entre los integrantes, decidimos utilizar GitHub.Esta herramienta con la que ambos teníamos experiencia y es fácil de preparar el entorno de desarrollo.

Todos los cambios que se realizaban se subían inmediatamente haciendo que los conflictos no surjan habitualmente.

https://github.com/luiseduardo123/TFG-DBcase

6. Resultados

Una vez realizada todas las funcionalidades y el nuevo diseño para la aplicación web se puede comprobar que es una aplicación fácil de usar, haciendo que la experiencia de usuario sea grata e intuitiva para la persona que utilice la aplicación,

El diseño que se ha creado con Bootstrap 4 permite que esta aplicación sea visualizada correctamente desde cualquier dispositivo, aunque debido a la forma en la que está implementado el diagrama, siempre haría falta el uso del ratón.

La fácil generación de diagramas y la amplia posibilidad de funcionalidades que se han implementado dentro del diagrama entidad relación satisfactoriamente, harán que los futuros alumnos puedan comprender el uso de las bases de datos de la mejor forma.

A continuación describimos brevemente el proceso de creación de un diagrama con las distintas opciones disponibles.

6.1 Creación de un diagrama

Una vez se ha iniciado sesión en el sistema el primer paso es la creación y el diseño de un diagrama entidad relación. Para ello se debe seleccionar la perspectiva que mejor se adapte al usuario entre las 6 disponibles en la aplicación.

Para poder añadir los elementos al diagrama de diseño el usuario dispone de dos opciones o formas de realizarlo:

- Haciendo clic derecho con el ratón sobre el panel de diseño. Se mostrará un menú desplegable el cual nos permite añadir cualquiera de los elementos disponibles.
- Con la barra lateral, pulsando en “Dibujar” y luego en los iconos que representan los elementos.

Cuando se desea insertar elementos, los datos a insertar se realizan mediante distintos cuadros de diálogo rellenando las características de los mismos. Por ejemplo si se quiere añadir una entidad se debe indicar el nombre, también se puede indicar opcionalmente si es una entidad débil con respecto a otra existente en el diagrama. En ese caso se debe seleccionar esa entidad fuerte e indicar el nombre de la relación que las une.

En el caso de que se quiera eliminar elementos del diagrama el procedimiento es el siguiente:

- Primero se debe seleccionar uno o más elementos (bien abarcando todos los elementos con el ratón haciendo clic en el botón  , o bien seleccionando de uno en uno y manteniendo pulsada la tecla ctrl).

Una vez seleccionado se puede eliminar los elementos de dos formas:

- Hacer clic derecho sobre el elemento a eliminar, tras lo que se mostrará un menú desplegable, por lo general la última opción permitirá eliminar un elemento.

- Pulsando la tecla de suprimir en el teclado.

Relaciones

Las relaciones son básicas para establecer la relación entre las entidades del diagrama.

Para poder asociar una relación con una entidad el usuario debe hacer clic derecho sobre la relación en cuestión y pulsar en la opción “añadir una entidad a la relación” del menú desplegable. Tras esto se abrirá un cuadro de diálogo en el que se especificará la entidad, se elegirá la asociación y cardinalidad si se desea establecer. También la aplicación permite indicar un nombre para el rol de la asociación.

Restricciones

El usuario si así lo considera necesario puede añadir restricciones a cualquier elemento del diagrama. Estas serán importantes a la hora generar el script SQL que se implementa en la base de datos.

Para añadir restricciones a los elementos primero el usuario debe hacer clic derecho sobre algún elemento existente en el diagrama y hacer clic en “restricciones” del menú desplegable.

Tras esa acción se mostrará un cuadro de diálogo que permite al usuario crear una lista de restricciones para ese elemento.

Dominios

La aplicación web ofrece once dominios que suelen ser los más usados pero también permite añadir al usuario nuevos dominios, si así lo considera.

Para poder crear un dominio se han habilitado dos formas de realizarlo:

- Pulsar clic derecho sobre un espacio libre en diagrama de diseño y pulsar en la opción “crear dominio” del menú desplegable.
- También en el apartado de Elementos/Dominios, hacer clic en dominios y luego en nuevo.

Al insertar un atributo es obligatorio indicar un dominio al atributo entre otros parámetros. Esta información es utilizada para generar el esquema físico, todas las columnas de una tabla deben tener un dominio.

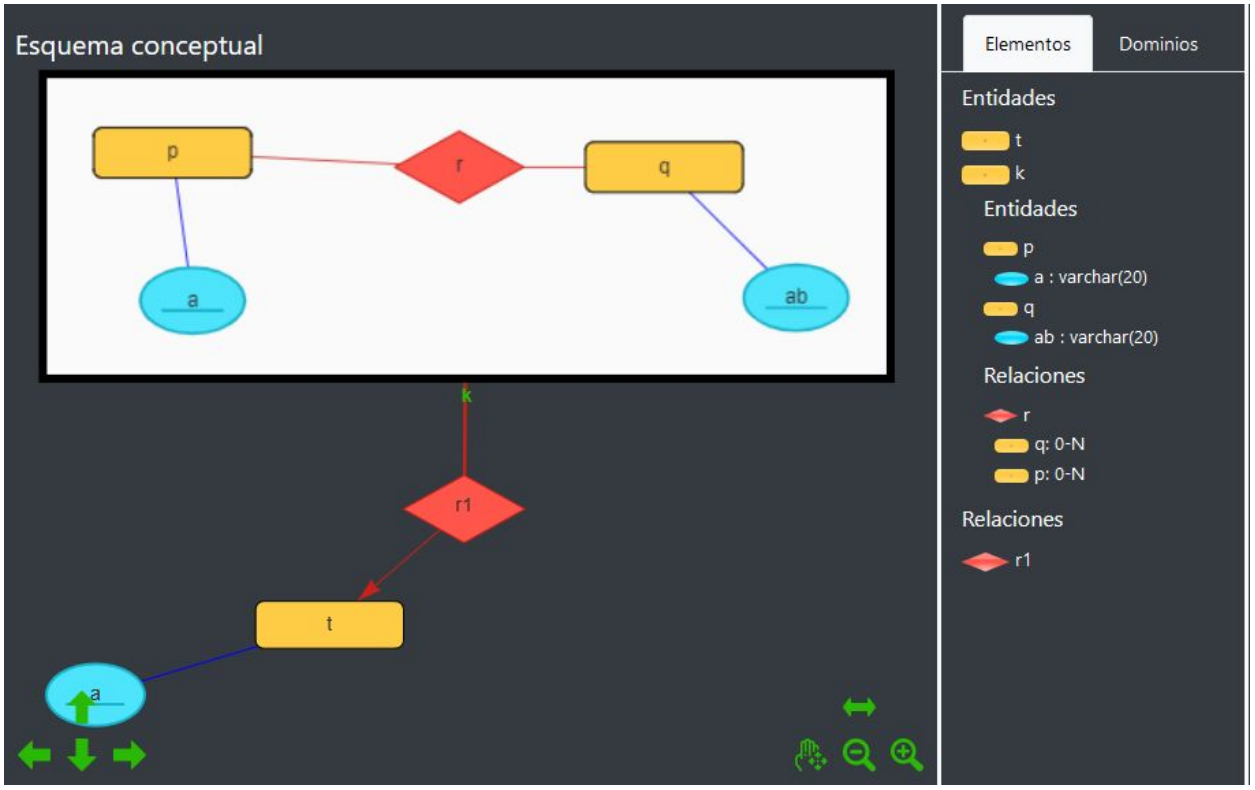

Figura 6.1 Diseño del mapa de diseño con la perspectiva esquema conceptual / Elementos-Dominios.

Generación de códigos

Una vez se ha realizado el diseño del diagrama, el proceso para continuar sería la generación de los esquemas lógico o físico.

Para esto el usuario debe pulsar en los botones de “Generar” dispuestos en cada panel. De producirse errores o advertencias en el diagrama, se mostrará de forma detallada cual es el error que surge.

Una vez generados los códigos, el usuario puede exportarlos como documentos de textos o sql para poder realizar cualquier acción que considere(añadir, modificar o eliminar).

O también la aplicación web tiene la posibilidad de conectarse a una base de datos y generar directamente el resultado obtenido en el diagrama.

Para esto solo se tiene que hacer clic en el botón “Ejecutar”, completar los datos de conexión, probar que la conexión se realizó correctamente y pulsar en “Ejecutar”.

DBCASE Web

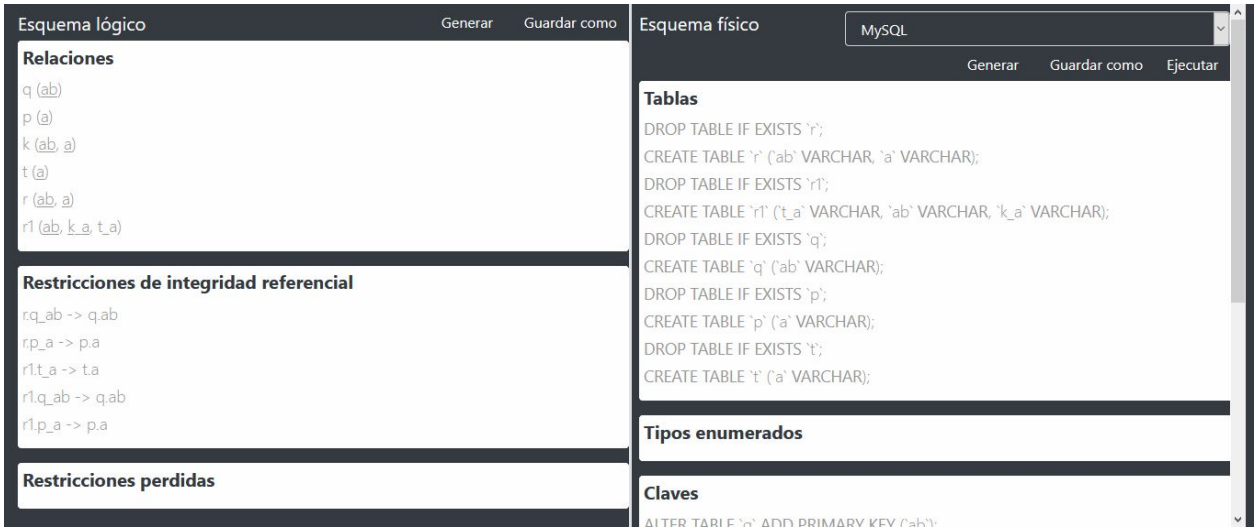

Figura 6.2 Diseño del mapa de diseño con la perspectiva esquema lógico / esquema físico.

7. Contribuciones individuales

Este proyecto fue idea de nuestro tutor Fernando Sáenz Pérez. Mi compañero y yo queríamos realizar un proyecto en programación web ya que era algo que nos apasionaba y ambos teníamos conocimientos en ello, ya que llevamos trabajando en este entorno durante varios meses.

Como ya hemos descrito anteriormente intentamos dividir el trabajo en tres fases, investigación y estudio del código, desarrollo de la aplicación web, y una fase de pruebas.

Podríamos decir que aunque se ha dividido en tres fases cada una de estas se ha realizado de forma constante durante todo el desarrollo del proyecto.

Muchas de estas acciones las realizamos juntos como la fase de investigación y estudio del código existente, así como también la fase de pruebas.La fase de desarrollo aunque teníamos una comunicación constante se ejecutó de una forma relativamente independiente.

Utilizamos Trello para poder dividir las historias de usuario en tareas de tal forma que queden tareas que afectarán sólo al backend y solo al frontend para que David hiciera las tareas que se realizaban en el backend de la aplicación y Luis desarrollara las tareas que se desarrollaran por lo general en el frontend de la aplicación.

LUIS EDUARDO PAUCAR CERRÓN

A la hora de comenzar este proyecto, el primer paso que se realizó fue reunirnos entre los integrantes del proyecto y el profesor para poder recabar información más a fondo sobre lo que exactamente se quería realizar.

Una vez aclarados los objetivos a realizar en el proyecto de forma detallada, empezamos con la fase de investigación y estudio. Para ello logramos contactar con el compañero que realizó este mismo proyecto en el año anterior, Miguel Arriba García para que nos pueda aportar algún tipo de información que nos pudiera servir a la hora de implementar el proyecto.

Con los aportes de Miguel durante unas semanas empezamos a ver el estado del proyecto, ver cómo estaba estructurado todo el código, las clases, las librerías y todos los aspectos relevantes de la aplicación ya existente.

Viendo el estado del proyecto y lo que debíamos hacer entre David y yo optamos por acoplar el código existente a un framework que nos ayudara a gestionar todo mejor y nos permitiera evitar configuraciones tediosas.

Una vez elegido Spring Boot como framework de desarrollo, ambos migramos el proyecto existente buscando la mejor forma de hacerlo e implementarlo.

Empezamos creando un proyecto Maven, donde una a una fuimos buscando cada dependencia y añadiendo al proyecto en el fichero pom.xml

Una vez construida la estructura del proyecto web ya podíamos empezar a implementar la nueva aplicación.

Creé un repositorio en Github para que podamos trabajar ambos, creé el tablero de Trello para poder realizar todas las historias de usuarios y llevar un control de las tareas.

Añadimos las historias de usuario y las dividimos en tareas para empezar con su implementación.

Volviendo al proyecto lo primero que hice fue añadir las librerías que sabía con anticipación que la aplicación web iba a tener (Bootstrap, JQuery, VisJs, entre otras).

Una vez hecho esto, realice el diseño de la portada de la aplicación, la pantalla de bienvenida, intentando hacerla sencilla pero a la vez adaptable a cualquier pantalla, y moderna.

Realicé la configuración desde la Consola de administración de Google para poder iniciar sesión en nuestra web con su servicio.

En la web instale el repositorio necesario y configuré la API de Google a la aplicación.

Procedí a instalar el repositorio que permite crear las traducciones, creando la clase de configuración y sus archivos donde se almacena cada idioma disponible.

Una vez hecho esto empecé con el diseño de toda la parte principal de la aplicación, el diseño de la página de realización del diagrama entidad relación, los paneles de esquema lógico y también la del esquema físico.

Empecé por el esquema conceptual ya que era la parte más larga y también la parte principal necesaria para poder empezar con las traducciones de los esquemas conceptuales a esquemas logico y fisico.

Empecé creando los archivos Javascript, diseñando cada elemento que existiría, modificando la librería para crear las aristas que no existían, y añadiendo dobles bordes a los elementos para poder utilizarlos en la aplicación, también se realizaron los menús desplegables que salen cuando se da clic derecho con todos los modales.

Una vez realizado el diseño del esquema conceptual desarrollé el diseño del panel de Elementos/Dominios, del panel de esquema lógico y también del esquema físico, los cuales una vez traducido en el backend, vuelven al frontend para mostrarlos en cada panel.

Una vez funcionando todo, se procedió al envío de un email al tutor del proyecto con los avances que se iban realizando, el cual nos respondía con un fichero de correcciones a realizar, donde todas las que tenían que ver con el frontend las iba corrigiendo.

Empecé el diseño del menú superior, implementando cada sección como las de descargar los ficheros y exportar como pdf, todas estas con librerías Javascript que añadimos al iniciar el proyecto.

Una vez terminado todo el proyecto o a falta de correcciones que el tutor nos indicaba empezamos con la redacción de esta memoria.

Para esto, entre David y yo decidimos dividirnos los distintos apartados, teniendo en cuenta que cada uno haría lo que había implementado. También decidimos realizar juntos algunas partes que considerábamos importantes como el resumen, la introducción y conclusión, entre otros.

Empecé redactando la parte de la implementación de todo el diseño explicando su realización y también adjuntando imágenes que permitieran visualizar de la mejor forma posible todo lo realizado.

Una vez acabado esto procedí a realizar los apartados de creación de un diagrama el cual pudiera servir como ejemplo y ayuda sobre como diseñar un diagrama, también explique los apartados de planificación del proyecto y el plan de trabajo.

YRVING DAVID CONDE CUBAS

Después de las reuniones con Miguel y debatir sobre la tecnología que utilizariamos en el proyecto, en la reunión previa al comienzo de la implementación y teniendo cuenta las habilidades y conocimientos de cada uno de nosotros, se decidió que Luis se encargaría de la parte frontend como detalla en el apartado anterior, mientras que la parte del backend sería mi responsabilidad.

Teniendo claro que la responsabilidad sobre la parte del backend del proyecto era mía, lo primero que realicé fue familiarizarme con el código predecesor de nuestro proyecto. Por tanto, realice pruebas, investigue las bibliotecas presentes en el proyecto, revisión de los parámetros enviados entre métodos, entre otras tareas para conseguir conocer el funcionamiento de la aplicación.

Una vez familiarizado con el proyecto de escritorio el siguiente paso fue intentar simular la ejecución de un diagrama mediante variables estáticas dentro del backend, esto se debió a que necesitábamos conocer la información necesaria para el correcto funcionamiento de los métodos encargados de la generación de los diagramas. Tras conseguir el éxito en estas pruebas procedí a montar la llamada Ajax con la información necesaria y parsear esta información en el controlador mediante las clases Node y Edge, apoyados por la clase Generic la cual nos permite mayor flexibilidad a la hora de enviar los datos mediante Ajax. Esto llevó a que se ajustará el controlador para que ahora ya no dependa de variables estáticas y tenga en cuenta la información que se recibe desde la vista.

Cuando conseguí tener la funcionalidad del esquema lógico, procedí a la investigación más a fondo de la generación del diagrama físico, esto me permitió encontrar el método que diferencian estos dos esquemas a nivel servidor. Realice las modificaciones oportunas para el correcto funcionamiento de esta funcionalidad, consiguiendo relativamente rápido los resultados esperados.

En este punto del proyecto, busqué solucionar errores existentes en el proyecto previo. Algunos de ellos eran referentes a las relaciones 1...n las cuales no se creaban correctamente, así como problemas con el formato utilizado para mostrar las restricciones de integridad. El problema indicado era que se concatena el nombre de la entidad al atributo y esto era incorrecto, estos fueron solventados en este punto.

Luego procedí a controlar la nueva funcionalidad añadida **la agregación**, esto me llevó a investigar sobre el funcionamiento de este nuevo elemento en el diagrama y realizar pruebas al respecto para encontrar la solución óptima y menos intrusiva con el código existente. Por tanto, decidimos crear una entidad intermedia como se explica en puntos anteriores. Esto funcionó en un primer momento, pero seguidamente a las pruebas se detectó problemas con las entidades de origen de los atributos externos a la agregación. Por tanto, tuve que diseñar una posible solución a este problema, lo cual me llevó a la creación de un nuevo atributo dentro de la clase Atributo.java este indica la entidad origen. Mediante este nuevo atributo creado y el HashMap que conserva la información de todos los elementos enviados desde la vista, utilizando como

clave el identificador pude encontrar una solución válida para que los resultados generados sean los correctos.

En este punto empecé con la realización de la funcionalidad de importación de las consultas generadas en el diagrama físico al DBMS. Por lo que me encargué de creación del modal así como de la llamada Ajax para la conexión con el controlador. Seguidamente, modifique los métodos en el servidor encargados de la importación al DBMS, siendo exitosas las pruebas posteriores.

Al haber finalizado las tareas indicadas anteriormente, procedí con la continuación de la memoria añadiendo información en diversos puntos, aunque centrándome en el punto referente al backend **4.5 Traducciones de código**.

8. Conclusiones y trabajo futuro

Hemos conseguido uno de los principales objetivos que se planteó al inicio del proyecto la reutilización de código de sus predecesores. Esto ha beneficiado enormemente la fase de pruebas ya que teniendo en cuenta que el backend de la aplicación se había testeado en las versiones anteriores, tuvimos muy pocos problemas en este aspecto. Esto no significa que no hubiera que solucionar problemas en este apartado, pero este objetivo cumplio las expectativas.

Llegar al punto de la reutilización del código no fue una tarea trivial, ya que retomar y continuar un proyecto previo del cual no has sido partícipe desde el inicio siempre es complejo. Nos facilitó el trabajo el hecho de que el código estuviera tan bien comentado y estructurado, permitiéndonos entender el flujo de la generación de los diagramas.

Tuvimos complicaciones varias respecto a bibliotecas, tanto a nivel usuario como a nivel servidor. A nivel usuario, como se indicó anteriormente se tuvieron complicaciones en seleccionar el API más adecuado para diseñar los diagramas, esto se debió a que ninguna de las API investigadas cubrían todos los requisitos iniciales por lo que llegamos a la conclusión de seleccionar la más cercana a ellos y modificar su código fuente. A nivel servidor, tuvimos inconvenientes con bibliotecas que estaban obsoletas, así como con la complejidad del código para la generación de los diagramas.

8.1 Futuras mejoras

Las ideas para ampliar y mejorar que se han ido generando a medida del avance del proyectos son las siguientes:

- **Deshacer-Rehacer (Ctrl-Z, Ctrl-Y).** Permitir al usuario poder moverse entre las diferentes acciones realizadas en la creación del diagrama.
- **Estadísticas de uso y acceso al sistema.** Añadir la capacidad de medir los accesos al sistema, así como las acciones realizadas, tipos de diagramas ejecutados, entre otras acciones medibles dentro de la aplicación.
- **Diseño basado en estadísticas.** Permitir el diseño basado en los diagramas que se han realizado con más frecuencia, el cual tendrá más prioridad en función del volumen de datos, las consultas y las frecuencias de acceso.
- **Simplificar y añadir la posibilidad de incorporar más de una agregación dentro del diagrama.** Modificar la lógica del proyecto para permitir que las relaciones no solo se puedan dar entre entidades y añadir la posibilidad de una agregaciones múltiples dentro del diagrama.
- **Constraint en MySQL.** Este lenguaje requiere añadir como CONSTRAINT una segunda restricción de integridad referencial. Hay que añadirla como CONSTRAINT y con nombre.
- **Funcionalidad selección múltiple o individual de elemento.** al dejar de arrastrar un grupo o elemento se debería mantener la selección, por si después de soltar el botón se vuelve a pulsar para arrastrar de nuevo.

9. Introduction

DBCASE Web continues the implementation of the DBCASE project whose development began in 2007, implementing new functionalities and changing the user interface with the objective to make it more user friendly and accessible via web.

The application has been developed for the students of the faculty that will allow them to better understand the operation of relational databases, without the need for any previous programming language knowledge, useful for students who are getting started with this subject .

Also, the tool is useful for users with extensive knowledge in database that will allow them to build a relational database easily and quickly.

The new version of the application is developed in Java programming language and structured based on the Spring Boot framework which allows, among other things, the ability to obtain web applications without complex configurations. The Maven repository manager has also been added, which allows better control of the existing dependencies in the project as well as adding new needed libraries.

10. Conclusions

One of the main objectives, that was raised at the beginning of the project, is the reuse of base code . This has greatly benefited the testing phase since considering that the application's backend had been tested in previous versions. This does not mean that there were no problems to be solved in that part, but this objective met expectations.

Reaching the code reuse point was not an easy task, since resuming and continuing a previous project in which you have not been a participant from the beginning is always complex. The fact that the code was so well commented and structured made our work easier, allowing us to understand the flow of the diagrams generation.

We had several complications regarding libraries, both at the user level and at the server level. At the user level, as indicated above, there were complications in selecting the most appropriate API to design the diagrams, this was due to the fact that none of the investigated APIs covered all the initial requirements, so we reached the conclusion of selecting the closest one to them and modify their source code. At the server level, we had problems with libraries that were obsolete, as well as with the complexity of the code for generating the diagrams.

Bibliografía

Enlaces y recursos útiles mencionados a lo largo de toda esta memoria.

[1] Documentación de Spring Boot

https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/

[2] Documentación y ejemplos de HTML, JQuery, CSS, Javascript

https://www.w3schools.com/

[3] Documentación de librería Vis Js

https://visjs.github.io/vis-network/docs/network/

[4] Documentación de JQuery

https://api.jquery.com/

[5] Documentación de JsPDF

http://www.rotisedapsales.com/snr/cloud_staging/website/jsPDF-master/docs/jspdf.js.html

[6] Documentación de librerías Java Swing

https://docs.oracle.com/javase/8/docs/api/index.html

[7] Documentación librería Gson

https://www.javadoc.io/doc/com.google.code.gson/gson/2.8.0/com/google/gson/Gson.html

[7] Documentación de Bootstrap 4

https://getbootstrap.com/docs/4.1/getting-started/introduction/

[8] Memorias antiguas versiones DBCASE

Alberto Milán Gutierrez, Miguel Martinez Segura, Francisco Javier Cáceres González, Rodrigo Denis Cepeda Mateos, Cristina Marco de Francisco, Tello Serrano Gordillo, Miguel Arriba García

[9] Apuntes de Bases de Datos

Javier Arrollo, Yolanda García, Virginia Francisco Gilmartín, Iván Martínez, Fernando Sáenz