CAMPUS
DE EXCELENCIA
INTERNACIONAL

Graduado en Ingeniería Informática

Universidad Politécnica de Madrid

Escuela Técnica Superior de Ingenieros Informáticos

TRABAJO FIN DE GRADO

Diseño y desarrollo de una plataforma web para la consulta, visualización y gestión de un repositorio semántico de interacciones planta-patógeno.

Autor: Roberto García Salgado

Director: Alejandro Rodríguez González

MADRID, JUNIO 2016

Agradecimientos:

En primer lugar, quiero agradecer a mi tutor el Sr. Alejandro Rodríguez González por haberme dado la oportunidad de realizar este Trabajo de fin de grado, aportándome siempre sus consejos y guiándome en todo el proceso de realización de la plataforma.

En segundo lugar, me gustaría agradecer al compañero Juan Camilo Mesa Polo, que ha realizado el Trabajo de fin de Master, paralelo a éste y que me ayudó enormemente en todo lo relativo a las consultas SPARQL.

En último lugar, quiero agradecer a mis compañeros de trabajo por haberme dado grandes consejos a la hora de la realización de los primeros pasos del proyecto, y de esta manera facilitar el proceso de creación.

1-ABSTRACT

The main objective of this project is the search and data display support with a web platform that consume data of a semantic repository through SPARQL queries. This platform have to be able to search data, and then show and make new interactions in the platform. So in this work we have been developed and desing of the web platform, more concretely the display of the interactions plant-pathogen of the sematic repository.
In this way the platform designed, must consume data repository through SPARQL queries, and later make a graphic display data module. This display will be performed with a graphic based in nodules, which must be attractive, intuitive and easy for the user.

About the development, we have developed a series of queries to get necesary data for display. For this we used C# language and a specialized library named dotNetRDF. For the webside itself, we used the standard HTML5 whit CSS3 as a style sheet. Finally for the necesary graphic representation we used a ibrary GoJS based in JavaScript which is responsable of show the nodules tree, and the interactions between represented data and user.

2-RESUMEN

El objetivo principal de este proyecto es el de dar soporte a la búsqueda y visualización de datos, mediante una plataforma web, que consuma los datos de un repositorio semántico, mediante consultas SPARQL. Dicha plataforma, por lo tanto, debe ser capaz de realizar búsquedas de datos, visualizarlos e introducir nuevas interacciones en la plataforma. De este modo, en este trabajo de fin de grado, se ha llevado a cabo el diseño y desarrollo de la plataforma web, más concretamente de la visualización del repositorio semántico de interacciones planta-patógeno.

De esta manera, la plataforma diseñada deberá consumir los datos de dicho repositorio mediante consultas SPARQL específicas, para posteriormente generar un módulo de visualización gráfica de dichos datos. Esta visualización se realizará mediante un gráfico basado en nodos, el cual debe ser lo más atractivo, intuitivo y sencillo posible para el usuario final que utilice la plataforma.

En cuanto al desarrollo propiamente dicho, encontramos que para la implementación de la lógica interna se han desarrollado una serie de consultas para la obtención de los datos necesarios para la visualización, para lo cual se ha utilizado el lenguaje de programación C# con una librería especializada para consultas SPARQL denominada dotNetRDF. Para la implementación de la página web propiamente dicha, se ha utilizado el estándar HTML5 junto con CSS3 como hoja de estilos. Por último, para llevar a cabo la representación gráfica necesaria, se ha utilizado una librería GoJS basada en JavaScript, la cual se encarga de la representación del árbol de nodos, y de las interacciones entre el usuario y los datos representados.

Índice

Índice de figuras

Índice de tablas

1 INTRODUCCIÓN Y OBJETIVOS

PHI-Base [1] es una base de datos XML (eXtensible Markup Language) desarrollada por Rothamsted Research, un instituto de investigación agrario situado en Reino Unido. Durante la realización del presente trabajo de fin de grado (TFG), se llevará a cabo la utilización de un repositorio semántico, desarrollado por investigadores del CBGP (Centro de Biotecnología y Genómica de Plantas) e investigadores de la ETSIINF-UPM basado en PHI-Base. Este repositorio genera tripletas RDF (Resource Description Framework) las cuales son desplegadas sobre este repositorio semántico virtuoso [2].

El objetivo principal del TFG, será la realización de una plataforma web. Dicha plataforma debe de ser capaz de consumir datos, obtenidos del repositorio nombrado anteriormente, de este modo debe realizar una representación gráfica y visual, de estos datos que se van obteniendo.

Para llevar a cabo estos objetivos, el primer paso es la obtención de los datos, para lo cual se van a llevar a cabo una serie de consultas SPARQL (SPARQL Protocol and RDF Query Language) al repositorio semántico. De esta forma se va a ir organizando y guardando en formato JSON (JavaScript Object Notation), la información obtenida mediante las consultas para su posterior utilización.

Por otro lado, ha sido necesaria la implementación de una página web, basada en HTML y CSS. Esta página web, contendrá además de los elementos básicos de cualquier página, como son una cabecera, que muestre la información básica, la propia representación gráfica, la cual será la base donde se llevará a cabo tanto la propia representación de los datos obtenidos, como todas las interacciones que realice el usuario final con dicha plataforma.

Como se ha dicho anteriormente, los datos obtenidos se guardan y se organizan en JSON, el formato seguido para llevar a cabo la organización de los datos. Está basada primero en la división de estos datos en nodos, dichos nodos representarán cada elemento que se va obteniendo del esquema general del repositorio semántico el cual se puede consultar en ANEXO-5. De cada nodo en cuestión se irán obteniendo primero los datos propios del nodo, posteriormente los datos adicionales que pudiese tener dicho nodo, y por último las relaciones que tiene el nodo con los demás nodos.

Una vez obtenidos los nodos, los datos se deben ir organizando además de en un formato JSON, de una manera determinada, para que dichos datos se puedan leer desde la librería encargada de la representación GoJS, de la cual se ampliará su información más adelante. Dicha librería se encargará tanto de la representación de los datos, como de todas las interacciones que se pueden realizar con estos, tanto los movimientos entre los distintos nodos, como la obtención de toda la información que el usuario final requiera.

Debido a toda la terminología concreta que se va a utilizar durante todo el proyecto, se ha decidido que antes de comenzar con el contexto del proyecto y los objetivos, es necesario

aclarar algunos de los conceptos clave para poder abordar con garantías todo el ámbito del proyecto.

Web semántica:

La web semántica es una extensión de World Wide Web (www), a la que se le otorga un significado semántico a la información y a sus servicios, lo que nos permite entender y satisfacer ciertas peticiones, ya sean realizadas por personas, o por máquinas que utilicen este contenido web.

El término en concreto, fue utilizado por primera vez en el año 2000 para referirse a esta extensión a la que se hace referencia anteriormente. Desde entonces, se han llevado a cabo una gran cantidad de investigaciones sobre el tema, con un gran número de resultados importantes en el campo.

Los elementos básicos de los que se nutre la web semántica, son las ontologías y anotaciones. En cuanto a las primeras, normalmente se representan con lenguajes como OWL (Ontology Web Laguage) o RDF, que describen formalmente los conceptos de las ontologías y se comparten por medio de un dominio. Por otro lado, las anotaciones, se describen normalmente por RDF, y nos permiten describir instancias de las propias ontologías y de esta manera poder asociarlas a un recurso Web.

En la actualidad se ha desarrollado un nuevo concepto, de datos enlazados, el cual tiene similitudes con los mecanismos utilizados para la creación de datos en RDF públicos en los que se utilizan protocolos HTTP (Hypertext Transfer Protocol) [3].

Repositorio Semántico Virtuoso:

Virtuoso es un motor de base de datos híbrido capaz de combinar las funcionales de las bases de datos tradicionales, con las bases de datos virtuales como XML, texto libre, RDF, servidor de aplicaciones web o funcionalidades de servidor de archivos en un único sistema.

Virtuoso no posee servidores dedicados para las funcionalidades anteriormente nombradas, sino que se trata de un servidor universal el cual es capaz de habilitar un proceso sobre un hilo para los diferentes servicios, por lo tanto es multi-hilo. Además de esto existe una caché compartida entre las diferentes bases de datos. Virtuoso es por lo tanto multiplataforma, multi-hilo, y además está disponible en la GPL de código abierto con licencia, aunque es preciso pagar las versiones de código cerrado [4].

RDF y Consultas SPARQL:

Con RDF nos encontramos con un formato de datos para grafos dirigidos y etiquetados, el cual permite representar la información en la web. En cuanto a las consultas ineludibles para la obtención de toda la información necesaria para el funcionamiento de la plataforma web, se van a llevar a cabo consultas basadas en SPARQl a un repositorio

RDF [5]. El lenguaje de consulta SPARQL, es un lenguaje estandarizado para la realización de consultas sobre RDF, de tal manera que cumpla los casos de uso y las necesidades específicas de acceso a datos RDF. Es una tecnología clave en el desarrollo de la web semántica que se constituyó como recomendación oficial del W3C el 15 de enero de 2008 [6].

SPARQL cuenta con las siguientes especificaciones básicas:

- La especificación de protocolo SPARQL para RDF, se define como un protocolo remoto para enviar las correspondientes consultas SPARQL y recibir los resultados.
- Las especificaciones de los resultados de las consultas SPARQL en formato XML se definen como un documento para representar resultados de consultas SELECT y ASK de SPARQL.
- Al igual que SQL, se debe diferenciar entre el lenguaje de consultas y el motor de almacenamiento y de recuperación de datos.
- Aunque en principio SPARQL solo incorporaba funciones para sentencias RDF, actualmente existen diferentes propuestas para incluir operaciones en otro tipo de datos.

1.1 Contexto del Proyecto.

En cuanto al contexto de este proyecto, se desarrolló debido a la necesidad de crear una plataforma, que en primer lugar, fuera capaz de ser consultada por cualquier persona, pero más concretamente por personas relacionadas con el campo de la biología y las ciencias biológicas.

Debido a esta necesidad, se requerían dos plataformas paralelas; una que se encargase de realizar las consultas propiamente dichas al repositorio semántico virtuoso de una forma interactiva mediante diferentes filtros, la cual se ha desarrollado por el compañero Juan Camilo Mesa Polo, como proyecto de fin de Máster. La segunda plataforma, a la que atañe este proyecto, se encarga de la visualización de estos resultados, provenientes de la plataforma anterior, de tal manera que se puedan observar de la forma más clara, y precisa.

Por lo tanto, el contexto en el que se diseña esta plataforma, atiende a satisfacer la necesidad de seguir un diseño simple y sobre todo visual, motivo por el cual, se decidió diseñarla mediante un sistema basado en nodos representativos de la información obtenida del repositorio semántico, de tal forma, que el usuario final (biólogo/a), fuera navegando a través de estos nodos, y pudiese tanto interactuar con éstos, como obtener toda la información necesaria de cada nodo.

1.2 Objetivo del Proyecto.

En cuanto al objetivo principal del trabajo consiste en llevar a cabo el diseño y desarrollo de una plataforma web, la cual debe ser capaz de consumir los datos de un repositorio, desarrollado como se ha mencionado previamente por investigadores del CBGP (Centro de Biotecnología y Genómica de Plantas) e investigadores de la ETSIINF-UPM.

Dicha plataforma web, debe ser capaz de consumir los datos mediante consultas SPARQL, y sobre todo centrarse en mostrar estos datos obtenidos mediante la consulta, en forma de nodos, como se ha dicho anteriormente. Por lo tanto, el trabajo se va a centrar en cómo se van a recibir estos datos por parte de las consultas, y sobre todo de qué manera se van a representar éstos, ya que preferiblemente se deben mostrar en forma de grafos interactivos, para que el usuario final sea capaz de navegar por éstos.
Por consiguiente, dicha plataforma web debe de ser muy visual e intuitiva, debido a que se pretende usar en múltiples ámbitos y por diferentes personas con distintos niveles de conocimientos. Por este motivo se va a llevar a cabo la implementación de un manual de usuario, además de una sección de ayuda dentro de la propia plataforma web además de implementarse también su propio manual de usuario, para asegurarse de su buen uso.

Además de esto, se pretenden implementar funcionalidades extras, de tal manera que sea posible mostrar la máxima información de los datos obtenidos, así como añadir enlaces que posibiliten ampliar esta información y realizar nuevas búsquedas.

Lista de objetivos concretos:

- Generar una plataforma web visual y que cumpla con los requisitos y especificaciones dados.

- Generar un módulo de visualización gráfica para los datos del repositorio, intuitivo y visual.

- Generar un módulo de búsquedas dinámicas.

- Generar una integración y paso de información entre la plataforma de búsquedas y la plataforma de visualización.

- Generar un módulo de administración de toda la plataforma, para tener un registro de todas las acciones que se realicen dentro de la propia plataforma.

- Generar un manual de usuario, a modo de guía para el usuario.
- Generar un manual del desarrollador, para poder desplegar la plataforma posteriormente.

2 DESARROLLO

2.1 TRABAJOS PREVIOS A LA MEMORIA

En este apartado se pretenden destacar, aquellos conceptos que se han ido adquiriendo a lo largo del grado, y el modo en que nos han ayudado a la realización de este proyecto.

En primer lugar, se quieren destacar algunas de las asignaturas que se consideran más relevantes, en relación con este proyecto concreto:

Diseño de aplicaciones web:

Esta asignatura optativa del cuarto curso se cursó debido al interés que se tenía en todo lo relacionado con el diseño web, y como complementación a los estudios ya realizados, ya que durante la realización del grado, no se incluye ningún tipo de asignatura que trate esta rama tan demandada actualmente.

Algunas similitudes que se pueden encontrar entre este proyecto y la asignatura en cuestión son las siguientes:

- Se obtuvieron conocimientos de HTML5 Y CSS: conocimientos básicos de estas dos tecnologías básicas en toda aplicación web.

- Además también se adquirieron conocimientos sobre JavaScrip: primeramente temas relacionados con JavaScript, para posteriormente entrar en más profundidad en ciertas librerías de JavaScript, como por ejemplo JQuery.

Ingeniería del software I e ingeniería del software II:

Estas asignaturas obligatorias del grado, se cursaron, la primera parte de la asignatura en el tercer curso, y la segunda parte en el cuarto curso.
En cuanto a ambas, se quieren destacar algunos conceptos que se han considerado relevantes a lo largo de la realización del proyecto:

- La gran importancia de planificar todas las acciones y los pasos que se van a ir siguiendo en todas las etapas del proyecto. Además de todos los conocimientos que se obtuvieron en referencia a la planificación.

- Los conocimientos adquiridos para el desarrollo de todas las etapas y necesidades a lo largo de un proyecto, como pueden ser la ingeniería de requisitos y la definición de éstos, los diagramas de casos de uso, los ciclos de vida de un proyecto o las normativas.

- La importancia de planificar y ejecutar las pruebas necesarias para que el proyecto sea capaz no solo de cumplir con los requisitos y especificaciones que se habían marcado, sino además de cumplir ciertos estándares de calidad.

- Adquisición de conceptos a la hora de la organización de toda la documentación del proyecto, lo cual ha ayudado enormemente a la hora de redactar la presente memoria.

Finalmente también se ha querido destacar, una asignatura muy importante a lo largo de la realización del grado, como es el practicum.

Trabajo y actividades realizadas en el practicum:

En primer lugar, y por ser el evento más reciente, se quieren destacar aquellos conocimientos que se adquirieron durante la realización del practicum, ya que se desarrolló dentro de un equipo en un proyecto enmarcado en digital, basado en el mantenimiento y mejora de una página web de una destacable cadena hotelera española.

De este modo se pueden señalar varias similitudes entre el TFG y este trabajo realizado:

- La utilización de Visual Studio: El entorno de trabajo empresarial para todo el mantenimiento de la Pagina web se llevaba a cabo en Visual Studio, conectándolo a un TFS (Team fundation Server) de Microsoft.

- La utilización de Html5 y Css3 en toda la página web, lo cual proporcionó unos conocimientos básicos respecto a estas dos tecnologías.

- La lógica de la página web estaba basada en C#, con lo que se adquirieron conocimientos básicos de este lenguaje de programación.

El trabajo realizado posteriormente al practicum:

Posteriormente a la realización del practicum, se ha seguido trabajando en el mismo proyecto hasta la actualidad, ya que el equipo en que estaba integrado, se ha seguido dedicando enteramente al desarrollo de una nueva versión de la web para esta cadena hotelera, aunque con ciertas variaciones.
Este nuevo proyecto ha proporcionado muchos conocimientos, sobre todo a la hora de comenzar un trabajo de tanta envergadura desde cero, y permitiendo conocer cómo progresivamente se van realizando las diversas fases dentro de éste.

2.2. TECNOLOGIAS UTILIZADAS EN EL PROYECTO.

En este apartado se abordarán todas las tecnologías utilizadas a lo largo de la realización del proyecto. Se destacarán tanto los lenguajes de programación utilizados, como las

librerías de apoyo. En este punto es importante destacar que la memoria de este proyecto no tiene vocación de enseñar al lector de dicha memoria, ningún lenguaje de programación, ya que existen libros más específicos que tratan estos temas, pero sí se pretende dar una aproximación a cada uno de ellos, para que de esta manera se vea justificado el uso de cada uno de estos lenguajes.

2.2.1 JavaScript:

JavaScrip (abreviatura tradicional "JS") [7] es un lenguaje de programación ligero e interpretado, el cual está orientado a objetos, aunque soporta estilos de programación funcional e imperativa. Nos encontramos frente a un lenguaje script multi-paradigma, y muy dinámico. Se utiliza principalmente para la creación de páginas web dinámicas, éstas son aquellas en las cuales incorporamos efectos, como pueden ser textos que aparecen o desaparecen, animaciones variadas, o acciones que realiza el usuario interaccionando con botones o ventanas.

En cuanto al aspecto de que JavaScript, es un lenguaje de programación interpretado, nos referimos al hecho de que no es necesario compilar los programas para que estos sean ejecutados. Por lo tanto, los programas que se escriban con JavaScript se pueden probar directamente en el navegador que se prefiera.

A pesar del nombre, JavaScript no tiene ninguna relación con el lenguaje de programación Java, además de ser una marca registrada de la empresa Sun Microsystems. El estándar de JavaScript es ECMAScript. Desde el año 2012, todos los navegadores modernos soportan completamente ECMAScript 5.1, sin embargo, los navegadores más antiguos soportan como máximo ECMAScript 3. Actualmente nos encontramos frente a la sexta revisión de javaScript.

Características Principales:

En este apartado se van a resumir algunas de las características principales de JavaScript. Para comenzar, se va a tratar la manera de incorporar el código JavaScript en las aplicaciones web. Esto se puede realizar de dos maneras distintas:

- Encerrando el código JavaScript entre etiquetas <script> e incorporándolas en cualquier parte del documento en cuestión, aunque siempre se recomienda incluirlo dentro de la cabecera del documento para que vaya contenido dentro de la etiqueta <head>.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DT
D/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Ejemplo de código JavaScript en el propio documento</title>
<script type="text/javascript" src="/js/codigo.js"></script>
</head>

<body>
<p>Un párrafo de texto.</p>
</body>
</html>
```

Figura 1: Código JavaScript [20]

Para que el resultado de la página sea válido, es necesario añadirle el atributo type (con el valor "text/javascript") a la etiqueta <script>, ya que los valores de este atributo están estandarizados.

El principal inconveniente de esta manera de incorporar código, es que si se quiere llevar a cabo una modificación en el código, se tiene que llevar a cabo en todas las páginas que lo incluyan.

- El otro método es definirlo en un archivo externo. Para ello se puede incluir este archivo externo de tipo JavaScript que los documentos de la página enlazarán mediante la etiqueta <script>, de esta manera se pueden crear tantos archivos de este tipo como se requieran.

En este caso además del atributo “type”, este método requiere definir también el atributo “src”, en el cual se indica la url correspondiente a la localización del atributo JavaScript correspondiente. Cada etiqueta puede enlazar solo un archivo, pero en una misma página se pueden añadir tantas etiquetas como se quiera.

La principal ventaja de esta manera de incorporar JavaScript es que simplifica el código de la propia página, además permite reutilizar código, y solventa el principal problema del método anterior ya que al realizar cualquier modificación se aplica a todas las paginas donde esté ese script en ejecución.

- Para finalizar, la manera menos utilizada para incluir JavaScript, consiste en incluir trozos de JavaScript dentro del código de la página.

  El mayor inconveniente de este método, es que ensucia el código de la propia página, y complica mucho el mantenimiento del propio código JavaScript. Solo se suele utilizar para definir eventos y casos excepcionales.

En cuanto a la sintaxis propiamente dicha de JavaScript, podemos decir que es muy similar a otros lenguajes de programación como Java o C. A continuación se van a destacar los aspectos más relevantes de ésta:

- No se tienen en cuenta los espacios en blanco y las nuevas líneas (similar a HTML), estos espacios en blanco son ignorados por el intérprete. Esto facilita el entendimiento al poder tabular el código perfectamente.
- Distingue entre mayúsculas y minúsculas (similar a HTML aunque con alguna diferencia), aunque importa el orden entre ambas.
- Una de las características más destacadas, es a la hora de la declaración de variables, ya que no es necesario indicar el tipo de datos que se va a almacenar. De esta manera una misma variable es capaz de almacenar diferentes tipos de datos durante la ejecución.
- No es necesario acabar cada sentencia con punto y coma (";"), como en otros lenguajes de programación, aunque sí se recomienda realizarlo por no perder la tradición o práctica.
- Proporciona la capacidad de incluir comentarios, que añaden información adicional al código. Aunque hay que extremar las precauciones, ya que estos comentarios aunque no se visualicen, sí se envían al servidor del usuario.

En cuanto a las grandes posibilidades ya mencionadas de JavaScript, hay que destacar que desde la aparición de Flash disminuyó su popularidad, ya que éste permitía realizar algunas acciones imposibles con JavaScript. Sin embargo, tras la llegada de AJAX, JavaScript volvió a tener toda la popularidad perdida.

Pese a su popularidad JavaScript también tiene algunas limitaciones, ya que fue diseñado para ser ejecutado en un entorno muy limitado que permitiera a la máquina del usuario confiar en los script, por lo tanto éstos tienen la limitación de no poder comunicarse con recursos externos al mismo dominio. Este aspecto limita mucho las acciones realizadas por JavaScript dentro del navegador. Además de esto, los scripts no pueden acceder a los archivos de la máquina del usuario, y por si fuera poco si JavaScript tiene algún error o su ejecución dura demasiado, los navegadores informan al usuario de que el script en concreto está consumiendo muchos recursos, y por lo tanto puede llevar a una denegación por parte del usuario hacia ese script.

Actualmente se han solventado algunas de estas limitaciones, firmando digitalmente el script y de esta manera solicitando al usuario el permiso para realizar ciertas acciones.

2.2.2 HTML

Nos encontramos frente al lenguaje HTML [8], el concreto se trata de la quinta revisión de este lenguaje básico de la World Wide Web (www), lo que conlleva la incorporación de nuevos elementos, atributos y comportamientos.

En esta revisión se incorporan un conjunto de tecnologías más amplias, lo que permite dotar a los sitios web y a las aplicaciones de una gran variedad y alcance.

Se diseñó para poder ser utilizado en todos los entornos de Open Web, y contiene numerosos recursos que se clasifican en diferentes grupos: semántica, conectividad, sin conexión y almacenamiento, multimedia, gráficos y efectos 2D/3D, rendimiento e integración o acceso a dispositivos.

Características Principales.

En un principio las paginas HTML solo incluían información sobre sus diferentes contenidos, ya sean textos o imágenes. Pero con el paso del tiempo y el desarrollo en el estándar HTML, se empezaron a incluir estilos al contenido (letra, colores, márgenes...). Debido a la aparición de los scripts, existía la creciente necesidad de separar los contenidos y el diseño de la página web. Por este motivo, en este punto aparece un mecanismo importante en HTML como es el mecanismo de CSS, el cual permite separar los contenidos como se tratará más adelante.

En cuanto al lenguaje en si HTML, como se ha expresado anteriormente, primeramente se planteó el problema de almacenar información en archivos digitales, con los primeros textos con formato. Para ello se realizaba una conversión de letras en números (ASCII). Posteriormente se tuvo que resolver el problema de almacenar el texto con formato, la solución frente a esta problemática, fue utilizar el propio archivo electrónico para que fuese el encargado de almacenar la información sobre el formato de los contenidos.

Uno de los retos iniciales a los que se tuvo que enfrentar la informática fue el de cómo almacenar la información en los archivos digitales. Como los primeros archivos sólo contenían texto sin formato, la solución utilizada era muy sencilla: se codificaban las letras del alfabeto y se transformaban en números.

De esta forma, para almacenar un contenido de texto en un archivo electrónico, se utiliza una tabla de conversión que transforma cada carácter en un número. Una vez almacenada la secuencia de números, el contenido del archivo se puede recuperar realizando el proceso inverso. Este proceso para indicar las diferentes partes que componen la información se realiza por medio de etiquetas.

Para realizar las etiquetas:

- Se utiliza la etiqueta de apertura con el carácter "<", seguido del nombre de la etiqueta sin espacios en blanco y terminando con el carácter ">".

- Para el cierre de la etiqueta se utiliza el carácter "<", seguido de "/", y seguido del nombre de la etiqueta, para finalizar con el carácter ">".

Ejemplo:

<nombre_etiqueta> ... </nombre_etiqueta>

Figura 2: Etiquetas básicas HTML [28]

Por consiguiente HTML es considerado un lenguaje de etiquetas o marcado, y las páginas web actuales se forman con cientos o incluso miles de estas etiquetas. Esto facilita mucho tanto su lectura como su escritura, aunque aumentando mucho el tamaño del documento.

Además de la característica de las etiquetas, HTML debe de estar dividido en dos grandes partes:

- La cabecera: la cual incluye toda la información sobre la propia página web (por ejemplo el título o el idioma).
- El cuerpo: el cual incluye todo el contenido propiamente dicho de la página web (por ejemplo texto, tablas, imágenes).

En cuanto a las novedades que aporta esta quinta revisión de HTML, hay que señalar que supuso una gran mejora en ciertas áreas de las más demandadas en la actualidad y con tecnologías más actuales como la geolocalización. Debido a esto, HTML5 redujo las dependencias de los plug-ins que tenemos que tener instalados para ejecutar dichas páginas web, este punto lastra mucho a otras tecnologías como Adobe Flash. Las novedades más destacables en esta revisión de HTML son:

- Se desarrollan etiquetas para contenido específico, como audio, video, etc.
- Utilización de una base de datos local, por lo que se podrá trabajar en un página web a través de una API con el cliente. Lo que facilitara la utilización de páginas web sin tener conexión a internet
- Incorporación de Canvas, este componente nos proporcionara la capacidad de dibujar y realizar animaciones en 2D y 3D Todos estos dibujos se consigue por medio de funciones provenientes de una API.
- También se incorpora web workers, estos son procesos que requieres un tiempo de procesamiento alto por parte del navegador, pero se realizan en segundo plano por lo que el usuario no tiene constancia de ellos, y puede utilizar la página a su vez.
- Aplicaciones web offline para trabajar en local.
- Como se ha dicho anteriormente la geolocalización es uno de los puntos fuertes de HTML5, ya que se puede localizar geográficamente por medio de una API (Application Programming Interface).

- Incorporación de nuevas APIs para la interfaz de usuario.
- Pone fin a la utilización de etiquetas de presentación, ya que todas las etiquetas que modifican estilos de la página son eliminadas, debido a la utilización de CSS.

Las etiquetas más importantes que se incorporan en esta quinta versión son:

- **article**: esta etiqueta sirve para definir un artículo, un comentario de usuario o una publicación independiente dentro del sitio.
- **header, footer:** estas etiquetas individuales ahorran tener que insertar IDs para cada uno, como se solía hacer anteriormente. Además, se pueden insertar headers y footers para cada sección, en lugar de tener que hacerlo únicamente en general.
- **nav:** la negación puede ser insertada directamente en el markup, entre estas etiquetas, que nos permitirán hacer que nuestras listas oficien de navegación.
- **section:** con esta etiqueta, una de las más importantes de las novedades, se puede definir todo tipo de secciones dentro de un documento. Por ponerlo de forma sencilla, funciona de una forma similar a la etiqueta div que nos separa también diferentes secciones.
- **audio y video:** estas son las dos más importantes etiquetas de HTML5, dado que nos permiten acceder de forma más simple a contenido multimedia que puede ser reproducido por casi todo tipo de dispositivos; marcan el tipo de contenido que estará en su interior.
- **embed:** con esta etiqueta se puede marcar la presencia de un contenido interactivo o aplicación externa.
- **canvas:** finalmente, esta etiqueta nos permite introducir un "lienzo" dentro de un documento, para poder dibujar gráficos por vectores para lo cual será necesario el uso de JavaScript.

2.2.3 CSS:

Css [9] es una hoja de estilos en cascada, el cual se usa para definir y crear la presentación de un documento estructurado y con estilo de HTML o XML. El fin de esta tecnología es el de poder separar la estructura del documento, de los estilos o presentación propiamente dicha. De esta manera, la información del estilo puede ser definida en un documento separado o en el propio HTML.

En este caso nos encontramos con la tercera revisión, de esta hoja de estilos.

Características Principales.

Antes de que se generalizara el uso de CSS, para realizar el diseño de páginas web se utilizaban etiquetas de HTML especiales (por ejemplo "<font>") para poder dotar de

estilo a los elementos de la página con los problemas que esto conllevaba. Con la llegada de CSS se puede dotar de estilos de una manera más efectiva y mejorada como se puede observar a continuación:

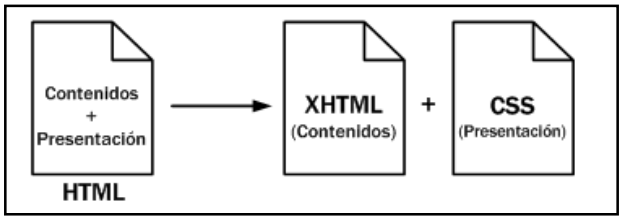


Figura 3: Integración HTML y CSS [21]

CSS nos permite por tanto separar el contenido de la página de la información de los estilos de este contenido. De esta manera conseguimos añadir los estilos correspondientes sin ensuciar lo código de la página. Para incluir este código CSS en la página se puede optar por varias maneras:

- Definirlos en una zona específica del propio documento HTML. Se define empleando la etiqueta <style> de HTML y solamente se puede incluir en la sección de la cabecera del documento HTML (sección <head>).

```
<style type="text/css">
  p { color: black; font-family: Verdana; }
</style>
</head>
```

Figura 4: Incorporación de código CSS en HTML [22]

- Definirlo en un archivo externo, para ello se incluyen en un archivo de tipo CSS todos los estilos necesarios, en la página HTML se enlaza mediante la etiqueta <link>. Se pueden crear todos los archivos CSS que sean necesarios.

```
<link rel="stylesheet" type="text/css" href="/css/estilos.css" media="screen" />
</head>
```

Figura 5: Enlace archivo externo CSS [22]

- Actualmente también se usa la etiqueta <style> para incluir los CSS.

```
<style type="text/css" media="screen">
  @import '/css/estilos.css';
</style>
</head>
```

Figura 6: Etiqueta Style para incluir CSS [22]

En este caso en concreto es necesario utilizar la regla especial @import seguido de la localización o url donde se encuentra el CSS.

Ejemplo:

@import "/css/estilos.css";

@import url ('/css/estilos.css');

- Para finalizar el último método de incluir CSS en un documento HTML, y al menos utilizada ya que posee las mismas limitaciones que el primer método, es la inclusión de CSS directamente en los elementos HTML.

```
<body>
<p style="color: black; font-family: Verdana;">Un párrafo de texto.</p>
</body>
```

Figura 7: Incluir CSS en documento HTML [22]

En cuanto a la terminología básica que se usa en CSS, para describir las partes correspondientes a los estilos. Hay que destacar las diferentes terminologías que forman un estilo de CSS básico. Por lo tanto una hoja de estilos está compuesta por "reglas", las cuales contienen cada uno de los estilos que componen estas hojas de estilos, cada regla está formada por una parte de "selectores" un símbolo de llave de apertura ("{"), después una declaración, y por último una llave de cierre ("}"). A continuación se especifican en más profundidad:

- Selector: indica el elemento o los elementos HTML a los que afecta la regla CSS en concreto
- Declaración: Especifica los estilos que se debe de aplicar al elemento del selector. Esta propiedad puede estar compuesto por uno o por varios elementos.
- Propiedad: se trata de la característica que se debe de modificar del elemento en concreto (por ejemplo tamaño de letra o color).
- Valor: establece el nuevo valor de la característica que se modificará en el elemento.

Por lo tanto un archivo CSS pueden contener un número ilimitado de reglas CSS, y cada regla a su vez varios selectores diferentes, con declaraciones diferentes.

En cuanto a las novedades que incorpora la tercera revisión de CSS, pretenden sobre todo que el diseñador tenga un mayor control sobre el diseño, para ello incorpora algunas novedades respecto a versiones anteriores como son:

- Permite incorporar varias imágenes de fondo en un mismo elemento, además de nuevas propiedades para los fondos.
- Incorpora también la posibilidad de agregar esquinas redondeadas en los elementos, posibilidad muy demandad por los diseñadores actuales.
- Posibilidad de incorporar bordes a los elementos.
- También permite la incorporación de varias sombras en los elementos y textos.

- Con CSS3 podemos añadir la propiedad de que los elementos se vuelvan semitransparentes. Ofreciendo la posibilidad de poner trasparencias en cada color y cada elemento por separado.
- Nos da la posibilidad de poner el texto en varias columnas, sin necesidad de realizar más divisiones en la página. Además de otras novedades en relación al texto.
- Modelo de cajas flexibles mediante diferentes posicionamientos: absoluto, relativo, y flotante.
- Aparición de nuevos selectores que permiten seleccionar elementos de una manera específica (normalmente el posicionamiento respecto a su elemento padre).
- Añade la posibilidad de crear animaciones.

En cuanto a las ventajas que encontramos con la utilización de CSS3, podemos señalar que proporciona un aspecto más dinámico en las páginas web. Y si nos centramos en los inconvenientes, CSS3 se ha adelantado a las nuevas especificaciones de los principales navegadores, de forma que estos deberán adaptarse a CSS3, aunque por el momento no existe ningún problema.

2.2.4 C#:

Con C# [10] nos encontramos frente a un lenguaje de programación, desarrollado por Microsoft, el cual se diseñó para compilar diversas aplicaciones que se ejecutan sobre .NET Framework. C# es muy simple y eficaz, gracias a la incorporación de un sistema orientado a objetos.

C# permite el desarrollo de aplicaciones rápidamente, sin perder funcionalidades y con cierto parecido al estilo del lenguaje de C.

En este caso se ha utilizado para dotar a la aplicación web de la lógica interna, además de eso también se utilizó para la realización de las consultas SPARQL necesarias a lo largo del desarrollo de la plataforma.

Características Principales.

En cuanto a las características, C# es un lenguaje de programación visual controlada por eventos, donde se crean programas o aplicaciones mediante el uso de un entorno de desarrollo integrado (IDE). Con este IDE se puede crear, ejecutar, probar y depurar los programas en este lenguaje de programación, reduciendo así los tiempos necesarios para obtener un programa funcional. En cuanto a la plataforma .NET permite utilización de múltiples lenguajes, incluso la integración con software antiguo para que trabaje con C#.

En cuanto a la plataforma de desarrollo, hay que destacar Visual Studio del cual se hablará más en profundidad en capítulos posteriores. Y en cuanto a algunos de los tipos de aplicaciones que se pueden desarrollar mediante C# se quieren destacar:

- Aplicaciones de consola: utilizan la entrada y salida de comandos estándar, en lugar de los formularios, utilizando la clase System.IO para controlar estas entradas y salidas. El nombre de las clases se pueden utilizar delante de métodos como "System.IO.Console.WriteLine ()", o la utilización de using al inicio de los programas.

- Aplicaciones de formularios: tienen una interfaz gráfica de usuario muy similar a la de Windows, con controles mediante botones y cuadros de diálogo para la entrada de los datos. Utiliza la clase System.Windows.Forms. Estas aplicaciones son fáciles de crear mediante Visual Studio, o desde cualquier editor de texto como el bloc de notas de Microsoft.

- Aplicaciones web APS.NEt: Se trata de aplicaciones web, las cuales se mostrarán finalmente en un navegador, en lugar de en consola o formulario. Este tipo de aplicaciones utilizan el espacio de nombres System.Web y clases como System.Web.UI para controlar la entrada y salida del explorador. Son aplicaciones sencillas de crear mediante Visual Studio.

Además de estos tres tipos que se han destacado también se pueden desarrollar aplicaciones de servicio Web ASP.Net, aplicaciones para dispositivos inteligentes, aplicaciones tradicionales de instalación e implementación, y aplicaciones ActiveX.

Por lo tanto las cualidades más representativas que definen al lenguaje de programación C# son:

- Destaca por su sencillez de uso, ya que C# elimina muchos elementos añadidos en otros lenguajes, y los cuales dificultan el uso y la comprensión de estos. Por ejemplo los ficheros de cabecera, ficheros fuente (IDL1), por esta razón se denomina a C# como auto contenido.

- Nos encontramos frente a un lenguaje de programación muy moderno, y por lo tanto incorpora elementos de última generación, los cuales son muy útiles para el programador, como por ejemplo: tipos decimales o booleanos, string, instrucciones para recorrer colecciones con facilidad (foreach).

- Es un lenguaje de programación orientado objetos, de propósito general. Nos permite la inclusión de funciones sin variable globales que no estén incluidos en una definición de tipos, por lo que esta orientación a objetos es mucho más

trasparente que en otros lenguajes de programación como C++. Soporta encapsulación, herencia y polimorfismo.

- También nos encontramos frente a un lenguaje orientado hacia componentes, ya que la sintaxis incluye elementos propios del diseño de componentes, sin necesidad de ser simulados. Como por ejemplo: definición de propiedades, eventos o atributos.

- Como otros lenguajes, posee un recolector de basura de CLR, lo que implica incluir instrucciones para destruir objetos.

- C# incorpora mecanismos de seguridad avanzados, como por ejemplo: control de acceso a tipos de datos, la no utilización de variables no inicializadas, acceso a tablas sin comprobar los rangos, y control de desbordamiento en operaciones aritméticas. Además incorpora instrucciones seguras, por lo que las evaluaciones de condiciones deben de ser de manera condicional y no aritmética.

- Este lenguaje incorpora, la extensión de operadores básicos, para facilitar la legibilidad de código y conseguir nuevos tipos de datos. Se permite la definición de la mayoría de los operadores que aplican a diferentes tipos de objetos. Por ejemplo: "++","-","+=","==", y "!=".

- C# también incorpora la extensión de modificadores, y por lo tanto la posibilidad de añadir metadatos al módulo resultante de la compilación de cualquier información adicional que se genere en el propio compilador, pudiendo esta información ser consultada en tiempo de ejecución.

- Nos encontramos frente a un lenguaje de programación eficiente, y en el que el código incluye numerosas restricciones para garantizar la seguridad, y no permite punteros como en otros lenguajes.

Destaca por su gran compatibilidad, para facilitar, por ejemplo, la migración de programadores especializados en C++ o Java. La sintaxis es muy similar a estos dos lenguajes de programación, por lo que facilita esta migración, además de poseer el CLR, lo que posibilita acceder al propio código nativo, como DLLs de la API de Win32.

En cuanto a la sintaxis de C#, nos encontramos frente a una sintaxis muy expresiva y sencilla. Resulta muy fácil de aprender, ya que está basada en signos de llave, y cualquier persona familiarizada con C, C++, o Java podrán empezar a trabajar rápidamente de una manera productiva con C# en un periodo relativamente corto de tiempo. Esta sintaxis simplifica algunas complejidades de C++ y proporciona algunas características eficaces como los tipos de valor que admiten "NULL", enumeraciones, delegados, expresiones

lambda y Acceso directo a memoria. Además las expresiones Language-Integrated Query (LINQ) convierten la consultas tapadas ofreciéndole mayor valor al lenguaje C#.

2.2.5 GoJs:

GoJs [11] es una biblioteca basada en JavaScript, la cual posee una gran variedad de características para la realización de diagramas interactivos personalizados y para la visualización a través de navegadores web modernos y diferentes plataformas de visualización.

Incorpora muchas características avanzadas para la interacción entre los diagramas y el propio usuario, por ejemplo: arrastrar y soltar, copiar y pegar, edición de texto en el diagrama, la información de herramientas, menús contextuales, de disposiciones automáticas, plantillas, enlace de datos y modelos, estado transaccional y deshacer gestión, paletas, vistas generales, controladores de eventos, comandos y un sistema de herramienta ampliable para las operaciones personalizados.

GoJs se ejecutará normalmente por completo en el navegador, ya que esto lo propicia la utilización de un elemento Canvas de HTML5 o SVG, sin ningún tipo de requisito por parte del servidor. Al no depender de ninguna biblioteca o marco, GoJs debería de funcionar en cualquier marco en el que se necesiten sus funcionalidades.

Por lo tanto esta librería lleva a cabo la construcción de entornos modelados personalizados y muy visuales. Por tanto, proporciona un editor del sistema y un monitor de estado para la visualización del código compartido y diferentes plantillas que se ofrecen desde GoJs. En las visualizaciones aporta múltiples alternativas para llevar a cabo estas representaciones de los datos por medio de diferentes diagramas. Por ejemplo, con la utilización de subárboles y subgrafos.

GoJs resulta muy simple a la hora de desarrollar los diagramas, para las diferentes y potentes posibilidades que nos ofrece. Por lo tanto la librería consta solo de algunas clases importantes que contienen multitud de características, que interactúan entre sí.

Debido a que GoJs es una biblioteca de JavaScript, depende de las características de HTML5, hay que tener declarado que nos encontramos frente a un documento HTML5 y además se debe cargar la librería propiamente dicha por medio de "<Script src = "go-debug.js">".

A continuación se van a destacar algunos elementos esenciales y características de esta librería:

- Primeramente cada diagrama se debe implementar por separado con una etiqueta " " en el documento HTML, con un tamaño específico:

```
<!-- The DIV for a Diagram needs an explicit size or else we will not see anything.
     In this case we also add a background color so we can see that area. -->
<div id="myDiagramDiv"
     style="width:400px; height:150px; background-color: #DAE4E4;"></div>
```

Figura 8: Etiqueta para diagrama [23]

- Todo el código utilizado durante la creación de los diagramas, utiliza código de las clases de GoJs, como por ejemplo: diagrama, nodo, panel, forma, y TextBlock.

- En cuanto a los nodos y enlaces que contiene un diagrama, nos encontramos ante los elementos que representan a los datos que se manejan a partir de un modelo. Por lo tango GoJS tiene una arquitectura basada en modelo-vista, donde los modelos son los encargados de almacenar los datos y la vista de representarlo por medio de objetos del tipo nodo y enlaces.

- Nos da la posibilidad de hacer click y arrastrar e interactuar de forma variada con todos los elementos de un diagrama. Además de esto, estos objetos tienen a su vez sus propias características y propiedades.

- A la hora de la creación y diseño de nodos, nos ofrece plantillas para la creación y configuración de diferentes objetos. Dándonos las posibilidades de darle forma al objeto, añadir texto (TextBlock), imágenes, panel horizontal y vertical.

- Para llevar a cabo el que los datos de nuestro modelo afecten a los nodos, se realiza mediante enlaces de datos. Estos enlaces nos permiten cambiar el aspecto y comportamiento de los nodos de forma automática.

- Existen diferentes pantallas de nodo, creadas con el fin de tener un esquema para montar un diagrama de organización completa. Algunos de las plantillas más utilizadas son TreeModel, y TreeLayout.

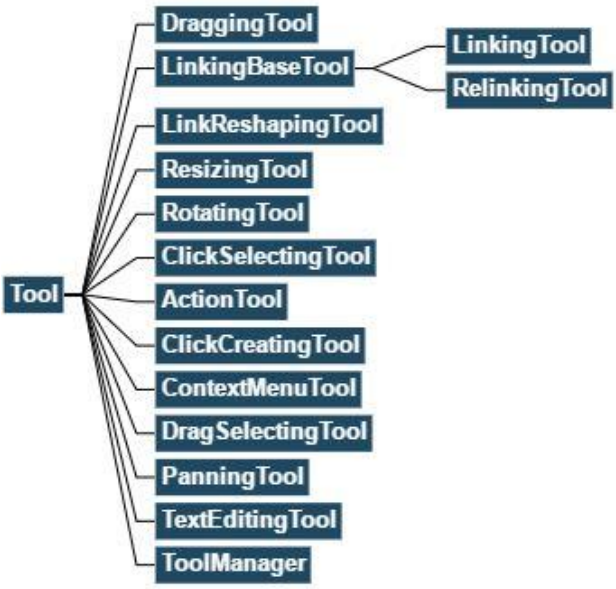


Figura 9: Ejemplo de Árbol GoJS [24]

2.3 PLATAFORMA DE TRABAJO.

En el siguiente apartado se pretende explicar la plataforma de trabajo que se ha utilizado a lo largo del proceso de creación de la plataforma de visualización. Debido al uso de tantas tecnologías tan diversas, y a la utilización del lenguaje C# se optó por la utilización de la plataforma de Microsoft Visual Studio [12].

2.3.1 Visual Studio.

Visual Studio es un entorno de desarrollo que engloba a un gran conjunto de herramientas y tecnologías de desarrollo de software, las cuales están basadas en componentes capaces de crear aplicaciones de gran eficiencia y alto rendimiento. Es capaz de soportar múltiples lenguajes de programación como c#, Visual Basic, Net, Java, Python, etc., además de múltiples entornos de desarrollo a nivel web como ASP.NEt MVC, Django, etc...

Por otro lado, Visual Studio está optimizado para un diseño basado en equipos, desarrollo e implementación utilizando la plataforma Visual Studio Online o Team Foundation Server (TFS) de Microsoft.

Figura 10: Logotipo Microsoft Visual Studio [25]

Historia:

A continuación se pretende dar un pequeño acercamiento a la historia de Visual Studio, destacando los hechos más representativos de ésta.

Este entorno de desarrollo fue lanzado en 1998 y por tanto Microsoft comenzó a migrar toda su estrategia de desarrollo alrededor a .NET Framework. El objetivo primordial de Microsoft a largo plazo con este entorno, trataba de unificar todas las herramientas en único entorno. Posteriormente se lanzó Visual Studio .NET (2002), versión que propició un cambio sustancial, puesto que se introdujo la plataforma .NET de Microsoft por completo. Un año después llego Visual Studio .NET (2003), que supuso una actualización menor respecto a la versión anterior, donde se añadió soporte para el desarrollo de aplicaciones móviles

Dos años después llegó Visual Studio (2005), con la novedad de su comercialización a través de internet, antes de su venta en formato físico. En esta versión la actualización más importante la recibieron los lenguajes de programación, significando su acercamiento a C++. En el año 2008 llegó una nueva versión, Visual Studio (2008), con la novedad de incorporar el nuevo framework (versión .NET 3.5), el cual estaba diseñado para conseguir destacar todas las ventajas del sistema operativo actual Windows Vista y posteriormente dando el salto a Windows 7 con sus consiguientes mejoras.

Posteriormente llego la edición del año 2010, la cual consistía en una actualización de la versión anterior. En el año 2012 se lanzó la versión Visual Studio (2012) la cual incorporaba soporte completo para desarrollar en Windows 8 y su interfaz Metro. Un año después se lanzó la siguiente versión Visual Studio (2013), la cual incorporó soporte para Windows 8.1 , así como para .Net 4.5.1 y Team Foundation Server 2013.

Para finalizar, de la última versión, y por lo tanto la más actual, Visual Studio (2015), se ha querido destacar por encima de todo el desarrollo multi-plataforma. Además de esto, se han integrado en el instalador todas las tecnologías que se pueden añadir a Visual Studio.

Características principales:

A continuación se van a presentar las características más destacables de Visual Studio, centrando la atención solo en destacar las relacionadas con la programación web, debido

a los múltiples ámbitos y extensas características que ofrece dicha plataforma, y también debido a la relación directa del proyecto con la programación web.

- Multitud y variedad de herramientas web:

  Visual Studio está basado en herramientas de código abierto, lo que ofrece gran flexibilidad y permite crear aplicaciones web modernas, por lo que una de las características principales de la plataforma, es el gran número de lenguajes de programación que soporta, como son HTML 5 , CSS 3 ,C #,JavaScript , JSON, PHP, Python, o ASP.NET entre otros.

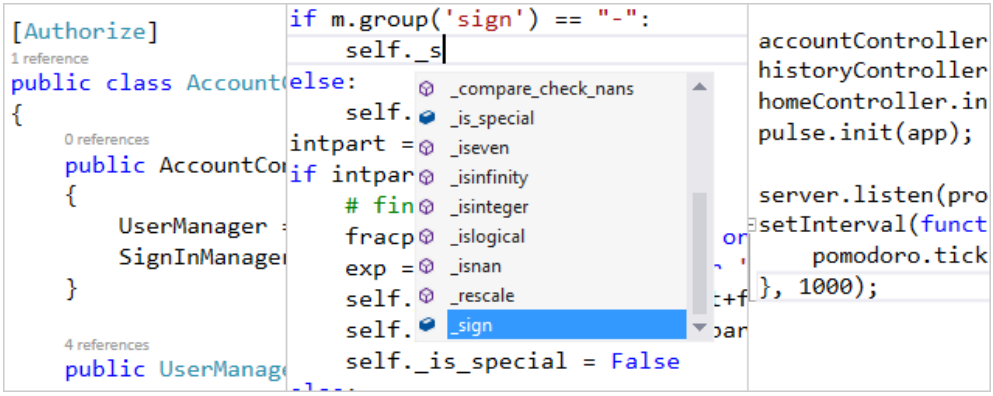


Figura 11: Ejemplo de diferentes lenguajes soportados por la plataforma [26]

- Web moderna:

  Las múltiples actualizaciones a lo largo del tiempo de Visual Studio están diseñadas para que la plataforma se encuentre siempre con las principales novedades en la programación Web, y por lo tanto que soporte las tecnologías más punteras y con mayor reclamo en la actualidad como son: Angular, jQuery, Bootstrap, Django, Backbone.js, Express. Por esta razón, Visual Studio tiene las herramientas de código abierto y la flexibilidad que necesita para crear e implementar aplicaciones web modernas.

  De esta manera se consigue incrementar enormemente la productividad con marcos web potentes y actuales. Además de esto, incorpora IntelliSense para el uso de código JavaScript por el lado del cliente lo que consigue autocompletar mucha cantidad de código de manera automática. También hay que destacar su compatibilidad con los marcos web más populares hoy día, como se ha recalcado anteriormente, como son sobre todo Angula y Bootstrap.

- Muchas plataformas

  Visual Studio destaca principalmente debido a que es posible su funcionamiento en múltiples plataformas, de esta manera ASP.Net 5 y .NET Core CLR son capaces de funcionar en Windows, Max y Linux sin ningún problema.

Por este motivo, los desarrolladores que creen un servicio o aplicación basada en ASP.NET basada en C# y mediante el core CLR en Visual estudio, pueden realizar la implementación basada en varias tecnologías. Incluidas tecnologías "antiguas", realizado la correspondiente migración. Además se permite las codificaciones del código en cualquier editor o sistema preferido para el desarrollador con la característica omnisharp.net.

- Aplicaciones web móviles:

  Soporta los estándares web más populares como: HTML, Css y JavaScript, para la creación de plataformas web para móviles tanto Android, iOs y Windows phone.

  Por lo tanto proporciona a los desarrolladores la capacidad de realizar aplicaciones web basadas en responsive.

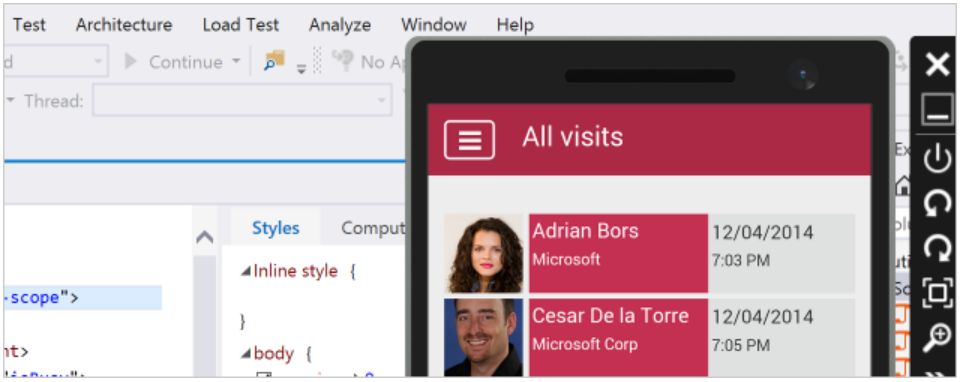


Figura 12: Ejemplo de desarrollo para móvil [26]

- Código abierto y administración de paquetes:

  Como se ha dicho anteriormente nos encontramos frente a código abierto, por lo que ASP.Net y .NET tienen repositorios en GitHub, los cuales proporcionan un soporte frente a incidencias de cualquier tipo, además de contribuciones.

  En cuanto a la administración de paquetes, soporta NuGet, npm y bowe, ofreciendo repositorios basados en bibliotecas de múltiples tipos para .NEt tanto del lado del servidor, como un administrador de paquetes, conexiones de JavaScript desde el cliente y distintas herramientas.

- Ecosistema extensible y Escaldo en la nube:

  Visual Studio, permite extenderlo con múltiples extensiones que ofrece su comunidad, y de esta manera ofrece la capacidad de personalizarlo como mejor se adapte a las necesidades del desarrollador. Algunas de las extensiones más reconocidas entorno al desarrollo web: Web development, Web Essentials, Web Compiler, Web Analyzer, PHP Tools, Bootstrap Snippet Pack.

Visual Studio incluye herramientas integradas que permite que el desarrollador implemente su aplicación web en cualquier host o escalarlo en Microsoft Azure. De esta manera se puede publicar y administrar los desarrollos web y las máquinas virtuales desde Visual Studio, sin necesidad de softwares adicionales.

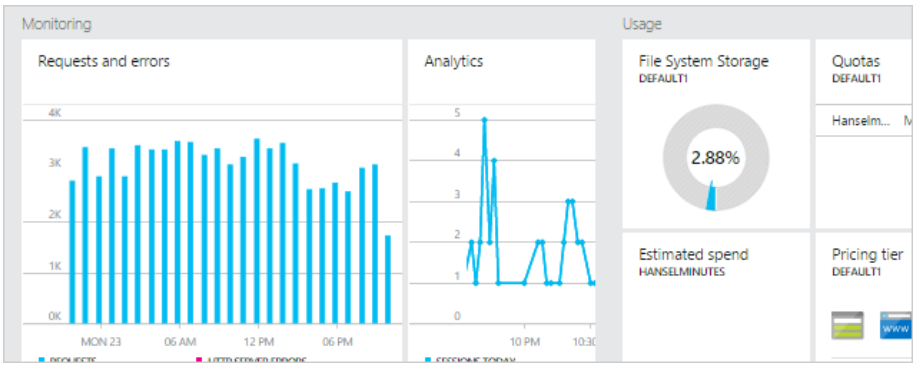


Figura 13: Ejemplo de desarrollo en la nube.

- Herramientas Potentes y un excelente editor de código [26].

  Para finalizar, destacar que nos encontramos frente a un entorno que contiene herramientas muy potentes que mejoran la productividad y un gran compilador. De esta manera se consiguen grandes resultados siempre basados en una estructura sencilla, fundamentada en carpetas, y que se actualiza constantemente con la última versión del código en cuestión.

  Se quiere destacar el gran editor de código que posee Visual Studio, en el cual se puede escribir, editar, explorar, depurar, realizar pruebas, localizar errores o administrar el código fuente. Además de la incorporación de Visual Studio Team Services o Team Foundation Server (TFS), junto al repositorio de Git.

2.4 PLANTEAMIENTO DEL PROYECTO.

En este apartado se expondrán tanto los objetivos del proyecto, como una descripción general para poner en contexto el proyecto, así como todas las todas las tareas que se han llevado a cabo para cumplir los objetivos finales del proyecto.

2.4.1 Objetivos.

El objetivo del presente TFG es el desarrollo de una aplicación web, que sea capaz de proporcionar un diagrama para la visualización completa de la información tomada de un

repositorio semántico. Esta representación se debe llevar a cabo de tal forma que el usuario sea capaz de interaccionar con el diagrama generado, y de esta manera lograr obtener todos los datos que requiera de dicha visualización. Toda está visualización e interacción con el usuario se pretende realizar de la manera más sencilla y atractiva posible.

Se quiere recalcar que aunque la plataforma a desarrollar puede ser usada por cualquier persona que se encuentre interesada, el principal número de usuarios que van a acceder a ésta, serán personas relacionadas con la biología.

2.4.2 Descripción

La aplicación web que se ha desarrollado, se encuentra disponible para su visualización en cualquier navegador actual. De esta manera, esta plataforma web nos permitirá recibir datos (correspondientes a una interacción, o a una interacción más otro nodo del árbol) que provienen de la plataforma de consulta, hacia la plataforma actual, la cual se encargará primero de tratar todos los datos recibidos. Por lo tanto, mediante los datos que se reciben se realizan consultas SPARQL a una web semántica, de donde irán obteniendo todos los datos necesarios para representar dicha información mediante nodos, de esta manera se representarán todos los datos obtenidos en forma de árbol. Con dicho árbol el usuario final será capaz de navegar tanto hacia delante, como hacia atrás, además de obtener toda la información que requiera de cada nodo representativo.

Para facilitar el uso de la plataforma, se obtienen todos los datos necesarios para llevar a cabo la formación de dos diagramas. Primero el diagrama completo de la interacción en la que nos encontramos, de esta manera el usuario siempre puede saber en qué rama del árbol se encuentra, además de ser el encargado de alimentar el segundo diagrama, el local sobre el cual tendrá lugar toda la interacción con el usuario.

Para finalizar se incluye un cuadro de resultados para que el usuario pueda ir consultando en todo momento la información adicional de los nodos, así como un botón para desplegar u ocultar el diagrama completo y de esta manera adaptarse a cada usuario.

2.4.3 Tareas Realizadas.

En el presente apartado se van a presentar los diferentes trabajos que se han realizado a lo largo del desarrollo del proyecto.

1. **Formación y periodo de documentación:**

   El primer paso a realizar consistió en la realización de un periodo de documentación y de formación para poder llevar a cabo los objetivos propuestos.

   Descripción:
   Durante este periodo, se realizó principalmente documentación en todo lo relacionado con la web-semántica, debido a su desconocimiento hasta el momento, ya que durante la carrera no se obtuvieron conocimientos previos en relación a este tema.
   Además de esto, se llevó a cabo un periodo de documentación enfocado a poder facilitar el trabajo posterior, de tal forma que se tuvieran siempre actualizados todos los conceptos y tecnologías, que iban a ser necesarios a lo largo de la realización del proyecto.

   Esfuerzo:
   Supuso un esfuerzo de 4 días.

2. **Especificación de requisitos funcionales:**

   Se realizó una especificación de los requisitos funcionales que debería de tener la web de visualización.

   Descripción:
   Se crea una lista de especificaciones donde se encuentran los requisitos funcionales que debe tener la web de visualización, para que ésta sea capaz de cumplir todas las funcionalidades establecidas y cómo debería comportarse el sistema en todo momento. Dentro de esta lista de requisitos, se puede establecer una división en dos grandes grupos:

   2.1. Requisitos funcionales:
   En este punto se englobarán los requisitos funcionales básicos para que la plataforma de visualización cumpla con los requerimientos básicos respecto a su funcionamiento.

   2.2. Requisitos no funcionales:
   En este punto se incluirán aquellos requisitos funcionales, los cuales establecen algunas funciones secundarias que se podrían incorporar a la plataforma de visualización, pero que no son tan críticos para el funcionamiento de la página web.

   Esfuerzo:

   Supuso un esfuerzo de 2 días.

3. Especificación de los Casos de Uso:

Se realizó una especificación de los Casos de uso del proyecto.

Descripción:
Primero se realizan los casos de uso para las dos aplicaciones web, tanto los filtros como la plataforma de visualización en conjunto.
Posteriormente nos centramos en los casos de uso de la plataforma de visualización, los cuales representan una descripción de los pasos que se deben de seguir para llevar a cabo el proceso de visualización cumpliendo los objetivos dados.

Esfuerzo:

Supuso un esfuerzo de 1 día.

4. Elección de los diferentes lenguajes de programación:

En este momento se llevó a cabo la elección de los diferentes lenguajes de programación y librerías que se pretendían usar a lo largo del proyecto.

Descripción:
En este punto se llevó a cabo un periodo de investigación y de análisis, para poder establecen los diferentes lenguajes de programación que eran más adecuados para toda la construcción de la plataforma, de tal manera, que fuera capaz de cumplir todos los objetivos establecidos para esta aplicación web.

4.1. Html5:

Para toda la programación Web.

4.2. Css3:

Complemento a HTML5 en la programación web, para dar estilo al código HTML.

4.3. JavaScript:

Para establecer toda la funcionalidad en la web.

4.4. C#:

Utilizado para toda la lógica interna de la aplicación web.

Además de esto también se llevó a cabo una investigación en profundidad para la utilización de una librería capaz de ayudar a la representación de los nodos en la plataforma web.

4.5.GoJs:

Librería de JavaScript, la cual sirve para representación de nodos en este caso sobre HTML.

Esfuerzo:

Supuso un esfuerzo de 2 días.

5. Diseño de baja fidelidad.

Se llevó a cabo un diseño en papel de la página web.

Descripción:

Se lleva cabo un diseño de alto nivel de la plataforma. Este diseño se lleva a cabo sobre papel, y sobre él se va diseñando toda la página web, así como la distribución de los elementos principales de la plataforma.

Este diseño de alto nivel, permitió la toma de decisiones sobre la mejor localización de los elementos dentro de la plataforma web.

Esfuerzo:

Supuso un esfuerzo de 5 horas.

6. Realización de una primera maqueta visual de la plataforma web.

En este apartado se llevó a cabo la creación de diseño de alta fidelidad, y posteriormente una maqueta de la plataforma web, sin funcionalidades, consistiendo solo en un modelo preliminar (solo visual).

Descripción:

Se realizó una maqueta exclusivamente visual de la plataforma web, la cual no contaba con ninguna funcionalidad, dotándola exclusivamente de código en HTML y CSS3.

Este punto sirvió para detectar fallos del diseño de alto nivel, y de esta manera tener todo el apartado visual cerrado y mucho más consolidado.

Esfuerzo:

Supuso un esfuerzo de 7 días.

7. **Desarrollo de un ejemplo completo del entorno visual.**

En este punto se llevó a cabo la creación de ejemplo completo con la librería GoJs.

Descripción:
Se realizó un ejemplo completo con nodos de ejemplo, para desarrollar todo el entorno y las funcionalidades, con las cuales se quería dotar al entorno visual para que fuese capaz de representar tanto un diagrama general, como un diagrama local.
Además de esto, se implementó un cuadro de texto, para posteriormente ir añadiendo la información correspondiente a cada nodo.

Esfuerzo:
Supuso un esfuerzo de 7 días.

8. **Construcción de consultas necesarias.**

Se lleva a cabo la construcción de las consultas básicas necesarias para el proyecto.

Descripción:
Realización de las consultas SPARQL necesarias al repositorio web para la obtención de los datos de necesidad para completar los nodos que se deben representar en la plataforma web.

Esfuerzo:
Supuso un esfuerzo de 3 días.

9. **Implementación de la lógica de la plataforma web.**

Se implementa la lógica de la página web, con el lenguaje de programación C#.

Descripción:
Se realiza toda la lógica web interna de la plataforma, para lograr llevar a cabo con los datos de entrada, las consultas al repositorio web. Posteriormente, mediante estas consultas se procederá a la creación de los nodos representativos (los cuales son el elemento clave de la página web), y el grafo visual que constituye el núcleo central del TFG.

Esfuerzo:
Supuso un esfuerzo de 14 días.

10. Implementación del paso de parámetros hacia entorno visual.

Se desarrolló al completo el paso de parámetros para después mostrar los nodos representativos.

Descripción:
Se llevó a cabo la realización de toda la implementación del paso de parámetros desde la lógica web en C#, hasta GoJs, para que de esta manera se alimente el modelo de esta librería, y se encargue de representar la información por medio de los nodos.

Esfuerzo:
Supuso un esfuerzo de 3 días.

11. Diseño e implementación de la página de ayuda.

Se diseña e implementa la página que servirá como ayuda en la plataforma de visualización.

Descripción:
Esta página en concreto se implementa a modo de ayuda, para que el usuario pueda consultarla en cualquier momento de duda que tenga durante la utilización de la plataforma.

Esfuerzo:
Supuso un esfuerzo de 1 día.

12. Diseño e implementación de la página de contacto.

Se diseña e implementa la página que servirá como contacto en la plataforma de visualización.

Descripción:
En esta página en concreto se añadirán todos los datos relevantes de la persona que se encargó de desarrollar la plataforma web, así como los datos de contacto para que el usuario se pueda poner en contacto con el desarrollador.

Esfuerzo:
Supuso un esfuerzo de 1 día.

13. **Desarrollo de las pruebas realizadas a la plataforma.**

    Se desarrolla un plan de pruebas a la plataforma de visualización.

    Descripción:
    En este apartado se llevaron a cabo todas las pruebas que se consideraron necesarias para que la aplicación web cumpla con todos los requisitos que se establecieron al principio del proyecto.

    Esfuerzo:
    Supuso un esfuerzo de 2 días.

14. **Desarrollo de la memoria del TFG y de la presentación.**

    Desarrollo de esta memoria del trabajo fin de grado y de la presentación.

    Descripción:
    Por último se llevó a cabo toda la documentación necesaria para la realización de la presente memoria y de esta manera conseguir que el proyecto al completo quede perfectamente documentado.

    Además también se desarrolla la presentación, que servirá de guía a la hora de defender el trabajo de fin de grado.

2.5 REQUISITOS DEL PROYECTO

En este apartado se pretende ampliar toda la información referida tanto a la comprensión de los casos de uso referidos a la plataforma web, como a los requisitos del proyecto.

2.5.1 Casos de uso:

Un caso de uso consiste en una descripción de los pasos o las actividades que deberán realizarse para llevar a cabo un determinado proceso. Los personajes o entidades que participarán en un caso de uso se denominan actores. En el contexto de ingeniería del software, un caso de uso se define como una secuencia de interacciones que se desarrollarán entre un sistema y sus actores en respuesta a un evento que inicia un actor principal sobre el propio sistema. Los diagramas de casos de uso sirven para especificar la comunicación y el comportamiento de un sistema mediante su interacción con los usuarios y/u otros sistemas.

Por lo tanto los casos de uso, permiten considerar y organizar los requisitos en el contexto de los objetivos y escenarios de uso dentro del sistema en el que nos encontramos.
Como ya se ha mencionado previamente, en cuanto a los casos de uso, nos encontramos con dos conceptos básicos: los actores y el escenario.

En cuanto a los tipos de actores que podemos encontrar en el diagrama de casos de uso, encontramos distintos tipos:

- Actores principales: tienen objetivos los cuales se satisfacen mediante el uso del sistema en el que nos encontramos.
- Actores de apoyo: proporcionan servicios al sistema en el cual nos encontramos estudiando.
- Actores pasivos: está interesado, le afecta o se ve afectado de alguna manera por el comportamiento del sistema del caso de uso.

En cuanto al escenario, nos encontramos frente a una secuencia específica de acciones e interacciones entre los actores y el sistema de objeto de estudio.
Es importante recalcar las relaciones entre los distintos casos de uso:

- Relación Incluir (include): siempre se ejecuta el caso de uso incluido, y una vez finalizado retorna al punto desde donde se llamó a esta relación.
- Relación Extender (extend): no siempre se tiene que ejecutar el caso de uso que extiende, y una vez finalizado el caso de uso retorna al punto desde donde se llamó a esta relación.
- Relación Generalizar (generalize): un caso de uso que es un caso particular de otro. Es una abstracción.

En primer lugar, se requiere especificar un diagrama de casos de uso en el cual se integren ambos sistemas, tanto el sistema en el cual se realizarán las consultas y se obtiene la información, como el sistema que atañe a este proyecto, la representación gráfica de las consultas realizadas en referencia a las interacciones [13].

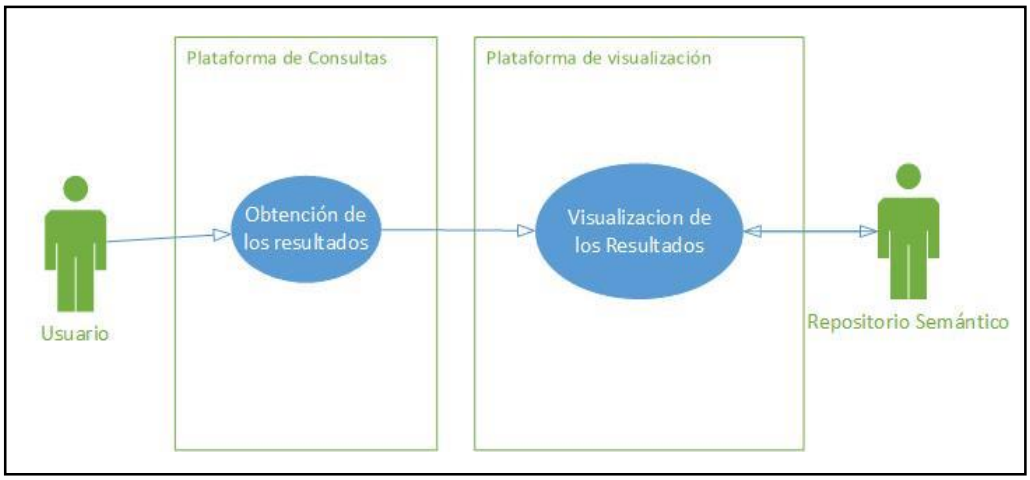


Figura 14: Diagrama casos de uso sistema general [28]

En este diagrama a modo de resumen de la integración entre ambas plataformas, se puede observar como actor principal al usuario, y como usuario de apoyo al repositorio semántico. El usuario realizará una serie de consultas, basándose en unos filtros visuales que se configurarán dentro de esta plataforma, una vez realizadas estas consultas, esta plataforma ofrecerá al usuario una serie de resultados.
Dentro de estos resultados, al usuario final se le ofrecerá, con aquellos que cumplan una serie de condiciones, la opción de visualizar esos resultados a través de la plataforma de visualización, y de este modo la plataforma de consultas enviará la información necesaria a la plataforma de visualización, en la cual nos debemos de centrar

Por lo tanto, en este punto vamos a centrarnos en el diagrama de casos de uso que implica a este proyecto en concreto. Se requiere desarrollar una plataforma web para la consulta visual de una serie de datos obtenidos de una web semántica, de tal manera que se quiere obtener un sistema lo más ágil posible para la realización de las comunicaciones entre los distintos dispositivos, con el mínimo posible de fallos e incidencias.

De este modo, el diagrama de caso de uso de la plataforma visualización web atendería al diseño mostrado en el ANEXO-1.

Descripción de los actores:

ACTOR	TIPO	OBJETIVO
Usuario	Principal	Envió de información, encargado de consultar resultado visual, consultar ayuda, consultar contacto, consultar la información de los nodos, navegar entre los nodos.
Repositorio Semántico	Apoyo	Base de datos semántica, contenedor de la información y de las relaciones

Tabla 1: Actores casos de uso [28]

Descripción de los casos de uso en formato breve:

CU1: Consulta con interacción:

Se trata de la recopilación de la información necesaria para obtener el parámetro de la interacción en la que se desea centrar la plataforma de visualización, para luego alimentar al método de consultas y gestión de resultados.

CU2: Consulta con interacción y otro nodo:

Por otro lado, se puede recopilar la información necesaria para obtener el parámetro de la interacción, pero en este caso también se debe obtener otro parámetro el cual representa el nodo dentro del árbol visual en el que se debe de centrar la plataforma de visualización, para luego ser capaz de alimentar al método de consultas y gestión de resultados.

CU3: Método de consultas y gestión de resultados:

En este caso se trata de la realización de una consulta a la plataforma de visualización web, pasándole como parámetro la interacción, o la interacción y además el nodo en el cual queremos centrar este sistema de visualización.

CU4: Consulta de resultados:

Este caso de uso, es gracias al cual el usuario es capaz de llevar a cabo la consulta de todos los resultados obtenidos en el CU3, y de esta manera navegar por el árbol de resultados, y obtener la información detallada de cada nodo.

CU5: Resultados:

En este apartado, es donde el método de consultas y gestión de resultados, volcará los resultados obtenidos dependiendo del tipo de interacción y nodo en el que nos encontremos.

CU6: Consulta de ayuda

En este caso, nos encontramos con una consulta por la cual el usuario podrá acceder a una sección de ayuda, acerca de cómo utilizar la plataforma web de visualización.

CU7: Página de ayuda

Representa la página web donde se aloja toda la información referente a la ayuda.

CU8: Consulta de contacto:

En este caso, encontramos una consulta por la cual el usuario podrá acceder a una sección de contacto, donde se encuentran datos del desarrollador y de contacto para cualquier tipo de incidencia futura.

CU9: Página de contacto:

Representa la página web donde se aloja toda la información referente a la ayuda.

Descripción de los casos de uso en formato esencial:

Caso de uso en el que el usuario, realiza una consulta, y después de obtener los datos de ésta, decide que requiere que uno de dichos resultados sea representado de forma visual.

Actor principal: usuario que realiza dicha petición a la plataforma de visualización.

Personal involucrado e intereses: es el usuario, ya que es el encargado de requerir la información, y el mayor beneficiado de este sistema de visualización.

Precondiciones: el usuario debe de elegir entre enviar a la plataforma de visualización o una interacción, o por otro lado una interacción pero además un "nodo" en el que se quiere centrar la visualización.

Garantías de éxito (post condiciones): se consigue llevar a cabo una visualización correcta de la información requerida por parte del usuario.

Escenario principal de éxito (o flujo básico):

1. El usuario después de realizar la consulta correspondiente, y seleccionar un resultado que cumpla las precondiciones, realiza la consulta a la plataforma de visualización (se envían los datos de interacción, o interacción más "nodo").
2. El sistema se encarga de comprobar que los datos enviados desde la plataforma de consultas son válidos.
3. Después de comprobar que todos los datos enviados son válidos, el sistema realizará las consultas necesarias para obtener el árbol de nodos al completo, además de la información de cada nodo.
4. Una vez obtenido el árbol, el sistema se encargará de construir el diagrama centrado en los nodos, además del diagrama al completo.
5. El sistema acaba y se queda a la espera de más peticiones.

Extensiones (o flujos alternativos):

*a= El sistema falla.
2ª. = El sistema falla debido a que los datos son correctos, o no se obtienen los resultados esperados.

Lista de tecnología y variaciones de datos:

Se consultara la base de datos semántica, que es la que contiene toda la información requerida por la plataforma de visualización.

2.5.2 Requisitos funcionales.

Son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que éste debe reaccionar a entradas del sistema y de cómo se debe comportar en ciertas situaciones. En conclusión, los requisitos del sistema describen lo que el sistema debe de hacer [14].

RF1: Obtener información necesaria a través de consultas
Requisitos: • La lógica interna de la aplicación se encargará de realizar todas las consultas necesarias a la base de datos semántica. • Por medio de este requisito indispensable, se obtendrá toda la información que se manejara en la plataforma web.

RF2: Tratamiento de la información obtenida en las consultas
Requisitos: • La plataforma web debe ser capaz de tratar todos los datos obtenidos durante las consultas, ya se trate de la información básica o más compleja del nodo, así como sus relaciones. • A través de esta información la lógica interna irá decidiendo qué proceso se realiza con la información, ya sea representarla, o usar dicha información para volver a lanzar el proceso de consultas.

RF3: Representación información de los nodos obtenidos
Requisitos: • La plataforma web debe de ser capaz de representar la información del nodo. • Además de la información sobre sus relaciones. • Por otro lado los nodos en los que se encuentre disponible, posibilitarán la visualización de información más extensa.

RF4: Representación conjunta de nodos
Requisitos: • La plataforma deberá de ser capaz, de representar un número determinado de nodos, en los que se encuentra el usuario en ese momento. • Además de representar estos nodos, debe de representar dichas relaciones entre ellos.

RF5: Posibilidad de navegar a través de los nodos
Requisitos: • La aplicación web debe de ser capaz de dar al usuario la posibilidad de interaccionar con la representación gráfica de los nodos. • De este modo el usuario debe de tener la posibilidad de desplazarse tanto hacia delante, como hacia detrás, por el árbol de los nodos.

RF6: Interacción con el usuario
Requisitos: • Uno de los requisitos indispensables, se trataba de la gran interacción que debe de tener la plataforma con el usuario, para de esta manera despertar interés en el usuario al utilizar dicha plataforma.

RF7: Sistema manejable
Requisitos: • Uno de los requerimientos básicos, para la plataforma web en cuestión, era la facilidad que debía de ofrecer ésta al usuario para su utilización, debido a que de este modo se consigue que ésta pueda ser utilizada por cualquier persona de cualquier ámbito.

2.5.3 Requisitos no funcionales.

Los requisitos no funcionales, como su propio nombre indica, no son necesarios para el funcionamiento final del sistema, pero su cumplimiento añade gran valor al producto final.

Los requisitos no funcionales a menudo se aplican al sistema en su totalidad y normalmente apenas se aplican características o servicios individuales del sistema [15].

RNF1: Página de ayuda
Requisitos: ▪ Se implementa una página de ayuda a la que el usuario puede acceder, para realizar las consultas pertinentes en relación a la aplicación web.

RNF2: Página de contacto
Requisitos: ▪ Se implementa una página de contacto a la cual el usuario puede acceder, para reportar cualquier tipo de incidencia o sugerencia futura.

RNF3: Estética
Requisitos: ▪ Un requisito muy importante en la plataforma visual, es que sea estética y atractiva para el usuario. ▪ Acciones realizadas a nivel estético, como por ejemplo destacar con un recuadro más oscuro el nodo actual en el que nos encontramos tanto en el diagrama completo, como en el diagrama general.

RNF4: Relación entre los nodos
Requisitos: ▪ Se implementa un letrero dentro de cada relación entre los nodos.

RNF5: Diagrama completo
Requisitos: ▪ Por medio de un método recursivo se obtiene el diagrama completo basado en la interacción dada. Este diagrama completo, se representará en la plataforma web a modo de aclaración, y a modo de ayuda. En el segundo caso, este diagrama puede servir de ayuda si el usuario en un momento determinado está en las últimas ramificaciones del árbol y pretende volver al principio, ya que no tendrá que ir dando pasos hacia atrás a través del árbol.

2.6 Diseño de la plataforma.

En este apartado se pretende ampliar la información en relación al diseño de la plataforma de visualización, para llevar a cabo el diseño de ésta. Se realizó en 3 fases distintas:

1- Prototipo de baja fidelidad.
2- Prototipo de alta fidelidad.
3- Maqueta de un caso de uso básico.

Una vez superadas dichas fases, se comenzó el desarrollo de la plataforma web al completo, dotándola de todas las funcionalidades necesarias para cumplir los objetivos.

A continuación se va a ampliar la información de cada una de las tres fases seguidas por la del diseño de la plataforma.

2.6.1 Prototipo de baja fidelidad.

El primer paso que se realizó respecto al diseño de la plataforma web, fue la realización de un modelo de baja fidelidad. Un prototipo de baja fidelidad, se encarga de destacar los aspectos generales del sistema, sin entrar en grandes detalles ni funcionalidades. Con un prototipo de baja fidelidad lo que se consigue es abarcar en todo lo posible la interacción que debe de tener el usuario con la plataforma que se quiere desarrollar.

Además este tipo de prototipos, nos permite detectar rápidamente errores en el diseño, y realizar los cambios que se crean convenientes rápidamente. Por otro lado, también complementa la identificación algunos casos de uso.

De este modo se llevó a cabo el diseño de este prototipo con las siguientes premisas:

- La plataforma debe de ser capaz de mostrar toda la información por medio de nodos.
- Tiene que tener un diseño atractivo y sencillo.
- Debe de ser capaz de mostrar todos los nodos necesarios

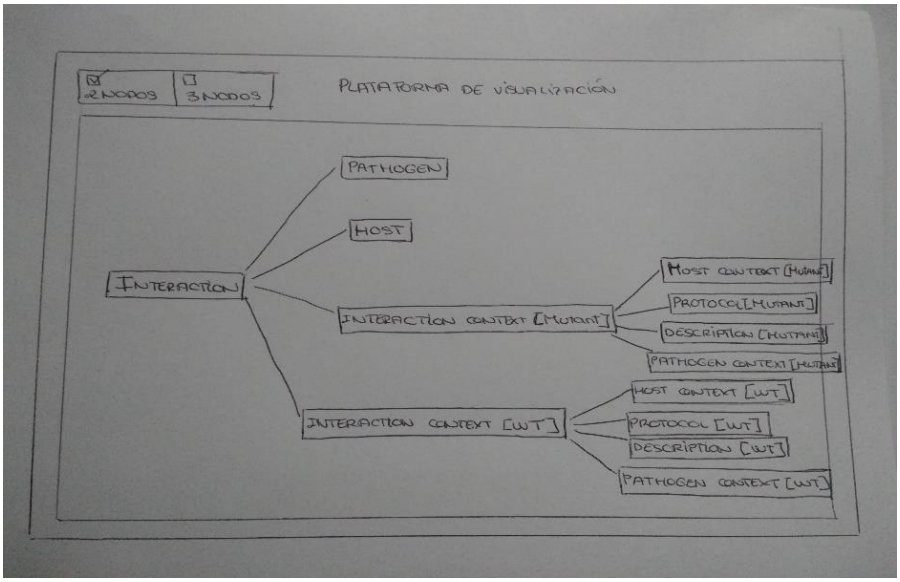


Figura 15: Prototipo de baja fidelidad [28]

Después de analizar el primer diseño que se realizó de la plataforma se llegó a la conclusión de que era necesario añadir otro diagrama extra, el cual se utilizará para representar el diagrama completo relativo a la interacción en concreto que se está representando. Este diagrama se decide añadir por los siguientes motivos:

- Para que el usuario sea capaz de saber en todo momento en que lugar del diagrama se encuentra.
- Para que el usuario sea capaz de volver al principio del diagrama sin necesidad de tener que ir retrocediendo nodo a nodo en el diagrama local.

Para visualizar el segundo prototipo de baja fidelidad consultar **ANEXO-2**.

2.6.2 Prototipo de alta fidelidad.

Después de analizar el prototipo de baja fidelidad, y realizar todos los cambios que se consideraron pertinentes, se llevó a cabo la creación de un prototipo de alta fidelidad. Nos encontramos ya ante un prototipo real y con el que se tiene una primera toma de contacto de cómo podría ser la versión final de la plataforma.

Para facilitar su visualización se puede ver en el **ANEXO-3.**

Para la realización de este prototipo de alta fidelidad se llevó a cabo un periodo de documentación y de aprendizaje para la utilización de la librería GoJs, que como se ha explicación con anterioridad se utiliza para llevar a cabo la construcción de los diagramas, por medio de la construcción de nodos. Por lo tanto, una vez superada la fase de documentación y aprendizaje, se llevó a cabo la creación de un árbol completo a modo de ejemplo siguiendo el esquema general que tiene el repositorio semántico.

Los elementos que se pueden observar en este prototipo de alto nivel son:

- Título: se crea un título a modo de cabecera de la página.
- CheckBox Diagrama Completo: se diseña un checkbox básico para mostrar y ocultar
- Checkbox 2 nodos: sirve para que el Diagrama local muestre solo hasta 2 nodos de distancia.
- Checkbox 3 nodos: sirve para que el Diagrama local muestre solo hasta 2 nodos de distancia.
- Diagrama completo: el primer diagrama con el que nos encontramos es el diagrama completo, en el cual se muestra el árbol completo de la interacción con la que se está trabajando, el cual debe marcar en todo momento el nodo en el que nos encontramos respecto al diagrama local.
- Diagrama local: el segundo diagrama nos muestra el diagrama local, el cual mostrará 2 o 3 nodos de distancia según la Checkbox marcada. Este diagrama es con el que interactúa el usuario principalmente.

Además de diseñar ambos diagramas, también se le añadió ciertas funcionalidades a estos diagramas, las cuales se presentan a continuación:

- Al diagrama completo se le añadió la funcionalidad de que su foco se moviera en relación al nodo que se encuentra seleccionado en el diagrama local.
- Además de esto, también se le dio la funcionalidad de poder seleccionar el nodo que se va a mostrar en el diagrama local, para por ejemplo no tener que ir retrocediendo nodo a nodo si se quiere volver al principio de la interacción.
- En cuanto al diagrama local tiene la funcionalidad de ir recorriendo el árbol de nodos tanto hacia delante como hacia detrás.

Las tecnologías utilizadas para la realización de este prototipo:

- HTML5 y CSS3: estas dos tecnologías son utilizadas para la creación de la página web al completo, y para dotarle de los estilos necesarios.
- GoJS: se utiliza esta librería para llevar a cabo la creación de los dos árboles de nodos, tanto el diagrama completo, como el diagrama local.

Una vez finalizado este diseño de alta fidelidad, se le presentó a modo de prueba al tutor académico del trabajo de fin de grado, para que ofreciera una valoración, después de la cual se llegó a la conclusión de que era necesario añadir un mecanismo que mostrara la

información adicional de cada nodo. Por lo tanto se llevó a cabo una maqueta de un caso de uso completo, añadiéndole dicha funcionalidad, y mejorando la estética de la plataforma al completo.

2.6.3 Maqueta de un caso de uso básico.

Se llevó a cabo el diseño de una maqueta sin ninguna funcionalidad en cuanto a la lógica interna de la aplicación, en la cual se añade el cuadro de resultados para mostrar la información extra de los nodos. Las tecnologías utilizadas en la creación de esta maqueta son las mismas que en el prototipo de alta fidelidad.

Para facilitar su visualización se puede ver en el **ANEXO-4.**

Los elementos que se pueden observar en esta maqueta son:

- Cabecera: se crea una cabecera que incorpora el título, el logo de la universidad, y tres enlaces (inicio, contacto,+info).
- CheckBox “Diagrama Completo”: se diseña un checkbox básico para mostrar y ocultar.
- Checkbox “2 nodos”: sirve para que el Diagrama local muestre solo hasta 2 nodos de distancia.
- Checkbox “3 nodos”: sirve para que el Diagrama local muestre solo hasta 2 nodos de distancia.
- Diagrama completo: El primer diagrama con el que nos encontramos es el diagrama completo, en el cual se muestra el árbol completo de la interacción con la que se está trabajando.
- Diagrama local: el segundo diagrama nos muestra el diagrama local, el cual mostrara 2 o 3 nodos de distancia según la Checkbox marcada, este diagrama es con el que interactúa el usuario principalmente.
- Cuadro de resultados: en el cual se añadirán la información adicional de cada nodo con el que interaccione el usuario.

2.7 DESARROLLO E IMPLEMENTACIÓN DE LA PLATAFORMA WEB.

En este apartado se va a tratar el desarrollo completo de la plataforma web, y cómo se ha llevado a cabo su implementación, explicando a grandes rasgos el desarrollo del código fuente de dicha aplicación.

Lo primero en lo que nos centraremos en este punto, es en la arquitectura del software que se va a utilizar para la implementación de la aplicación web.

Arquitectura del software:

Para el desarrollo de la plataforma web, se decidió optar por un estilo de arquitectura de software denominada modelo vista controlador (MVC), la cual nos ofrece independencia entre los datos de la propia aplicación, la lógica de control y la interfaz para el usuario. Esto lo consigue ofreciendo tres componentes distintos: el modelo, la vista y el controlador. A continuación se detallan cada uno de ellos:

- El modelo: es el encargado de realizar el papel de almacenamiento de datos, por lo que éste tiene que ser independiente al sistema de almacenamiento. Define la funcionalidad del sistema.
- La vista: se encarga de recibir los datos del modelo y de mostrárselos al usuario. Suele estar asociado a un controlador y a un modelo únicamente.
- El controlador: es el encargado de recibir los eventos de entradas y de implementar la lógica principal del software [16].

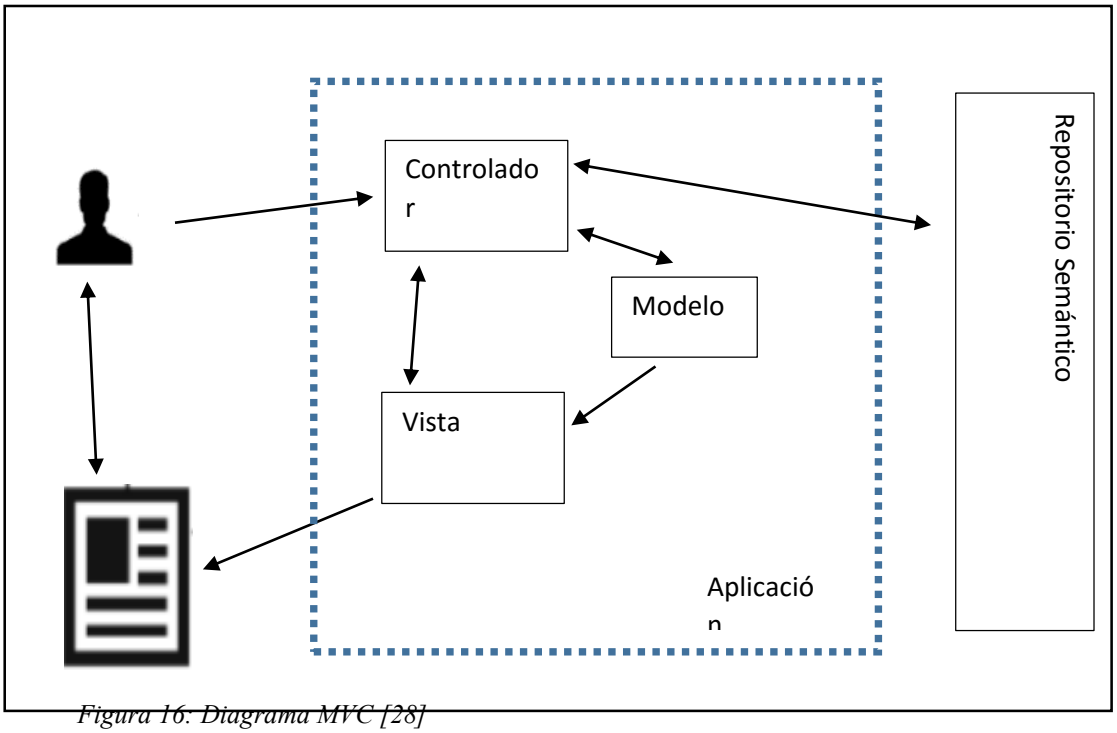


Figura 16: Diagrama MVC [28]

Modelo-vista-controlador en la plataforma actual.

Como se puede observar a continuación en el explorador de soluciones, nos encontramos con la estructura principal del proyecto en la que se pueden observar tanto la carpeta Models, como la de Controllers, y la de Views.

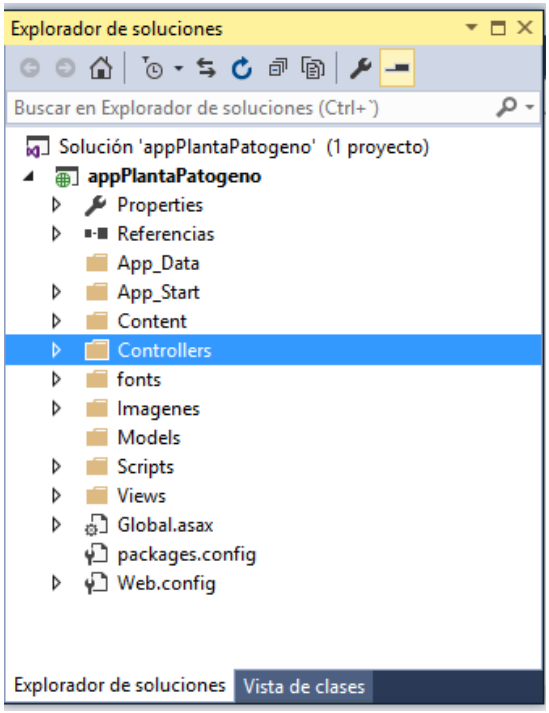


Figura 17: Explorador de soluciones proyecto[28]

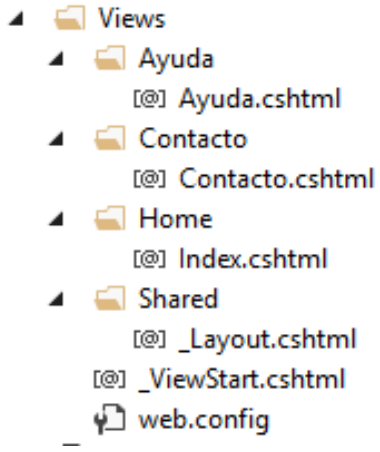
Además de estas carpetas para MVC [17], nos encontramos algunas otras:

- Imágenes: donde se alojan las imágenes utilizadas en todo el proyecto.
- Content: donde se alojan los contenidos, como por ejemplo el site.css, que contiene la hoja de estilos de la plataforma web.
- Referencias: lugar en que se encuentran todas las referencias utilizadas para el funcionamiento de todas las tecnologías utilizadas en el proyecto.
- Scripts: aquí se encuentran todos los scripts de los JS utilizados en el proyecto.
- Global.asax: también conocido como el archivo de la aplicación ASP.NET, es un archivo que se encarga de responder a eventos en el nivel de aplicación provocados por ASP.NET o por HttpModules.
- Web.config: se trata de un archivo XML en el que se encuentran las principales opciones de configuración de una aplicación ASP.NET.ET.
- APP_Start: en este directorio nos encontramos frente a los archivos de código que se ejecutan al iniciar la aplicación web, para realizar toda la configuración necesaria para ASP.Net MVC.
- APP_Data: se trata de un conjunto de archivos XML, los cuales sirven de almacén de datos, para dotar a la aplicación de una pequeña base de datos local.

Vista:

En lo referente a la vista, en el explorador de archivos del proyecto se pueden observar las 3 vistas principales del proyecto: la vista de ayuda, la vista de contacto, y la de Index.

Estas vistas contienen el código Html de la aplicación web, además de pequeños fragmentos de Razor, los cuales sirven para tomar los datos que provienen del modelo.



Ayuda.cshtml: se trata de la vista correspondiente a la página de ayuda.

Contacto.cshtml: consiste en la vista correspondiente a la página de contacto.

Index:cshtml: en la vista de index, se encuentra el código principal de la aplicación.

_Layout.cshtml: sirve para la declaración de código común para todas las vistas del proyecto.

_ViewStart.cshtml: establece la vista de inicio.

Figura 18: Vistas del proyecto [28]

Controlador:

En el controlador nos encontramos con los tres controladores correspondientes a las vistas principales nombradas anteriormente. Estos son: AyudaController.cs, ContactoController.cs y HomeController.cs. Nos encontramos frente a archivos ".cs", en el lenguaje de programación C#.

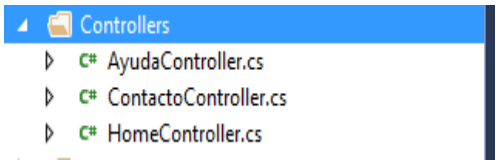


Figura 19: Controladores del proyecto [28]

- AyudaController.cs: se encuentra la lógica correspondiente a la página de ayuda, para el cambio de página entra las demás.
- ContactoController.cs.: se encuentra la lógica correspondiente a la página de contacto, para el cambio de página entra las demás.
- HomeController.cs: nos encontramos al archivo que contiene la lógica principal de toda la aplicación web, la encargada de realizar las consultas SPARQL al repositorio semántico.

Modelo:

En cuanto al modelo nos encontramos frente al elemento que se encarga de almacenar los datos obtenidos de las consultas, y debido a la utilización de datos no muy complejos, se decidió incluir por medio de un modelo estándar y predefinido.

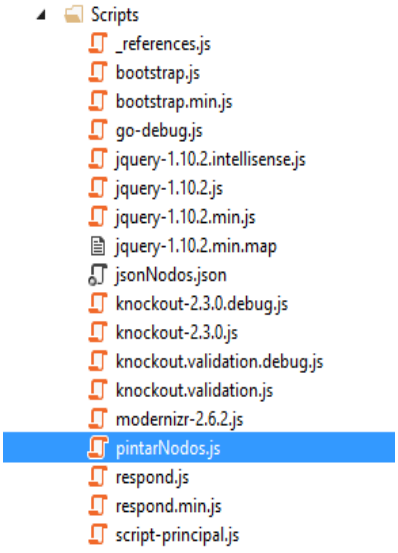
```
ViewData["Nodos"] = json;

ViewData["Nodo"] = json2;

    }
    return View();
}
```

Figura 20: Modelo del proyecto [28]

De esta manera se consigue trasladar los datos desde el controlador a su vista correspondiente de una manera sencilla y efectiva.

Scripts:

Este apartado se ha querido destacar, debido a la creación de otro de los elementos principales en el proyecto, el cual se encarga de llevar a cabo la representación gráfica de los nodos en la vista, el script: pintarNodos.js



pintarNodos.js: En dicho script se encuentra todo el código basado en GoJS, y JavaScript que se encarga de otorgar a la aplicación toda la funcionalidad necesaria.

Figura 21: Scripts del proyecto (destacado pintarNodos.js) [28]

Una vez resumido cómo se ha organizado el proyecto en general, pasaremos a explicar los elementos más importantes en el desarrollo de todo el proyecto, como son:

- HomeController.cs
- pintarNodos.js
- Index.cshtml
- _Layout.cshtml

HomeController.cs

Método principal de la aplicación web, desarrollado enteramente en C#, aunque con la utilización de la librería dotNetRDF, la cual es una librería de código abierto para C#, que nos proporciona una potente y ágil herramienta para realizar las consultas SPARQL al repositorio semántico. Debido a que se decidió incluir el diagrama completo se desarrolló un método recursivo para obtener el diagrama de la interacción al completo.

El código se estructura de la siguiente manera:

Declaración de variables y obtención de valores de URL:

El primer paso en la lógica en C#, es la encargada de declarar las variables globales que se van a utilizar durante el desarrollo de la aplicación web, posteriormente se obtienen los dos valores provenientes de la url: interaction y class_type. Por medio de la herramienta que ofrece C#, "Request.QueryString".

Primera Consulta SPARQL:

Para la realización de la primera consulta SPARQL, lo primero que se debe de realizar es la declaración de los prefijos utilizados para todas las consultas que se realizan a lo largo de la lógica.

Prefijos:

```
PREFIX RO: <http://www.obofoundry.org/ro/ro.owl#>
PREFIX SIO: <http://semanticscience.org/resource/>
PREFIX EDAM:  <http://edamontology.org/>
PREFIX PHIO: <http://linkeddata.systems/ontologies/SemanticPHIBase#>
PREFIX PUBMED:  <http://linkedlifedata.com/resource/pubmed/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX up:  <http://purl.uniprot.org/core/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
```

Figura 22: Prefijos consultas SPARQL [28]

Después se desarrolló la consulta básica que se va a utilizar para ir obteniendo los datos necesarios, para conseguir construir el diagrama completo de la interacción.

```
SELECT DISTINCT ?disn_1 ?label ?rel ?valor
 WHERE {
?disn_1 ?rel ?valor .
?disn_1 rdfs:label ?label
 FILTER(( ?disn_1 = <Interaccion>))}
```

Figura 23: Consulta SPARQL [28]

Una vez diseñada esta consulta, se declaran los parámetros necesarios para la realización de la consulta por medio de dotNetRDF [18]:

- Se debe de declarar el endpoint donde se desea realizar la consulta:

```
SparqlRemoteEndpoint endpoint = new SparqlRemoteEndpoint(new Uri("http://linkeddata.systems:8890/sparql"));
```

Figura 24: Endpoint para la consulta [28]

- Posteriormente se realiza la consulta:

```
SparqlResultSet resultQuery1 = endpoint.QueryWithResultSet(query);
```

Figura 25: Consulta al endpoint establecido [28]

- Los datos obtenidos se guardan para, posteriormente, realizar una llamada al método recursivo.

Método recursivo:

Este método, es el encargado de realizar todas las llamadas necesarias para construir todos los nodos correspondientes al diagrama completo de la interacción. Para ello se utiliza de guía el esquema correspondiente al repositorio semántico, el cual se puede ver en el ANEXO-3.

Para ello se toman los datos provenientes de la primera consulta realizada, partiendo de esos datos obtenidos. Dichos datos se recorren de la siguiente manera:

- Primero se recorren los datos obtenidos para de cada uno de ellos, para obtener los siguientes valores: disname, label, relación, y el valor.

```
string disn =
string label
string rel =
string valor
```

Figura 26: Valores de cada nodo [28]

- Una vez obtenidos esos datos en relación a cada resultado obtenido, se recorren resultado a resultado hasta obtener el valor del nodo, por medio de un método “foreach()”.
- Se recorren los resultados de nuevo para obtener la descripción del nodo, si es que existe. Estos datos serán los que posteriormente se representarán en el cuadro de resultados.
- Una vez obtenido el nodo y su información, se vuelven a recorrer dichos resultados para obtener las relaciones que tiene dicho nodo. En este punto en concreto, se debe de llamar el método a sí mismo (como método recursivo) por cada relación obtenida de dicho nodo, para de esta forma profundizar en

el árbol. Estas relaciones se recorren de la misma manera, por medio de un método "foreach()", para de esta forma recorrer todas las relaciones antes de acabar.

- Una vez se acaba de profundizar en el árbol, se retrocede hasta volver a la interacción y finalizar el método recursivo. Durante la realización se fue consultando paralelamente el esquema guía del repositorio semántico de ANEXO-5.
- Durante todo el proceso se van guardando todos los datos que se van obteniendo en el formato "JSON" para que posteriormente el método pintarNodos.js sea capaz de representar tanto el diagrama local, como el diagrama completo.

Index.cshtml

Esta vista es la más representativa del proyecto, ya que es la encargada de representar el diagrama local, cuadro de resultados, y diagrama completo. Todo ello en HTML, excepto alguna sección en Razor [29].

Elementos principales:

- El código Razor, encargado de la lógica correspondiente a la navegación entre las diferentes vistas que componen la aplicación web.
- A continuación nos encontramos frente al código correspondiente al título, la cabecera, y el logo.
- Después nos encontramos con el diagrama local, el cual se declara de la siguiente manera:

```
<div id="sample">
    <div id="localDiagram" style="height:350px;width:100%;border:1px solid black;margin:2px"></div>
```

Figura 27: Declaración de diagrama local [28]

Esto se realiza así, para que posteriormente sea tratado como un elemento Canvas donde poder representar los nodos y sus relaciones.

- A continuación se incorpora el cuadro de resultados, en el cual se mostrará la información adicional de cada nodo.
- Posteriormente se creó un botón correspondiente a la ocultación del "diagrama completo".
- Por último, se declara el diagrama completo, de una manera similar al diagrama local.

```
<div id="fullDiagram" class="visible"style="height:550px;width:100%;border:1px solid black;margin:2px;"></div>
```

Figura 28: Declaración de diagrama completo [28]

- Para finalizar, en esta vista se montan los datos obtenidos del modelo:

```
<textarea id="mySavedModel" style="width:100%;height:250px;display:none">
    @ViewData["Nodos"]
</textarea>
<textarea id="Nodo" style="width:100%;height:250px;display:none">
    @Html.Raw(ViewData["Nodo"])
</textarea>
```

Figura 29: Carga de datos del modelo [28]

De esta manera el método pintarNodos.js consumirá los datos desde la vista.

pintarNodos.js

Otro de los elementos más importantes en el desarrollo del proyecto, sería el script encargado de realizar toda la representación de los nodos, mediante la librería GoJS.

Dicho script se compone de las siguientes secciones:

- Primero se declaran los parámetros relativos al diagrama completo, mediante el identificador, que se le ha dado en el código html "myFullDiagram", dotándole de las características más importantes, como:
    - La escala a la que se encuentra el diagrama.
    - Solo lectura (isReadOnly).
    - Establecimiento del contenido, el cual es un árbol, dotándolo de la orientación y los nodos que se pueden seleccionar a la vez (uno).
    - Llamada al método que se encarga de la navegación en el diagrama completo.
- A continuación se declara el diagrama local, de una manera similar al anterior por medio del identificador "myLocalDiagram" que se le dio en el código HTML.
- Después se declararan los parámetros más relevantes de todos los nodos que van a ser representados en ambos diagramas:
    - Información centrada en el nodo.
    - Key, la cual representa el nombre del nodo.
    - Texto, el cual representa la información adicional de cada nodo.
    - Los estilos correspondientes al nodo.
- Después se realizan los métodos encargados de resaltar el nodo en el que nos encontramos en el diagrama completo. Dicho método debe de actualizarse cada vez que se realiza cualquier click en un nodo del diagrama. Además se añadirán las características y estilos que correspondan para realizar la señalización del nodo en cuestión.

- Posteriormente, se lleva a cabo el método encargado de seleccionar el nodo central que se va a representar en el diagrama completo, junto a los dos nodos siguientes y anteriores (si existiesen), que es capaz de representar este diagrama.
- Por último hay que destacar el método encargado de consumir los datos que se encuentran en la vista "Index.cshtml" en formato de JSON mediante el método "go.Mdel.fromJson".

```
function setupDiagram() {
    myFullDiagram.model = go.Model.fromJson(document.getElementById("mySavedModel").value);
}
```

Figura 30: Carga de los datos de la vista [28]

_Layout.cshtml:

Del mismo modo, y aunque se trate de un elemento secundario del proyecto, se ha querido destacar la clase del Layout, debido a su gran utilidad a la hora de que nos permite declarar elementos comunes a todas las páginas. Así mismo, aunque se trate de un proyecto relativamente pequeño, en un proyecto mayor podría servir de mucho, siendo utilizada para evitarnos muchas declaraciones innecesarias en multitud de vistas, motivo por el cual se ha querido destacar gracias al gran valor que aporta al proyecto.

- En este caso se ha utilizado para la declaración de la hoja de estilos, de esta manera conseguimos que la declaración se realice para las tres vistas que componen el proyecto.

```
@Styles.Render("~/Content/Site.css")
```

Figura 31: Carga de CSS desde layout [28]

- Además nos proporciona un mecanismo para declarar todos los scripts necesarios para las tres vistas del proyecto.

```
@Scripts.Render("~/Scripts/go-debug.js")
@Scripts.Render("~/Scripts/pintarNodos.js")
@Scripts.Render("~/Scripts/script-principal.js")
@Scripts.Render("~/Scripts/jquery-1.10.2.js")
```

Figura 32: Carga de Scripts desde layout [28]

2.8 Pruebas

En este apartado se pretende aportar un pequeño resumen de las pruebas que se han ido realizando tanto a lo largo de todo el proceso de desarrollo e implementación de la plataforma, como pruebas posteriores.

2.8.1 Pruebas realizadas durante el desarrollo.

Durante el desarrollo de toda la plataforma, se han ido realizando diferentes pruebas, tanto unitarias por sección, como pruebas completas cuando se iban desarrollando secciones.

Debido a que se fue realizando por diferentes fases, primero se desarrolló la página web, posteriormente la plataforma de visualización de nodos sin datos, y posteriormente la lógica interna de la aplicación para la obtención de los datos, motivo por el que se van a dividir las pruebas en estos apartados.

Pruebas desarrollo de la página web.

En cuanto a la propia página web, se desarrolló al comienzo del proyecto y debido a la poca funcionalidad aplicada en esta fase del proyecto, las pruebas se realizaron básicamente en el intercambio de las diferentes vistas que componen la plataforma (inicio, ayuda, contacto), y a la hoja estilos que se le aplicó a toda la página web.

Además de dichas pruebas, también se llevaron a cabo pruebas a algunas funcionalidades desarrolladas en JavaScrip, como la interacción realizada con el botón “plataforma completa”.

Pruebas desarrollo de la visualización de nodos.

Posteriormente se desarrolló la plataforma de visualización basada en la librería específica para este propósito GoJS. En esta fase de desarrollo, sí se llevaron a cabo pruebas más exhaustivas, debido a que la plataforma de visualización era clave para el resto del proyecto. Por lo tanto, se realizaron las siguientes pruebas para permitir la representación de toda la información necesaria en la visualización.

Sección	Prueba realizadas
Representación de los nodos	Se llevaron a cabo las diferentes pruebas para que los nodos se representasen de la manera óptima.
Propiedades de los nodos	Se llevaron a cabo las pruebas necesarias para que cada nodo tuviese las propiedades necesarias.
Interacción con los nodos	Fue necesario el desarrollo de todos los tipos de interacciones que se pueden realizar con los nodos, y por lo tanto se tuvieron que llevar a cabo las pruebas pertinentes para cumplir con los objetivos.
Representación diagrama	Por último se realizaron pruebas a la creación de los diagramas tanto local como completo con datos ficticios para que cumpliesen con todas las especificaciones.

Tabla 2: Pruebas plataforma de visualización [28]

Pruebas lógica interna.

Una vez desarrollada tanto la página web, como la plataforma de visualización, llegó la fase de desarrollo de la lógica interna que se iba a manejar en la plataforma. Para ello se hizo uso del lenguaje C# como se ha dicho anteriormente, junto a la herramienta dotNetRDF. En esta fase del desarrollo, nos encontramos frente al mayor número de pruebas realizadas a la plataforma, debido a su gran complejidad a la hora de obtener todos los datos necesarios para el funcionamiento óptimo de la plataforma.

Sección	Prueba realizadas
Primera consulta	Primero se realizaron las pruebas pertinentes a la herramienta dotNetRDF, hasta conseguir realizar las primeras consultas RDF.
Método recursivo	Se fueron realizando pruebas, a cada paso del desarrollo del método recursivo, hasta que se consiguió montar el diagrama completo de una interacción.
Árbol Completo	Una vez acabadas las pruebas al método recursivo, se llevaron a cabo pruebas exhaustivas con diferentes interacciones para garantizar su funcionamiento óptimo.
Prueba completa a la lógica	Por último se realizaron pruebas completas a la lógica de la aplicación, pasando los parámetros por la url, similares al sistema final.

Tabla 3: Pruebas realizadas a la lógica de la plataforma [28]

Una vez desarrolladas las diferentes secciones, se realizaron pruebas a la interacción necesaria para la trasferencia de los datos obtenidos en la lógica interna, hasta la plataforma de visualización, pasando previamente por la vista principal en la página web.

2.8.2 Pruebas realizadas al finalizar el desarrollo.

Una vez se llevó a cabo el desarrollo completo de la plataforma, se llevó a cabo una batería de pruebas completa, para garantizar todas las funcionalidades establecidas en los objetivos del proyecto, dichas pruebas se realizaron comprobando todos los resultados con el repositorio y los datos que éste proporciona.

2.9 MANUAL DE USUARIO.

En este apartado se pretende exponer un pequeño manual que proporcionará al usuario una guía rápida para poder manejar rápidamente la plataforma de visualización y de esta manera exponer todas las posibilidades que aporta la plataforma desarrollada. Para que este manual no resulte complejo se han realizado capturas de la plataforma para ilustrar las acciones.

De esta forma, se van a exponer dos casos de uso distintos; uno mostrando solo el diagrama local, y otro mostrando ambos diagramas, tanto el diagrama local como el completo.

Elementos Comunes:

Se van a establecer los elementos comunes entre los distintos casos de uso.

La cabecera es el elemento común entre los distintos casos de uso. En la página de inicio la cabecera se puede observar con los tres enlaces siguientes: inicio, contacto y ayuda.

Figura 33: Cabecera General [28]

Una vez se accede a la página de contacto cambiará el título de la cabecera.

Figura 34: Cabecera selección contacto [28]

Así mismo al acceder a la página de ayuda, también cambiará el título de la cabecera.

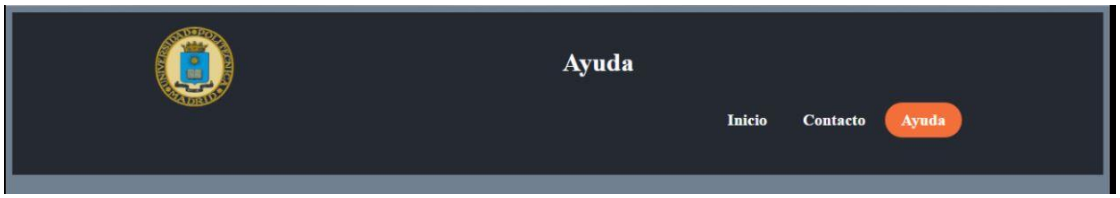


Figura 35: Cabecera selección ayuda [28]

Opción 1 (solo diagrama local):

En este caso se accede a la página mediante la url y un único valor en la url, el correspondiente a la interacción que se quiere consultar.

URL=http://localhost:52312/?interaction=http://linkeddata.systems/SemanticPHIBase/Resource/interaction/INT_00003

Según accedemos a la plataforma se puede observar el primer diagrama, el cual corresponde al diagrama local, que representa hasta los dos nodos de distancia, y cómo se encuentra el foco centrado en la interacción.

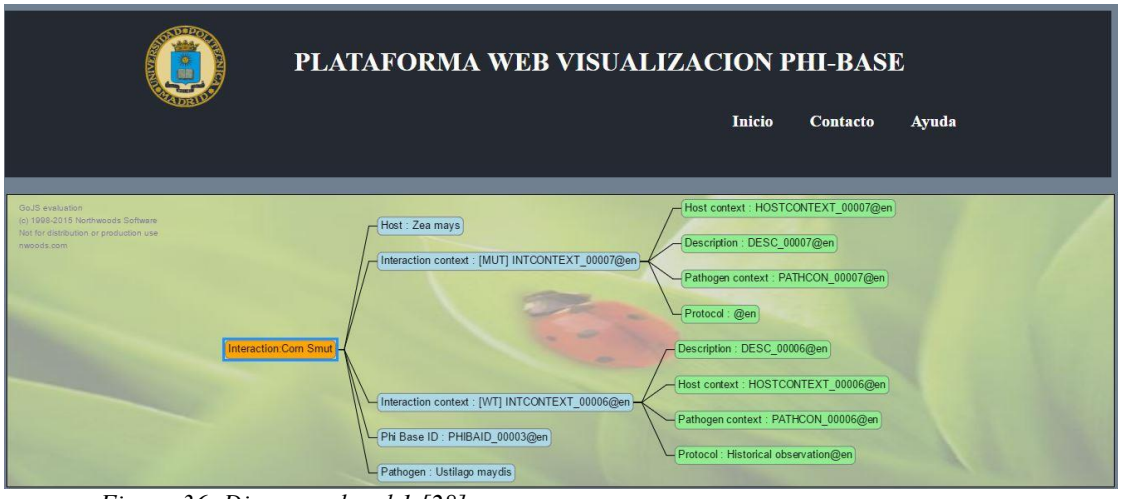


Figura 36: Diagrama local 1 [28]

A continuación del diagrama local se puede observar el cuadro de resultados, el cual se encarga de mostrar la información adicional (si la hay), del nodo en el que se encuentra el foco en el diagrama local.

Figura 37: Cuadro de resultados 1 [28]

De esta manera si realizamos un click en cualquier nodo el cual no contenga información, mostrará el siguiente mensaje: "No hay información disponible", como se puede observar a continuación.

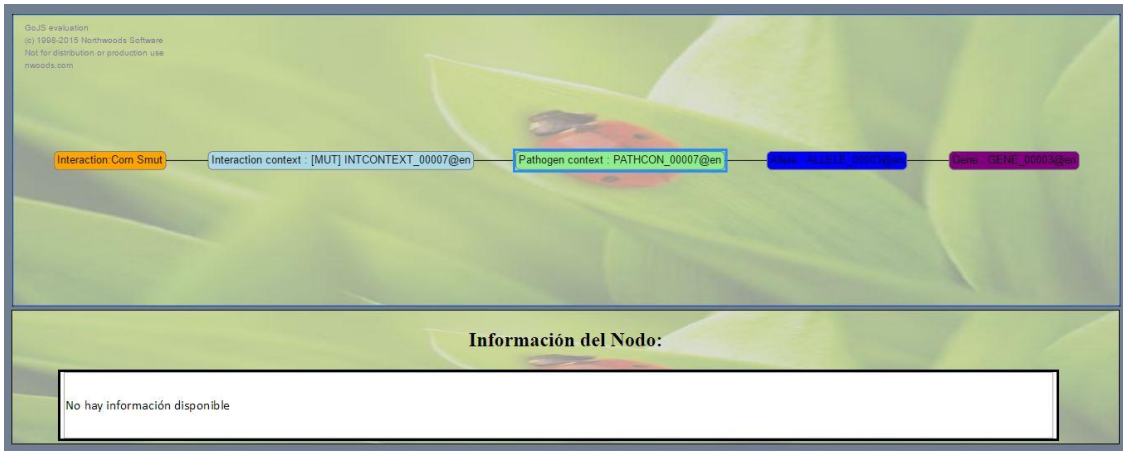


Figura 38: Diagrama local y cuadro de resultados [28]

Sin embargo si realizáramos un click en cualquier nodo que contenga información, en el cuadro de resultados se mostrará dicha información, como por ejemplo si se realiza este click sobre el nodo "PHI Base ID", como se puede observar a continuación:

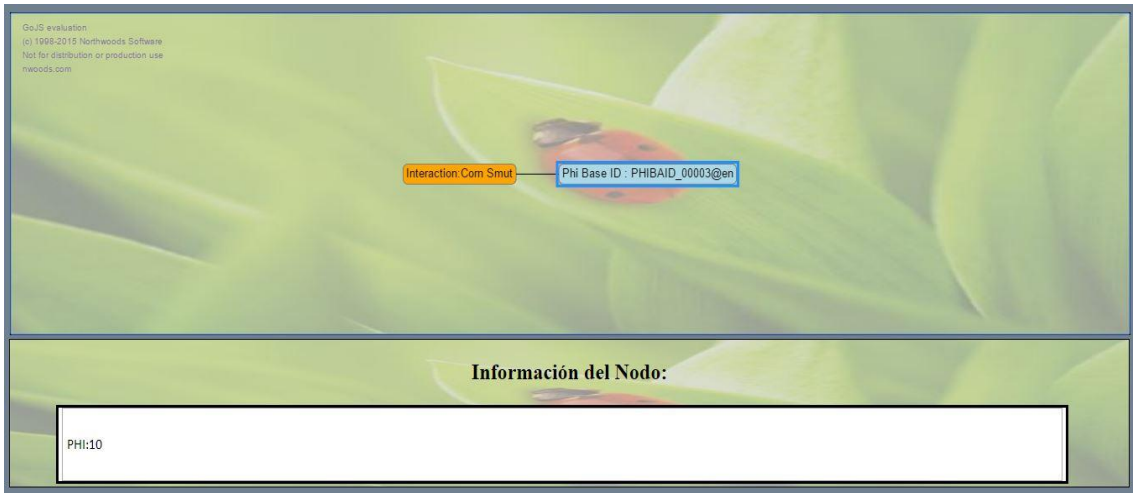


Figura 39: Cuadro de resultados con información adicional [28]

A continuacion nos encontramos con un botón, para la interacción con el usuario encargado de mostrar u ocultar el diagrama completo. Por lo tanto si se realiza click sobre el boton, se debe de ocultar el diagrama completo para mejorar la visualizacion por parte del usuario.

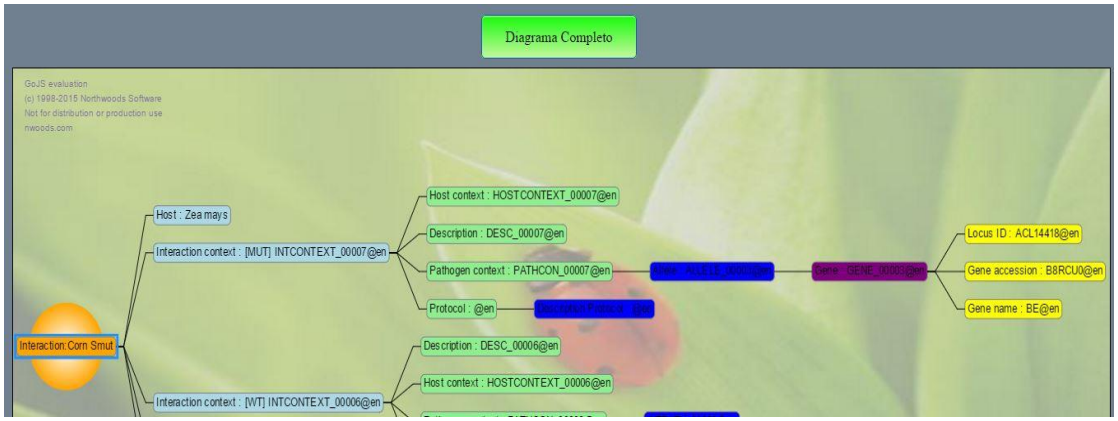


Figura 40: Botón diagrama completo [28]

Al hacer click sobre el botón, el resultado es la desaparición del diagrama completo, como se ha dicho anteriormente, tal y como se puede ver a continuación.

Figura 41: Click botón diagrama completo [28]

Por último, nos encontramos frente al último elemento de la plataforma web, el diagrama completo de la interacción. Se puede observar como al iniciar la aplicación web, el diagrama se encuentra con el foco en la interacción debido a los parámetros que provienen de la url.

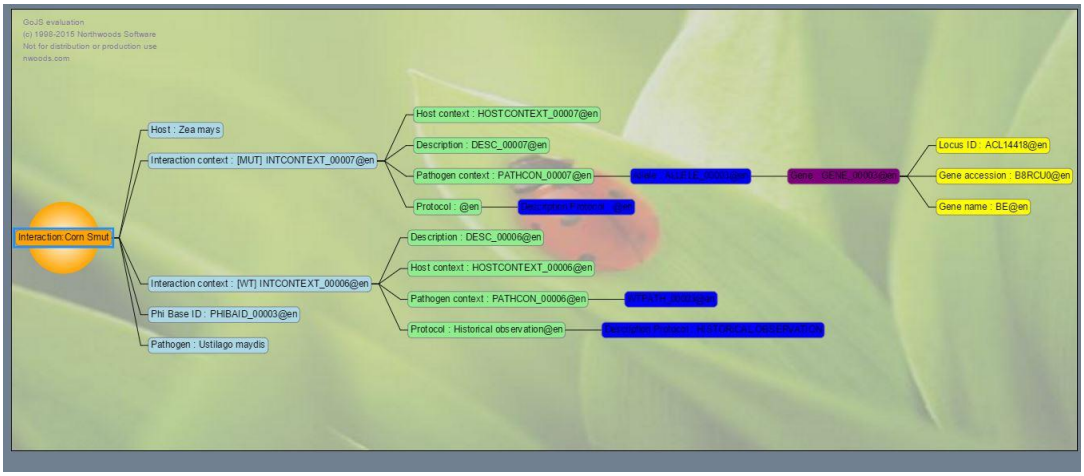


Figura 42: Diagrama completo 1 [28]

De esta manera si navegamos por el diagrama local, por ejemplo a la description,

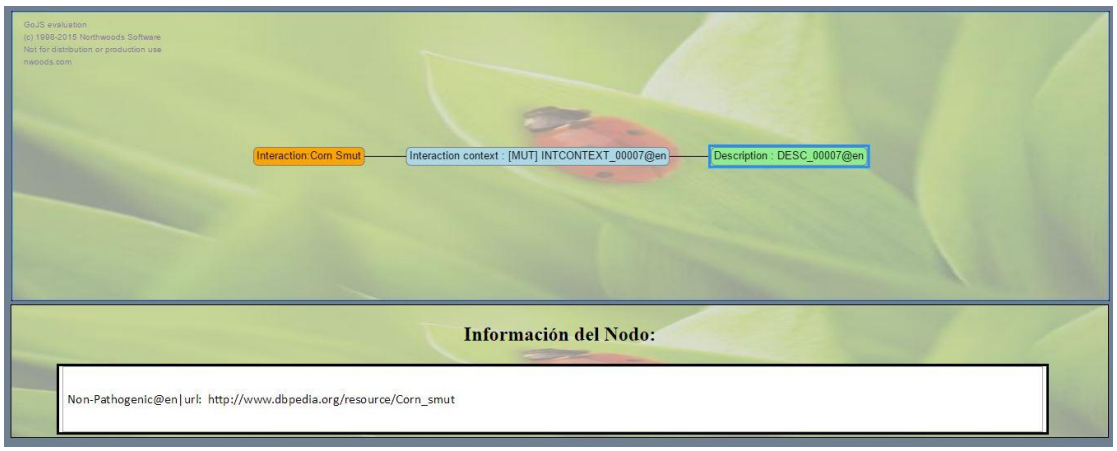


Figura 43: Interacción entre diagrama local y diagrama completo [28]

el diagrama completo se va actualizando al mismo tiempo, para que el usuario pueda localizarse en todo momento.

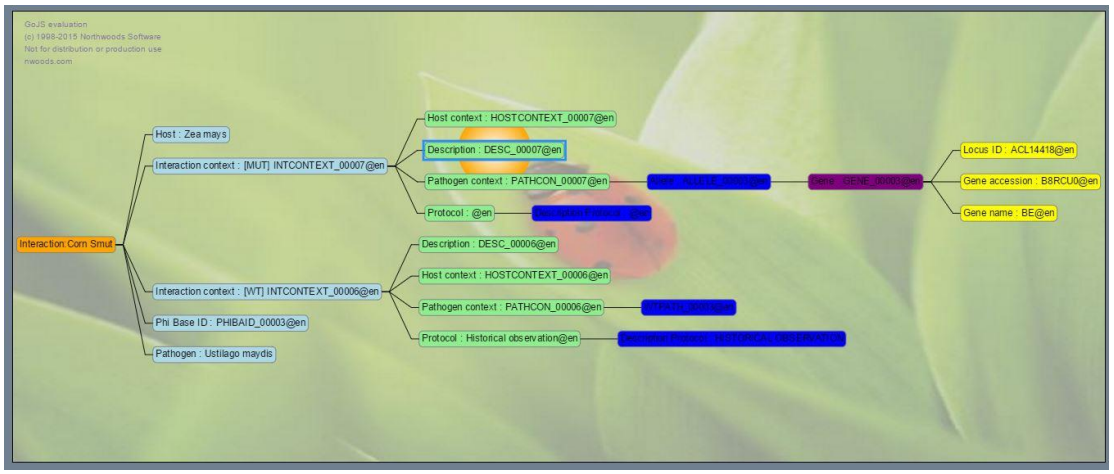


Figura 44: Actualización diagrama completo [28]

Además de esto, si se realiza un click en cualquier nodo del diagrama completo se actualizará el diagrama local a la vez, de esta manera se puede navegar con total libertad por el diagrama al completo.

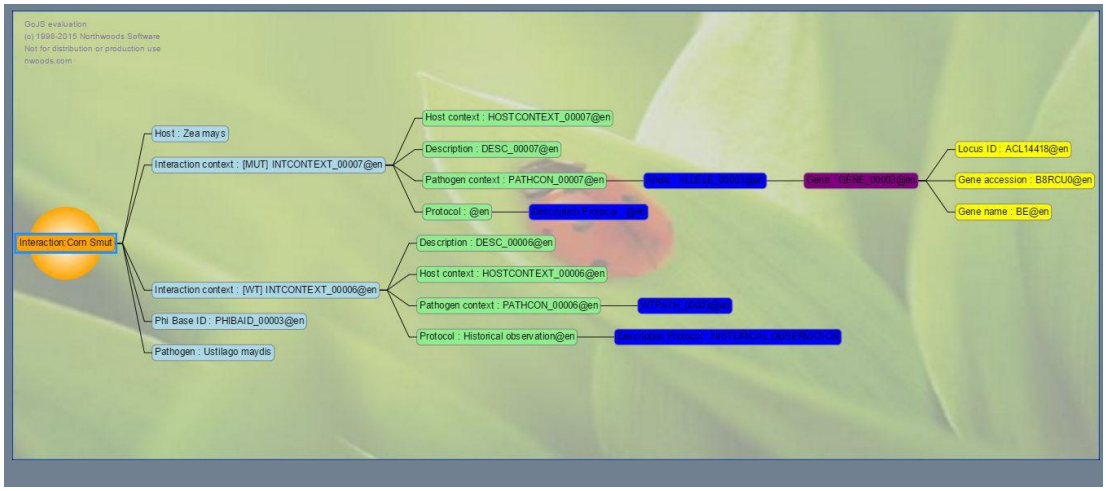


Figura 45: Interacción diagrama completo y diagrama local [28]

Se actualiza el diagrama local a la vez.

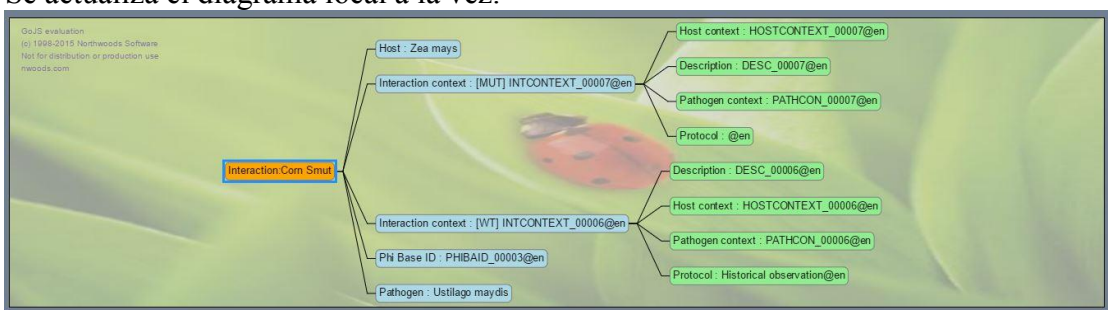


Figura 46: Actualización diagrama local [28]

Opción 2 (Interaction + nodo):

Se realiza el acceso a la plataforma añadiendo dos parámetros a la url, la interacción y el valor del nodo en el que se quiere centrar la plataforma.

URL=http://localhost:52312/?interaction=http://linkeddata.systems/SemanticPHIBase/Resource/interaction/INT_00002&class_type=http://linkeddata.systems/SemanticPHIBase/Resource/gene/GENE_00002

En esta segunda opción se aplican las mismas funcionalidades que en el apartado anterior, tanto las interacciones en el diagrama local, como las que se realizan con el diagrama completo, además de la funcionalidad del botón de “diagrama completo”, con la única diferencia en relación al modo en que se estructuran tanto el diagrama local, como el diagrama completo al iniciar la plataforma.

El diagrama local aparece centrado en el segundo parámetro que se le dio a la url en el momento de arrancar la plataforma, como se puede apreciar a continuación.

Figura 47: Diagrama local 2 [28]

Respecto al diagrama competo, el foco se centra del mismo modo, en el segundo parámetro, al igual que el diagrama local.

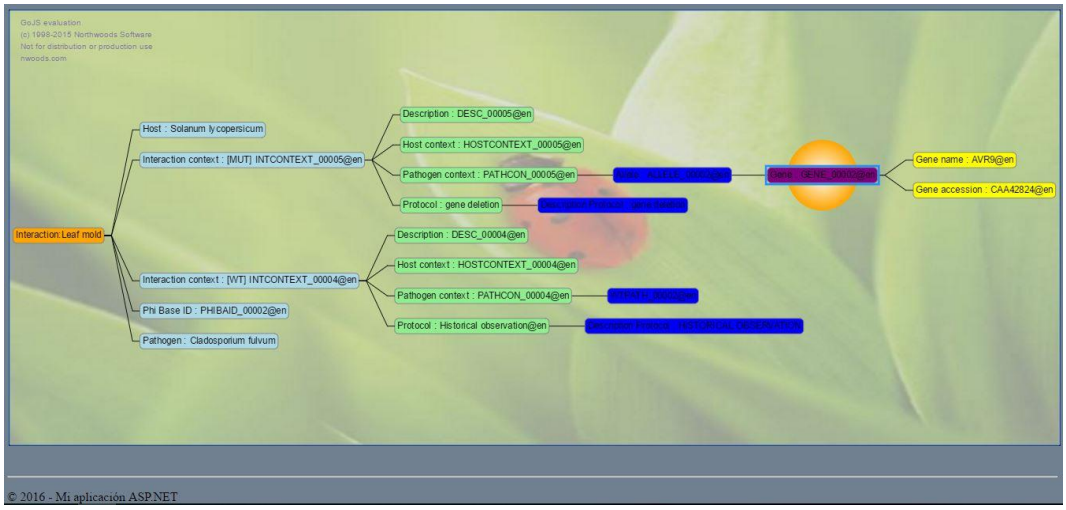


Figura 48: Diagrama completo 2 [28]

El presente manual de usuario de las funcionalidades básicas de la plataforma, tanto lo relativo a la opción 1, como a la opcion2, se encuentra disponible para la consulta del usuario dentro de la plataforma web, en la sección de ayuda.

2.10 INTEGRACIÓN ENTRE PLATAFORMAS.

El proyecto al completo consta de dos grandes módulos que se han desarrollado de forma paralela, los cuales son:

- La plataforma 1. Es la encargada de realizar las consultas propiamente dichas y de esta manera proporcionar al usuario una forma fácil y visual para realizar un filtrado de los resultados que quiere obtener.
- La plataforma 2. Es la correspondiente a este trabajo de fin de grado, la encargada de una vez obtenidos dichos resultados, proporcionar al usuario una plataforma de visualización para dichos resultados, de forma que el usuario pueda interaccionar con dichos datos.

Por lo tanto la interacción entre dichas plataformas se realizará por medio del paso de ciertos parámetros desde la plataforma uno, hacia la plataforma dos. Este paso de parámetros se llevará a cabo por medio de la url, mediante los parámetros de: "interaction" y "class_type". Un ejemplo de url concreto sería:

http://localhost/?interaction=<http://linkeddata.systems/SemanticPHIBase/Resource/interaction/INT_03115>&class_type=<http://linkeddata.systems/SemanticPHIBase/Resource/host/HOST_03115>

De este modo, existirán dos tipos de formas de realizar el paso de parámetros entre ambas plataformas:

- Realizando el paso de parámetros únicamente de la interacción que haya seleccionado el usuario.
- Realizando el paso de parámetros de la interacción que haya seleccionado el usuario, y además el parámetro correspondiente a cualquier otro nodo dentro de dicha interacción.

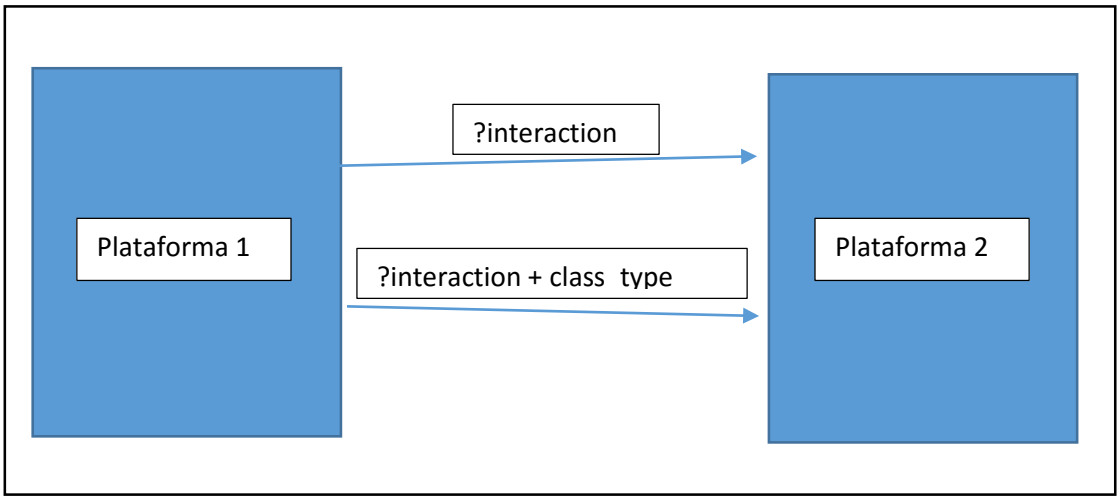


Figura 49: Paso de parámetros entre plataformas [28]

2.10.1 Arquitectura general del sistema

En este apartado, se pretende dar visibilidad general a la arquitectura completa del sistema. De esta manera se pretende aclarar el funcionamiento general de todo el sistema, del módulo de búsqueda junto con el módulo de visualización.

Se pretende ilustrar mediante un esquema, cómo el módulo de búsqueda será el encargado de la obtención de los datos necesarios mediante las consultas, tras lo cual dichos datos generan una serie de resultados con los que el usuario puede interaccionar. Una vez el usuario final seleccione qué datos decide mostrar en la plataforma de visualización, dicho módulo se encargará de procesar los datos necesarios, para posteriormente enviarlos a la plataforma de visualización.

En cuanto a la plataforma de visualización, será la encargada de comprobar que dichos datos son correctos, y una vez comprobados, se ocupará de ir recogiendo poco a poco toda la información que resulte necesaria para que, finalmente, se pueda mostrar la visualización de los datos seleccionados por el usuario, permitiendo que este usuario sea capaz de navegar e interaccionar con la visualización final de los datos.

El esquema de la arquitectura general se puede consultar en el ANEXO-6.

3 CONCLUSIONES Y RESULTADOS.

En esta memoria relativa al trabajo de fin de grado, se ha tomado la decisión, debido a su dimensión y al gran número de líneas de código, de no añadir dicho código en su totalidad. La plataforma web diseñada, ha cumplido con los objetivos de llevar a cabo una representación en forma de nodos de los datos obtenidos del repositorio semántico. Dicha representación se realiza de una manera efectiva, rápida y con la que cualquier usuario con un mínimo de conocimiento del esquema de dicho repositorio semántico, puede sentirse cómodo al utilizarla.

En cuanto a los objetivos, que se plantearon al comienzo del proyecto, hay que señalar que se han ido cumpliendo a lo largo del desarrollo de éste, tanto a la hora de generar la plataforma visual, como de conseguir que dicha plataforma sea lo más intuitiva y visual posible. Además de los objetivos del proyecto, también las planificaciones de todos los hitos que se marcaron en el comienzo del proyecto se han ido cumpliendo correctamente.

Desde un punto de vista personal, el haber realizado este trabajo de fin de grado, me ha aportado una gran variedad de conocimientos nuevos, sobre todo en lo relacionado con la programación web, debido a la gran multitud de tecnologías que se han utilizado en el proyecto. Además de esto, hay que sumarle los conocimientos adquiridos sobre la web semántica, los cuales al no haber cursado ninguna asignatura relacionada con este tema, era muy limitados.

3.1 Líneas futuras.

En relación a las líneas futuras relacionadas con el proyecto, hay que destacar que tanto las tecnologías utilizadas, como la metodología permiten posibles actualizaciones hacia versiones más modernas, sin ningún tipo de problema.

Además se quiere destacar una mejora importante que se puede llevar a cabo en la plataforma, y la cual debido a su complejidad se descartó en un primer momento, y que consistiría en la inclusión de una funcionalidad en la plataforma, en la cual el usuario tendría la posibilidad de poder seleccionar interacciones que estén relacionadas con el mismo “Host” y “Pathogen”.

Por último, se quiere destacar que se pretende presentar el proyecto conjunto con las dos plataformas operativas, para una publicación en el Special Issue de la revista *Journal of Web Semantics* como un caso de estudio [27].

4 ANEXOS

4.1 ANEXO-1

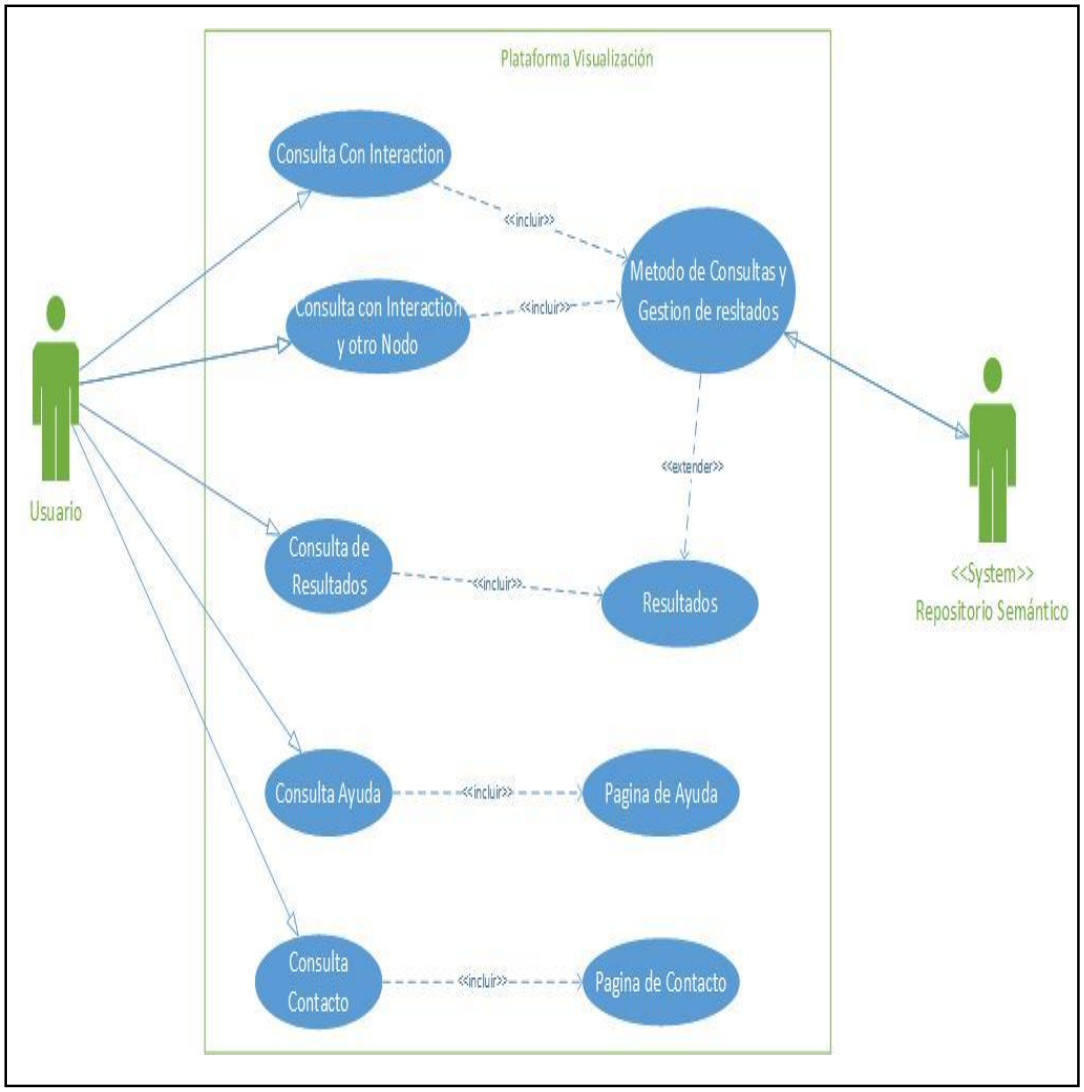


Figura 50: Diagrama casos de uso plataforma de visualización [28]

4.2 ANEXO-2

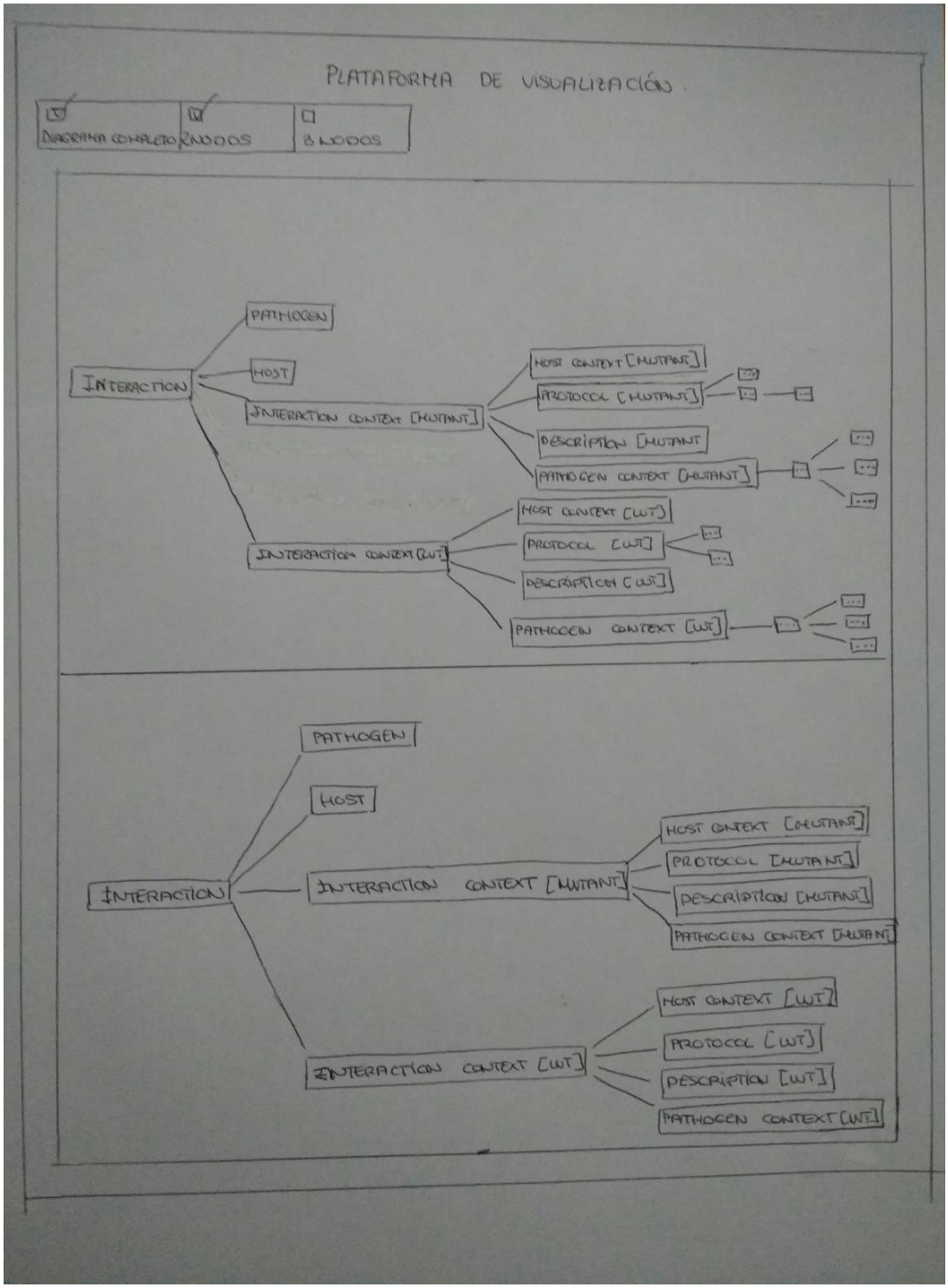


Figura 51: Prototipo de baja fidelidad ampliado [28]

4.3 ANEXO-3

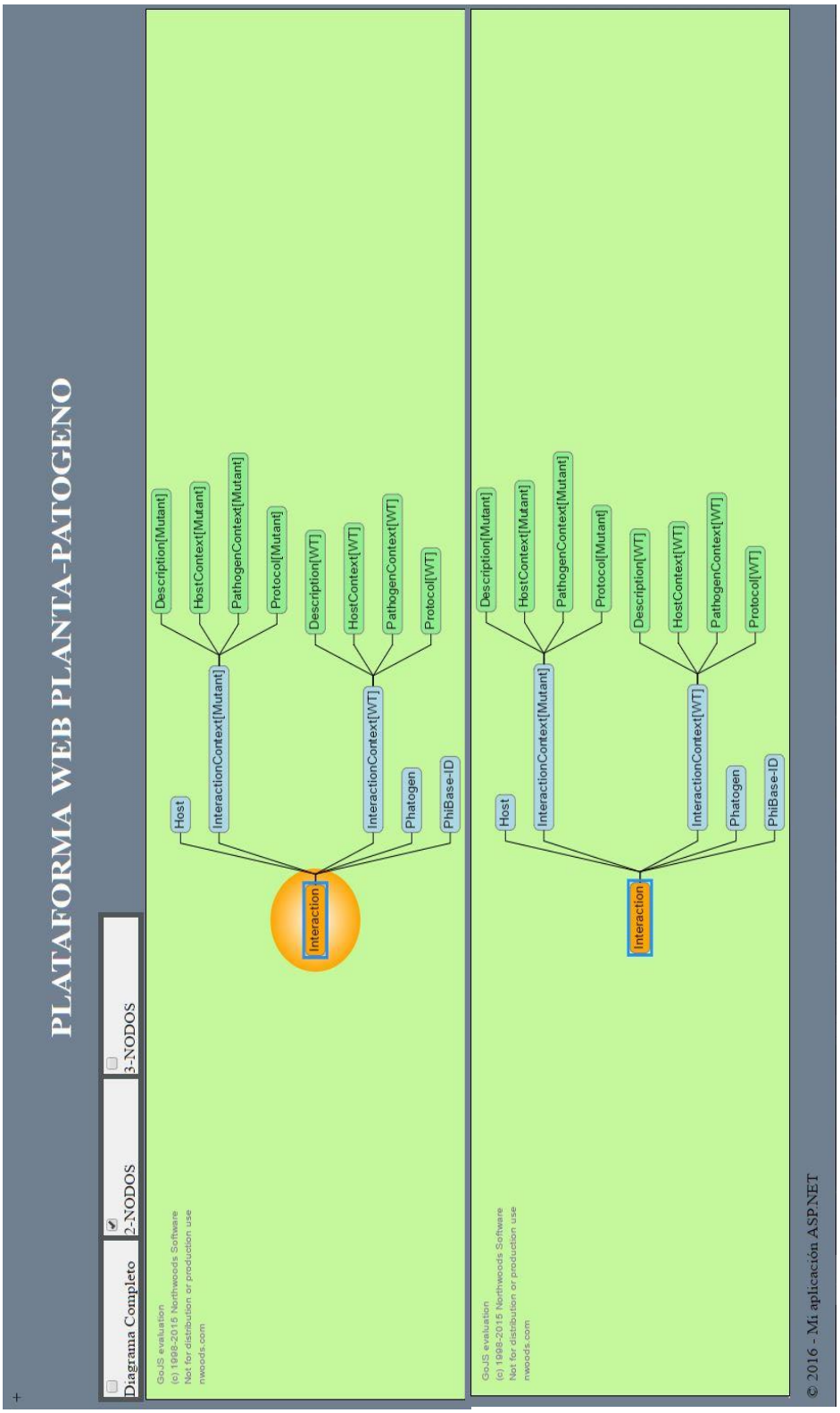


Figura 52: Prototipo de alta fidelidad [28]

4.4 ANEXO-4

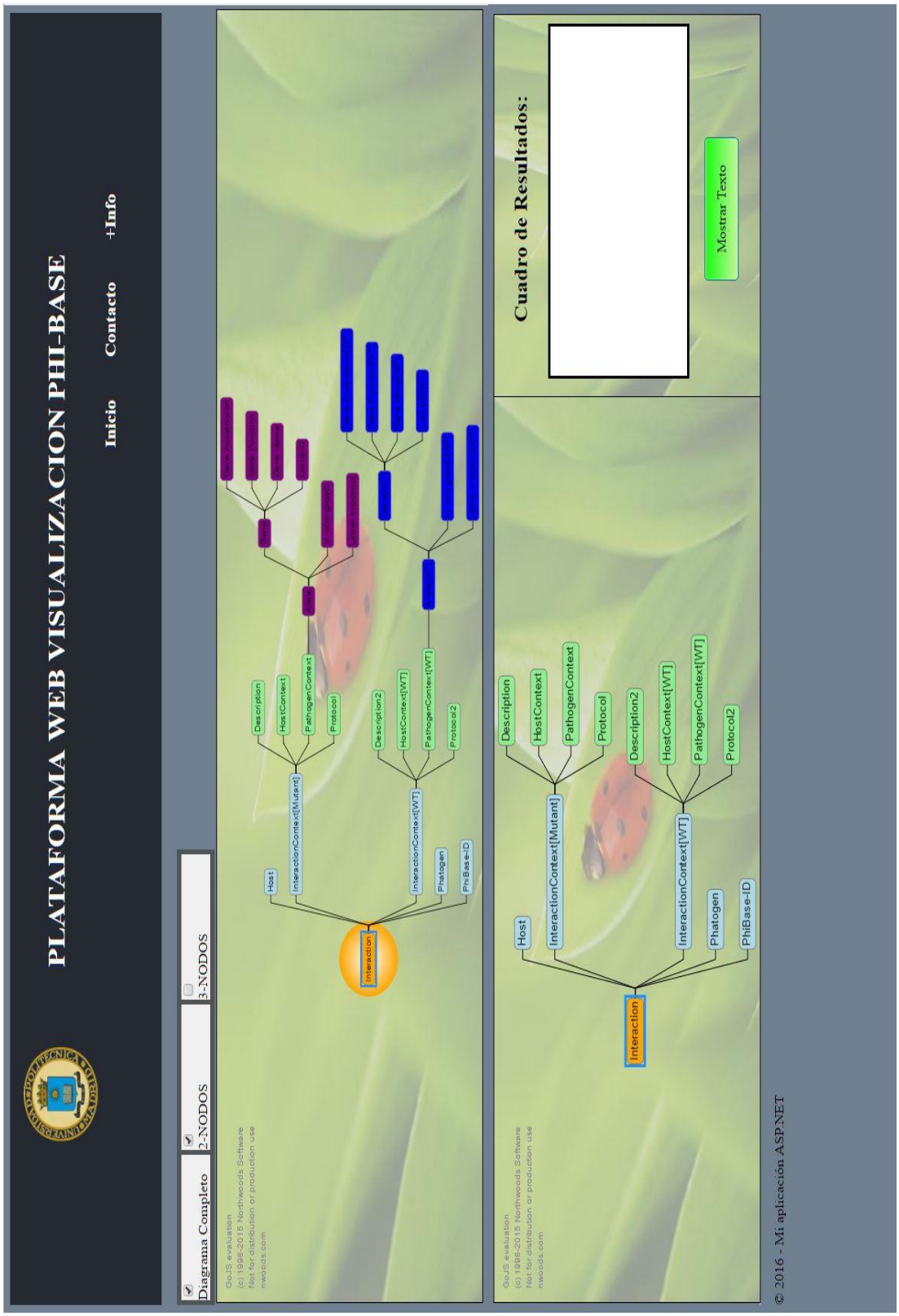
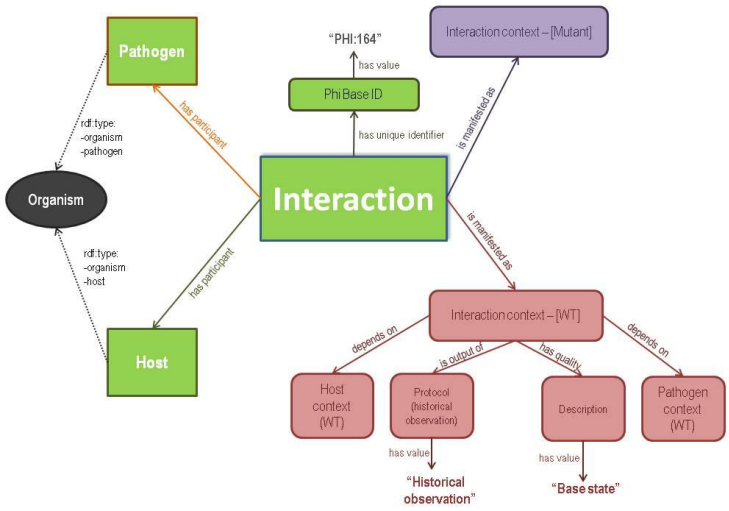


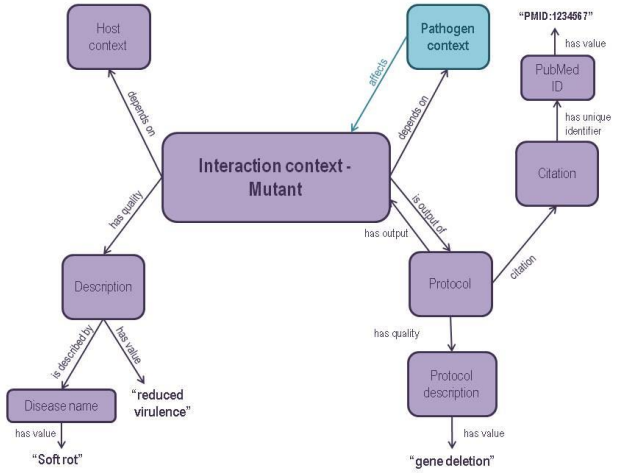
Figura 53: Maqueta de la plataforma [28]

4.5 ANEXO-5

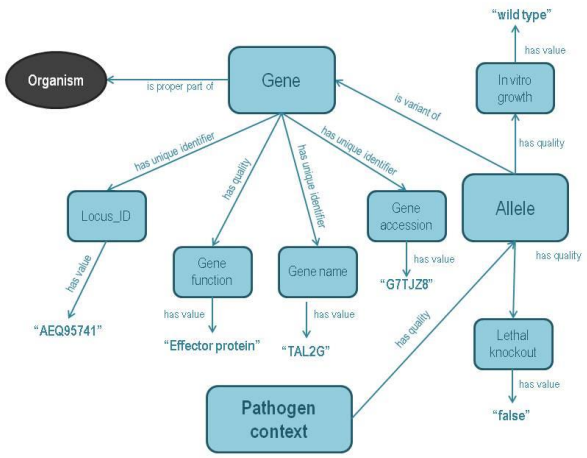
A



B



C



4.6 ANEXO-6

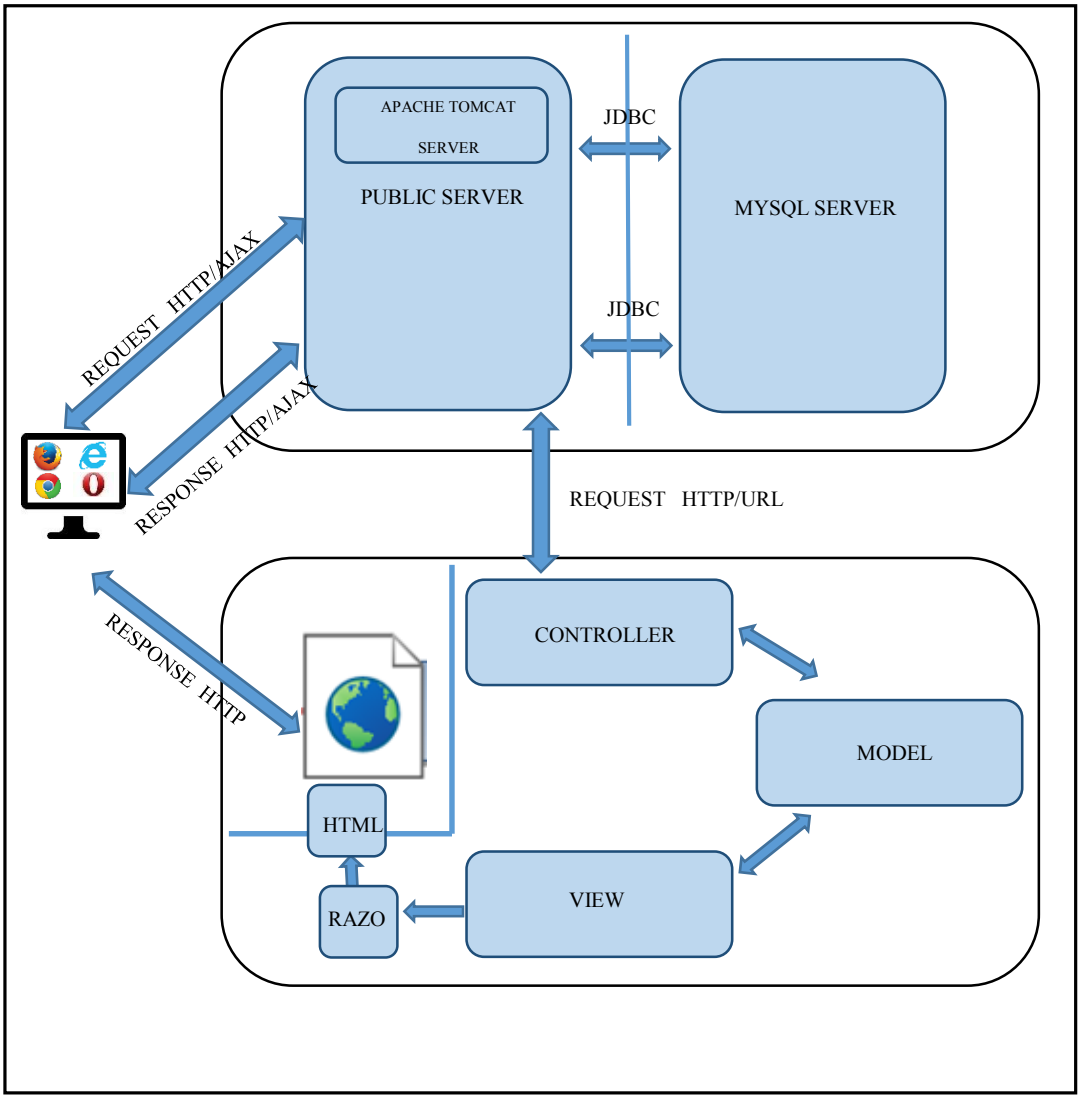


Figura 54: Esquema general [28]

5 BIBLIOGRAFÍA.

[1] "PHI-base : Pathogen Host Interactions", *Phi-base.org*, 2016. [Online]. Disponible: http://www.phi-base.org/. [04- Feb- 2016].

[2] A. Rodríguez-Iglesias, A. Rodríguez-González, A. Irvine, A. Sesma, M. Urban, K. Hammond-Kosack and M. Wilkinson, "Publishing FAIR Data: An Exemplar Methodology Utilizing PHI-Base", *Frontiers in Plant Science*, vol. 7, 2016.

[3] " Semantic web" 2016. [Online]. Disponible: http://www.atana.org/es/descargas/Servicios/Promocion_comercial_desarrollo_del_Sector_TIC/SemWeb-PresenteFuturoParte1.pdf. [15- May- 2016].

[4] "XML y Web Semántica" 2016. [Online]. Disponible: http://openaccess.uoc.edu/webapps/o2/bitstream/10609/23462/7/krisjaenTFC0613memoria.pdf. [04- May- 2016].

[5] "RDF - Semantic Web Standards", *W3.org*, 2016. [Online]. Disponible: https://www.w3.org/RDF/. [13- Apr- 2016].

[6] "SPARQL Lenguaje de consulta para RDF", *Skos.um.es*, 2016. [Online]. Disponible: http://skos.um.es/TR/rdf-sparql-query/. [07- Apr- 2016].

[7] I. JavaScript, "Introducción a JavaScript", *Librosweb.es*, 2016. [Online]. Disponible: https://librosweb.es/libro/javascript/. [08- May- 2016].

[8] I. XHTML, "Introducción a XHTML", *Librosweb.es*, 2016. [Online]. Disponible: http://librosweb.es/libro/xhtml/. [17- May- 2016].

[9] "CSS - CCM", *CCM*, 2016. [Online]. Disponible: http://es.ccm.net/contents/css-2026809048. [19- May- 2016].

[10] "C#", *Msdn.microsoft.com*, 2016. [Online]. Disponible: https://msdn.microsoft.com/es-es/library/kx37x362.aspx. [23- May- 2016].

[11] "GoJS Diagrams for JavaScript and HTML, by Northwoods Software", *Gojs.net*, 2016. [Online]. Disponible: https://gojs.net/latest/index.html. [28- May- 2016].

[12] "Visual Studio - Microsoft Developer Tools", *Visualstudio.com*, 2016. [Online]. Disponible: https://www.visualstudio.com/. [29- May- 2016].

[13] Apuntes en relación a casos de uso, asignatura INGENIERIA DEL SOFTWARE I Y II.

[14] Apuntes requisitos, asignatura INGENIERIA DEL SOFTWARE I Y II.

[15] Apuntes requisitos, asignatura INGENIERIA DEL SOFTWARE I Y II.

[16] "Modelo vista controlador (MVC). Servicio de Informática ASP.NET MVC 3 Framework", *Si.ua.es*, 2016. [Online]. Disponible: http://si.ua.es/es/documentacion/asp-net-mvc-3/1-dia/modelo-vista-controlador-mvc.html. [27- May- 2016].

[17] "Administrar sitios Web ASP.NET en Visual Studio", *Msdn.microsoft.com*, 2016. [Online]. Disponible: https://msdn.microsoft.com/es-es/library/1th9sxc4(v=vs.80).aspx. [28- May- 2016].

[18] "dotNetRDF - Semantic Web, RDF and SPARQL Library for C#/.Net", *Dotnetrdf.org*, 2016. [Online]. Disponible: http://www.dotnetrdf.org/. [26- May- 2016].

[19] "Journal of Web Semantics: cfp: Special Issue on Visualization and Interaction for Ontologies and Linked Data", *Journalofwebsemantics.blogspot.com.es*, 2016. [Online]. Disponible: http://journalofwebsemantics.blogspot.com.es/2016/05/cfp-special-issue-on-visualization-and.html. [29- May- 2016].

[20] " Cómo incluir JavaScript en documentos XHTML (Introducción a JavaScript)", *Librosweb.es*, 2016. [Online]. Disponible: http://librosweb.es/libro/javascript/capitulo_1/como_incluir_javascript_en_documentos_xhtml.html. [27- May- 2016].

[21] “Combinación HTML y CSS” 2016. [Online]. Disponible: http://librosweb.es/libro/xhtml/capitulo_1/html_y_css. [19- May- 2016].

[22] "1.6. Cómo incluir CSS en un documento XHTML (Introducción a CSS)", *Librosweb.es*, 2016. [Online]. Disponible: http://librosweb.es/libro/css/capitulo_1/como_incluir_css_en_un_documento_xhtml.html. [23- May- 2016].

[23] "Get Started with GoJS", *Gojs.net*, 2016. [Online]. Disponible: https://gojs.net/latest/learn/index.html. [23- May- 2016].

[24] "GoJS Class Hierarchy Tree", *Gojs.net*, 2016. [Online]. Disponible: https://gojs.net/latest/samples/classHierarchy.html. [28- May- 2016].

[25] *Microsoft.com*, 2016. [Online]. Disponible: https://www.microsoft.com/spain/prensa/resources/ContentImages/Contents/2010/03/logovisual.jpg. [28- May- 2016].

[26]"Modern, Open, Web Development Tools | Visual Studio", *Visualstudio.com*, 2016. [Online]. Disponible: https://www.visualstudio.com/features/modern-web-tooling-vs. [23- May- 2016].

[27]"Journal of Web Semantics: cfp: Special Issue on Visualization and Interaction for Ontologies and Linked Data", *Journalofwebsemantics.blogspot.com.es*, 2016. [Online]. Disponible: http://journalofwebsemantics.blogspot.com.es/2016/05/cfp-special-issue-on-visualization-and.html. [30- May- 2016].

[28] Autor: Roberto García Salgado.

[29]"ASP.NET Razor Markup", *W3schools.com*, 2016. [Online]. Disponible: http://www.w3schools.com/aspnet/razor_intro.asp. [31- May- 2016].

Este documento esta firmado por

	**Firmante**	CN=tfgm.fi.upm.es, OU=CCFI, O=Facultad de Informatica - UPM, C=ES
	Fecha/Hora	Mon Jun 06 19:42:49 CEST 2016
	Emisor del Certificado	EMAILADDRESS=camanager@fi.upm.es, CN=CA Facultad de Informatica, O=Facultad de Informatica - UPM, C=ES
	Numero de Serie	630
	Metodo	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature)