Universidad Zaragoza

Trabajo Fin de Grado

Navegación visual por datos semánticos guiada por agente software

Visual exploration of semantic data guided by software agent

Autor

Sergio Sota Martínez

Directores

Carlos Bobed Lisbona
Guillermo Esteban Pérez

Escuela de Ingeniería y Arquitectura

2016

Navegación visual por datos semánticos guiada por agente software
Resumen

Desde los inicios de la Web 2.0, los usuarios llevan años no solo consumiendo sino proveyendo información, aumentando el contenido existente en la Web. Desde la aparición de la Web 3.0, se busca que las computadoras sean capaces de entender esa información tradicionalmente orientada al consumo humano. La Web 3.0 dota a la información de semántica y relaciones entre los datos para conseguir que los propios ordenadores sean capaces de entender y procesar la información de la Web, favoreciendo nuevas técnicas de búsqueda y clasificación, aliviando así las tareas del usuario.

Para estructurar los datos y dotarles de semántica, el W3C [16] ha publicado diversas recomendaciones, como el uso de un lenguaje descripción de recursos (RDF) y un lenguaje de especificación de esquemas (Ontología). Tras estructurar los datos de la web, dotarles de semántica e interconectarlos, se obtienen los Datos Enlazados o Linked Data [14]. La Web Semántica trabaja con los datos enlazados, creando la llamada *Web of Data*.

Los datos enlazados son creados para ser entendidos y procesados por máquinas, no por personas. El hecho de estar estructurados, permiten crear herramientas más inteligentes, como navegadores o exploradores. En este proyecto se ha utilizado el paradigma y los datos de la Web 3.0 para crear una herramienta capaz de navegar por los datos semánticos y transformarlos de tal forma que el ser humano sea capaz de trabajar con ellos sin perder la ventaja de los datos enlazados sobre las máquinas.

El objetivo de este trabajo es integrar y mejorar distintas herramientas para desarrollar un agente software que disponga de los medios necesarios para facilitar la navegación sobre datos enlazados. Para que el usuario interactúe con el agente, la interfaz utilizada se basará en una navegación por grafo. Esta visualización debe ser capaz de comprimir visualmente la vasta información de la *Web of Data*, simplificando la navegación por la misma y facilitando la comprensión semántica de los datos de interés del usuario. Para ello se parte de VOX, un agente software que permite hacer búsquedas semánticas sobre la DBPedia. Se mejoran los aspectos visuales (cambiar las metáforas de interacción, mejorar y añadir elementos visuales para mejorar la comprensión del usuario, ordenación espacial de la búsqueda), se crean elementos extra para el apoyo a las búsquedas (creación de migas de pan, búsqueda por palabras clave) y se añade un registro de acciones para mejorar las búsquedas iniciales de los usuarios.

Agradecimientos

Ante todo, me gustaría darle las gracias a mi familia por esta siempre ahí. Por poder contar con ellos en momentos difíciles y por poder sonreír en los buenos momentos. Sin ellos no habría llegado donde me encuentro ahora mismo ni podría haber hecho todo lo que he hecho y lo que me ha definido como soy actualmente.

Me gustaría darle las gracias a mi pareja, la cual me ha estado apoyando a lo largo de toda la carrera y me ha hecho vivir muy buenos momentos que me han servido para crecer como persona.

Gracias a mis amigos por estar ahí y hacerme pasar siempre buenos ratos y ayudándome a olvidar todo el estrés que he tenido.

Gracias a los profesores por tener la paciencia y el temple para soportar mis preguntas y mis dudas a lo largo de todos estos años de estudio que se acercan a su fin, por ahora.

También gracias a mis dos directores: Carlos Bobed y Guillermo Esteban por darme la oportunidad y permitir que este proyecto salga adelante. Sin ellos no hubiese podido enfrentarme a todos los retos que ha supuesto este proyecto ni a todos los quebraderos de cabeza que traía consigo.

Gracias a todos por estar ahí cuando os necesitaba.

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

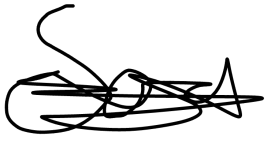
D./Dª. Sergio Sota Martínez,

con nº de DNI 25193796X en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) Grado, (Título del Trabajo) Navegación visual por datos semánticos guiada por agente software,

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 31 de Agosto de 2016



Fdo: Sergio Sota

Índice general

Capítulo 1

Introducción

Este proyecto consiste en el desarrollo de un agente software, cuya función es permitir la visualización y la navegación de los datos enlazados mediante un grafo. La visualización debe ser capaz de comprimir visualmente la vasta información de la *Web of Data*, simplificando la navegación por la misma y facilitando la comprensión semántica de los datos de interés del usuario. La Web Semántica promueve la interoperabilidad entre los sistemas informáticos, mediante la creación de datos enlazados [14] (los datos de la web, dotados de semántica e interconectados). Así, se pueden crear *Agentes inteligentes*, capaces de buscar información o navegar por datos concretos sin la necesidad de un operador humano. La Web Semántica se basa en dos puntos:

- La descripción del significado: Donde los datos se representan mediante (RDF) y los conceptos se definen en el esquema de datos (Ontologías). Estas dos ideas están explicadas en sus respectivas secciones.
- La manipulación de las descripciones, efectuada mediante lógica o motores de inferencia, usando en este caso razonadores semánticos.

La motivación y la problemática del proyecto radican en la falta de herramientas que faciliten la exploración de datos enlazados, tanto para desarrolladores (usuarios intentando comprender la información que hay detrás de un repositorio de datos) como para usuarios finales (usuarios buscando información en dicho repertorio), las cuales permitan a los usuarios entender la información de la Web Semántica de forma sencilla e intuitiva y permitirle una navegación ágil y dinámica. Actualmente la representación y navegación visual es un problema abierto, no existe ninguna aproximación al problema de la representación con exactitud y de forma eficiente.

1.1. Objetivo y alcance

El objetivo de este proyecto ha sido crear una herramienta que facilite la navegación sobre datos enlazados y dé apoyo al usuario en su navegación, mientras se mejora la representación de datos de una manera eficiente. Esta herramienta de navegación, que integra distintos módulos, permite

añadir nuevas técnicas de búsqueda fácilmente, así como interacción hablada con el usuario.

Para esto se parte de VOX [8], un agente software que permite hacer búsquedas semánticas basadas en palabras clave sobre la DBPedia, comunicándose con el usuario mediante una interfaz multimodal.

El esquema de funcionamiento original que se usaba es el que se ve en la Figura 1.1. El usuario tenía dos opciones: escribir las palabras clave, o decir una consulta oralmente. En este caso, la consulta se le mandaba al agente para que la *parseara* y obtuviese las palabras clave. Después se lanzaban contra SENED, proyecto fin de carrera de Guillermo Esteban [9].

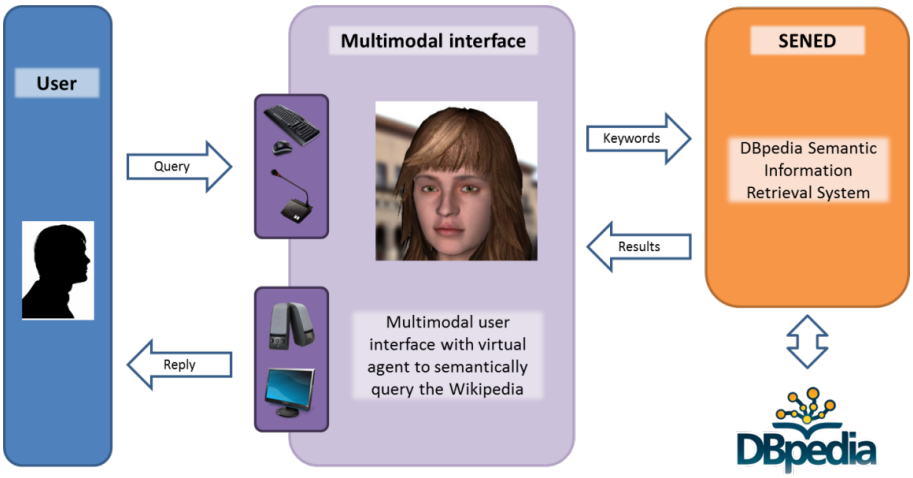


FIGURA 1.1: Esquema funcionamiento VOX

SENED realiza una búsqueda por palabras clave sobre un dominio predefinido en la DBPedia. Al lanzar las consultas en un conjunto acotado de datos de la DBPedia, éstas son más rápidas. Los resultados que nos devuelve son recursos relacionados con las palabras clave de entrada. Estos recursos están *rankeados* según la frecuencia relativa de la aparición de las palabras.

El trabajo realizado en VOX, se continuó mediante la implementación de un prototipo preliminar para la navegación sobre el grafo de datos de manera complementaria. En este prototipo se sustituyó la búsqueda de palabras clave por un recurso dado introducido manualmente. El grafo permitía realizar una navegación básica mediante distintos recursos, a través de las propiedades de los mismos.

Partiendo de este prototipo se crea la herramienta que presenta este proyecto. En ésta se mejora la interfaz gráfica del grafo para favorecer la navegabilidad por los datos, se implementan mejoras de navegación y se vuelve a implantar la búsqueda por palabras clave y el avatar.

De forma general, los objetivos son:

- Integración de VOX con el prototipo de navegación sobre el grafo.
- Mejorar la visualización, ampliando metáforas visuales y las formas de interacción con el usuario.

- Añadir herramientas de navegación para facilitar la misma al usuario, a través de ordenación de recursos, creación de migas de pan y posición espacial del grafo.
- Crear un *framework* para probar y mejorar técnicas de rankeado de propiedades y recursos de acuerdo a la búsqueda del usuario, así como las de aprendizaje y posible sugerencia de nuevos temas por parte del avatar.

1.2. Contenidos de la memoria

En el **Capítulo 2** se explica el contexto tecnológico y las herramientas y tecnologías usadas en este proyecto para llevar a cabo los objetivos definidos. En el **Capítulo 3** se van a observar algunos ejemplos de aproximaciones a representaciones de datos (ejemplos de algunos programas que se aproximan a nuestro proyecto) y los análisis críticos correspondientes. Finalmente, en el **Capítulo 4** se muestra el aspecto final del proyecto, centrándose en cada módulo de forma individual y cómo se complementan.

Capítulo 2

Contexto Tecnológico

En este capítulo se resumen las características más importantes de las tecnologías usadas en el proyecto para entender el contexto y la estructura del mismo. La Figura 2.1 ilustra la arquitectura básica de las tecnologías de la Web Semántica.

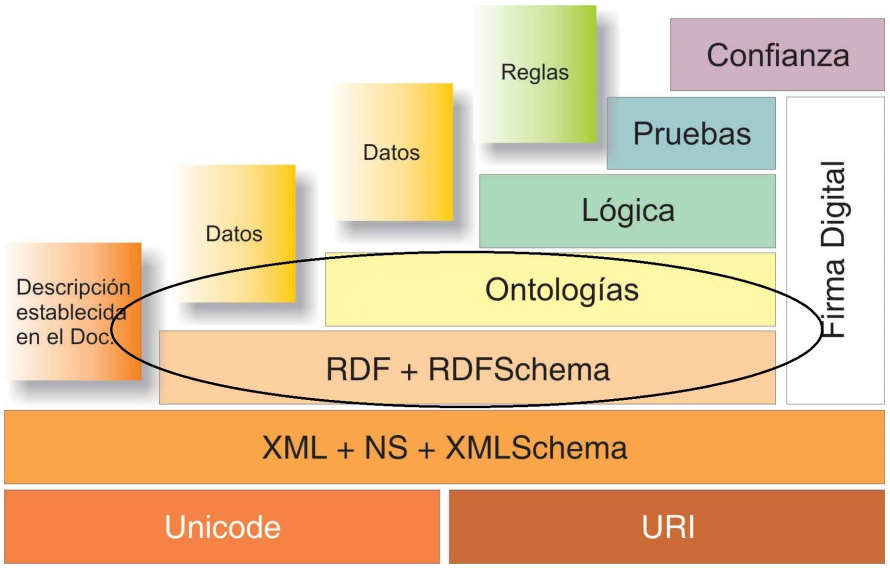


FIGURA 2.1: Esquema arquitectura básica Web Semántica

La parte inferior del esquema muestra los elementos a bajo nivel. A continuación se van a exponer los dos más importantes en este proyecto:

- XML [18]: eXtensible Markup Language o XML es un formato basado en texto para representar información estructurada: documentos, datos, configuraciones, etc... Derivó de un antiguo estándar: SGML. Se propone como un estándar para el intercambio de información entre distintas plataformas, siendo usado en bases de datos, editores de texto, hojas de cálculo, etc... Actualmente existen otros formatos de serialización distintas a RDF/XML. Las más usadas son JSON-LD [13] y Turtle [17].
- URI [12]: Universal Resource Identifier o URI son cadenas de caracteres que identifican recursos en la red de manera unívoca, es decir,

es usada para una identificación única (ningún recurso puede tener la misma URI que otro).

En la parte superior del esquema, se encuentran los elementos de alto nivel, que amplían la semántica mediante la definición de ontologías y la adición de reglas. Para entender el proyecto que nos ocupa, esta memoria se centrará en RDF y ontologías.

2.1. RDF

Resource Description Framwork o RDF [15] es un conjunto de especificaciones del W3C cuya finalidad es servir como modelo de datos estándar para el intercambio de información en la Web. RDF marca cada recurso como una URI. Los recursos están relacionados entre sí mediante relaciones, las cuales están representadas mediante URIs y definen las propiedades entre los datos. En este modelo de datos, los recursos se relacionan mediante propiedades formando tripletas <A,R,B>, donde A y B son recursos distintos y R es la propiedad que los relaciona.

En la Figura 2.2 se muestra un ejemplo de un grafo RDF. Este grafo corresponde a la frase "existe una persona llamada Sergio Sota cuya dirección es Calle Univers N7". Se usan URIs para representar:

- Clases: La clase sería persona, identificado por `diis:contacto#Persona` y definido mediante la propiedad `type`.
- Individuos: o también llamadas instancias. En este ejemplo Sergio Sota es una instancia identificado por `diis:contacto#ssota`.
- Propiedades: En este ejemplo, una propiedad es `diis:contacto#nombre`.
- Valores de propiedades: La dirección `Calle_Univers_N7` es un valor de propiedad.

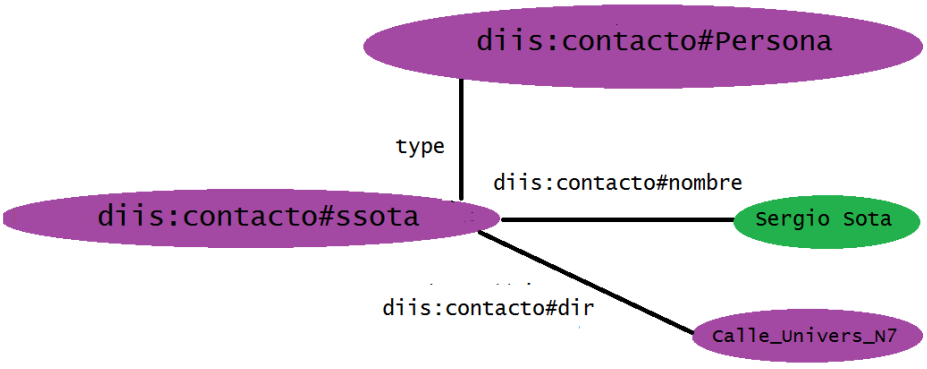


FIGURA 2.2: Ejemplo Red Semántica con RDF

Se puede representar en distintos formatos, no solo en forma de grafo. Uno de los más usados es una variación de XML, llamada RDF/XML. La representación en RDF/XML del ejemplo de la Figura 2.2 es la del Listado 2.1:

LISTADO 2.1: Ejemplo RDF/XML

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:diis="diis:contacto#">
  <rdf:Description rdf:about="diis:contacto#ssota">
     <rdf:type
     rdf:resource="diis:contacto#Persona"/>
    <diis:nombre>Sergio Sota</diis:nombre>
    <diis:dir rdf:resource="Calle_Univers_N7"/>
  </rdf:Description>
</rdf:RDF>
```

Con esta información se puede ver cómo son los datos entrelazados y cómo se relacionan entre ellos mediante propiedades.

2.2. Ontologías

Para trabajar correctamente con RDF, se necesita definir una ontología. Una ontología es un esquema que da significado a los datos. El concepto de ontología la definió *Thomas R. Gruber* como "la especificación explícita de una conceptualización" [4]. Una conceptualización es una abstracción, una vista simplificada del mundo que queremos representar.

Esta definición fue ampliada por *Studer et al.* [10]: "Una ontología es la especificación explícita y formal de una conceptualización compartida." Una 'conceptualización' se refiere a un modelo abstracto de algunos fenómenos del mundo. 'Explícita' significa que los tipos de conceptos usados y las restricciones en su uso, deberían estar definidos. 'Formal' se refiere al hecho de que la ontología debe ser procesable por las máquinas. 'Compartida' es la idea de que la ontología refleja un conocimiento consensuado, aceptado por un grupo.

En resumen, las ontologías permiten la definición de esquemas comunes para representar el conocimiento ubicado, dentro de un ámbito dado. Los elementos básicos de las ontologías son:

- Conceptos (clases): Son las ideas o conceptos básicos que se intentan formalizar. Cada clase define una agrupación de individuos con elementos comunes, también llamada categoría.
- Roles (propiedades): Son las propiedades o características de las clases. Permiten expresar los atributos de las clases y permiten establecer relaciones con otros elementos pertenecientes a otras clases
- Individuo: Como su nombre indica, son objetos específicos de una clase en concreto.

En este caso, usamos la DBPedia como repositorio de datos cuya ontología está expresada en OWL.

2.2.1. OWL

OWL [7] o *Web Ontology Lenguage* es un lenguaje para la Web Semántica, diseñado con el fin de representar el conocimiento semántico, sobre conceptos y relaciones de conceptos. Se basa en XML Schema[1] y RDF Schema[2] añadiendo más vocabulario para enriquecer y definir clases y propiedades. Usa como base los lenguajes de Lógicas Descriptivas [1], lo cual permite verificar la integridad del conocimiento.

2.3. Herramientas y lenguajes utilizados

Como lenguaje de programación se usó Java con JDK 8 con entorno de programación NetBeans. Para el apartado visual, se usaron varias librerías gráficas.

- JMonkeyEngine[3]: JMonkeyEngine es un motor de videojuegos libre, orientado al desarrollo de videojuegos 3D. Usamos esta API para representar el agente virtual 3D que acompañará a la búsqueda del usuario.
- CAD [6]: El acrónimo de Computer-aided Desing. Esta librería es muy popular y usada, como AutoCAD. Permite el diseño de elementos 2D y 3D. En este proyecto es usada como base para nuestra librería 2D gráfica, Xcad. Esta librería se usa como base para poder crear los elementos gráficos necesarios para el grafo y poder colocarlos en su sitio correspondiente.
- SWING [11]: SWING es un Framework MVC (Modelo - Vista- Controlador) para desarrollar interfaces gráfica en JAVA. Es usado en nuestro proyecto para crear el entorno de las migas de pan. La ventaja frente a la librería XCad, es la sencillez de adición de nuevos elementos y el bajo consumo, tanto en memoria como en CPU, de los mismos.

[1] XML Schema: https://www.w3.org/standards/xml/schema
[2] RDF Schema: https://www.w3.org/TR/rdf-schema/
[3] Página Web de JMonkey: http://jmonkeyengine.org/

Capítulo 3

Estado del Arte

Es este capítulo se estudian algunas de las distintas aproximaciones, que ha habido al problema de representar datos de una Red Semántica, visualmente y de forma simple. De esta forma se puede enmarcar la propuesta del presente proyecto.

Actualmente, aunque existen herramientas que ofrecen diversas aproximaciones a la solución del problema, no existe ninguna aplicación, académica o comercial, satisfactoria. Esto se debe a las dificultades existentes en representar la información de la *Web of Data* de forma sencilla. Es un problema aún abierto. Casi todas las herramientas que existen son prototipos dentro del ámbito de la investigación y están orientadas principalmente a investigadores. La mayoría no están orientadas a usuarios ajenos a la investigación académica. A continuación vamos a citar algunas herramientas, con sus pros y sus contras, para poder situar nuestro proyecto.

3.1. *LodLive*

LodLive [5] es una herramienta de búsqueda exploratoria y de visualización de *Linked Data*. Permite distintos tipos de búsquedas, siendo la más relevante la búsqueda por recurso. Esta herramienta utiliza como repositorio de datos tanto la DBPedia [2] como FreeBase [3].

Introduciendo el recurso `Isaac Newton`, obtenemos la visualización de la Figura 3.1. Prácticamente no hay información en pantalla, solo sabemos el nombre de los recursos y la propiedad entre ellos. Los elementos circulares alrededor de los nodos principales son las propiedades.

Esta herramienta agrupa la información de forma concentrada, creando varios subniveles de relaciones, haciendo que la búsqueda de relaciones concretas, sea poco visual, al obligar al usuario a mirar relación por relación hasta encontrar la deseada (se necesita mover el cursor encima de cada relación para que aparezca el nombre). La búsqueda de las palabras clave no es profunda. Busca los recursos que contenga esas palabras clave, sin tener en cuenta los posibles recursos relacionados, al que el usuario puede estar interesado en acceder. Esto hace que encontrar un punto de partida favorable, pueda ser complicado.

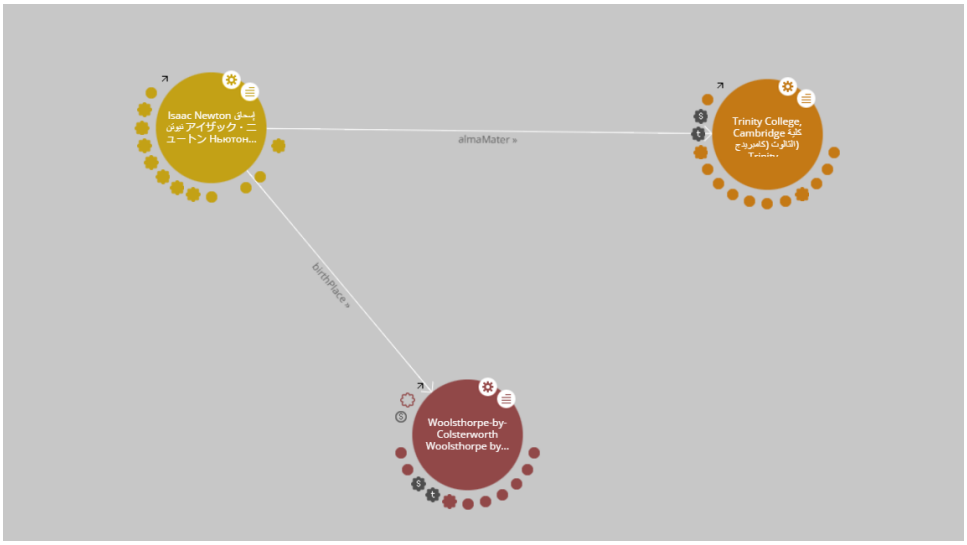


FIGURA 3.1: Búsqueda en LodLive del recurso `Isaac_Newton`

3.2. *RelFinder*

RelFinder es un buscador de relaciones entre elementos. Permite estudiar las relaciones que se encuentran entre dos recursos, para obtener puntos de nexo o propiedades similares (ver Figura 3.2)

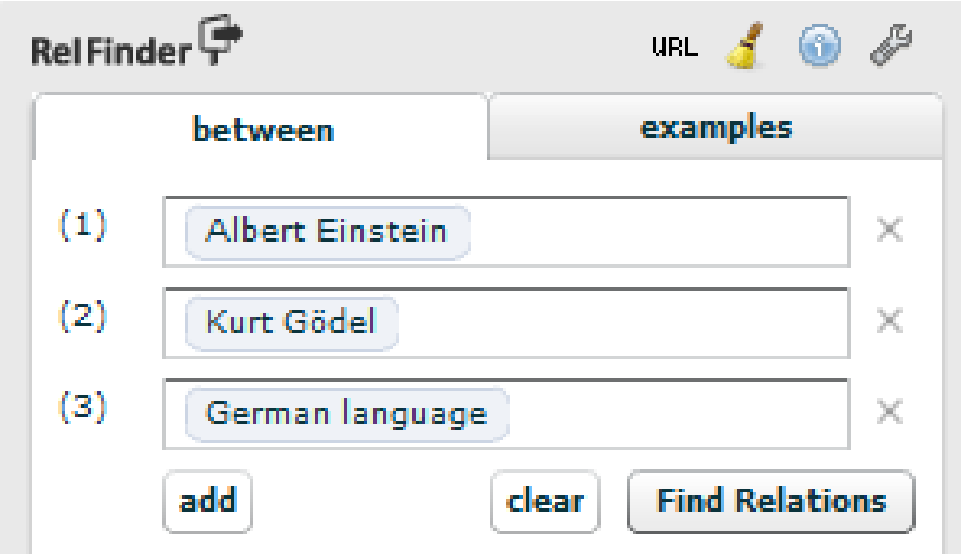


FIGURA 3.2: Buscador RelFinder

En este ejemplo se han buscado las relaciones entre tres recursos: `Albert Einstein, Kurt Gödel y German Language`. Los resultados, como se puede apreciar en la Figura 3.3, presentan cierta complejidad visual, llegando en ocasiones a resultar caótica cuando se aplica a un conjunto grande de recursos de entrada. Hay que tener en cuenta que este tipo de aplicaciones aplican algoritmos de búsqueda en profundidad que, dada la extensión del grafo (de forma arbitraría), puede desembocar en tiempos de carga excesivos.

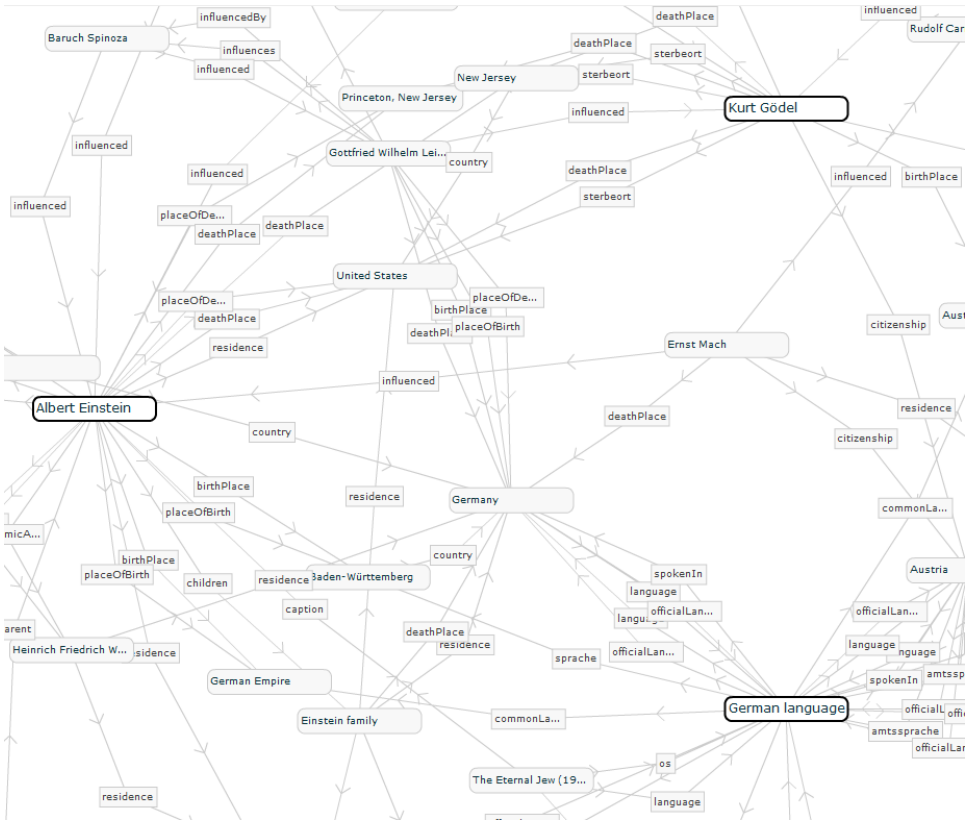


FIGURA 3.3: Resultado de *RelFinder* con tres recursos de partida

Esta herramienta es buena para, como se ha dicho antes, encontrar relaciones entre recursos, pero no permite una búsqueda exploratoria a partir de un elemento dado.

3.3. *gFacet*

Esta última herramienta está orientada a visualizar los datos. En la Figura 3.4 se puede observar su interfaz principal. En el ejemplo se ha buscado el recurso `German Physicist`, y como resultado se ofrece un conjunto de recursos que están dentro de esa categoría. Esta aplicación permite buscar relaciones sobre un conjunto de recursos (como en este ejemplo, `doctoral_students`) y cuando se hace click en un elemento, se filtra el resultado en todos los recuadros para ver cuál es el recurso que comparte la relación (Ver Figura 3.5). De esta manera se filtran los resultados. No es nada fácil de usar ya que los recursos son enlaces a categorías de la Wikipedia y se necesitan mirar cada enlace por separado para ver cuál es.

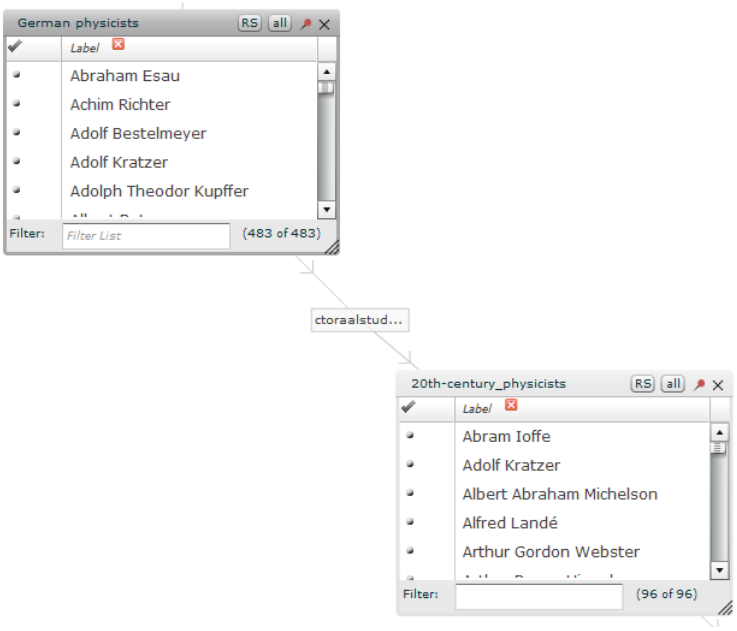


FIGURA 3.4: Búsqueda `German Physicist` y `doctoral_students`

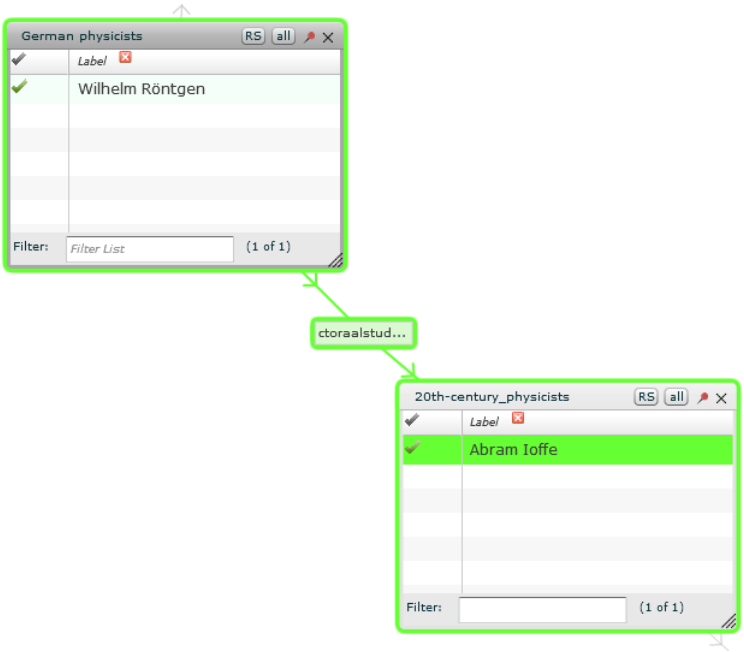


FIGURA 3.5: Recurso de `German Physicist` que esta relacionado con recurso de `doctoral_students`

Todas estas herramientas están orientadas a investigadores o usuarios expertos, lo cual sesga el conjunto de usuarios de esta aplicación (sólo un usuario entrenado puede usar la herramienta). Nuestro proyecto, simplificando, hace comprensible la información y ofrece herramientas para conseguir una navegación más fluida y sencilla, permitiendo que sea accesible a un mayor conjunto de usuarios.

Capítulo 4

Resultado del proyecto

En este capítulo se habla del trabajo realizado en el proyecto. Se divide en dos secciones:

- Apartado visual: Todos los módulos referentes a aspectos visuales (Exploración con grafo, migas de pan).
- Apartado no visual: Todos los módulos referentes a aspectos fuera del ámbito visual (búsqueda por palabras clave, registro de acciones).

En ambos apartados, se van a encontrar, tanto mejora o integración de elementos ya existentes; como creación de nuevos elementos. En la Figura 4.1 se puede observar la interfaz final del proyecto.

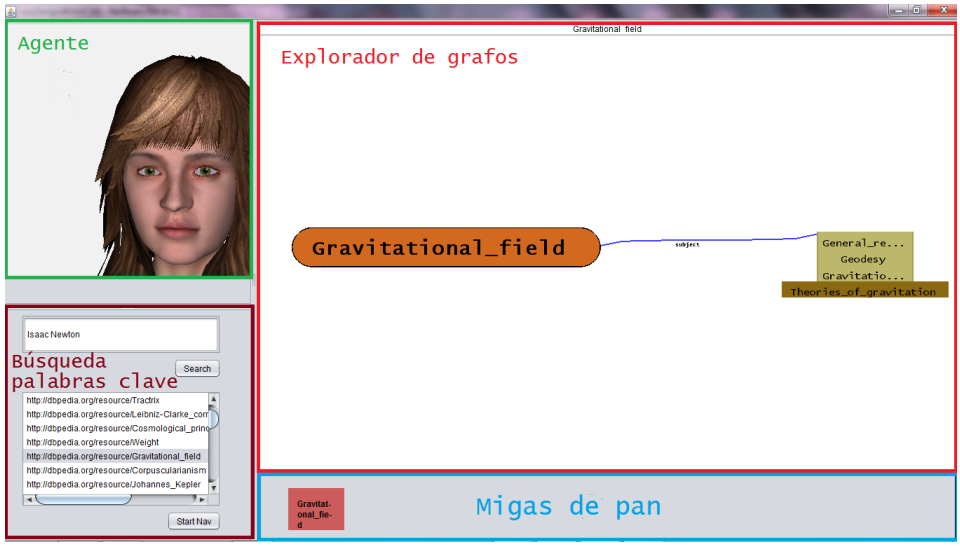


FIGURA 4.1: Interfaz final

Las partes marcadas indican distintos elementos principales del proyecto, cuya función es dar apoyo al usuario y mejorar la navegación por el grafo de información. Estos elementos se explicarán en sus secciones correspondientes mostradas a continuación.

4.1. Apartado visual

En esta sección se van a ver todos los aspectos visuales del proyecto.

- Explorador de grafo: realizado a partir del prototipo base, modificando las metáforas del color y añadiendo nuevos elementos y funcionalidades.
- Migas de pan: módulo de apoyo creado para facilitar la exploración del grafo, añadiendo un contexto al estado actual.
- Avatar: Módulo de VOX integrado en este proyecto que ofrece una mejora a la interacción con el usuario mediante el uso de voz sintética.

4.1.1. Explorador de grafos

Es el principal elemento de exploración de la herramienta y permite navegar entre los diferentes recursos que están relacionados entre sí. Esta sección está separada en dos subsecciones: la primera tratará los elementos visuales del grafo; y la segunda la navegación por el grafo.

4.1.1.1. Elementos visuales

Existen varios elementos visuales en el explorador. Algunos de ellos tienen una funcionalidad simple (representar elementos), mientras que otros son más complejos (ranking de elementos). Todos los elementos de esta sección se encontraban en el prototipo excepto **cuadro propiedad**, que fue añadido para simplificar la ordenación de recursos, bajo una misma propiedad. Todos ellos han sido modificados y mejorados para obtener los objetivos marcados.

- Recursos (ver Figura 4.2): Los **recursos** son los principales elementos visuales del grafo. Simbolizan un recurso RDF identificado con una URI. Cada recurso tiene distintas propiedades que se ordenan en distintas páginas. Los elementos circulares inferiores son las páginas de propiedades. Un recurso posee un estado binario *abierto* o *cerrado* indicando su estado en el grafo (sólo el elemento actual se encuentra *abierto*). Sus colores son naranja y gris respectivamente.

FIGURA 4.2: Ejemplo del nodo de un recurso, `Isaac_Newton`

- Cuadro propiedad (ver Figura 4.3): Los **cuadros propiedad** agrupan todos los recursos RDF relacionados con el recurso actual que comparten una propiedad común, identificados por URIs. En la Figura 4.3

son todos los recursos con la propiedad *Subject* (indicada en la curva que los une). Como el número de recursos relaciones puede ser alto, los cuadros propiedad solo muestran cuatro elementos cada vez. Con las flechas se avanza de cuatro en cuatro para poder acceder a todos. Estos recursos están ordenados por un ranking de relevancia presente en SENED.

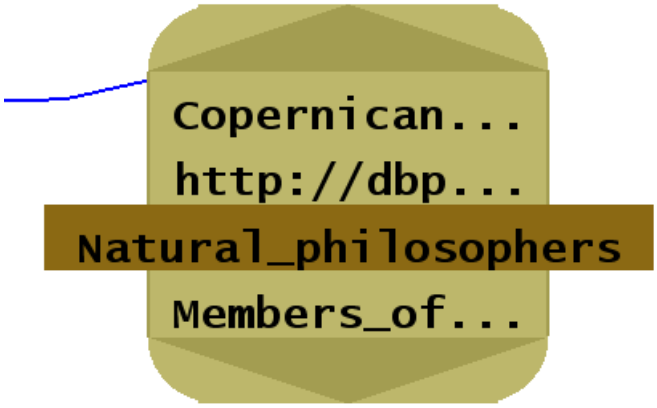


FIGURA 4.3: Cuadro propiedad de la propiedad `subject`

- Relaciones (Ver Figura 4.4): Estas **relaciones** unen un recurso con el cuadro propiedad correspondiente. Están denotadas con una etiqueta indicando la relación que las une. Estas relaciones también son URIs y son las propiedades de RDF. Este elemento, junto a su **cuadro propiedad** correspondiente, solo será visible cuando el recurso relacionado este *abierto*.

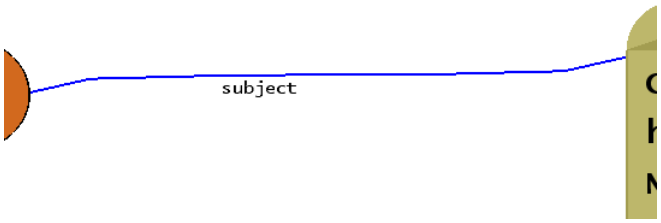


FIGURA 4.4: Relación `subject` de `Isaac_Newton`

Estas tres figuras de ejemplo representan una tripletas <A,R,B>: El recurso sería `Isaac_Newton`, la propiedad sería `subject` y el recurso relacionado sería uno del conjunto del **cuadro propiedad**.

Al realizar click en un recurso del cuadro propiedad, éste se *abrirá* quedando a partir de ese momento siempre disponible en la interfaz (hasta que se inicie una nueva búsqueda), *cerrando* a la vez el anterior recurso abierto.

4.1.1.2. Navegación

A continuación se va a mostrar un caso de navegación por el grafo tomando como punto de partida el recurso `Michael_Faraday`. El estado inicial del grafo se observa en la Figura 4.5. El recurso abierto está en el centro y a los lados se encuentran los recursos relacionados unidos por una arista, que denota la relación existente.

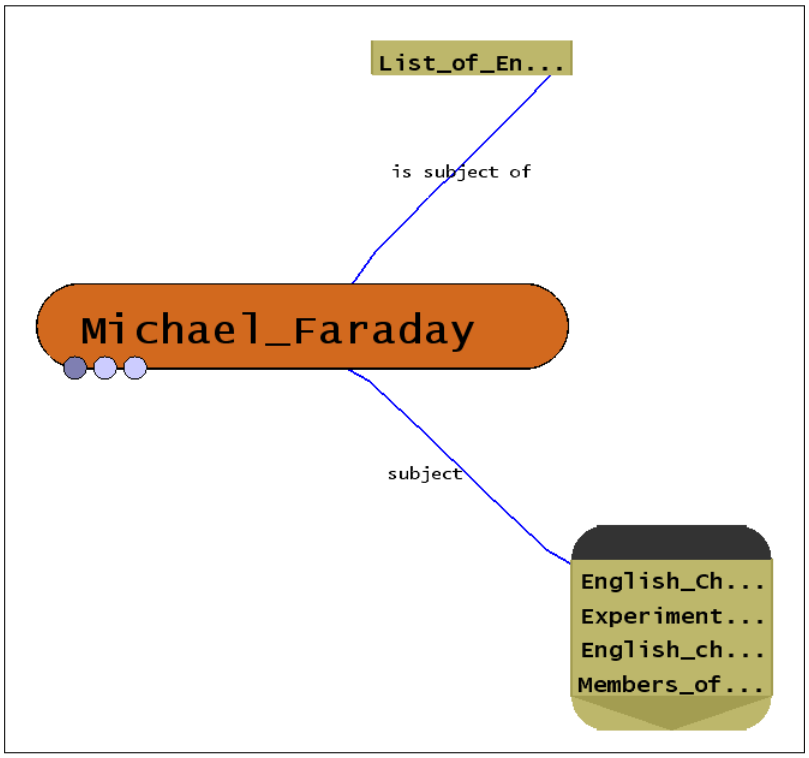


FIGURA 4.5: Navegación paso 1: Inicio navegación con el recurso, `Michael_Faraday`

Tras hacer click en un recurso, pasamos al estado del grafo de la Figura 4.6. El nuevo recurso, `Category:Experimental_physicists` muestra los recursos relacionados con él. El antiguo recurso se cambia a *cerrado*, cambiando su color y eliminando los recursos relacionados con el mismo.

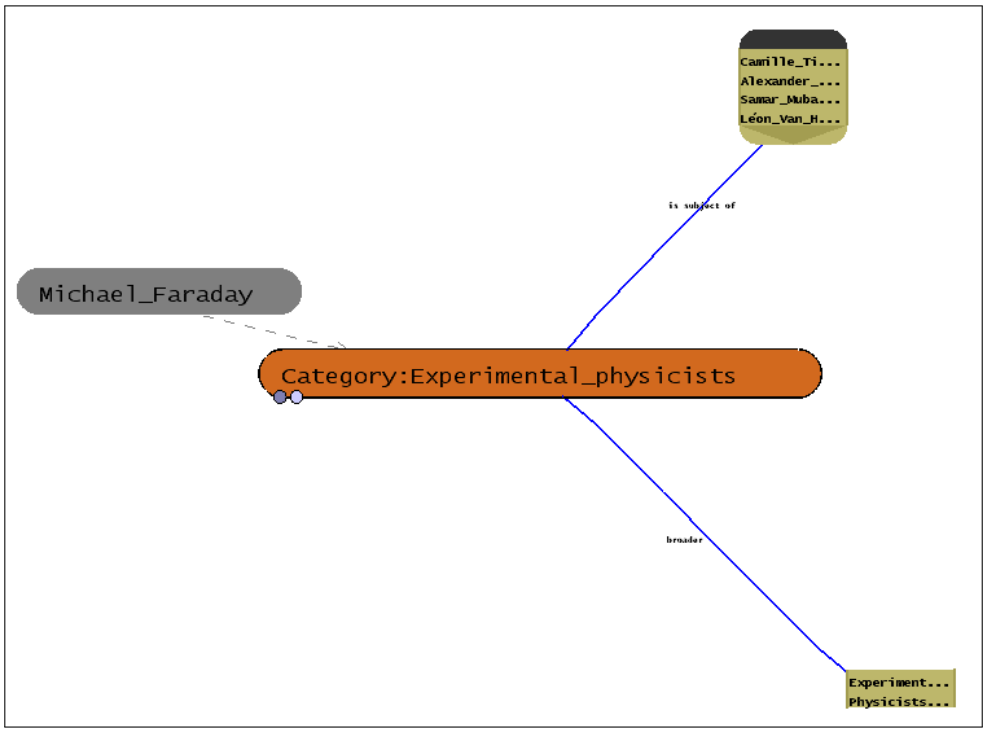


FIGURA 4.6: Navegación paso 2: Se viaja por la relación `subject` al recurso `Experimental_physicists`

Si hacemos click en el recurso *cerrado* `Michael_Faraday`, se abrirá de nuevo, como se observa en la Figura 4.7 y el anteriormente abierto, `Category:Experimental_physicists` cambiará su estado a *cerrado*.

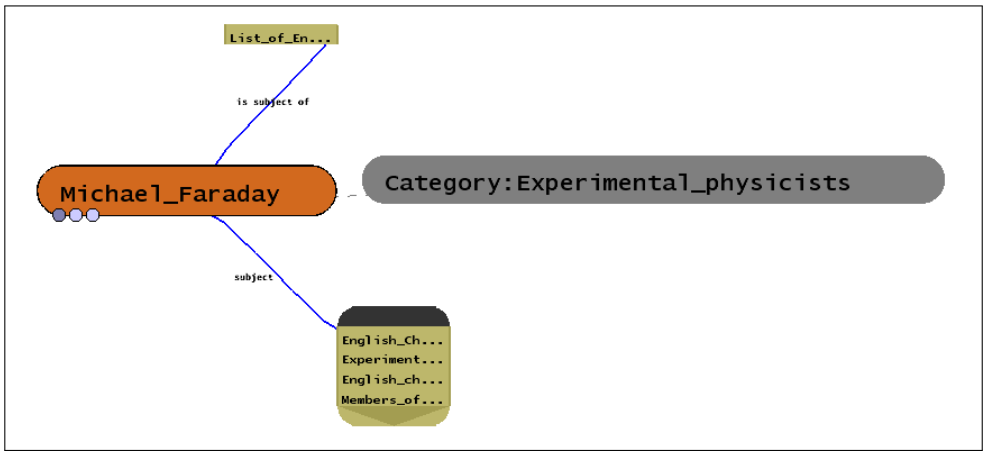


FIGURA 4.7: Navegación paso 3: Volvemos al recurso `Michael_Faraday`

También se puede iniciar la navegación por otro recurso distinto a `Category:Experimental_physicists`, por ejemplo, buscando un nuevo recurso en la propiedad *subject*. En la Figura 4.8 se puede apreciar cómo se ha *abierto* un nuevo recurso, pero se mantienen aún los anteriores, permitiendo volver a ellos en cualquier momento.

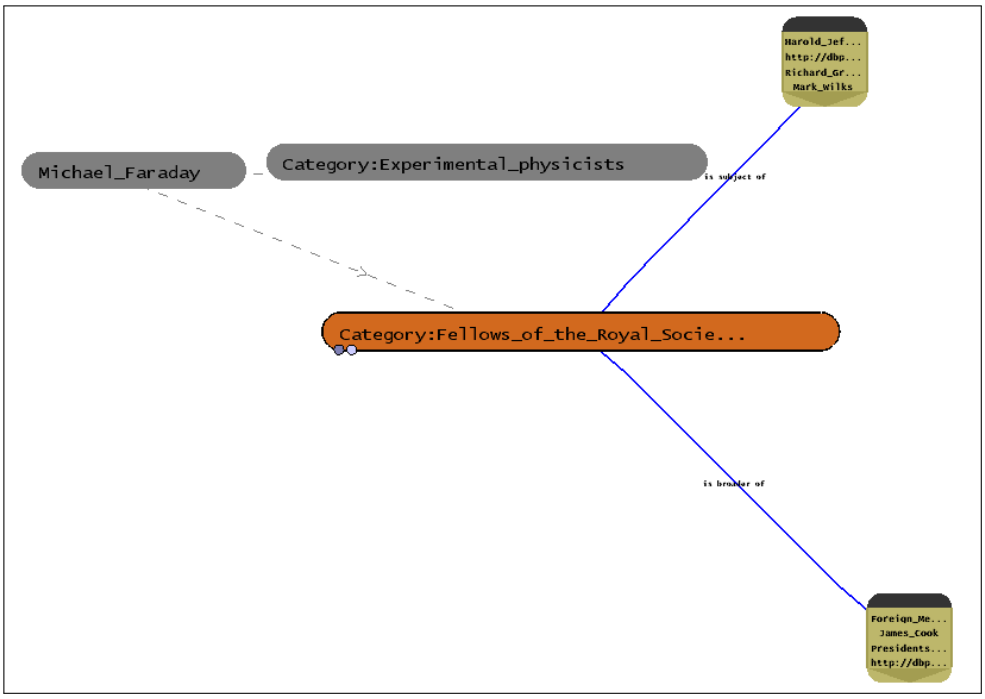


FIGURA 4.8: Navegación paso 4: Se navega por la propiedad `is subject of` al recurso `Category:Fellows_of_the_Royal_Socie...`

Para facilitar la compresión temporal de la apertura de elementos, los nuevos recursos que se *abran* se ubicarán siempre a la derecha.

Los recursos se pueden reposicionar libremente por el espacio, para posicionarlos en un lugar dado, o conseguir una mejor visibilidad con solo arrastrarlo. Esto permite al usuario reordenar los elementos, personalizando la visualización y adaptándolos a su gusto (acercando los elementos para más información en pantalla o alejándolos para separar cada uno). Además de mover los recursos, esta herramienta permite reposicionar libremente la cámara, para poder recorrerlo con facilidad sin tener que navegar entre distintos nodos.

El explorador de grafos posee además la capacidad de hacer zoom. Esto permite hacerse una idea global del camino recorrido, u obtener una mejor visión de un recurso dado. Debido al zoom, puede ocurrir que si hay poco zoom, el nombre del recurso actual no se distinga bien (el zoom se usa para ver el estado global). Además, debido a la longitud de algunos recursos, sus nombres se han acortado para estar dentro del estilo de los nodos. Por eso se añadió una herramienta en la parte superior del grafo para ver el nombre del recurso actual. (Figura 4.9)

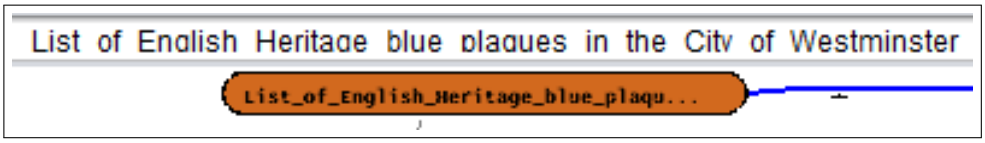


FIGURA 4.9: Ejemplo de un nodo no visible con el nombre del recurso visible

4.1.2. Migas de pan

Es un elemento secundario que sirve de apoyo a la navegación por el grafo. Esta herramienta es muy utilizada en la navegación hipertextual, para proveer de contexto al estado actual (cómo has llegado navegando hasta dónde estás). Surge de la necesidad de que el usuario sepa en cada momento dónde se encuentra y cuál ha sido su historial de navegación de forma simple y sencilla, permitiendo un retorno a puntos anteriores de la búsqueda con un solo click.

En este proyecto se parte de la misma idea pero con un ligero cambio: cuando en una página Web vuelves a un lugar anterior, se borra el camino que sale de ese lugar. En nuestro caso, ese camino se mantiene hasta que inicies una nueva búsqueda desde algún recurso existente en las *migas de pan*, para que el usuario siempre tenga una referencia visual de su búsqueda. Como en la sección *Explorador de grafos*, la presente se va a dividir en dos apartados: visual y navegación. Todos los elementos han sido creados íntegramente para este proyecto, sin partir ni del prototipo ni de VOX.

4.1.2.1. Aspectos visuales

Las *migas de pan* poseen elementos visuales distintos, más simplificados, que el grafo:

- Recursos (Ver Figura 4.10): Los recursos, al igual que los del grafo, son elementos RDF identificados por una URI. Estos elementos identifican los nodos abiertos en el grafo, siendo un vínculo hacía los mismos. Igual que en el grafo, están identificados con un estado binario, indicando si son el actual o no. En este caso, el color rojo significa que es el actual y el naranja que no lo es.

FIGURA 4.10: Recurso abierto con su miga correspondiente

- Relación (Ver Figura 4.11): Las relaciones indican la propiedad por la cual se ha viajado de un recurso a otro. Van de izquierda a derecha y el texto superior es el nombre de la relación, la cual también es una URI.

FIGURA 4.11: Relación de miga de pan

Con estos dos elementos se pueden observar perfectamente las tripletas <A,R,B> por las que el usuario ha ido navegando. Como se observa en la Figura 4.12, el recurso A sería `Albert_Einstein`, el recurso B sería `Category:People_from_Zurich` y la propiedad que las une, R, sería `subject`.

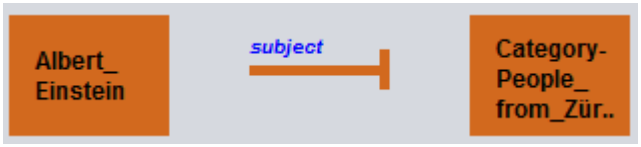


FIGURA 4.12: Tripleta ARB en miga de pan

4.1.2.2. Navegación

Las *migas de pan* trabajan conjuntamente con el grafo, actualizándolo cuando se ejecutan acciones en las *migas* o viceversa. Cada vez que se *abre* un elemento en el grafo, se crea un nuevo elemento y se pone a la derecha. Si ese elemento ya ha salido en la búsqueda actual, se repite de nuevo y se le añade un valor extra, indicando el número de veces que ha aparecido. A su vez, cuando se hace click en un elemento del grafo que está en las *migas*, estas se actualizan haciendo foco al nuevo elemento. Cuando se hace click en las *migas* el grafo se actualiza y centra el nuevo recurso.

Empezando una búsqueda (Ver Figura 4.13), se observa como aparece un elemento en la zona de *migas de pan* referenciando a `Albert_Einstein`, con color rojo. Este es el elemento actual de la navegación, el cual corresponde con el elemento actual del grafo.

FIGURA 4.13: Navegación paso 1: Iniciamos la navegación en el recurso `Albert_Einstein`

En la Figura 4.14 se puede observar una navegación con tres recursos abiertos (`Albert Einstein`, `Category: People from Zürich`

`y Category: Zürich`). Las flechas entre elementos indican la relación que une a los recursos. En este caso, *Albert Einstein* es `subject` de `Category: People from Zürich` y esté a su vez es `broader` de `Category:Zürich`

FIGURA 4.14: Navegación paso 2: Se tienen varios elementos en las *migas*

En la Figura 4.15 se puede ver que al seleccionar un recurso cerrado en las *migas de pan*, éste se *abre* en el grafo, se centra en la pantalla y, a la vez, se pone como activo en las *migas de pan*.

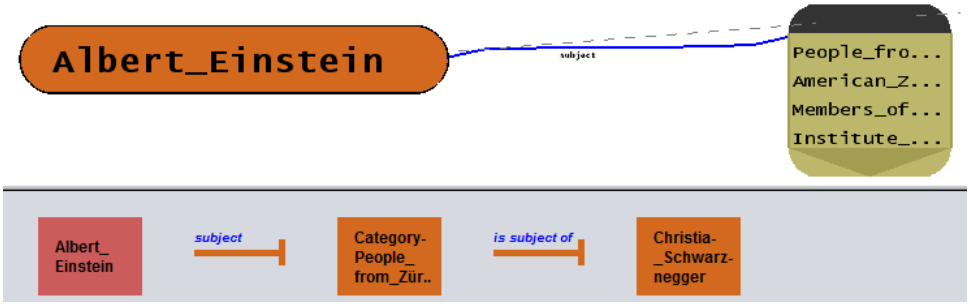


FIGURA 4.15: Navegación paso 3: Volvemos atras en las migas para observar como se actualizan los elementos

Cada recurso guarda el camino por el que se ha llegado a él (siempre se guarda el último camino). Cada vez que se *abre* un nuevo recurso que no está en las *migas* actuales, se coge el camino del recurso *abierto* y se sustituye con las *migas* actuales. Se hace de esta manera para que las *migas* sean coherentes con la navegación. Cuando se salta a un recurso que no está en las actuales, se entiende que el usuario quiere continuar la navegación por otro lugar distinto, así que se le mantiene la navegación que llevaba cuando llegó a ese recurso.

A continuación el resultado para ilustrar este caso especial, partiendo de la Figura 4.16 y viendo el resultado en la Figura 4.17.

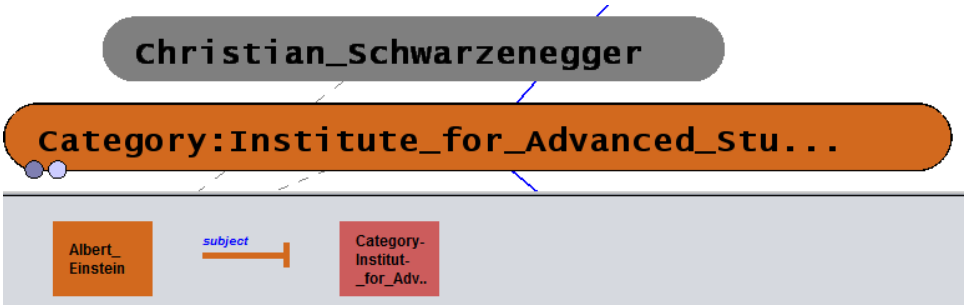


FIGURA 4.16: Navegación paso 4: Se quiere hacer click en `Christian_Schwarzenegger`, el cual está fuera de las migas actuales, desde el recurso actual

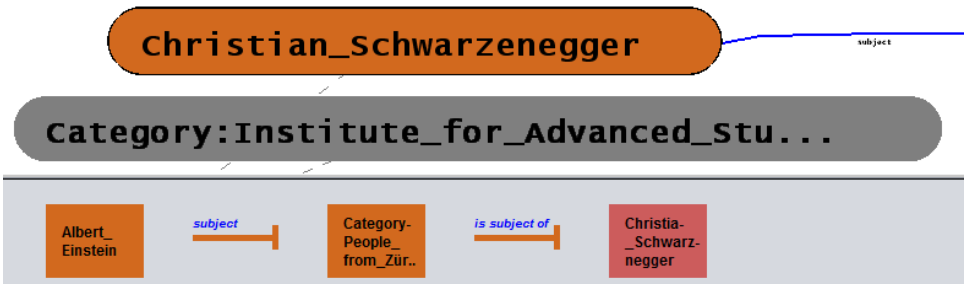


FIGURA 4.17: Navegación paso 5: Se observa como al cambiar de recurso activo, las migas se han actualizado

Como se puede apreciar, la pila se ha sustituido por la que se llevaba al acceder por primera vez a `Christian_Schwarzenegger`. A partir de esta se podría comenzar otra navegación sin problemas, teniendo siempre las *migas* bien actualizadas.

4.1.3. Avatar

El objetivo del avatar es apoyar al usuario mediante voz diciendo posibles resultados de la búsqueda o nombrando los recursos actuales a los que accede. (Ver Figura 4.18)

FIGURA 4.18: El *avatar*

Este *avatar* tiene integrados gestos faciales, tanto de ojos como de labios, sincronizados con la voz que emite para crear coherencia visual. Actualmente su tarea es decir el nombre del recurso en el que se encuentra. En trabajos posteriores, se quiere mejorar la funcionalidad, integrando el aspecto conversacional que existía en VOX. Con esto se conseguiría ayudar a los usuarios, ya que estos lo perciben de manera positiva a la hora de buscar [8]. Para este fin, se busca caracterizar las búsquedas de los usuarios y disparar consultas en *background*, para sugerir resultados relevantes como trabajo futuro.

4.2. Apartado no visual

A continuación se van a presentar los dos elementos no visuales del proyecto. Estos son las búsquedas por palabras clave y el registro de acciones.

4.2.1. Búsqueda por palabras clave

Este módulo permite al usuario introducir una serie de palabras clave y que el sistema devuelva un conjunto de recursos relacionados con las palabras clave introducidas. Además, permite definir un dominio dado con la siguiente sintáxis: `Palabras_clave = dominio`. (Ver Figura 4.19)

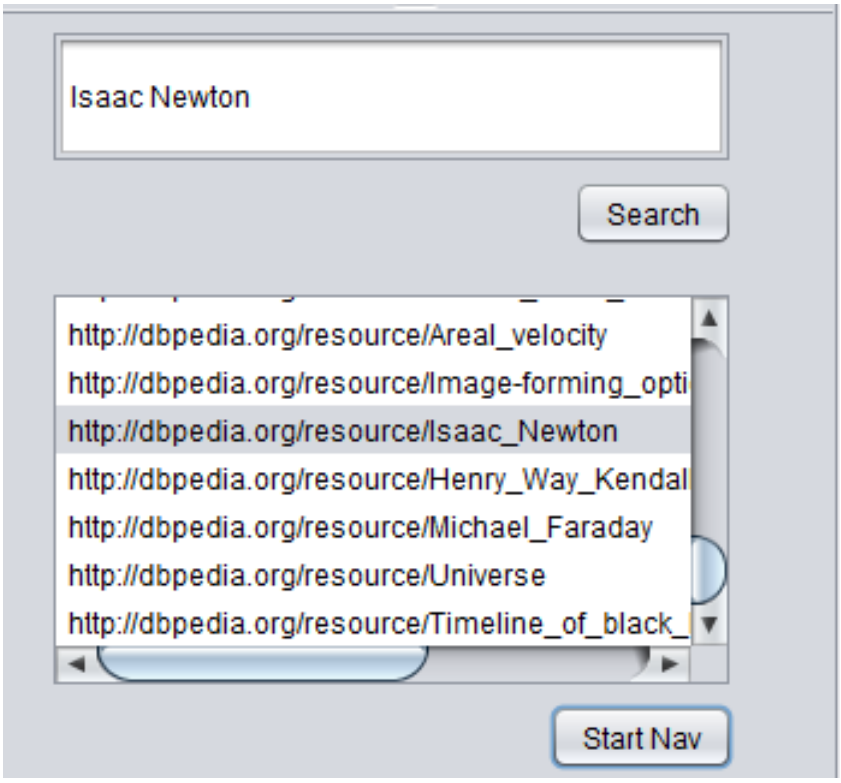


FIGURA 4.19: Ejemplo de búsqueda por palabras clave

Esta búsqueda se hace contra SENED, explicada en Objetivo y alcance. La parte relacionada con este proyecto fue añadir a la interfaz los elementos

necesarios (cajas de texto, botones) para poder introducir las palabras clave para poder hacer búsquedas.

Esta característica existía en VOX, aunque con una ligera variación: aparte de introducir las palabras clave mediante teclado, el usuario tenía la opción de decir la consulta mediante voz y era el *parser* quien analizaba la consulta y extraía las palabras clave. En este proyecto se ha optado por introducir las palabras clave directamente por texto.

La forma de mejorar el *rankeado* de los datos y ordenarlos de acuerdo a ese *ranking* es trabajo futuro de investigación. Para más información sobre SENED, consultar el PFC de Guillermo Esteban[9].

4.2.2. Registro de acciones

El registro es un elemento que se encarga de registrar todas las interacciones del usuario con el sistema, desde iniciar una navegación, a navegar entre recursos o viajar a otros de forma rápida. Estos datos son recogidos, y marcados con una marca temporal, para el posterior estudio de los mismos. Este estudio no entra en el objetivo de este trabajo así que no se especifica pero consistirá en estudiar los datos para mejorar la navegación del usuario. Para guardar los datos y no perder información, se escribe en un fichero tras cada nueva línea que tenga que escribir. De esta forma, aunque el usuario cierre la aplicación de manera abrupta, no perderá nada de la navegación en ningún momento.

En el Listado B.1 se puede ver un extracto del documento generado por el programa.

LISTADO 4.1: Extracto del registro

```
25-07-2016_21:39:24 - iniciar Isaac_Newton ||
25-07-2016_21:39:27 - navegar Isaac_Newton |is subject of|
    List_of_English_Heritage_blue_plaques_in_the_City_of_Westminster
25-07-2016_21:39:30 - navegar
    List_of_English_Heritage_blue_plaques_in_the_City_of_Westminster
    |subject| Edward_Gibbon
25-07-2016_21:39:32 - clickar Isaac_Newton ||
25-07-2016_21:39:34 - pasarLista Isaac_Newton ||
25-07-2016_21:39:36 - pasarLista Isaac_Newton ||
25-07-2016_21:39:38 - navegar Isaac_Newton |subject|
    Category:Post-Reformation_Arian_Christians
25-07-2016_21:39:41 - navegar
    Category:Post-Reformation_Arian_Christians |is subject of|
    Patrick_Pakingham
25-07-2016_21:39:43 - navegar Patrick_Pakingham |subject|
    Category:Arianism
```

Existen distintas acciones de las que se hace registro:

- ***Iniciar***: Se usa al principio de la navegación para obtener el recurso de partida.
- ***Navegar***: Se usa cuando va de un recurso a otro a través de una relación.

- ***Clickar***: Se usa para cuando hace click en un recurso que no está directamente relacionado con el actual.
- ***Página***: Se usa cuando se pasa a la siguiente página de relaciones.
- ***Pasar lista***: Se usa para cuando se mueve la lista de recursos dentro de una relación.

Todas las acciones que se guardan son anonimizadas y marcadas por sesión. Cada vez que se usa el programa, éste crea un fichero nuevo que guarda las acciones. Estas acciones se pueden extender y añadir más en el futuro según se vayan modificando las necesidades.

Con el uso de este registro, se quiere aprender de la forma en la que los usuarios buscan y navegan para favorecer búsquedas exploratorias o en profundidad en función de la caracterización del usuario y la búsqueda, por ejemplo, a partir de las trazas que dejen los mismos.

Capítulo 5

Conclusiones

La conclusión más importante que se obtiene del trabajo de este proyecto, es que se ha conseguido realizar una implementación, de un navegador de Web Semántica mediante búsqueda por palabras clave y apoyado por un agente virtual. Hemos dado un paso en el problema de dificultad de navegación con una interfaz sencilla y con distintos módulos que sirve como apoyo. En particular, el sistema diseñado e implementado posee las siguientes características:

- Se ha integrado el funcionamiento de VOX con el prototipo de navegación sobre el grafo.
- Se ha mejorado el proceso de búsqueda, permitiendo navegar, no sólo por los resultados própiamente dichos de la búsqueda por palabras clave, sino por los resultados más relevantes en torno a los mismos.
- Permite una navegación ágil por los Datos Enlazados desde un nodo de partida con un mapa de navegación orientado al usuario, añadiendo módulos de apoyo tales como las migas de pan o la reordenación espacial de elemento.
- Se ha mejorado la visualización, ampliando las metáforas visuales.
- Recopila datos de usuario necesarios para mejorar, en el futuro, el funcionamiento de SENED.
- Se ha creado un *framework* para probar y mejorar técnicas de *rankeado* de propiedades y recursos.

5.1. Trabajo futuro

Tras el desarrollo de este proyecto, aparecieron nuevos objetivos para continuar como trabajo futuro. Estos objetivos intentarán suplir nuevas necesidades aparecidas como fruto del trabajo o mejoras estéticas debido al uso de diferentes herramientas:

- Mejorar la navegación usando la información obtenido del usuario con el registro de acciones.
- Migrar el sistema de OpenGL a Swing. Concretamente el apartado del Explorador de grafos. Esto se debe a que Swing nos ofrece una mayor portabilidad y una interfaz homogénea. Esto está explicado con más detalle en el Apéndice A.
- Añadir nuevas funcionalidades al avatar, entre las que se encuentran: la caracterización de las búsquedas del usuario y la sugerencia de búsquedas relevantes.
- Conseguir el hito de llegar a VOX2, el siguiente estado del proyecto.

5.2. Conclusiones personales

Trabajando con tecnologías de Web Semántica, he adquirido conocimientos sobre el funcionamiento de la Web Semántica y algunas partes de su esquema (RDF y Ontologías), entendiendo cómo funcionan los datos enlazados y por qué son tan útiles y con un buen potencial. He entendido cómo se hacen las búsquedas contra conjuntos de datos, como DBPedia, usando lenguajes de búsqueda como SPARQL.

He aprendido a trabajar con interfaces gráficas y a lidiar con los problemas inherentes en ellas. He asentado conocimientos que he ido adquiriendo en la carrera mientras lidiaba con la programación de las clases y los datos que he estado usando en este proyecto. He adquirido competencias críticas respecto a interfaces visuales y he aprendido a ver y controlar el consumo de recursos que producen.

De la parte de desarrollo he mejorado mis conocimientos de Java y he conocido dos grandes librerías gráficas: CAD y SWING.

Como fruto de este trabajo he adquirido y mejorado bastantes competencias, tanto personales como profesionales. He mejorado mi trabajo personal frente a un repositorio online, para tener control de versiones y facilidad de trabajo en distintos dispositivos. Las facultades de autoaprendizaje que poseo se han visto mejoradas así como la gestión de proyectos, tomando consciencia de los retos, limitaciones y riesgos, que existen al abordar el desarrollo de estos.

5.3. Tiempos

La evolución de este proyecto se ha visto condicionada y limitada por diferentes factores. En primer lugar la compatibilidad del desarrollo del proyecto con el periodo lectivo ha impedido la completa implicación a tiempo completo durante la mayor parte del mismo. Había elementos que requerían de otras personas, así que eso incremento los tiempos y fechas de algunas secciones.

A continuación se expone la planificación seguida en el desarrollo del proyecto y las horas invertidas en cada tarea.

CUADRO 5.1: Reparto horas por tarea

Tarea	Horas invertidas
Estudio	40
Aprendizaje JOGL	10
Migración librerías y solución bugs	20
Creación modelo por propiedad	35
Nueva GUI	5
Metáforas de color y mejora gráfica	35
Creación TextRenderer	40
Integración VOX	10
Implementación archivo configuración	5
Creación migas de pan	30
Estudio y redacción de la memoria	75
Total	305

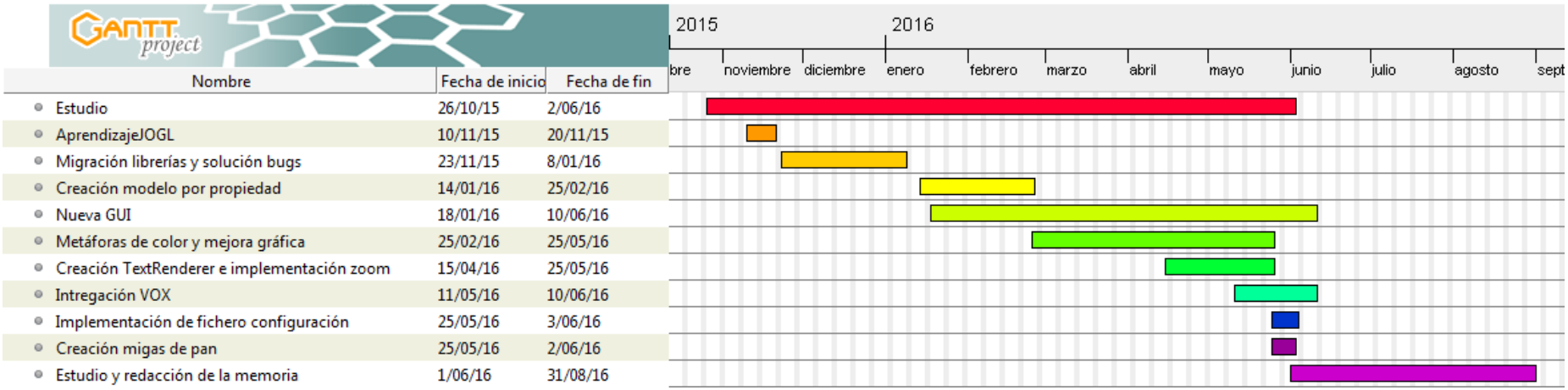


FIGURA 5.1: Cronograma

Bibliografía

[1] F. Baader y W. Nutt. «Basic Description Logics». En: *The description logic handbook* (2003), págs. 43-95.

[2] *DBPedia*. URL: http://wiki.dbpedia.org/ (visitado 20-08-2016).

[3] *Free Base*. URL: http://wiki.freebase.com/ (visitado 20-08-2016).

[4] T. K. Gruber. «A Traslation Approach to Portable Ontology Specifications». En: *Knowledge Acquisition* 5 (1993), págs. 199-220.

[5] *LodLive*. URL: http://en.lodlive.it/ (visitado 25-08-2016).

[6] K. L. Narayan, K. Mallikarjuna Rao y M.M.M. Sarcar. *Computer Aided Design and Manufacturing*. Prentice, 2008.

[7] *OWL 2 Web Ontology Language Document Overview*. URL: http://www.w3.org/TR/owl2-overview/ (visitado 20-08-2016).

[8] F. J. Serón y C. Bobed. «VOX System: A Semantic Embodied Conversational Agent exploiting Linked Data». En: *Multimedia Tools and Applications. Springer* 1 (ene. de 2016), págs. 381-404.

[9] *Sistema de extracción de información semántica de la DBpedia*. URL: https://zaguan.unizar.es/record/8795?ln=es (visitado 20-08-2016).

[10] R. Studer, V. R. Benjamins y D. Fensel. «Knowledge Engineering: Principles and Methods». En: *Data and Knowledge Engineering* 25 (1998), págs. 161-197.

[11] *SWING Guide*. URL: http://docs.oracle.com/javase/7/docs/technotes/guides/swing/ (visitado 20-08-2016).

[12] *URI Overview*. URL: https://www.w3.org/Addressing/URL/URI_Overview.html (visitado 20-08-2016).

[13] *W3C JSON*. URL: https://www.w3.org/TR/json-ld/ (visitado 20-08-2016).

[14] *W3C LinkedData*. URL: https://www.w3.org/standards/semanticweb/data (visitado 20-08-2016).

[15] *W3C RDF*. URL: https://www.w3.org/RDF/ (visitado 20-08-2016).

[16] *W3C Semantic Web*. URL: https://www.w3.org/standards/semanticweb/ (visitado 20-08-2016).

[17] *W3C Turtle*. URL: https://www.w3.org/TR/turtle/ (visitado 20-08-2016).

[18] *XML Essentials*. URL: https://www.w3.org/standards/xml/core.html (visitado 20-08-2016).

Índice de figuras

Anexo A

Problemas tecnológicos

A lo largo de este proyecto han surgido varios problemas de índole tecnológico. Estos problemas han dificultado el avance en ciertos momentos del mismo, planteando distintas soluciones o nuevas aproximaciones. Se va a usar la palabra mapa como el lienzo, el mundo, la parte visual. El espacio pixel se referirá a la pantalla física, fuera del mundo visual.

A.1. Renderizar texto

Uno de los problemas con los que tuvo que lidiar este proyecto, fue renderizar texto. Nuestros recursos y propiedades están denotados con caracteres, así que era necesario poder añadir texto con facilidad. En el prototipo del grafo se escribían las letras mediante *Polylines*.Las *Polylines* son letras con polígonos (Figura A.1). Cada letra tiene asociados un conjunto de puntos. Entonces, cuando se quiere imprimir en pantalla, se crean líneas entre esos puntos (con un orden ya definido) para formar la letra correspondiente. El problema de este método es que carecía de dos elementos: no se podían rellenar las letras de color; y producía errores de escalado en el zoom. Este último problema es debido a que el zoom no era constante en el número de aumentos, sino que cuanto menos zoom tenía, los pasos eran más pequeños y viceversa.

FIGURA A.1: Letras creadas con Polyline

Para solventar esto, se estudió el usar otro tipo de letras. OpenGL ofrece distintas soluciones al problema de poner texto en pantalla.

- Bitmaps: Los bitmaps crean las letras mediante un mapa de bits. Se define uno por cada letra y se usa cada vez que se escribe. Esta forma

escalaba bien y era barata en cuanto a recursos. El problema era que solo se podía tener un tipo de fuente. Como se quería poder tener varios tipos de fuente para distintos elementos, se eliminó.

- Texturas: Esta opción crea una textura por cada texto que se quiere incluir. De esta forma, se puede colocar la textura en el mapa y cuando se escale, se escalará el resto. El problema era que se tenía que hacer una textura por cada texto, lo cual es algo imposible teniendo en cuenta el contenido de la DBPedia.
- TextRenderer: La última opción era hacer un *render* de un texto. Esta opción permitía cambiar la letra en tiempo de ejecución y no gastaba tantos recursos en generar texturas infinitas. El problema era que no aceptaba el zoom ya que se escribía en espacio de pixel y no en espacio de pantalla.

A.1.1. TextRenderer

TextRenderer[1] es una clase de OpenGL que permite hacer render de texto en tiempo de ejecución. Tiene un pipeline propio de funcionamiento (Figura A.2), ya que hay que preparar el entorno para hacer un render.

FIGURA A.2: Pipeline TextRenderer

beginRendering() prepara a OpenGL para empezar a renderizar en el entorno. Como parámetros se le tiene que pasar el tamaño de la ventana actual. *draw* dibuja, en una posición dada, una cadena de texto en concreto. *endRendering()* finaliza el ciclo de render, devolviendo el estado de OpenGL al anterior de ejecutar *beginRendering()*.

El problema que tiene esta clase es que dibuja en espacio pixel, no en el mundo como tal. Si se le pide dibujar en la posición X = 220, Y = 100, va a dibujar en el pixel (220,100) de la pantalla. Esto significa que cuando se mueva el mapa, el renderizado se mantendrá en la misma posición. Además, al dibujar en espacio pixel, no se dibujará en la posición correcta deseada, ya que las coordenadas del mapa no son las mismas que las coordenadas pixel (Figura A.3).

[1]API TextRenderer : http://download.java.net/media/jogl/jogl-2.x-docs/com/sun/opengl/util/awt/TextRenderer.html

FIGURA A.3: Texto sin transformar coordenadas

Para solventar ese problema, tuvimos que transformar las coordenadas pixel a coordenadas de mapa. Para ello se recurrió a la librería GLU.

A.1.1.1. GLU

GLU[2] añade funcionalidades de alto nivel a OpenGL. Algunas de ellas:

- Escalar imágenes 2D
- Soporte para NURBS (*non-uniform rational B-spline*, usado para representar objetos 3D con curvas o superficies) y teselación (crear patrones de textura sin repeticiones)
- Transformación de coordenadas

GLU tiene la capacidad de hacer proyecciones para transformar coordenadas pixel a coordenadas de mapa y viceversa. Con esto se solventaba el problema de obtener las coordenadas en el sistema que nos interesaba.

LISTADO A.1: Método gluProject y gluUnProject

```
gluProject(GLdouble objX, GLdouble objY, GLdouble objZ,
    const GLdouble * model, const GLdouble * proj, const
    GLint * view, GLdouble* winX, GLdouble* winY, GLdouble*
    winZ);

gluUnProject(GLdouble winX, GLdouble winY, GLdouble winZ,
    const GLdouble * model, const GLdouble * proj, const
    GLint * view, GLdouble* objX, GLdouble* objY, GLdouble*
    objZ);
```

En el Listado A.1 se observa el método completo. *objX, objY y objZ* son las posiciones X,Y,Z del mundo. *model* es la matriz del modelo, *proj* es la matriz de proyección y *view* es la matriz *viewport*. Estas tres matrices las tiene el objeto OpenGL que está en uso. Por último, *winX, winY y WinZ* son las coordenadas X,Y,Z del espacio pixel.

Con esto, se puede observar que *gluProject* transforma del mapa al pixel; y *gluUnProject* transformar del pixel al mapa. Con estas dos funciones, el texto puede ser dibujado en la posición correcta aunque sea fuera del mapa. Se obtiene la posición deseada a dibujar (encima de un nodo, por ejemplo), se transformar con *gluProject* y se dibuja en el resultado.(Figura A.3).

[2]GLU: https://www.opengl.org/archives/resources/faq/technical/glu.htm

FIGURA A.4: Texto con transformación

A.2. Problemas OpenGL

OpenGL posee una característica negativa: su gasto en memoria y su trabajo en gráfica es alto. Además que tiene una curva de aprendizaje bastante dura (es poco intuitivo). Como solo se usa en un módulo (explorador de grafos) se planea mover el módulo a Swing, dejando el trabajo de gráfica solo para el avatar y consiguiendo unificar la interfaz ya que actualmente se combina Swing y OpenGL.

Con Swing se espera tener un rendimiento similar, pero se podrán usar las características de una mejor portabilidad y una interfaz más homogéneo, además de facilitar la inserción de texto y el control del zoom y del movimiento del mapa. Además, la gráfica estará dedicada al avatar solo.

Anexo B

Formato del registro acciones

En este anexo se va a tratar el formato del registro visto en **Resultado del proyecto**.

LISTADO B.1: Extracto del registro

```
25-07-2016_21:39:24 - iniciar Isaac_Newton ||
25-07-2016_21:39:27 - navegar Isaac_Newton |is subject of|
   List_of_English_Heritage_blue_plaques_in_the_City_of_Westminster
25-07-2016_21:39:30 - navegar
   List_of_English_Heritage_blue_plaques_in_the_City_of_Westminster
   |subject| Edward_Gibbon
25-07-2016_21:39:32 - clickar Isaac_Newton ||
25-07-2016_21:39:34 - pasarLista Isaac_Newton ||
25-07-2016_21:39:36 - pasarLista Isaac_Newton ||
25-07-2016_21:39:38 - navegar Isaac_Newton |subject|
   Category:Post-Reformation_Arian_Christians
25-07-2016_21:39:41 - navegar
   Category:Post-Reformation_Arian_Christians |is subject
   of| Patrick_Pakingham
25-07-2016_21:39:43 - navegar Patrick_Pakingham |subject|
   Category:Arianism
```

Este registro tiene un formato de línea distinto según la acción que registre. La parte común a todos es: `Hora - Accion`. Estos dos elementos son independientes de la acción realizada. Según la acción, la última parte de la línea será distinta. En esta parte, los elementos se separan con `|`.

- Iniciar: Con esta acción, lo único que le sigue es el recurso de inicio. En este caso, la línea sería `Hora - Accion Recurso ||`
- Navegar: En esta acción, se involucran dos recursos y una propiedad. El formato sería: `Hora - Accion R1 |Propiedad| R2`
- Clickar: Esta acción se realiza cuando el usuario hace click en un nodo del grafo (no en un elemento de **cuadro propiedad**). Su formato es: `Hora - Accion R||`
- Pasar Lista: En este caso, igual que el anterior, solo se guarda el recurso actual. Su formato es: `Hora - Accion R||`

- Pagina: Esta acción se guarda cuando se selecciona una nueva página y guarda la información en el siguiente formato: Su formato es: `Hora - Accion R||`

Como se ha dicho, este registro será extensible y se le podrán añadir más acciones según necesidad. Tan solo habrá que acceder a la clase correspondiente a `Accion` y añadir la nueva. Después llamar al `log` en donde se desee guardar la nueva línea de registro.

El nombre de cada fichero de acciones está definido en un documento de configuración del proyecto. Cada vez que guarda un elemento nuevo, coge el nombre definido y le añade un número correspondiente al siguiente documento de la carpeta.

Anexo C

Fichero de configuración y colores

En este anexo se muestra el fichero de configuración del proyecto. Para dar opciones de personalización al usuario, se decidió que algunas características del programa fuesen modificables mediante un fichero. Este fichero es el llamado Fichero de configuración.

Este fichero debe tener un nombre en concreto: **congif.conf** y una estructura dada (Listado C.1)

LISTADO C.1: Configuracion estandar usada

```
carpetaRegistro = Registroaccion
nombreFIchero = actionLog
numeroCurvas = 2
colorNodoNOT = Grey
colorNodoON = Chocolate
colorNodoOVER = IndiantaRed
ColorMigaON = Chocolate
ColorMigaOVER = IndianaRed
ColorBFTriangulo = DarkerKhaki
colorBFCuadrado = DarkKhaki
colorBFApagado = Grey
colorRecON = DarkKhaki
colorRecOVER = SaddleBrown
```

Todas estas opciones es obligatorio que esten pero no se necesitan en un orden definido. Existen diversas opciones (la mayoría visuales) que se van a explicar a continuación:

- *carpetaRegistro*: Esta primera opción especifica la carpeta de destino del registro de la sesión que se va a iniciar. SI la carpeta no existe la crea.
- *nombreFichero*: Esta opción permite imponer el nombre para el fichero de registro. Este nombre irá seguido de un número según los elementos que haya en la carpeta destino. Por ejemplo, si hay cuatro elementos en la carpeta, el fichero se llamará *actionLog_5*.

- *numeroCurvas*: El número de curvas que salen del recurso central también son personalizables (hasta un máximo de 5).
- *colorNodoNOT, colorNodoON y colorNodoOVER*. Estas tres opciones permiten cambiar el color de los recursos centrales. El color *NOT* se refiere al color cuando no es el recurso activo, el color *ON* indica el color del recurso actual y el color *OVER* modifica el color para cuando se pasa el cursor por encima.
- *colorMigaON, colorMigaOVER*: Estas dos opciones modifican los colores de los elementos gráficos de las migas. *ON* es el color de todas las migas que no son las actuales y *OVER* el color de la miga actual.
- *colorBFTriangulo, colorBFCuadrado, colorBFApagado*: Denotan el color de los botones de los objetos *cuadro relación*. *Triangulo* se refiere al pequeño triángulo dentro de cara botón, *Cuadrado* se refiere al resto del botón y *Apagado* es el color que se usa cuando el botón esta desactivado (debido a que ya no hay más elementos en esa dirección).
- *colorRecON, colorRecOVER*: Estas dos últimas opciones se refieren a los recursos dentro de un **cuadro relación**. En este caso, *ON* es el color de elemento sin pasar el cursos sobre él, y *OVER* es el color al pasar el cursos por encima.

Todas estas opciones permiten al usuario modificar el aspecto visual a su gusto, dando una mayor libertad al usuario (aunque deberá reiniciar el programa para que surja efecto).

C.1. Colores

En esta sección se va a colocar la lista con los nombres de los colores actuales. Aparte de estos, se puede usar cualquier paleta de colores expresada en RGB.

	R	G	B	
Black	0	0	0	
White	1	1	1	
Red	1	0	0	
Green	0	1	0	
Blue	0	0	1	
IndianRed	0.8	0.36	0.36	
Salmon	0.98	0.5	0.447	
Coral	1	0.5	0.313	
Chocolate	0.823	0.421	0.117	
Peru	0.803	0.521	0.247	
SaddleBrown	0.545	0.411	0.07	
DarkKhaki	0.741	0.717	0.419	
DarkerKhaki	0.641	0.616	0.319	
Grey	0.2	0.2	0.2	
Greyer	0.5	0.5	0.5	

FIGURA C.1: Tabla de colores de la aplicación