Universidad de Valladolid

E. T. S. DE INGENIERÍA INFORMÁTICA

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Desarrollo de una Interfaz Gráfica de Consulta para el Proyecto de Datos Abiertos "Census 2001"

Autor:
Iván Loma Treceño

Tutores:
Miguel Ángel Martínez Prieto
Javier David Fernández García

Deseo expresar mi más sincero agradecimiento

a mis tutores, por guiarme y ayudarme
en todo momento en este proyecto,

a mi familia por haberme apoyado durante todos
estos años y por ayudarme a llegar hasta aquí,

a todos mis amigos y compañeros
en especial a Mercedes, Álvaro y Luis Miguel,
con los que he compartido
la mayoría de mis mejores y peores momentos y
sin los cuales no hubiese podido recorrer este largo camino.

Gracias a todos.

Resumen:

El actual "diluvio de datos" está inundando la Web con grandes volúmenes de información representados en RDF, dando lugar a la Web de Datos. Esto, unido a las nuevas iniciativas, tanto de gobiernos internacionales como locales, sobre la publicación de datos públicos en formatos estándar puede permitir la creación de nuevas herramientas con las que fomentar la transparencia, el escrutinio público y la innovación. Uno de estos grandes volúmenes de datos publicados es el censo español, en el cual el acceso se restringía generalmente a puntos de vista limitados de los datos, pero que actualmente ha sido ha convertido desde un formato plano inoperable a RDF.

Este proyecto presenta una primera aproximación hacia una interfaz gráfica que permita realizar consultas SPARQL simples sobre dicho censo, en formato RDF, mediante la generación de un patrón de grafo que represente la consulta deseada. El principal objetivo de este proyecto es permitir a cualquier usuario realizar consultas simples sobre el censo, garantizando así la transparencia, el escrutinio público y la posible innovación (realizando estudios estadísticos, ofreciendo nuevos servicios, etc.) con los datos censales.

Abstract:

Current data deluge is flooding the Web with very large RDF datasets, resulting in the so-called Web of Data. This fact, together with initiatives from international and local governments addressing the publication of open data using standards, encourage the creation of novel tools to promote transparency and innovation. The Spanish Census is a perfect example of such a very large open dataset, in which the access was traditionally restricted to a limited view of the total data, hence it has been converted to the interoperable RDF format.

This project is a first step towards a graphical interface providing simple SPARQL query resolution over the census in RDF format, while directly drawing the graph pattern representing the query. The main objective of this project is to fill the existing gap with the users and allow these simple queries over the census, and thus achieving the objectives of transparency and potential innovation with the census data, e.g. through statistical studies, new services on top of the novel infrastructure, etc.

ÍNDICES

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

Capítulo 1
INTRODUCCIÓN

1. INTRODUCCIÓN

1.1. Descripción del Proyecto

El siguiente documento presenta el sistema *"Interfaz Gráfica de Consulta para el Proyecto de Datos Abiertos Census 2001"*, realizado por Iván Loma Treceño como parte del Trabajo Fin de Grado.

Este proyecto se encuadra dentro del Proyecto de Datos Abiertos Census 2001[24] cuyo objetivo es incentivar la publicación de datos abiertos por el Gobierno de España, en el marco de la transparencia e interoperabilidad fijada por el movimiento Open Data. El principal objetivo a perseguir ha sido desarrollar una primera aproximación hacia una interfaz gráfica de consulta SPARQL sobre el conjunto de datos RDF fijo y ya disponible, como son los datos del censo español de 2001.

El proyecto en concreto, consiste en el análisis, documentación y desarrollo de una interfaz gráfica para la consulta del censo 2001 RDF, sobre la plataforma Flash. En este primer capítulo introductorio, daré una descripción general del proyecto, expondré los objetivos del mismo y explicaré como se organizan los contenidos de la documentación.

1.2. Alcance del Proyecto

Hasta la fecha, el gran potencial de la Web de Datos no está siendo explotado en su totalidad. Una de sus razones, junto con la falta de soluciones escalables, es la usabilidad desde el punto de vista de un usuario "común". Los datos se encuentran, o deben encontrar, en formatos estándares como RDF [25], mientras que las consultas son del tipo SPARQL [26]. Sin embargo, la interfaz de consulta SPARQL requiere de un entrenamiento previo que no siempre se puede exigir. El uso de interfaces gráficas surge como una solución trivial, pero su diseño en muchas ocasiones forma a ser parte del problema más que de la solución. Por una parte, la solución debe ser simple y fácil de usar para obtener una buena experiencia de usuario. Por otra, debe ser suficientemente potente como para no perder expresividad en la consulta, y poder captar también a usuarios más avanzados.

Por lo tanto este proyecto está orientado a facilitar la realización consultas sobre un conjunto de datos RDF utilizando el lenguaje SPARQL. Para ello, tiene como objetivo el desarrollo de una primera aproximación hacia una interfaz gráfica que permita al usuario "común" realizar consultas sobre un conjunto de datos RDF definido, como es en este caso el censo español de 2001.

Como ya se ha mencionado, la utilización de esta interfaz está destinada a todo tipo de usuario que desee realizar algún tipo de consulta sobre el censo español de 2001, aunque principalmente estará destinada a aquellas personas que deseen realizar un aprendizaje práctico sobre el lenguaje SPARQL o a aquellas que deseen realizar estudios estadísticos.

Además este proyecto está destinado a todos aquellos que deseen obtener información sobre el estudio y desarrollo de interfaces gráficas desarrolladas en ActionScript y el aprovechamiento para ello de herramientas ya existentes como es Adobe Flash Builder 4.6 y UML 2 [3].

Propone un paso más para incentivar la publicación de datos abiertos por el Gobierno de España, en el marco de la transparencia e interoperabilidad fijada por el movimiento Open Data.

1.3. Objetivos

Seguidamente, se describen los principales objetivos que este trabajo fin de grado pretende cumplir:

Objetivos específicos del proyecto:

- Desarrollar una interfaz gráfica que permita construir un grafo de consulta SPARQL sobre el Censo español de 2001 en formato RDF.
- El sistema manejará internamente la representación SPARQL de la consulta gráfica.
- El sistema actuará de intermediario con el sistema que representa la información censal en RDF para ejecutar y devolver el resultado de las consultas.
- Desarrollar dicha interfaz con Adobe Flash Builder y su lenguaje ActionScript.
- Aplicar los principios generales de Análisis y Diseño Orientados a Objetos, con el fin de facilitar la reutilización y portabilidad del código resultante
- Intentar concienciar en cierta medida a las personas, gobiernos, etc sobre las ventajas de la filosofía Link Open Data.

Objetivos generales:

- Conocer y utilizar un entorno de trabajo específico, en nuestro caso el desarrollo de aplicaciones con Adobe Flash Builder, centrándonos en el lenguaje de programación ActionScript.
- Poner en práctica nuestras habilidades para el trabajo, y mejorarlas en la medida de lo posible.
- Simular un proceso de trabajo profesional siguiendo todos los pasos de análisis, diseño e implementación, tratando de cumplir con todos los requisitos establecidos en un período de tiempo establecido.
- Conocer y comprender la filosofía Link Open Data [21], la web de datos y sus ventajas y sus inconvenientes.
- Usar todos los conocimientos adquiridos a lo largo de nuestra carrera universitaria para la elaboración del Trabajo Fin de Grado.

1.4. Contenido de la memoria

La memoria se organiza en capítulos y estos a su vez en diferentes puntos:

- Capítulo 1. Introducción: Sección en la que nos encontramos actualmente. Se trata de una breve presentación del tema y objetivos que se abordarán en el proyecto. Por último aparece el contenido de la memoria.
- Capítulo 2. La Web de Datos: Sección donde se expondrá un resumen general de la Web de Datos, así como de su formato RDF estándar y de su lenguaje de consulta SPARQL y del estado del arte en editores SPARQL.
- Capítulo 3. Proyecto "Census 2001 RDF": En esta sección se expondrá un resumen sobre dicho proyecto describiendo sus características y realizando una comparativa con el censo 2001 original.
- Capítulo 4. Desarrollo del proyecto: Este capítulo contiene toda la documentación necesaria para hacer un seguimiento del desarrollo del proyecto. Aparecen la planificación del tiempo, la gestión de riesgos, los modelos de Análisis (incluye el SRS y los casos de uso), de Diseño (diferenciando las capas Vista, Lógica y Persistencia) e Implementación (herramientas utilizadas para desarrollar el proyecto).
- Capítulo 5. Pruebas: Sección donde se expondrá un resumen general de las pruebas realizadas sobre la aplicación para encontrar fallos y mejorar la misma.
- Capítulo 6. Manuales de usuario: En este capitulo se muestran los distintos manuales para instalar el software, así como el manual de utilización de la propia aplicación.
- Capítulo 7. Conclusiones: Esta sección pretende mostrar posibles nuevas funcionalidades y futuros cambios a la vez que se hace una reflexión sobre los conocimientos adquiridos.
- Capítulo 8. Bibliografía: Sección en la que se muestran las fuentes bibliográficas, así como referencias Web, consultadas durante la elaboración del proyecto
- Apéndice: Recopilación del software y hardware utilizado. Además de un apartado donde se expondrán los valores permitidos en los campos de las relaciones.

1.5. Contenido del CD-ROM

El contenido del CD-ROM adjunto al proyecto es el siguiente:

- **Memoria:** El documento impreso completo, en un archivo único con el nombre "memoria.pdf" (formato .pdf).
- **Software:** Directorio que contiene todo el código fuente del software desarrollado (formato .mxml .as y .xml).
- **Ejecutables:** Directorio con los ejecutables para la inicialización del software desarrollado.
- **Manual de Usuario:** Breve manual que nos indica como utilizar la aplicación.

Capítulo 2
LA WEB DE DATOS

2. LA WEB DE DATOS

2.1. Introducción

La Web Semántica está viéndose superada por una versión menos estricta conocida como "la Web de Datos" [20]. Esta iniciativa pretende facilitar el acceso, publicación y reutilización de datos, en oposición al tradicional punto de vista "basado en documentos" de la Web clásica. El fin último es poder construir aplicaciones más ricas que aprovechen la gran cantidad de datos expuestos. El principal valedor de dicho modelo es la filosofía Linked Open Data [21], que aboga por disponer de datos abiertos y enlazados que están expresados en formato RDF [25] para que sistemas y agentes software puedan explotar estos datos de forma automática (recopilándolos, agregándolos, interpretándolos, publicándolos, etc.) utilizando vocabularios consensuados y ontologías.

2.1.1. Principios de la Web de Datos

El término Linked Data fue propuesto por Tim Berners-Lee, el creador de la World Wide Web. Dicho término se refiere a una forma de publicar y enlazar datos estructurados en la Web utilizando RDF, un lenguaje para representar información sobre recursos propuesto por el Consorcio de la World Wide Web en el área de la Web Semántica. El valor y la utilidad de los datos enlazados es mayor tanto en cuanto éstos estén más interconectados con otros datos en la Web de Datos [13].

Los cuatro principios de diseño en los que se basa la Web de Datos Enlazados son:

1. Utilizar URIs (Uniform Resource Identifier) como nombres únicos para los recursos.
2. Utilizar el protocolo HTTP para nombrar y resolver la ubicación de los datos identificados mediante esas URIs.
3. Representar los datos en RDF (un modelo de datos semi-estructurado que permite expresar sentencias como triples (sujeto; predicado; objeto) que pueden verse como grafos etiquetados) y utilizar SPARQL como lenguaje de consulta de dichos datos.
4. Incluir enlaces a otras URIs para permitir la localización de más datos enlazados.

Los datos en la Web de Datos se describen mediante el uso de términos que aparecen en vocabularios consensuados u ontologías.

2.1.2. Generación de nuevos datos en la Web de Datos

El primer paso para la generación de nuevos datos consiste en identificar qué información se quiere publicar de forma abierta como datos enlazados. También es importante informarse sobre otros datos relacionados ya presentes en la Web de Datos para así transformar, enlazar y publicar los datos propios con los ya presentes con el fin de que éstos puedan ser explotados automáticamente por sistemas software.

En el segundo paso se trasforma todos los datos seleccionados al formato RDF, es decir, se abre y publica la información almacenada en bases de datos, ficheros, hojas de cálculo, etc. en un formato reutilizable por todos.

Y como último paso, los datos transformados se enlazan con otros datos ya disponibles en la Web de Datos. En este último paso entra en juego el proyecto Liked Open Data, el cual reúne una gran cantidad de colecciones de datos enlazadas entre si con las que podemos relacionar nuestros datos. En 2011 este proyecto contaba con 295 colecciones de datos (DBpedia, GeoNames, etc.) y más de 31000 millones de triples RDF, creciendo día a día.

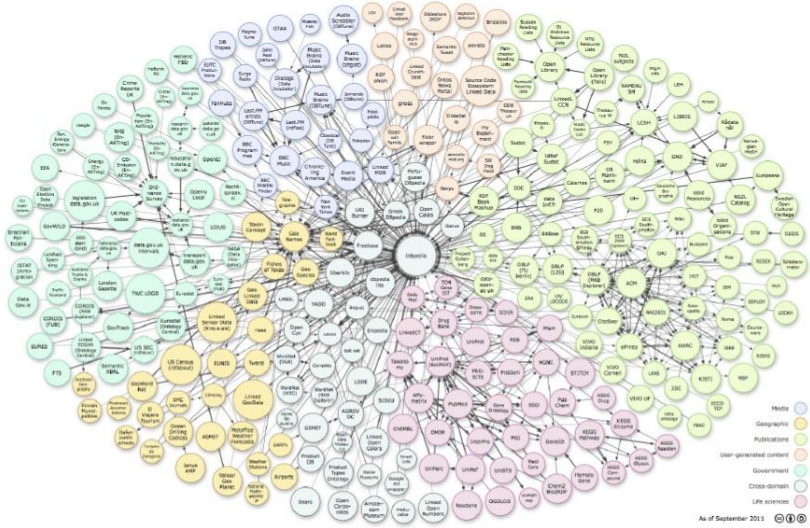

Figura 2.1: Colecciones de datos en el proyecto Linked Open Data (2011)

2.1.3. Datos enlazados en el ámbito gubernamental

En la actualidad, en la "nube de datos" de la filosofía Linked Open Data coexisten datos provenientes de muy diversos dominios (biología, datos geográficos, publicaciones, etc.).

En particular, el movimiento de datos abiertos se ha extendido con éxito a los datos gubernamentales, produciéndose un cambio histórico de filosofía hacia la transparencia. El uso de los datos enlazados en el Sector Público tiene un doble fin. El primero es proporcionar datos públicos más accesibles a la ciudadanía en un formato reutilizable y, el segundo, proporcionar un punto de acceso único a la información gubernamental en el que los datos están conectados y en el que es posible utilizarlos de forma automatizada por sistemas software. La apertura y disponibilidad de estos datos creará nuevas oportunidades de negocio al permitir a terceros crear nuevos servicios de valor añadido utilizando los datos públicos de forma integrada.

Algunos de los ejemplos de Open Data en los gobiernos de España son los siguientes:

- El portal http://datos.gob.es/datos/ que organiza y gestiona el Catálogo de Información Pública de la Administración General del Estado. Posee 581 "puntos de información".

- El gobierno de Euskadi donde en el portal http://opendata.euskadi.net/ provee de acceso a los datos públicos del Gobierno Vasco en formato reutilizable con el fin de que terceros creen servicios derivados de esos datos.
- El gobierno de Asturias provee de acceso a distintos datos de dicho gobierno en el portal http://risp.asturias.es.
- El Ayuntamiento de Zaragoza provee de una serie de datos en el portal http://datos.zaragoza.es para fomentar la reutilización de dicha información por parte de la ciudadanía, las empresas y otros organismos, ofreciendo así un aumento de la transparencia de la administración, el incremento de la participación ciudadana y la posibilidad de crecimiento económico en distintos sectores.
- La Junta de Castilla y León provee en el portal de datos abiertos http://www.datosabiertos.jcyl.es/ de acceso a distinta información sobre el sector público en Castilla y León con el fin de facilitar su reutilización.

No cabe duda de que la generación y utilización de tal cantidad de datos enlazados procedentes de fuentes heterogéneas creará nuevas oportunidades de investigación y de negocio y en caso de los datos gubernamentales además proporcionará transparencia e interoperabilidad.

2.2. RDF: Framework para la Descripción de Recursos.

El Framework para la Descripción de Recursos (RDF) [25] es un lenguaje utilizado para representar la información sobre "recursos" en la World Wide Web. Proporciona un marco común para la expresión de dicha información, para que ésta pueda ser intercambiada entre aplicaciones, sin pérdida de significado, a través de la web.

RDF se basa en la idea de identificar las cosas usando identificadores Web (llamados Identificadores Uniformes de Recursos o URI), y la descripción de los recursos en términos de propiedades simples y valores de propiedad. Esto permite expresar las sentencias como triples (sujeto; predicado; objeto) que pueden verse como grafos etiquetados. En la siguiente figura se muestra un claro ejemplo de un conjunto de datos RDF y su grafo asociado:

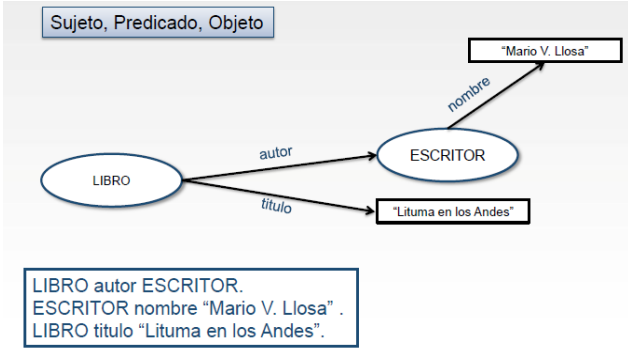

Figura 2.2: Grafo RDF

RDF nos permite enlazar datos mediante el uso de hipervínculos, es decir, creando un triple donde el objeto es una URI del dato que queremos enlazar.

Además, RDF nos permite utilizar un tipo de nodo “variable”, Blank Node. Este tipo de nodo nos permite conectar distintas partes del grafo sin la necesidad de especificar un URI.

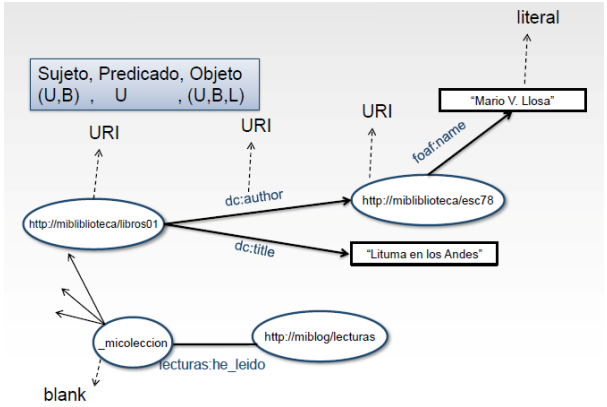

Figura 2.3: Grafo RDF posibles valores triple

En la anterior figura se muestra los posibles tipos de valores que pueden tomar el Sujeto, el Predicado y el Objeto (U = Uri; B = Blank Node; L = Literal).

2.3. SPARQL: Lenguaje de Consulta

SPARQL [26] es el lenguaje de consulta estandarizado para la consulta de grafos RDF, el formato con el que se modelan los datos en la Web de Datos. Se trata de un lenguaje con bastante similitud a SQL, la diferencia principal es que SQL supone que los datos están implementados en tablas y SPARQL supone que los datos están implementados en grafos, de modo que el cambio de filosofía dificulta la labor de aprendizaje. SPARQL esta recomendado por el W3C [31] desde el 15 de Enero de 2008.

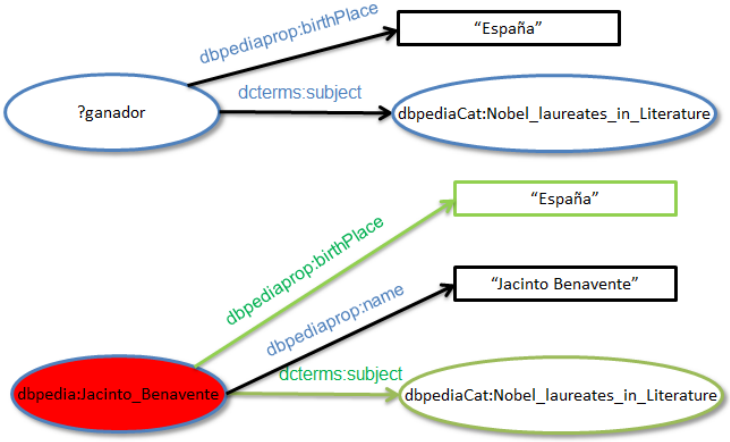

Figura 2.4: Ejemplo: Emparejamiento de Grafo de consulta

2.3.1. Sintaxis básica

Como ya se ha dicho este lenguaje tiene bastante similitud con SQL pero con la diferencia principal de que actúa sobre grafos en vez de tablas. En este apartado se describe la sintaxis básica del lenguaje SPARQL.

2.3.1.1 Formas de Consulta

Existen las siguientes formas de consulta:

- **SELECT:** Nos permite realizar consultas sobre un conjunto de triples utilizando variables, devolviendo como resultado los valores que coincidan con el grafo.

  ```
  SELECT ?ganador
  WHERE {
    ?ganador dcterms:subject dbpediaCat:Nobel_laurates_in_Literature .
    ?ganador dbpediaprop:birthPlace "Espana".
  }
  ```

  Esta consulta es la consulta asociada al grafo de la figura 2.4, la cual busca todos los ganadores del premio "Nobel de Literatura" cuyo lugar de nacimiento sea España. Mediante el SELECT obtenemos las distintas URIs de estos ganadores.

- **ASK:** Nos permite preguntar si un triple o un conjunto de triples existe en el dominio, devolviendo como resultado yes o no.

  ```
  ASK {
    ?ganador dcterms:subject dbpediaCat:Nobel_laurates_in_Literature .
    ?ganador dbpediaprop:birthPlace "Espana".
  }
  ```

  La anterior consulta comprueba si existe un ganador del premio Nobel de Literatura que sea español, devolviendo true en caso de que exista al menos uno y false si no existe ninguno.

- **CONSTRUCT:** Nos permite crear nuevos triples, en combinación con otros existentes. Devuelve como resultado los nuevos triples.

  ```
  CONSTRUCT <http://www.ejemplo.org/Yo> miVoc:deseoLeer ?ganador {
    ?ganador dcterms:subject dbpediaCat:Nobel_laurates_in_Literature .
    ?ganador dbpediaprop:birthPlace "Espana".
  }
  ```

  En este caso la consulta construye una serie de triples con cada uno de los ganadores del premio Nobel de Literatura españoles. Uno de los resultados:

```
<http://www.ejemplo.org/Yo>  miVoc:deseoLeer  dbpedia:Jacinto_Benavente
```

- **DESCRIBE:** Nos obtener toda la información acerca de una URI.

  DESCRIBE `dbpedia:Pablo_Neruda`

  Esta consulta devolverá toda la información de DBpedia [14] acerca de Pablo Neruda.

2.3.1.2 Modificadores

Los modificadores nos permiten ordenar, distinguir, reducir, limitar e ignorar una cantidad de los resultados de la consulta. Existen los siguientes modificadores:

- **ORDER BY:** Ordena los resultados.
- **GROUP BY:** Agrupa los resultados con mismo valor.
- **DISTINCT:** Elimina los resultados duplicados.
- **REDUCED:** Permite la eliminación de alguna de los resultados duplicados.
- **LIMIT:** Limita el número de soluciones devueltas.
- **OFFSET X:** Elimina las primeras X soluciones.

Un ejemplo de consulta SPARQL con modificadores:

```
SELECT DISTINCT ?name
        WHERE { ?x foaf:name ?name .}
ORDER BY ?name
LIMIT 3
OFFSET 1
```

La anterior consulta ordenará los nombres por orden alfabético, eliminara el primer resultado y devolverá los tres primeros resultados.

2.3.1.3 UNION y OPTIONAL

- **UNION:** Esta palabra clave nos permite combinar dos patrones de grafo, ya sean disjuntos o no, devolviendo todos los resultados posibles para ambos grafos.

```
SELECT ?escritor
WHERE{
    {
    ?escritor dcterms:subject  dbpediaCat:Nobel_laurates_in_Literature.
    ?escritor dbpediaprop:birthPlace "Chile" .
    }
UNION {
    ?escritor dcterms:subject dbpediaCat:Nobel_laurates_in_Literature.
    ?escritor dbpediaprop:birthPlace "España" .
    }
}
```

En la anterior consulta devolverá como resultado todos los ganadores del premio Nobel de Literatura nacidos tanto en Chile como en España.

- **OPTIONAL:** Nos permite añadir el resultado a la solución cuando la información existe, pero no rechazar la solución cuando la información no este disponible.

```
SELECT ?nombre ?nombreAlternativo
WHERE
{
  ?escritor dcterms:subject  dbpediaCat:Nobel_laurates_in_Literature.
  ?escritor  dbpediaprop:name  ?nombre .
  OPTIONAL{?escritor dbpediaprop:alternativeNames ?nombreAlternativo .}
}
```

Esta consulta devuelve el nombre de los ganadores del premio Nobel de Literatura y sus nombreAlternativo cuando éstos existan, pero no elimina los resultados que no tengan un nombreAlternativo.

2.3.1.4 Filtros

A la hora de realizar una consulta se pueden incluir filtros (idioma, tipo de datos, que sea una URI o un literal, aritméticos, etc.) mediante la palabra clave **FILTER**. Los distintos filtros son: lang, datatype, regex, isURI, isLiteral, isBlank, (aritméticas).

```
SELECT  ?title ?price
WHERE   {
    ?x  dc:title  ?title .
    ?x ns:price ?price . FILTER (?price < 30)
}
```

La anterior consulta devuelve el titulo y precio de aquellos productos cuyo precio no supere el valor 30.

2.4. Estado del Arte en editores de SPARQL

Existen distintas aplicaciones que nos permiten realizar consultas SPARQL directamente, de forma textual o dibujando su grafo, sobre la web de datos. Por ello distinguimos tres tipos de aplicaciones:

2.4.1. Interfaces Exploratorias

Este tipo de aplicaciones son en realidad buscadores de información donde se introduce una URI y muestra toda la información disponible que existe en la web de datos. Las principales aplicaciones de este tipo son:

- **Tabulator [28]:** Se trata de un buscador y editor para la web de datos. Actualmente esta disponible como aplicación Web o como componente de Firefox. En las últimas versiones nos permite crear consultas SPARQL mediante la exploración de la web de datos.

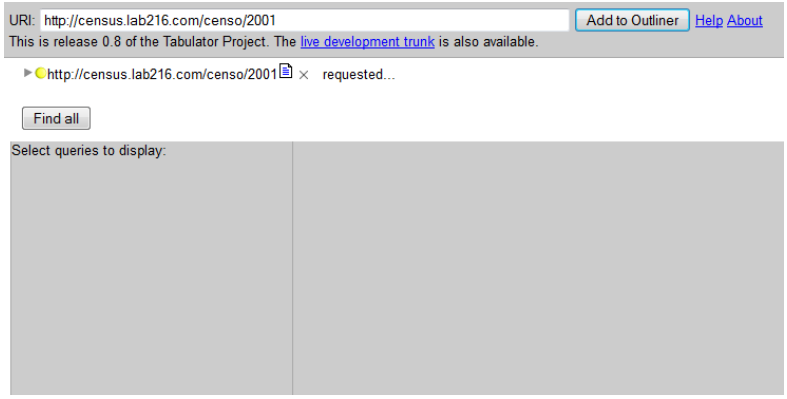

Figura 2.5: Aplicación Tabulator

- **DISCO [15]:** Se trata de un buscador para la web de Datos. Se puede navegar a través de los resultados obtenidos mediante los hiperenlaces.

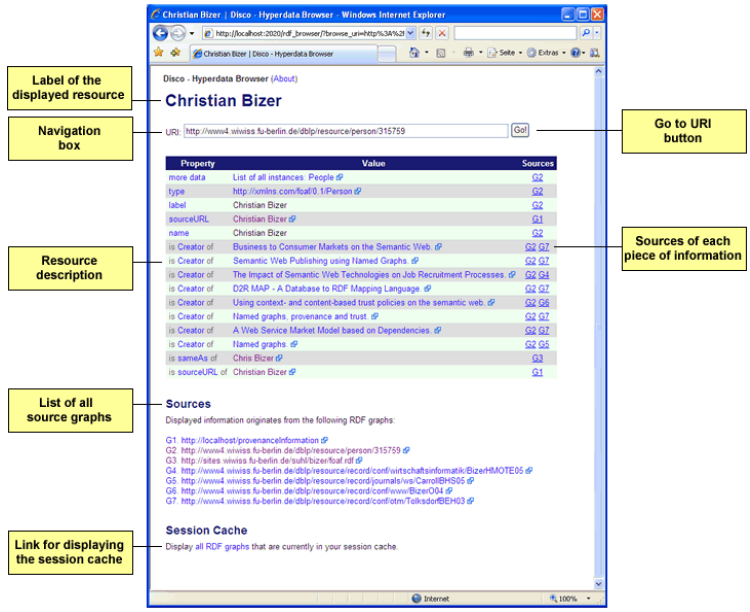

Figura 2.6: Aplicación DISCO

2.4.2. Editores SPARQL textuales

Este tipo de aplicaciones nos permiten realizar consultas SPARQL de forma textual, ofreciéndonos facilidades para la creación de las consultas. Algunas de las aplicaciones de este tipo son:

- **Twinkle [29]:** Aplicación de escritorio que permite realizar consultas SPARQL textuales. Destinada tanto para usuarios principiantes (ofrece ayudas y consultas predefinidas) como expertos.

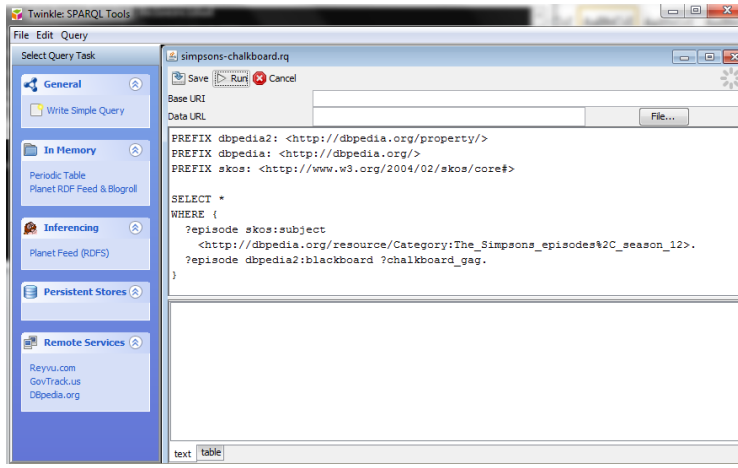

Figura 2.7: Aplicación Twinkle

- **SPARQL GUI [27]:** Aplicación de escritorio para realizar consultas SPARQL. Nos permite cargar RDF desde ficheros externos o URI.

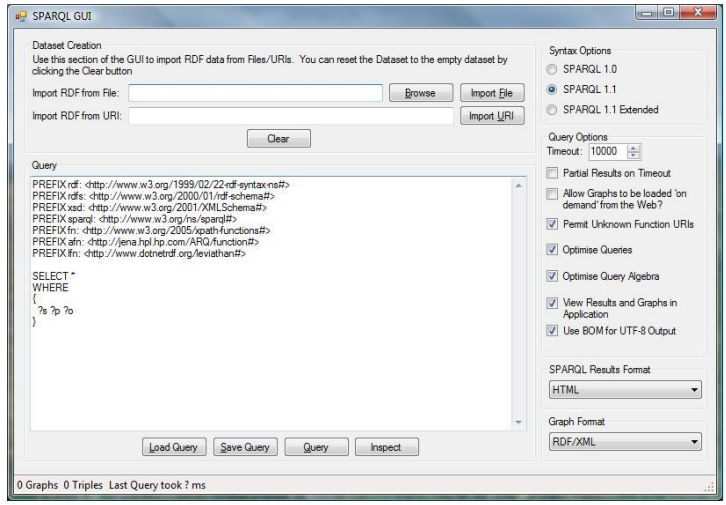

Figura 2.8: Aplicación SPARQL GUI

2.4.3. Editores SPARQL gráficos

Este tipo de aplicaciones nos permiten realizar consultas SPARQL mediante la generación de su grafo de consulta, manejando internamente el código SPARQL generado. Las principales aplicaciones de este tipo son:

- **gFacet [17]:** Aplicación basada en facetas (categorías), que nos permite explorar los datos RDF combinando una visualización en modo de grafo con las técnicas de filtrado de categorías (cada nodo representa una categoría). No se tiene acceso al SPARQL generado.

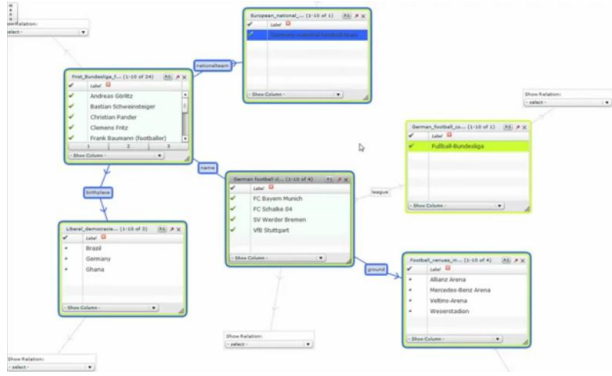

Figura 2.9: Aplicación gFacet

- **iSPARQL [19]:** OpenLink iSPARQL es una aplicación que permite construir y ejecutar consultas SPARQL tanto de forma textual como gráfica (generando el grafo de consulta). Está conectado con el poderoso endpoint de Virtuoso.

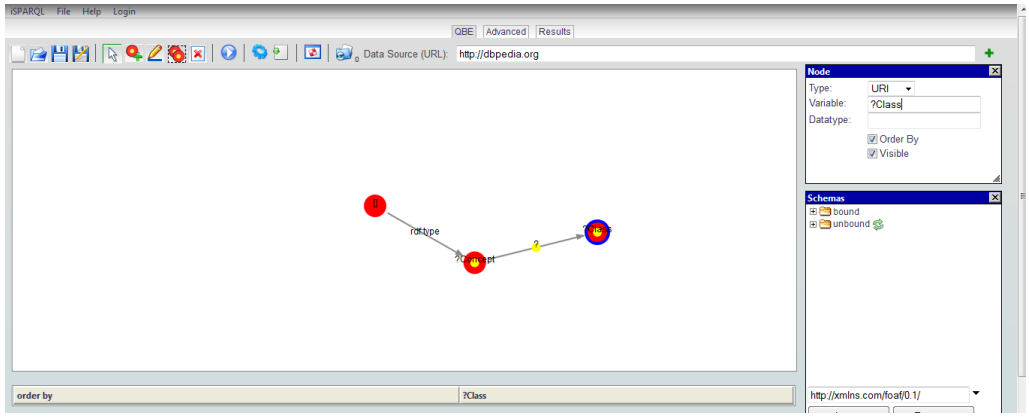

Figura 2.10: Aplicación iSPARQL

Esta última aplicación es una de las soluciones más similares al trabajo que se pretende realizar en este proyecto, ya que combina una interfaz gráfica de consulta con su representación en el lenguaje SPARQL.

Capítulo 3
PROYECTO “CENSUS 2001 RDF”

3. PROYECTO "Census 2001 RDF"

3.1. El censo Español

Desde que fue creado en 1768, el censo ha considerado a los individuos como una unidad de análisis y su estudio como una de las más poderosas retroalimentaciones para el gobierno. El censo [4] se lleva a cabo con una periodicidad de 10 años, reflejando importantes hechos históricos como la gripe española de 1918, la guerra civil, etc.

Se pueden agrupar los fines y usos del censo en: conocimiento, estudio e integración y normalización. El censo ha sido diseñado, principalmente, para contar la población, viviendas y edificios. Este conocimiento produce una imagen estructural de la población en términos de demografía, economía, sociología y cultura, y proporciona una herramienta de peso para las políticas gubernamentales. Los datos recolectados son la base para el estudio de la evolución de la población y para el desarrollo de estadísticas por muestreo.

Por otra parte, el inmenso esfuerzo realizado para recopilar toda esta información obliga a la integración y normalización de las herramientas gubernamentales, tales como mapas y cartografías, comunicaciones entre agencias locales, etc. Todas estas mejoras benefician la administración pública.

3.2. Censo 2001 original

El censo de 2001 español es el 16º censo oficial del país y se basó en un modelo de censo clásico, es decir, una consulta exhaustiva del territorio. Sin embargo, el censo de 2001 contó por primera vez con el apoyo del Padrón municipal, registrando a los residentes de cada municipio. El proceso seguido se resume de la siguiente forma:

1. Preparación: Se establece la estrategia y probada en un ensayo piloto.
2. Recopilación de datos: Es la tarea más ardua de la elaboración del censo, por ello en esta fase se contratan "agentes censales". Fue uno de los primeros censos en el mundo en permitir rellenar los formularios por internet.
3. Tratamiento informático: Se procesan todos los formularios, que después son destruidos para mantener el secreto estadístico, ya que está prohibido conocer a la persona correspondiente a los datos particulares.
4. Publicación de los resultados.

El proceso total se llevo acabo gracias al esfuerzo de más de 42.000 personas, con presupuesto total de 165.278.328 euros, de los cuales se dirigió el 70% de censo de los agentes censales (datos del INE [18]).

No obstante, a pesar de todo el esfuerzo realizado y el dinero gastado, los datos finales son publicados a través de:

- Archivos de microdatos. Son archivos ASCII que muestra el total de los datos en registros y representa cada campo con una longitud fija. El archivo principal (5% de muestreo), incluye un registro por persona anónima. Su análisis no es trivial.

```
0000010001121967033110841992210031967196719684199219944102399441201 2 71144
5601030101000401012003033341005001001040004000002001001000000000000001001000000000000000000
02711445216 61971030210841994
0000010002021998003610841994131011998199819981 1 11 11
20 1010301010004010120030333410050010010400040000000100100000000000001001000000000000000
027114452161971030210841994 04 2040201010000000
0000010003081971030610841994221041971197119754199219944104 04
5601030101000401012003033341005001001040004000002001001000000000000001001000000000000000000
02711445216 11967033210841992
0000010004051993008110841091131021993199319931 1 01 01
20 1010301010004010120030333410050010010400040000000100100000000000001001000000000000000
027114452161971030210841994 04 2040201010000000
0000020001121930070610811993310021930193019301 199311 06 06
```

Figura 3.1: Microdatos proporcionados por el INE

- Cifras de población que resumen los resultados más importantes.

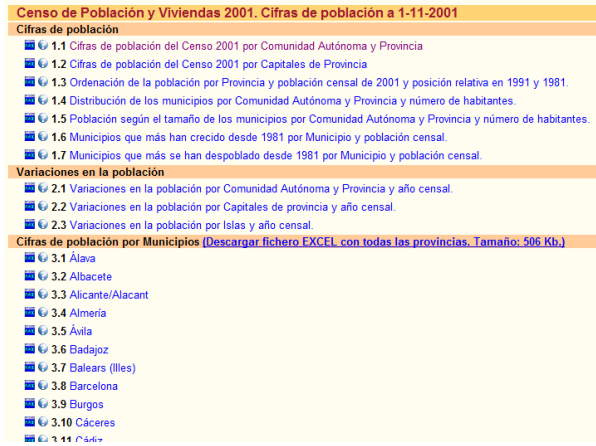

Figura 3.2: Cifras de población (INE)

- Un sistema de consulta básica que permite al usuario construir un conjunto limitado de preguntas condicionadas por la interfaz del sistema de consulta.

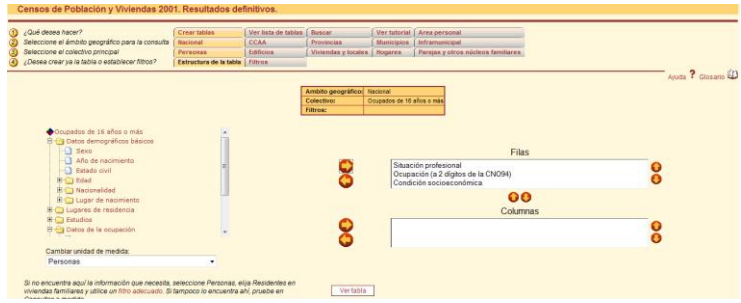

Figura 3.3: Sistema de consultas (INE)

En términos generales, la calidad y la utilidad de los datos publicados, en la conclusión de todo el proceso, es cuestionable.

3.3. Censo 2001 RDF

Censo 2001 RDF [24] es la conversión RDF del formato original en texto plano (archivos de microdatos) proporcionado por el Instituto Nacional de Estadística (INE) de España, representando una muestra del 5% del censo 2001 español. Este trabajo ha sido realizado por Javier D. Fernández, Miguel A. Martínez-Prieto y Claudio Gutiérrez [4].

Para realizar dicha transformación lo primero fue identificar la información que se desea transformar a RDF. En este caso el número de identidades presentes está bien acotado y se muestra en el siguiente diagrama:

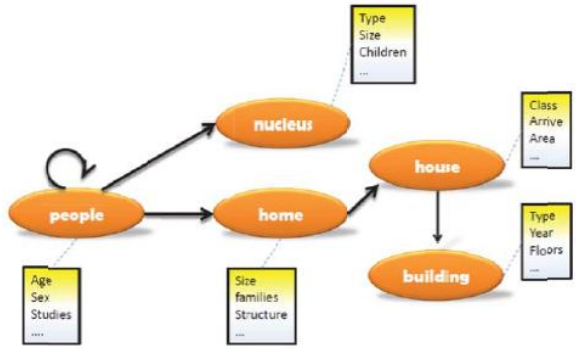

Figura 3.4: Identidades presentes en el censo

El segundo paso es transformar la información identificada al formato RDF de tal modo que se ha asignado una URI a cada persona, hogar, vivienda y edificios únicos y núcleos.

- `http://census.lab216.com/censo/2001/personas/<id>`, para personas, donde <id> está formado por 6 caracteres que identifican el número de hogar y 4 que identifican a la persona dentro del hogar (NORDF).
- `http://census.lab216.com/censo/2001/hogares/<id>`, para hogares, donde <id> está formado por 6 caracteres que identifican el número de hogar.
- `http://census.lab216.com/censo/2001/viviendas/<id>`, para viviendas, donde <id> está formado por 6 caracteres que identifican el número de vivienda.
- `http://census.lab216.com/censo/2001/edificios/<id>`, para edificios, donde <id> está formado por 6 caracteres numerados secuencialmente que identifican al edificio único.
- `http://census.lab216.com/censo/2001/nucleos/<id>`, para núcleos, donde <id> está formado por 6 caracteres que identifican al hogar y 2 caracteres que identifican el número dentro del hogar.

Además se ha asignado un vocabulario que se ajusta a los campos (Anexo C) definidos en la especificación del censo 2001 del INE, dividiendo estos en varios dominios de aplicación:

- `http://census.lab216.com/censo/2001/persona#<CAMPO>`, para predicados de las personas, donde <CAMPO> es una variable de los datos individuales.
- `http://census.lab216.com/censo/2001/hogar#<CAMPO>`, para predicados de los hogares, donde <CAMPO> es una variable de los datos del hogar.

- `http://census.lab216.com/censo/2001/vivienda#<CAMPO>` para predicados de las viviendas, donde <CAMPO> es una variable de los datos de la vivienda.
- `http://census.lab216.com/censo/2001/edificio#<CAMPO>` para predicados de los edificios, donde <CAMPO> es una variable de los datos del edificio.
- `http://census.lab216.com/censo/2001/nucleo#<CAMPO>`, para predicados de los núcleos, donde <CAMPO> es una variable de los datos del núcleo.

Y por último, se han añadido los siguientes predicados para enlazar distintos sujetos:

- `http://census.lab216.com/censo/2001/persona#HOGAR`, enlaza personas y hogares.
- `http://census.lab216.com/censo/2001/persona#NUCF`, enlaza persona y NUCLEO.
- `http://census.lab216.com/censo/2001/hogar#VIVIENDA`, enlaza hogar y vivienda.
- `http://census.lab216.com/censo/2001/vivienda#EDIFICIO`, enlaza vivienda y edificio.

Además, los predicados de la persona PAD, MAD, CON, HOM, MUJ Y PP, se redefinen para asociar directamente la URI del padre, madre, cónyuge, hombre del núcleo, mujer del núcleo y persona de referencia, ahorrando la repetición de datos y la presencia de las variables originales en estos apartados.

De esta forma se convirtió el censo original (5% de la muestra) en texto plano al formato RDF generando las siguientes estadísticas:

- 2.039.274 personas (5% de la población)
- 118.255.550 triples
- 147 predicados
- Tamaño Censo en N3: 14.991 MB

Figura 3.5: Formato N3 del censo 2001 en RDF

3.3. Conclusiones

El principal problema es que la mayoría de los ingentes datos gubernamentales se ocultan tras un difícil acceso, se encuentran en múltiples formatos y sistemas pensados para uso interno, haciéndolos prácticamente inservibles para terceros.

Transformar estos datos a un formato de datos abiertos (Open Data) como RDF genera valor, transparencia e interoperabilidad. Entre los múltiples beneficios que el Open Data ofrece, cabe destacar:

- Generación de valor: permite la obtención de productos y servicio derivados, de beneficio para empresas, intermediarios y ciudadanía en general.
- Transparencia: dota de un carácter transparente y de verdad a la administración pública, permitiendo la reutilización y análisis de los datos.
- Interoperabilidad: facilita la interoperabilidad de datos, productos y servicios, tanto dentro de la administración, como para el público en general.
- Ordenación Interna: promueve un trabajo organizado de la administración y útil para el futuro.

Por lo tanto el principal objetivo del proyecto census 2001 RDF es demostrar cómo la gran cantidad de datos generados por las administraciones, que son prácticamente inservibles para terceros, pueden ser convertidos en formatos abiertos, que generar valor no sólo a terceros, sino a la propia administración en tanto que son interoperables,. Dando así un paso muy importante en el fomento del desarrollo de la filosofía de un gobierno abierto.

3.4. El Trabajo Fin de Grado

Como ya se ha mencionado, este trabajo fin de grado pretende realizar una primera aproximación hacia una interfaz gráfica de consulta SPARQL, centrándolo en el caso del Censo 2001 RDF, ya que para este caso el número de entidades en el sistema está bien acotado (Figura 3.4).

La interfaz de consulta SPARQL deberá seguir el vocabulario establecido en el proyecto "Census 2001 RDF" para elaborar los patrones de búsqueda. Una vez que el usuario haya "dibujado" el subgrafo que desea buscar, el sistema debe de convertirlo internamente a SPARQL, realizar la consulta y devolver el resultado.

Se dispone de un prototipo no funcional [23] desarrollado en el proyecto "Census 2001 RDF" de la interfaz gráfica, el cual será utilizado como punto de partida de la interfaz gráfica de este trabajo fin de grado.

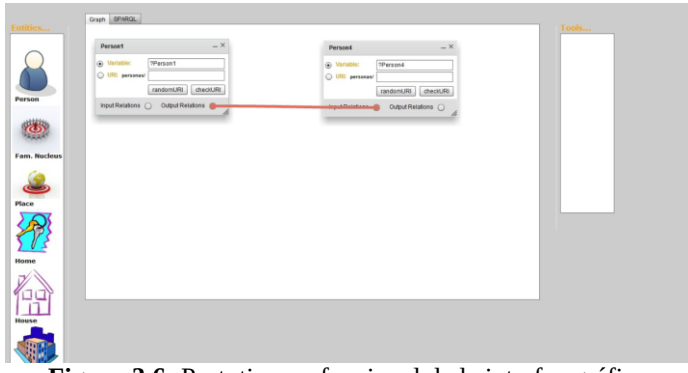

Figura 3.6: Prototipo no funcional de la interfaz gráfica

Capítulo 4
DESARROLLO DEL PROYECTO

4. DESARROLLO DEL PROYECTO

PLANIFICACIÓN Y GESTIÓN:

El objetivo del de este apartado es recopilar toda la información necesaria para llevar a cabo la gestión, control y seguimiento del proyecto, desarrollando una planificación temporal e investigando los posibles riesgos pueden aparecer en el desarrollo del sistema.

4.1. Gestión de Riesgos

En este apartado se describen los principales riesgos [8] que se han encontrado así como una descripción detallada y el estudio de su probabilidad, impacto y en qué parte del proceso pueden darse.

Los riesgos que aparecen en el siguiente listado atienden a la siguiente clasificación de alto nivel:

- Proyecto: Restricciones de recursos, interfaces externas, relaciones con los proveedores, políticas internas, problemas de coordinación interna del equipo o del grupo, financiación no adecuada.
- Proceso: Proceso software no documentado, falta de revisiones efectivas de colegas, no prevención de defectos, proceso de diseño pobre, gestión de requisitos, planificación ineficaz.
- Producto: Falta de experiencia en el dominio, diseño complejo, interfaces definidas deficientemente, sistemas de legado poco comprendidos, requisitos vagos o incompletos.

De la misma manera el impacto de dichos riesgo atiende a:

- Muy grave: si se produjera el riesgo, se fracasaría en el objetivo.
- Grave: supone una degradación del rendimiento del proceso y del sistema final.
- Leve: consiste en un fracaso de menor grado en un objetivo no primordial.
- Despreciable: se trata de un inconveniente menor.

4.1.1. Lista de riesgos

Nombre del riesgo	**Retraso en la planificación**
Enunciado	A causa a una mala planificación pueden producirse demoras.
Contexto	Este riesgo puedo darse a lo largo de todo el proceso debido a una planificación errónea.
Categoría del riesgo	De proceso, de gestión.
Probabilidad	60%
Impacto	Grave
Fase/actividad	Inicio / Elaboración / Construcción / Transición
Consecuencias	Retraso en la entrega de los distintos artefactos así como del producto final.

Figura 4.1: Riesgo: Retraso en la planificación

Nombre del riesgo	Falta de experiencia
Enunciado	Debido a la falta de experiencia en el manejo de Adobe Flash Builder es probable que se produzcan fallos a lo largo del desarrollo y que se produzca un avance más lento del esperado.
Contexto	Este riesgo puede darse a lo largo de todo el proceso.
Categoría del riesgo	De producto.
Probabilidad	50%
Impacto	Leve
Fase/actividad	A lo largo de todas las fases.
Consecuencias	Ralentización en el avance de las diferentes fases.

Figura 4.2: Riesgo: Falta de experiencia

Nombre del riesgo	Modificación de los requisitos
Enunciado	Debido a cambios y/o falta de entendimiento con los tutores puede ser necesario modificar el documento de requisitos.
Contexto	Este riesgo puede darse a lo largo de todo el proceso pero es mucho más probable en las fases iniciales. Debe tenerse en cuenta que en las fases finales esta modificación tendrá un mayor impacto.
Categoría del riesgo	De proceso.
Probabilidad	10%
Impacto	Leve en las fases iniciales, crítico en las fases finales.
Fase/actividad	A lo largo de todas las fases.
Consecuencias	Las consecuencias dependerán de en qué fase del proyecto se produzca el riesgo y de la importancia del requisito.

Figura 4.3: Riesgo: Modificación de los requisitos

Nombre del riesgo	Proceso de diseño pobre
Enunciado	Debido a la falta de experiencia el proceso de diseño puede no ser del todo correcto.
Contexto	A lo largo del proceso de diseño.
Categoría del riesgo	De proceso.
Probabilidad	60%
Impacto	Muy Grave
Fase/actividad	Elaboración – Diseño
Consecuencias	Un mal diseño o un diseño pobre podrían llevar a graves problemas en la fase de implementación e incluso a una repetición del proceso de diseño.

Figura 4.4: Riesgo: Proceso de diseño pobre

Nombre del riesgo	Falta de revisiones efectivas
Enunciado	Tras cada una de las iteraciones del proceso unificado deben hacerse revisiones, existe el riesgo de que no se hagan o no sean efectivas.
Contexto	A lo largo de todo el proceso unificado.
Categoría del riesgo	De proceso.
Probabilidad	30%
Impacto	Despreciable
Fase/actividad	A lo largo del proceso unificado.
Consecuencias	Si las revisiones no son las correctas el error se arrastraría a través de las diferentes iteraciones.

Figura 4.5: Riesgo: Falta de revisiones Efectivas

Nombre del riesgo	Escasez puntual de tiempo
Enunciado	Debido a momentos puntuales de carga excesiva de trabajo es posible que no pueda dedicar el tiempo estipulado a cualquiera de las partes.
Contexto	Puede aparecer en cualquier etapa del desarrollo.
Categoría del riesgo	De proyecto.
Probabilidad	80% en la fase de Inicio, 40% en el resto de fases
Impacto	Muy grave en fases avanzadas
Fase/actividad	Cualquier fase
Consecuencias	Puede provocar una demora en los tiempos previstos.

Figura 4.6: Riesgo: Escasez Puntual de tiempo

4.1.2. Planes de acción y monitorización

A continuación se exponen los diferentes planes de acción para cada uno de los riesgos identificados en el apartado anterior. Se proporciona el escenario en el que podría darse el riesgo, el proceso de monitorización del mismo junto a los puntos de comprobación oportunos y el tipo de estrategia y los pasos a seguir en caso de que el riesgo se desencadene.

Nombre del riesgo	Retraso en la planificación
Escenario	En el caso de que se produzca el riesgo, debe modificarse la planificación tratando de causar el menor perjuicio posible.
Punto de comprobación	Debe comprobarse a lo largo de todo el proceso de desarrollo.
Estrategia	Reservar el riesgo.
Plan de acción	Revisión periódica del calendario establecido y disponibilidad de tiempo.
Monitorización	Modificación de fechas.

Figura 4.7: Plan de Acción: Retraso en la planificación

Nombre del riesgo	Falta de experiencia
Escenario	En el caso de que se produzca el riesgo se tratara de mitigar la inexperiencia.
Punto de comprobación	Debe comprobarse al inicio de cada fase ya que la falta de experiencia puede darse en diferentes aspectos del desarrollo.
Estrategia	Reservar el riesgo.
Plan de acción	Tratar de formarse y usar la documentación disponible y herramientas de apoyo para el aprendizaje de las nuevas tecnologías.
Monitorización	Revisión de conocimientos adquiridos.

Figura 4.8: Plan de Acción: Falta de experiencia

Nombre del riesgo	Modificación de los requisitos
Escenario	En el caso de que el riesgo se diera las consecuencias variarán. Dependerá de en qué fase del proyecto se produzca y de la importancia del requisito. Deben entonces variar los puntos del proyecto que tienen relación con dichos requisitos.
Punto de comprobación	Debe comprobarse a lo largo de todo el desarrollo del proceso, especialmente en las fases iniciales.
Estrategia	Investigar el riesgo.
Plan de acción	Revisión de la especificación de los requisitos y el modelo de los casos de uso.
Monitorización	Entrevistas o encuentros con los tutores.

Figura 4.9: Plan de Acción: Modificación de los requisitos

Nombre del riesgo	Proceso de diseño pobre
Escenario	En caso de producirse un mal diseño o un diseño pobre podría ser necesario volver a la fase de diseño.
Punto de comprobación	Será necesario realizar comprobaciones durante toda la etapa de diseño y en la finalización del mismo.
Estrategia	Reducción del riesgo.
Plan de acción	Realizar una revisión exhaustiva en cada uno de los hitos de la etapa de diseño, para asegurar que se está conforme con lo realizado hasta el momento. También será necesario disponer de unos documentos de análisis correctos.
Monitorización	Seguimiento de los entregables en cada uno de los hitos de la etapa de diseño.

Figura 4.10: Plan de Acción: Proceso de diseño pobre

Nombre del riesgo	Falta de revisiones efectivas
Escenario	En caso de darse dicho riesgo, se deberá realizar una nueva revisión exhaustiva.
Punto de comprobación	A lo largo de todo el proceso de desarrollo, antes de cambiar de iteración.
Estrategia	Protección del riesgo.
Plan de acción	Realizar una nueva revisión exhaustiva con todo aquello que no se hubiera tenido en cuenta en revisiones anteriores.
Monitorización	Antes de cambiar de iteración debe comprobarse que se han hecho las revisiones convenientemente.

Figura 4.11: Plan de Acción: Falta de revisiones efectivas

Nombre del riesgo	Escasez puntual de tiempo
Escenario	En caso de darse dicho riesgo se debería reanudar el trabajo lo más pronto posible. Asignando más horas en caso de que fuese necesario.
Punto de comprobación	A lo largo de todo el proyecto.
Estrategia	Reservar el riesgo.
Plan de acción	Estimar el tiempo semanal que se dispone y tener cierta holgura en él para minimizar los contratiempos. Es posible dedicar más tiempo del estipulado en caso que se prevea una escasez.
Monitorización	Seguimiento de los retrasos acumulados en caso de haberlos.

Figura 4.12: Plan de Acción: Escasez puntual de tiempo

4.2. Planificación del Proyecto

La planificación [8] siempre es algo que puede ser flexible dependiendo de las circunstancias particulares. Además, es una fuente de imprecisiones, así que debe ser tomada con precaución. No obstante, a continuación se expone el plan de desarrollo inicial de este proyecto.

4.2.1. Estimaciones de proyecto

Al no contar con demasiada experiencia en proyectos similares, tomaré como base para la calendarización del proyecto las diferentes prácticas y PFCs realizados.

En cuanto a costes económicos, hemos de tener en cuenta que el desarrollador del producto es estudiante y que el proyecto se enmarca dentro del Trabajo Fin de Grado, por lo que pueden considerarse nulos. Lo mismo es aplicable a los costes indirectos de software y hardware, ya que a pesar de que Adobe Flash Builder [10] es una herramienta de pago (cada licencia Standard vale unos 250€) se ha conseguido una licencia gratuita de estudiante.

El proyecto comenzará después de la entrega de la práctica de la última asignatura pendiente, aproximadamente el 14/05/2012; y deberá concluir con anterioridad a la convocatoria que tendrá lugar en septiembre de 2012.

4.2.2. Planificación del tiempo

Se ha optado por seguir la metodología RUP [2], realizando las distintas fases de Inicio, Elaboración, Construcción y Transición.

Fase	Iteración	Duración	Hitos
Inicio	1	15 días (lun 14/05/12)	Plan de desarrollo, documento de riesgos y documento de requisitos.
Elaboración	1	25 días (lun 04/06/12)	Modelos de análisis, diseño y diseño de la interfaz.
Construcción	1	20 días (lun 09/07/12)	Versión alfa del sistema y realización de pruebas.
	2	10 días (lun 06/08/12)	Entrega de la versión beta del sistema y de la versión inicial del manual de usuario.
Transición	1	10 días (lun 20/01/12)	Entrega de la versión final del sistema junto con la documentación.

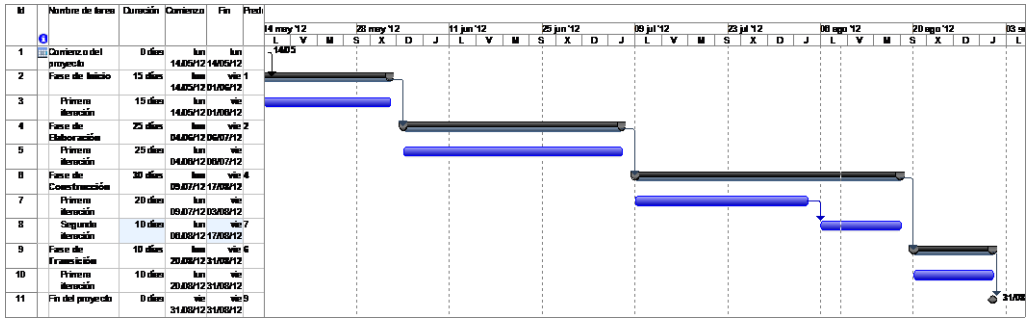

Figura 4.13: Plan temporal de Fases

4.2.2.1.Planificación de actividades:

Dentro de cada fase se realizarán las siguientes actividades:

- Fase de Inicio:
    1. Curso Introducción a la Web de datos: 10 horas
    2. Trabajo Relacionado y Estudios Previos: 50 horas
        a. Afianzamiento de los conceptos del curso; afianzar SPARQL en la medida de lo posible: 10 horas

   b. Puesta al día del trabajo relacionado en interfaces SPARQL: 10 horas
   c. Estudio de Requisitos y plan de trabajo: 10 horas
   d. Puesta al día con Flash: 20 horas
3. Redacción de la documentación de esta fase (Planificación, riesgos, requisitos): 10 horas.

- Fase de Elaboración:
  1. Desarrollo y documentación del modelo de análisis, diseño y despliegue: 40 horas

- Fase de Implementación:
  1. Interfaz visual inicial: 20 horas
  2. Modelado de entidades individuales: 20 horas
  3. Predicados y enlaces: 20 horas
  4. Sugerencias y recomendaciones de URIs: 20 horas
  5. Selección, Unión y OPTIONAL: 20 horas
  6. Transformación lógica a SPARQL: 20 horas
  7. Intermediación con Virtuoso : 20 horas
  8. Mostrar resultados: 20 horas
  9. Pruebas (Funcionalidad, eficiencia y usabilidad) y correcciones: 20 horas
  10. Documentación asociada: 10 horas

- Fase de Transición:
  1. Corrección de errores y versión final del producto: 5 horas
  2. Manuales de Usuario e Instalación y finalización de la memoria: 10 horas

4.3. Seguimiento de la planificación

A continuación se describe en qué situación se encontraba el proceso de desarrollo, así como los artefactos generados, destacando el grado de cumplimiento de los objetivos de las fases anteriores e indicando los posibles problemas que hayan surgido y cómo van a ser tratados, lo largo de las distintas fases.

- Fase de Inicio

La fase de inicio ha tenido una duración de 30 días, es decir, 10 días más de lo inicialmente planificado. Durante esta iteración han tenido lugar los siguientes riesgos:

- Aunque el curso de introducción a la web de datos se realizó a principios de marzo, debido a una falta puntual de tiempo, producida por la realización de la práctica de una asignatura, no se pudo comenzar el proyecto hasta finales de mayo.
- Debido a la falta de tiempo, también se alargó el aprendizaje de las tecnologías a utilizar provocando el retraso final de 10 días.

Este retraso se intentará paliar en fases posteriores dedicando más horas al proyecto. A pesar del retraso todos los artefactos pertenecientes a esta fase fueron creados satisfactoriamente.

- Fase de Elaboración

La fase de elaboración ha tenido una duración de 22 días. Esta duración es 3 días más corta de lo estimado, pero no se pudo reducir el tiempo total de retraso (10 días) debido a los siguientes riesgos:

- Se produjo una cierta relajación, debido a las vacaciones de verano.
- Al inicio de la iteración, se produjo un análisis y diseño pobres, que se intentaron corregir mediante sucesivas consultas a los tutores para que no entorpeciese el posterior desarrollo.
- Finalmente, se viene arrastrando un retraso en la planificación de la fase de inicio que no se ha podido recuperar.

A pesar de que se lleve un retraso de 7 días, los documentos iniciales de análisis, de diseño y de datos se han podido redactar completamente. También se han validado dichos artefactos por los tutores.

Debido a la experiencia que se poseía en el lenguaje de programación ActionScript, unida al aprendizaje obtenido durante estas fases en la herramienta de programación (Adobe Flash Builder), se intentará mitigar el retraso acumulado en la siguiente fase (Construcción).

- Fase de Construcción

La fase de construcción ha tenido una duración de 23 días, es decir, 7 días menos de lo inicialmente planificado, consiguiendo mitigar el retraso acumulado de las etapas anteriores. Este adelanto se ha conseguido debido a la experiencia y aprendizaje del entorno de programación y a la dedicación de más horas semanales al proyecto.

Por último, cabe destacar que en esta fase se ha comprobado que todos los casos de prueba se cumplen correctamente, con lo cual se puede considerar acabada la implementación de la interfaz. Se han validado todos los artefactos obtenidos hasta la fecha y se ha iniciado la escritura de los manuales de instalación y usuario.

- Fase de Transición

La fase de transición fue planificada con una duración de 5 días. Esta planificación fue correcta, se evitaron los riesgos y se pudieron cumplir en tiempo los objetivos expuestos en el Plan de Desarrollo de Software. De entre los objetivos conseguidos destaca la entrega final del Trabajo Fin de Grado.

En definitiva, el grado de cumplimiento de los objetivos de este proyecto es satisfactorio.

ANALISIS:

El análisis [1] consiste en la comprensión y modelado de la aplicación y del dominio en el cual funciona. La entrada inicial de la fase de análisis es una descripción del problema que hay que resolver y proporciona una visión general conceptual del sistema propuesto.

El objetivo de este apartado es reconocer los elementos básicos del programa tal como lo percibe el usuario. La salida del análisis es un modelo formal, que muestra de una forma abstracta y resumida, los diferentes elementos del dominio, y que sirve como base para la fase de diseño del sistema.

Nos vamos a centrar en los objetivos principales de la aplicación y de los límites de la misma. Trataremos de identificar las necesidades reales del usuario final de la aplicación mediante los requisitos. La identificación de los requisitos es una parte fundamental debido a que si no son correctos puede conllevar a problemas en la propia utilización de la aplicación.

4.4. Entrevistas

La forma de establecer los requisitos que debía cumplir el sistema ha sido mediante la observación de sistemas similares anteriores y las entrevistas con los propios tutores del proyecto, quienes propusieron unos requisitos iniciales. A partir de estas entrevistas, se identificaron los requisitos finales y se estableció una prioridad para los mismos, de acuerdo a las necesidades de los usuarios finales y a las funcionalidades y restricciones del sistema que se va a construir.

4.5. Especificación de Requisitos Software (SRS)

El propósito de este apartado, es desarrollar la especificación del comportamiento del sistema. El análisis presentado en este documento pretende cubrir las necesidades descritas por los tutores. Contiene los requisitos funcionales y no funcionales.

La especificación de los requisitos debe ser completa, no tener definiciones contradictorias, y debe ser expresada con exactitud, ya que si hay ambigüedad se corre el riesgo de que sean interpretados de forma diferente por los usuarios y por los desarrolladores.

4.5.1. Requisitos funcionales

Estos requisitos [7] constituyen la declaración de los servicios que el sistema debe proporcionar, cómo debe reaccionar ante una entrada particular y cómo se debe comportar ante situaciones particulares. En resumen, describen la funcionalidad del sistema y de qué forma va a utilizarlo el usuario.

En los siguientes subapartados detallo los requisitos funcionales encontrados para el sistema. En los cuales trato de expresar con exactitud los servicios que el sistema debe proporcionar y cómo se debe comportar ante situaciones particulares.

FRQ-001	**Ayuda**
Descripción	El sistema deberá proveer al usuario de una ayuda donde se explique como usar la aplicación y que además contendrá un tutorial con consultas predefinidas.
Importancia	Importante

Figura 4.14: FRQ Ayuda

FRQ-002	**Entidades**
Descripción	El sistema permitirá trabajar con entidades Persona, Núcleo Familiar, Hogar, Vivienda, Edificio y Lugar (una ciudad, un país,…)
Importancia	Vital

Figura 4.15: FRQ Entidades

FRQ-003	**Entidades Variable o URI**
Descripción	El sistema permitirá dar un nombre de variable a una entidad añadida o fijar su URI.
Importancia	Vital

Figura 4.16: FRQ Entidades Variable o URI

FRQ-004	**Comprobar URI**
Descripción	El sistema permitirá comprobar que una URI insertada es válida.
Importancia	Opcional

Figura 4.17: FRQ Comprobar URI

FRQ-005	**URI Aleatoria**
Descripción	El sistema podrá rellenar una entidad con una URI válida aleatoria, para facilitar el aprendizaje.
Importancia	Opcional

Figura 4.18: FRQ URI Aleatoria

FRQ-006	**Literales**
Descripción	El sistema permitirá añadir Objetos Literales, a los que también se podrá dar el valor variable o bien se rellenará con el valor deseado (por ejemplo, Número de Hijos: 3)
Importancia	Vital

Figura 4.19: FRQ Literales

FRQ-007	Relaciones
Descripción	El sistema permitirá enlazar entidades y Objetos Literales mediante predicados (relaciones).
Importancia	Vital

Figura 4.20: FRQ Relaciones

FRQ-008	Relaciones variable o URI
Descripción	Los predicados podrán establecerse como variables o con una URI determinada.
Importancia	Importante

Figura 4.21: FRQ Relaciones variable o URI

FRQ-009	Sugerencias en campos de relaciones
Descripción	Existirá un sistema de sugerencia para las URIs de los predicados.
Importancia	Importante

Figura 4.22: FRQ Sugerencias en campos de relaciones

FRQ-010	Seleccionar resultado
Descripción	El sistema permitirá seleccionar qué se desea devolver en la consulta, es decir, cuáles de las variables se retornan como resultado.
Importancia	Vital

Figura 4.23: FRQ Seleccionar resultado

FRQ-010	UNION y GROUP BY
Descripción	El sistema facilitará las consultas UNION, y los modificadores GROUP BY.
Importancia	Opcional

Figura 4.24: FRQ UNION y GROUP BY

FRQ-010	Count
Descripción	Aunque no está dentro de la primera especificación SPARQL, el sistema incluirá la opción de agregación COUNT, tal y como se hace en el sistema VIRTUOSO
Importancia	Opcional

Figura 4.25: FRQ Count

FRQ-010	Consulta SPARQL
Descripción	El sistema manejará internamente la representación SPARQL de la consulta gráfica, pudiendo verla y modificarla.
Importancia	Vital

Figura 4.26: FRQ Consulta SPARQL

FRQ-010	Resultado Consulta
Descripción	El sistema actuará de intermediario con el sistema que representa la información censal en RDF para ejecutar y devolver el resultado de las consultas.
Importancia	Vital

Figura 4.27: FRQ Resultado Consulta

4.5.2. Requisitos no funcionales

Los requisitos no funcionales definen propiedades y restricciones del sistema. Especifican las propiedades del sistema que tienen que ver con el rendimiento, velocidad, uso de memoria, fiabilidad, etc. Imponen condiciones a los requisitos funcionales.

NFR-001	Lenguajes de programación
Descripción	El sistema deberá será programado en el entorno de desarrollo de Adobe Flash Builder y el lenguaje ActionScript 3.0.
Importancia	Vital

Figura 4.28: NFR Lenguajes de programación

NFR-002	Facilidad de uso
Descripción	El sistema deberá facilitar la ergonomía en el uso de la aplicación, con el objeto que este pueda ser utilizado por personal no especializado.
Importancia	Importante

Figura 4.29: NFR Facilidad de uso

NFR -003	Estándar navegadores
Descripción	El sistema deberá ser accesible a través de los principales navegadores Web: Internet Explorer, Fire Fox..
Importancia	Importante

Figura 4.30: NFR Estándar navegadores

NFR -004	Servidor
Descripción	El sistema deberá funcionar en alguno de los servidores de la Escuela de Informática.
Importancia	Importante

Figura 4.31: NFR Servidor

4.5.3. Limitaciones

LIM-001	Tipo Consultas
Descripción	Se acotan las consultas únicamente a SELECT (No CONSTRUCT ni ASK).

Figura 4.32: Limitación Consultas

LIM-002	Modificar SPARQL
Descripción	Aunque se puede ver la consulta SPARQL generada, y se puede modificar, una modificación manual en SPARQL no se reflejará en el grafo.

Figura 4.33: Limitación Modificar SPARQL

4.6. Modelo de Casos de Uso (SRS)

El objetivo de este subapartado es detallar el Modelo [6] de casos de uso de la aplicación. Utilizamos los casos de uso para describir el uso del sistema y cómo los usuarios interaccionan con nuestra aplicación. En los casos de uso se detallan paso a paso las acciones que realiza el sistema y que producen un resultado para un actor en particular.

Contiene el diagrama de casos de uso, donde se representan los casos de uso existentes, las relaciones entre los mismos y con los actores del sistema y una descripción de cada uno de los casos de uso.

4.6.1. Definición de actores

En este subapartado se enumeran los actores del sistema y se da una breve explicación de los mismos. Los actores son todas aquellas entidades externas que puedan interactuar con la aplicación.

- Usuario: Este actor representa al usuario del sistema, quien se encargará de ejecutar la aplicación y es quien utiliza el sistema.

4.6.2. Diagrama de casos de uso

El objetivo de este subapartado es detallar el Modelo de casos de uso del juego. Contiene el diagrama de casos de uso, donde se representan los casos de uso existentes, las relaciones entre los mismos y con los actores del sistema y una descripción de cada uno de los casos de uso.

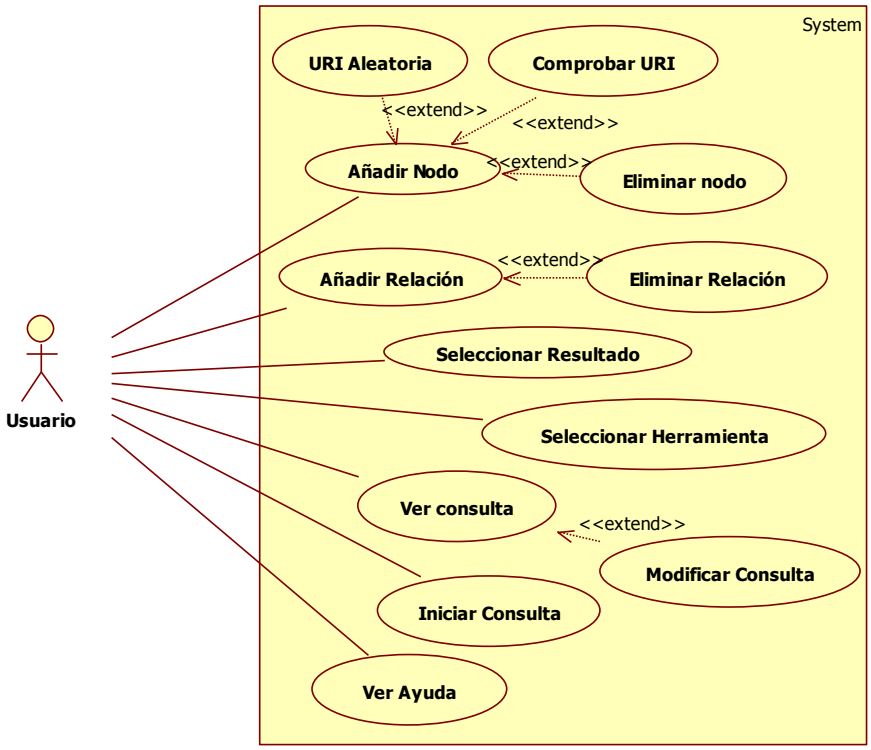

Figura 4.34: Diagrama de Casos de Uso

4.6.2.1. Detalle de los casos de uso

Casos de Uso de Usuario:

- **Ver ayuda [UC-0001]:** El Usuario quiere consultar la ayuda disponible para interactuar con la aplicación.

- **Añadir Nodo [UC-0002]:** El Usuario quiere añadir un nodo entidad (persona, hogar, vivienda,…) para el diseño del grafo de consulta.

- **URI Aleatoria [UC-0003]:** El Usuario quiere añadir una URI aleatoria correcta al nodo.

- **Comprobar URI [UC-0004]:** El Usuario quiere comprobar si una URI introducida es correcta.

- **Eliminar Nodo [UC-0005]:** El Usuario desea eliminar uno de los nodos añadidos anteriormente.

- **Añadir Relación [UC-0006]:** El Usuario añadir una relación entre dos nodos de la pantalla.

- **Eliminar Relación [UC-0007]:** El Usuario desea eliminar una de las relaciones añadidas anteriormente.

- **Seleccionar Resultado [UC-0008]:** El Usuario desea seleccionar o deseleccionar como resultado de la consulta uno de los elementos (nodo o relación) añadidos anteriormente.

- **Seleccionar Herramienta [UC-0009]:** El Usuario desea cambiar la herramienta utilizada actual (Añadir Nodos, Establecer Relaciones, Seleccionar Resultado).

- **Ver Consulta [UC-0010]:** El Usuario desea ver la consulta SPARQL asociada al grafo creado.

- **Modificar Consulta [UC-0011]:** El Usuario desea modificar la consulta SPARQL asociada al grafo creado.

- **Iniciar Consulta [UC-0012]:** El Usuario desea lanzar la consulta SPARQL y obtener los resultados.

4.6.2.2. Descripción de los casos de uso

UC-0001	**Ver ayuda**
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el Usuario desee obtener ayuda acerca del sistema.
Secuencia Normal	1. El Caso de Uso comienza cuando el usuario solicita "Ver ayuda" 2. El caso de uso finaliza cuando el sistema muestra la ayuda.
Poscondición	Se muestra la ayuda

Figura 4.35: UC Ver Ayuda

UC-0002	**Añadir Nodo**
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el Usuario desee añadir un nodo entidad a la pantalla.
Precondición	Está seleccionada la herramienta "Añadir Nodos".
Secuencia Normal	1. El Caso de Uso comienza cuando el usuario selecciona "Añadir Nodo Entidad" 2. El sistema crea un nuevo nodo y lo añade a la pantalla. 3. El usurario puede modificar la variable o completar una URI. 4. El caso de uso finaliza.
Poscondición	El sistema ha creado un nuevo nodo en la pantalla.

Flujos Alternativos	En el paso 4, si el usuario desea: - Añadir una URI aleatoria: Punto de extensión: "URI Aleatoria [UC-0003]" - Comprobar valided de la URI introducida: Punto de extensión: "Comprobar URI [UC-0004]" - Eliminar el nodo: Punto de extensión: "Eliminar Nodo [UC-0005]"

Figura 4.36: UC Añadir Nodo

UC-0003	**URI Aleatoria**
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el Usuario desee establecer una URI aleatoria correcta en un nodo.
Precondición	Existe al menos un nodo en la pantalla.
Secuencia Normal	1. El Caso de Uso comienza cuando el usuario selecciona "URI aleatoria" en un nodo. 2. El caso de uso finaliza cuando el sistema genera una URI aleatoria y la muestra.
Poscondición	Se asigna una URI aleatoria al nodo.

Figura 4.37: UC URI Aleatoria

UC-0004	**Comprobar URI**
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el Usuario desee comprobar la validad de una URI introducida.
Precondición	Existe al menos un nodo en la pantalla y se ha introducido una URI para éste.
Secuencia Normal	1. El Caso de Uso comienza cuando el usuario selecciona "Comprobar URI" en un nodo. 2. El caso de uso finaliza cuando el sistema muestra si la URI es valida o no.
Poscondición	Se sabe si la URI introducida es valida o no.

Figura 4.38: UC Comprobar URI

UC-0005	**Eliminar Nodo**
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el Usuario desee eliminar un nodo de la pantalla.
Precondición	Existe al menos un nodo en la pantalla.
Secuencia Normal	1. El Caso de Uso comienza cuando el usuario selecciona "Eliminar Nodo" en un nodo. 2. El caso de uso finaliza cuando el sistema elimina el nodo de la pantalla.
Poscondición	El nodo ha sido eliminado de la pantalla.

Figura 4.39: UC Eliminar Nodo

UC-0006	Añadir Relación
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el Usuario desee añadir una relación entre dos nodos.
Precondición	Está seleccionada la herramienta "Establecer Relaciones".
Secuencia Normal	1. El Caso de Uso comienza cuando el Usuario arrastra un nodo hacia otro. 2. El sistema comprueba entre que tipos de nodos se esta estableciendo la relación. 3. El sistema crea la relación entre nodos y la añade en pantalla. 4. El caso de uso finaliza.
Poscondición	Se crea una relación entre los nodos.
Flujos Alternativos	En el paso 2, si el sistema comprueba que no puede haber relación entre esos nodos el caso de uso finaliza. Después del paso 4, si el usuario desea eliminar la relación: - Punto de extensión: "Eliminar Relación [UC-0017]"

Figura 4.40: UC Añadir Relación

Nota: No todos los nodos (entidades) pueden establecer relaciones entre sí. Las posibles relaciones entre entidades son las siguientes:

- **Entre Persona y**: Persona u Hogar o Núcleo Familiar o Lugar o Literal.
- **Entre Hogar y**: Vivienda o Literal.
- **Entre Vivienda y**: Edificio o Literal.
- **Entre Edificio y**: Literal.
- **Entre Núcleo Familiar y**: Persona o Literal.
- **Los Lugares y Literales**: no podrán iniciar ninguna relación, serán siempre nodos fin en una relación.

UC-0007	Eliminar Relación
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el Usuario desee eliminar una relación existente entre nodos.
Precondición	Está seleccionada la herramienta "Añadir Nodos" y existe al menos una relación establecida entre nodos.
Secuencia Normal	1. El Caso de Uso comienza cuando selecciona eliminar relación. 2. El sistema elimina la relación entre los nodos. 3. El caso de uso finaliza.
Poscondición	La relación entre los nodos ha sido eliminada.

Figura 4.41: UC Eliminar Relación

UC-0008	**Seleccionar Resultado**
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el Usuario desee seleccionar un elemento para que sea parte del resultado de la consulta.
Precondición	Está seleccionada la herramienta "Seleccionar Resultado" y existen elementos añadidos a la pantalla.
Secuencia Normal	1. El Caso de Uso comienza cuando el usuario selecciona uno de los elementos de la pantalla. 2. El sistema selecciona el elemento y muestra las opciones de contar y agrupar en caso de que el elemento sea un nodo. 3. El Caso de Uso finaliza.
Poscondición	El elemento se encuentra en el estado "seleccionado".
Flujos Alternativos	En el paso 2, si el elemento ya estaba seleccionado: - El sistema cambia el estado "deseleccionado" del elemento y oculta las opciones de contar y agrupar en caso de que el elemento sea un nodo. En el paso 3, si el elemento seleccionado es un nodo: - El usuario podrá seleccionar las opciones de contar y agrupar. - El Caso de Uso finaliza.

Figura 4.42: UC Seleccionar Resultado

UC-0009	**Seleccionar Herramienta**
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el Usuario desee cambiar el estado de la aplicación
Secuencia Normal	1. El Caso de Uso comienza cuando el Usuario selecciona una de las herramientas del sistema. 2. El sistema cambia la herramienta seleccionada: - Añadir Nodos: El sistema permitirá añadir nuevos nodos y colocarlos por la pantalla, o eliminar cualquier elemento de la pantalla. - Establecer Relaciones: El sistema permitirá arrastrar los nodos para establecer relaciones. - Seleccionar Resultado: El sistema permitirá seleccionar los distintos elementos como resultado. 3. El Caso de Uso finaliza.
Poscondición	El sistema se encuentra en el estado seleccionado.

Figura 4.43: UC Seleccionar Herramienta

UC-0010	**Ver Consulta**
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el Usuario desee ver la consulta SPARQL asociada al grafo creado.
Precondición	Existe un grafo creado y es correcto.
Secuencia Normal	1. El Caso de Uso comienza cuando el Usuario selecciona "Ver SPARQL". 2. El sistema transforma el grafo diseñado a una consulta SPARQL. 3. El caso de uso finaliza cuando el sistema muestra dicha consulta.
Poscondición	La consulta asociada al grafo es mostrada.
Flujos Alternativos	En el paso 2, si el grafo no esta bien diseñado: - El Caso de Uso finaliza cuando el sistema muestra un mensaje de error al trasformar la consulta. Después del paso 3, si el usuario desea modificar la consulta: - Punto de extensión: "Modificar Consulta [UC-0011]"

Figura 4.44: UC Ver Consulta

UC-0011	**Modificar Consulta**
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el Usuario desea modificar la consulta SPARQL generada por el grafo.
Precondición	Se ha realizado el caso de uso "Ver Consulta [UC-0010]".
Secuencia Normal	1. El Caso de Uso comienza cuando el Usuario escribe en la consulta mostrada por el sistema. 2. El sistema modifica la consulta con lo introducido por el usuario y lo muestra 3. El caso de uso finaliza
Poscondición	Se muestra la consulta SPARQL modificada por el usuario.

Figura 4.45: UC Modificar Consulta

UC-0012	**Iniciar Consulta**
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el Usuario desee obtener los resultados de la consulta.
Precondición	Existe un grafo creado.
Secuencia Normal	1. El Caso de Uso comienza cuando el Usuario selecciona "Iniciar consulta". 2. El sistema obtiene la consulta del grafo creado. 3. El sistema envía la consulta al servidor. 4. El caso de uso finaliza cuando el sistema muestra los resultados obtenidos.
Postcondición	Los resultados de la consulta SPARQL asociada al grafo son mostrados en pantalla.
Flujos Alternativos	En el paso 2, si la consulta a sido modificada anteriormente por el usuario: - El sistema recoge la consulta modificada y el caso de uso continúa.

Figura 4.46: UC Iniciar Consulta

4.7. Modelo de Dominio

Este apartado ofrece una breve descripción de las clases involucradas en el modelo de dominio [6] obtenido de la interfaz gráfica de consulta.

Este apartado se limita a dar una descripción de la funcionalidad de sistema, de tal forma que se satisfagan todos y cada uno de los requisitos software desarrollados en los apartados anteriores. Se describen las clases que componen la aplicación y las relaciones entre ellas, así como una breve descripción de la responsabilidad de cada una de ellas.

4.7.1. Diagrama de dominio

En función de los requisitos de sistema y casos de uso presentamos el diagrama de clases inicial.

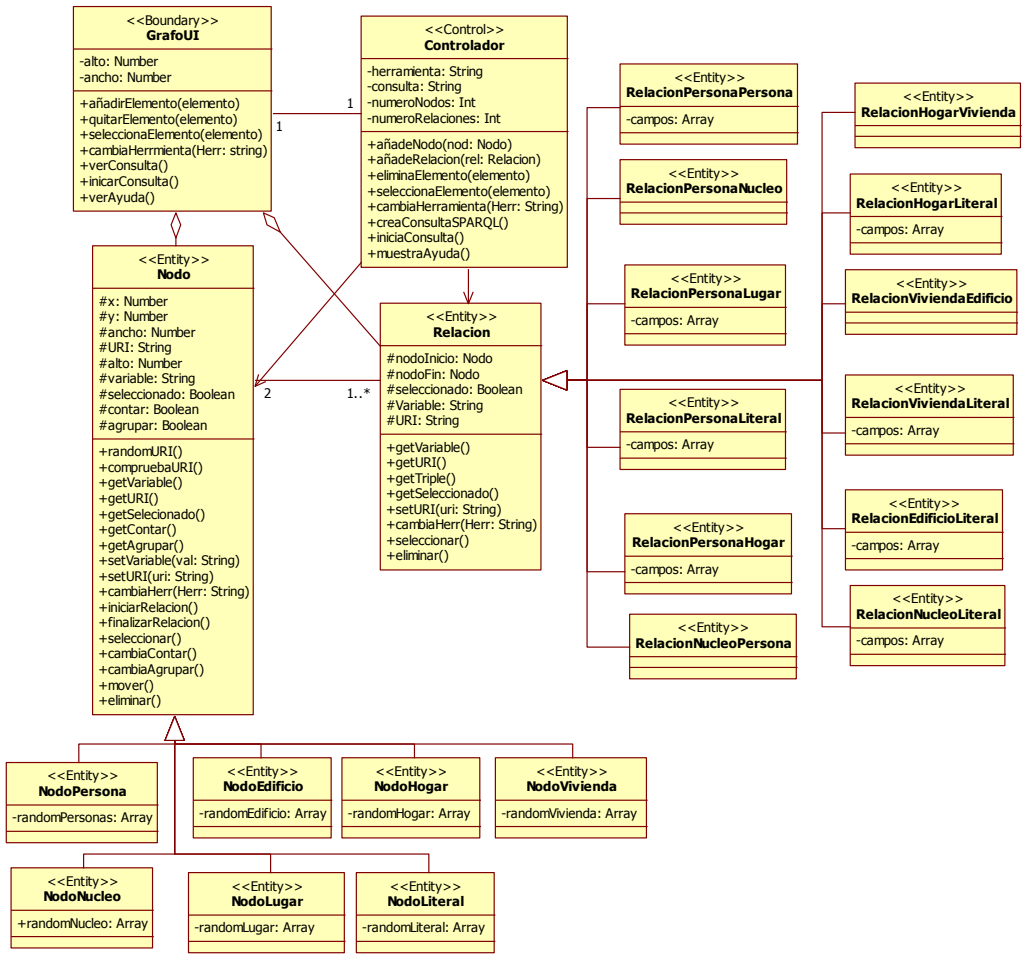

Figura 4.47: Modelo Inicial del Dominio

4.7.2. Explicación clases diagrama de dominio

Clase GrafoUI:
La clase GrafoUI es una clase Boundary por lo que representa la interfaz con la que se comunicará el usuario. Almacena su tamaño.

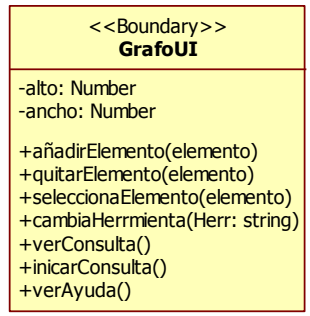

Figura 4.48: Clase GrafoUI

Atributos:
- Alto: Number
- Ancho: Number

Métodos:
+ añadirElemento(elemento:Sprite) → Añade un nodo o relación a la pantalla.
+ quitarElemento(elemento:Sprite) → Elimina un nodo o relación a la pantalla.
+ seleccionarElemento(elemento:Sprite) → Selecciona un nodo o relación como resultado de la consulta.
+ cambiaHerramienta() → Cambia la herramienta urilizada (Añadir nodos, establecer relaciones o seleccionar resultado).
+ verConsulta () → Muestra la consulta SPARQL asociada al grafo.
+ iniciarConsulta () → Inicia la consulta SPARQL creada.
+ verAyuda () → Muestra la ayuda de la aplicación.

Clase Controlador:
La clase Controlador es una clase Control por lo que será el intermediario entre la interfaz y las clases entidad.

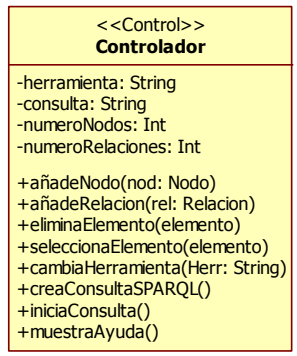

Figura 4.49: Clase Controlador

Atributos:

- herramienta:String → Herramienta utilizada (Añadir nodos, establecer relaciones o seleccionar resultado).
- consulta:String
- numeroNodos: Int
- numeroRelaciones: Int

Métodos:

+ añadeNodo(nod:Nodo) → Añade un nodo a la pantalla.
+ añadeRelacion(rel:Relacion) → Añade una relación entre nodos.
+ eliminaElemento(elemento:Sprite) → Elimina un nodo o relación a la pantalla.
+ seleccionaElemento(elemento:Sprite) → Selecciona un nodo o relación como resultado.
+ cambiaHerramienta() → Cambia la herramienta utilizada (Añadir nodos, establecer relaciones o seleccionar resultado) de los elementos añadidos a la pantalla.
+ creaConsultaSPARQL() → Crea la consulta SPARQL asociada al grafo.
+ iniciaConsulta () → Inicia la consulta y devuelve el resultado.
+ muestraAyuda() → Muestra la ayuda de la aplicación.

Clases Nodo y descendientes:

La clase Nodo y sus descendientes son clases entity, y representan a cada una de las entidades posibles (Persona, Hogar, Vivienda, Edificio, Núcleo, Lugar y Literal) que se pueden encontrar en el grafo.

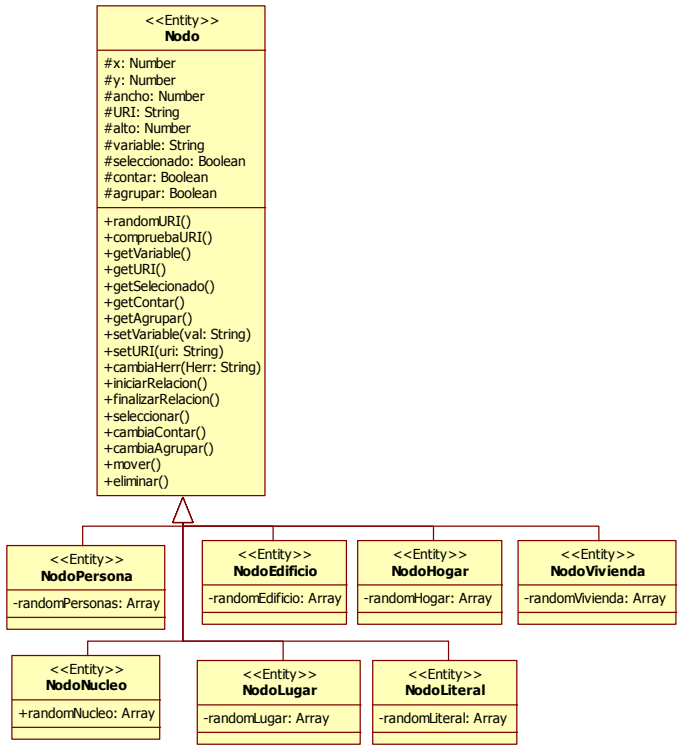

Figura 4.50: Clases Nodo y descendientes

Atributos clase Nodo:

x: Number
y: Number
ancho: Number
alto: Number
variable: String
URI: String
seleccionado: Boolean
contar: Boolean
agrupar: Boolean

Métodos clase Nodo:

+ randomURI() → Devuelve una URI correcta aleatoria.
+ comprobarURI() → Comprueba si la URI introducida es correcta.
+ getVariable() → Devuelve el valor de que tiene la variable del nodo.
+ getURI() → Devuelve el valor de que tiene la URI del nodo.
+ getSeleccionado() → Devuelve si el nodo está seleccionado o no.
+ getContar() → Devuelve si se debe obtener la cuenta (count) como resultado.
+ getAgrupar() → Devuelve si se debe agrupar el resultado por la variable del nodo.
+ setVariable() → Modifica el valor de que tiene la variable del nodo.
+ setURI() → Modifica el valor de que tiene la URI del nodo.
+ cambiaHerr() → Cambia el estado del nodo, según sea la herramienta utilizada de la aplicación (Añadir nodos, establecer relaciones o seleccionar resultado).
+ iniciaRelacion() → Establece el nodo como punto de inicio de una relación.
+ finalizaRelacion() → Establece el nodo como punto de finalización de una relación.
+ seleccionar() → Selecciona el nodo como resultado de la consulta.
+ cambiaContar() → Cambia el valor de contar.
+ cambiaAgrupar() → Cambia el valor de agrupar.
+ mover() → Desplaza el nodo por la pantalla.
+ eliminar() → Elimina el nodo de la pantalla.

Atributos clases descendientes:

- random*: Array → Contiene distintas URIs correctas.

Métodos clases descendientes:

Clase Relación y descendientes:

Cada instancia de alguna de las clases Relación representa una relación entre dos nodos de la pantalla.

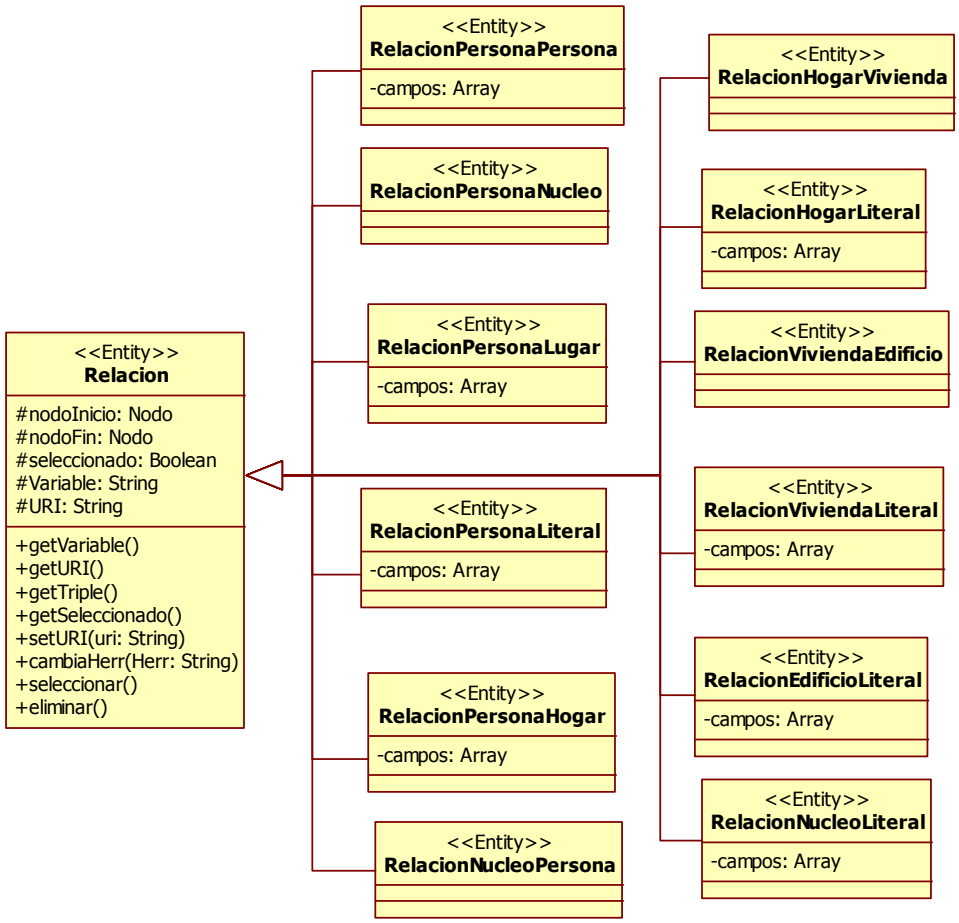

Figura 4.51: Clases Relación y descendientes

Atributos clase Relación:

- # nodoInicio: Nodo
- # nodoFin: Nodo
- # seleccionado: Boolean
- # Variable: String
- # URI: String

Métodos clase Relación:

- + getVariable() → Devuelve el valor de que tiene la variable de la relación.
- + getVariable() → Devuelve el valor de que tiene la URI de la relación.
- + getTriple() → Devuelve el triple formado por la relación y sus dos nodos.
- + getSeleccionado() → Devuelve si la relación está seleccionada o no.

+ cambiaHerr(herr:String) → Cambia el estado de la relación, según la herramienta seleccionada de la aplicación (Añadir nodos, establecer relaciones o seleccionar resultado).
+ seleccionar() → Selecciona la relación como resultado de la consulta.
+ eliminar() → Elimina la relación de la pantalla.

Atributos clases descendientes:

- campos Array → Contiene los distintos campos disponibles en dicha relación.

Métodos clases descendientes:

4.8. Principales Diagramas de secuencia

A continuación se mostrarán alguno de los principales diagramas de secuencia del sistema:

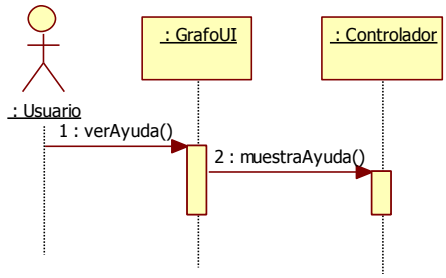

Figura 4.52: DS Ver Ayuda

Este es el diagrama de secuencia del caso de uso Ver ayuda. En el escenario representado, el usuario selecciona la ver la ayuda en la interfaz y el sistema la muestra.

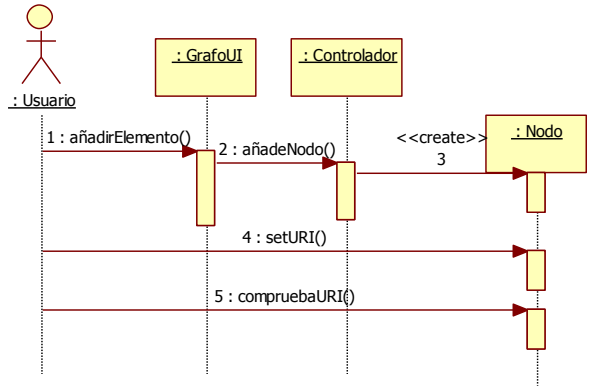

Figura 4.53: DS Añadir Nodo, establecer su URI y Comprobar URI

En el diagrama de secuencia anterior se muestra el escenario en el cual un usuario añade un nuevo nodo, le asigna una nueva URI y después la comprueba.

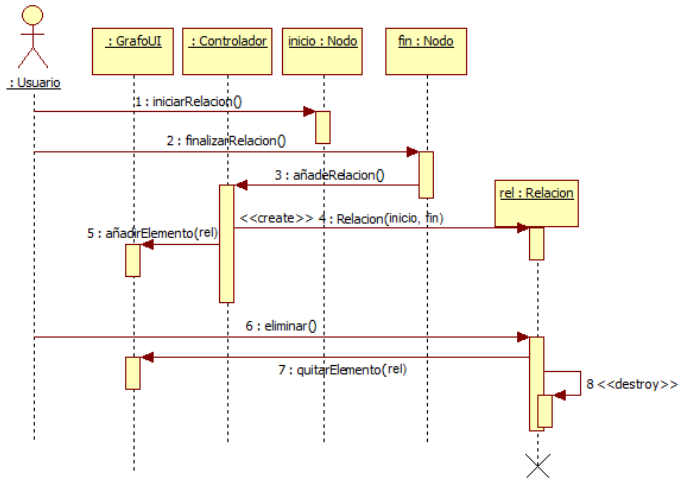

Figura 4.54: DS Añadir y Eliminar Relación

Este diagrama de secuencia pertenece a los casos de uso Añadir y Eliminar relación. En el escenario representado, el usuario crea una nueva relación arrastrando el nodo de inicio hasta el nodo fin y una vez creada la relación el usuario la elimina.

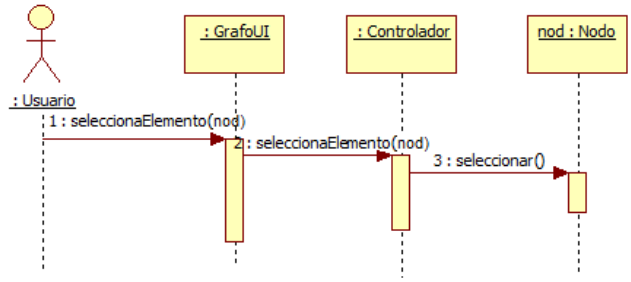

Figura 4.55: DS Seleccionar Nodo

Este es el diagrama de secuencia del caso de uso Seleccionar Resultado. En el escenario representado, el usuario selecciona como resultado uno de los nodos en la interfaz y el controlador notifica la selección al nodo que mostrará las opciones de contar y agrupar.

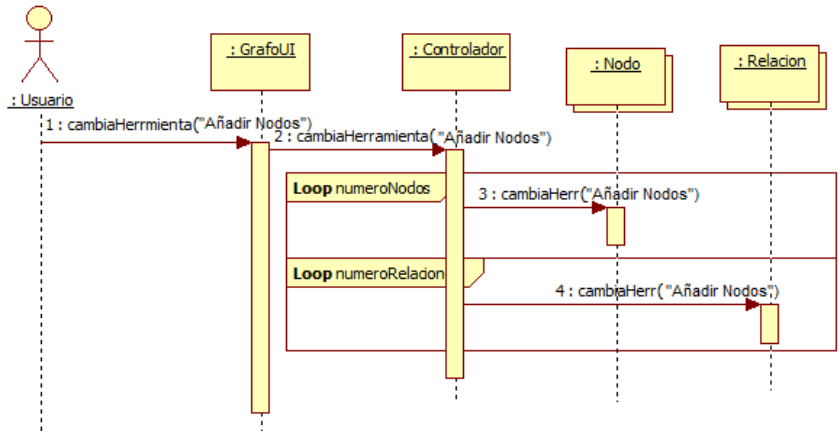

Figura 4.56: DS Seleccionar Herramienta

Este es el diagrama de secuencia del caso de uso Seleccionar Herramienta. En el escenario representado, el usuario selecciona la herramienta "añadir nodos" en la interfaz, y el controlador notifica el cambio a todos los nodos y relaciones.

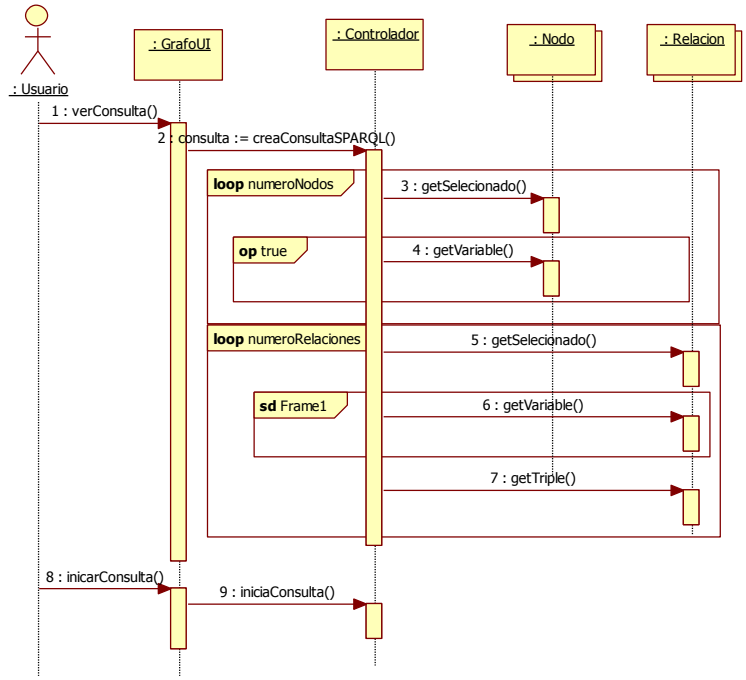

Figura 4.57: DS Ver e Iniciar Consulta

Este es el diagrama de secuencia de los casos de uso Ver e Iniciar Consulta. En el escenario representado, el usuario selecciona "Ver consulta" en la interfaz, y el controlador recorre todos los elementos para crear la consulta asociada, después el usuario lanza la consulta con el fin de obtener los resultados.

DISEÑO:

La fase de diseño [1] se define como el proceso por el cual se traducen las especificaciones de los requisitos en una representación del software que se quiere construir. El diseño es una especie de “puente” entre el análisis del problema y la implementación del mismo. A lo largo de esta fase se transforma el modelo de dominio de la información creado durante el análisis en las estructuras de datos necesarias para implementar el software. Además, el diseño puede entenderse como una materialización de los requisitos.

4.9. Arquitectura del Sistema

La arquitectura de un sistema abarca diferentes dimensiones:

- Arquitectura Lógica: Describe el sistema en términos de su organización conceptual en capas, paquetes, frameworks importantes, clases, interfaces y subsistemas.
- Despliegue de la arquitectura: Describe el sistema en términos de la asignación de los procesos a unidades de procesos y describe la configuración de la red.

Al tratarse de una interfaz gráfica web, el procesamiento se compartirá entre varios ordenadores, lo que nos lleva a una arquitectura Cliente-Servidor. Asignaremos a ambos las diferentes lógicas de la aplicación. Al tratarse de una aplicación web, estaremos ante un cliente delgado:

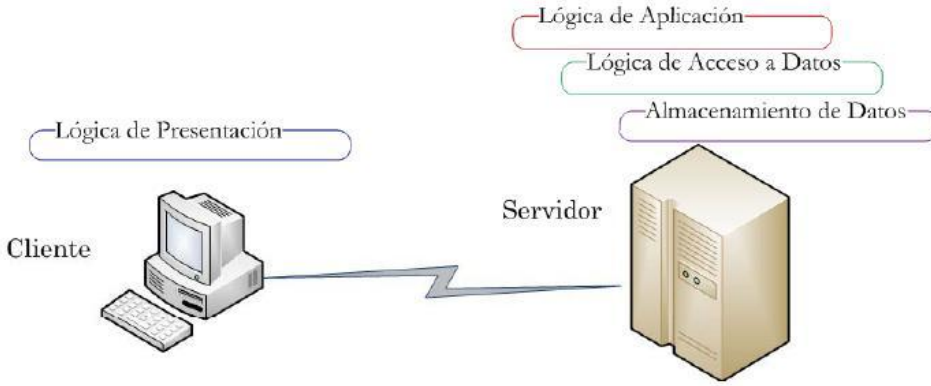

Figura 4.58: Cliente Delgado

4.9.1. Arquitectura Lógica:

Para el desarrollo de la interfaz gráfica de consulta, he optado por la aplicación del modelo arquitectónico de tres capas para establecer el modelo de diseño del sistema.

La programación por capas es un estilo de programación en la que el objetivo primordial es la separación de la lógica de negocio de la lógica de diseño, un ejemplo básico de esto es separar la capa de datos de la capa de presentación al usuario.

La ventaja principal de este estilo, es que el desarrollo se puede llevar a cabo en varios niveles y en caso de algún cambio sólo se ataca al nivel requerido sin tener que revisar entre código mezclado.

En dichas arquitecturas a cada capa se le confía una misión simple, lo que permite el diseño de arquitecturas escalables (que pueden ampliarse con facilidad en caso de que las necesidades aumenten).

La arquitectura propuesta se basa en tres capas bien diferenciadas.

1. **Capa de presentación**: realizada por el navegador, que en este caso es un cliente ligero debido a que no tiene capacidad de proceso o tiene muy poca.
2. **Capa de negocio** (lógica de dominio): aquí ira todo el código que define las reglas de negocio (cálculos, validaciones). Surge de los procesos que se encuentran en el análisis.
3. **Capa de datos**: el código que permite acceder a las fuentes de datos.

El sistema de capas será relajado para permitir las dependencias entre la de Interfaz y la de Persistencia, en pro de la eficiencia.

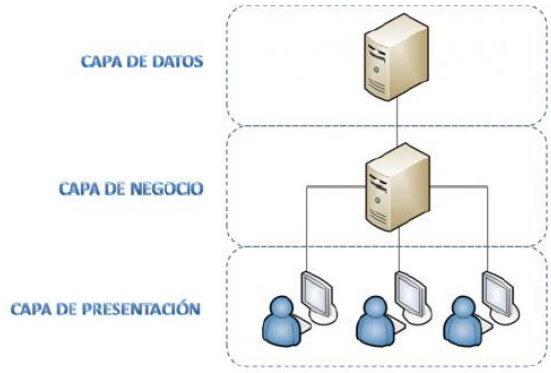

Figura 4.59: Arquitectura de 3 capas

También se ha decidido usar el patrón Modelo Vista Controlador en su versión pasiva, para que sea el controlador quien informe a la vista de los cambios del modelo y así ésta pueda actualizarse.

El Modelo Vista Controlador (MVC) es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos:

- **Modelo**: Esta es la representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de estos y permite derivar nuevos datos.
- **Vista**: Esta presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.
- **Controlador**: Este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista.

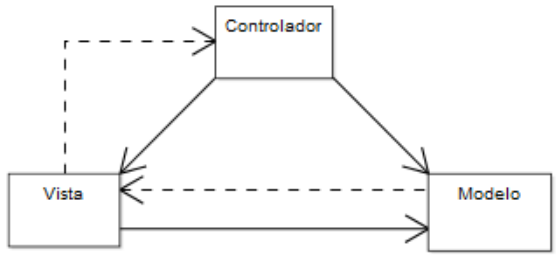

Figura 4.60: Patrón MVC

4.9.1.1. Esquema Arquitectura Final

Dado que el patrón arquitectónico escogido ha sido el de tres capas, y a su vez, hemos aplicado un patrón modelo-vista-controlador en la capa de presentación, la arquitectura final del sistema será la siguiente:

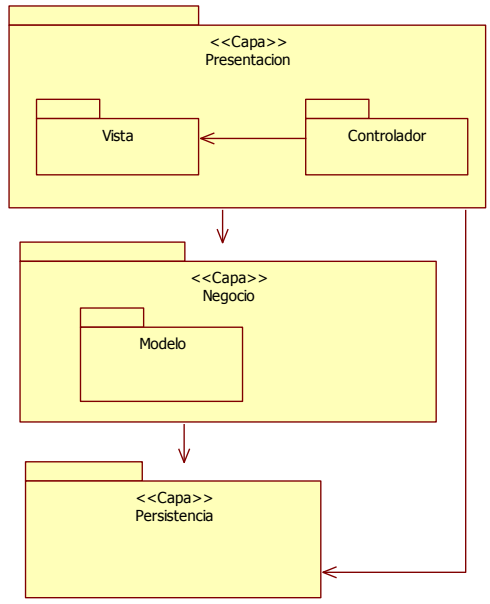

Figura 4.61: Arquitectura final del sistema

La capa de presentación se encarga de mostrar y recoger datos del usuario, la capa de negocio contiene la lógica del modelo vista en el modelo de análisis, y la capa persistencia se ocupa del almacenamiento persistente de los datos.

1- Capa de presentación:
Se compone de dos paquetes bien diferenciados:

- Vista: Se ocupa de la interacción con el usuario. Es quien recibe las entradas del mismo y le presenta los resultados. Se encarga de crear las interfaces y mostrar los distintos elementos de ésta.
- Controlador: Desempeña el papel homónimo del patrón Modelo-Vista-Controlador. Recibe por tanto las peticiones del usuario del paquete GUI, maneja y controla las acciones sobre los objetos de la capa de negocio e informa a la vista de los cambios en éstos.

2- Capa Lógica o de Negocio:
Alberga las entidades de dominio y algunas clases de ayuda derivadas del diseño, englobados en el paquete Modelo. Recibe las peticiones del usuario y envía las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica del negocio) pues es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados.

3- Capa de Persistencia:
Es la encargada del acceso a los datos guardados en los archivos XML y en el acceso a los datos del censo en RDF. Recibe solicitudes de petición y consulta.

4.9.2. Diagrama de Despliegue:

A continuación se muestra un diagrama de despliegue para dar una idea de la que puede ser la estructura del proyecto:

- El usuario interactuará con un navegador Web alojado en un equipo de trabajo.
- El cliente conectará con el servidor Jair, mediante una conexión TCP usando el protocolo HTTP, donde se encuentra hospedado el sistema.
- El servidor Jair conectará con el servidor Gutenberg, mediante un protocolo http, ya que en él se encuentra la capa de persistencia (ficheros XML y Censo RDF).

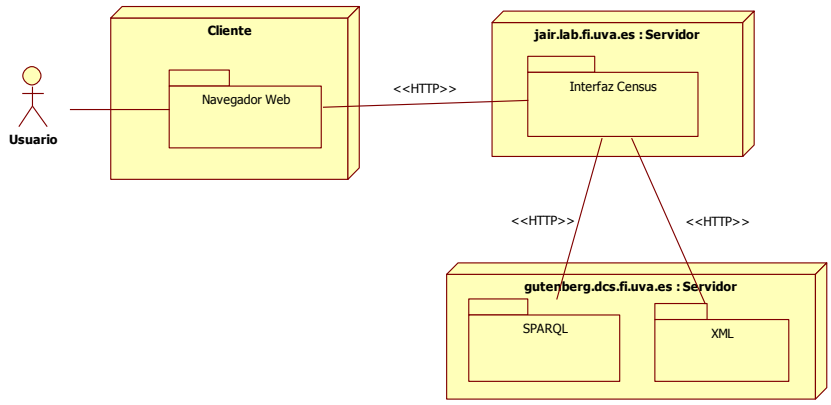

Figura 4.62: Diagrama de Despliegue.

4.10. Descripción de los paquetes de diseño

4.10.1. Capa de Presentación o interfaz

Se entiende por interfaz como el artefacto que se usa para acceder e interactuar con un sistema. Se trata a la vez un límite y un espacio común entre sistema y usuario. Este artefacto nos informa qué acciones son posibles, el estado actual del sistema y los cambios producidos, y nos permite actuar con o sobre él.

Las interfaces Web tienen ciertas limitaciones en las funcionalidades que se ofrecen al usuario. Hay funcionalidades comunes en las aplicaciones de escritorio como dibujar en la pantalla o arrastrar-y-soltar que no están soportadas por las tecnologías Web estándar. Los desarrolladores Web generalmente utilizan lenguajes interpretados o de script en el lado del cliente para añadir más funcionalidades, especialmente para ofrecer una experiencia interactiva (páginas dinámicas).

Recientemente se han desarrollado tecnologías que mejoran enormemente la experiencia del usuario con la página. Es el caso de Adobe flash y su lenguaje asociado ActionScript, tecnología utilizada en este proyecto, con la que se pueden construir aplicaciones que se salen del modelo tradicional Web de envíos sucesivos, proporcionando una interfaz de usuario mejorada gracias a la comunicación asíncrona entre la interfaz de usuario y el servidor Web.

Una interfaz mal diseñada se dice que genera problemas de usabilidad. El mal diseño de las interfaces puede generar aumento de costos, algunos de ellos son medibles y otros no. Actualmente, hasta el 45% del código de una aplicación está dedicado a la interfaz, persiguiendo en todo momento la facilidad de uso. Sin embargo, se dedica algo menos del 10% del presupuesto global de un proyecto al desarrollo de la interfaz, lo que indica que aumentar los recursos destinados al desarrollo de la interfaz es una excelente inversión, teniendo en cuenta la relación costo/beneficio medible y segura, aún sin tener en cuenta los beneficios no medibles en dinero como el aumento de la satisfacción.

Dado que el entorno de programación de Adobe Flash Builder nos facilita crear la interfaz de usuario, es decir, nos permite añadir los diferentes botones, campos de texto, movieclips, etc. directamente sobre lo que será la pantalla, no ha sido uno de mis objetivos el diseño de interfaces de usuario.

- **Diagrama de clases de la capa Presentación:**

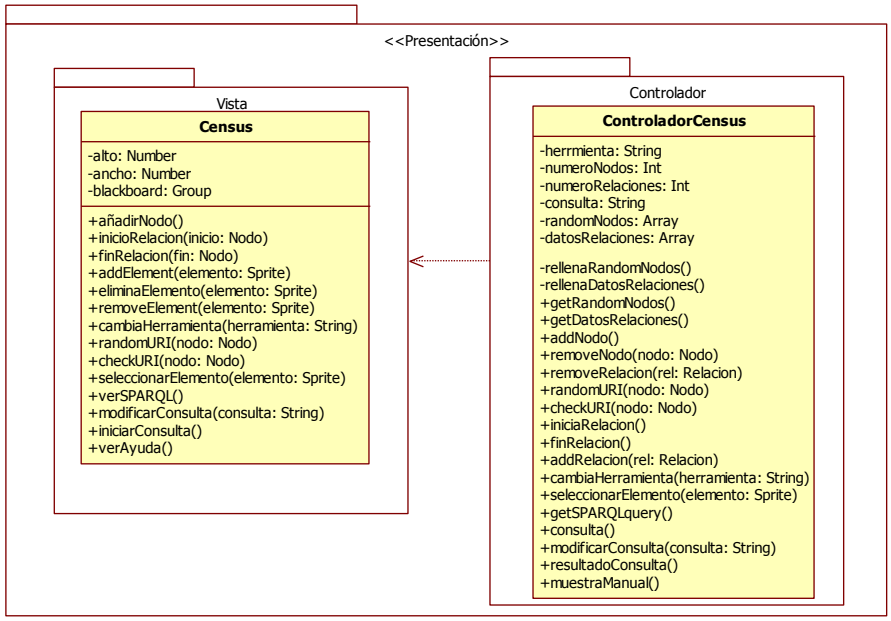

Figura 4.63: Diagrama Clases Capa Presentación

Breve descripción de las clases:

Nombre: Census
Descripción: Se trata de la interfaz de visualización. Es el contenedor principal de todos los elementos gráficos que serán visualizados al ejecutar la aplicación.
Atributos: • *alto: Number*. Indica la altura de la pantalla. • *ancho: Number*. Indica el ancho de la pantalla. • *blackboard*: Group. Lugar donde serán añadidos los elementos.
Métodos: • *añadirNodo().* Añade un nodo a la pantalla. • *inicioRelacion(inicio:Nodo).* Comienza una relación desde el nodo inicio. • *finRelacion(inicio:Nodo).* Finaliza la creación de una relación en el nodo fin. • *addElement(Elemento: Sprite).* Añade el elemento a la pantalla. • *eliminaElemento(Elemento: Sprite).* Elimina el elemento de la aplicación. • *removeElement(Elemento: Sprite).* Borra el elemento de la pantalla. • *cambiaHerrmienta(herramienta: String).* Cambia la herramienta que se está utilizando en la aplicación. • *randomURI(nodo:Nodo).* El nodo toma como valor una URI aleatoria. • *checkURI(nodo:Nodo).* Comprueba si la URI del nodo es correcta o no. • *seleccionaElemento(Elemento: Sprite).* Selecciona como resultado el elemento. • *verSPARQL().* Muestra la consulta SPARQL asociada al grafo creado. • *modificarConsulta(con:String).* Modifica la consulta y lo muestra por pantalla.

- *iniciarConsulta().* Inicia la consulta SPARQL generada y muestra el resultado.
- *verAyuda().* Muestra el manual de ayuda de la aplicación.

Figura 4.64: Clase Census

Nombre: ControladorCensus

Descripción: Se trata del controlador de la aplicación. Sera el encargado de comunicar el modelo de negocio con la interfaz.

Atributos:

- *herramienta: String.* Herramienta actual de la aplicación (Añadir nodos, Establecer relaciones, Seleccionar resultado).
- *numeroNodos: Int*. Número de nodos añadidas en pantalla.
- *numeroRelaciones: Int*. Número de relaciones añadidas en pantalla.
- *Consulta: String*. Consulta SPARQL.
- *randomNodos*: Array. Array que recoge los valores aleatorios de las URI de los Nodos.
- *randomRelaciones*: Array. Array que recoge los valores de los campos de las distintas relaciones.

Métodos:

- *rellenaRandomNodos().* Función que recoge los datos de las URI aleatorias del archivo XML que se encuentra en el servidor.
- *rellenaDatosRelaciones().*Función que recoge los datos de las relaciones del archivo XML que se encuentra en el servidor.
- *getRandomNodos().* Devuelve el array con las URIs aleatorias.
- *getDatosRelaciones().* Devuelve el array con los datos de las relaciones.
- *addNodo(). Crea un nodo y lo añade a la pantalla*
- *removeNodo(nodo:Nodo). Elimina el nodo y lo borra de la pantalla.*
- *removeRelacion(rel:Relacion). Elimina la relación y la borra de la pantalla.*
- *randomURI(nodo:Nodo).* El nodo toma como valor una URI aleatoria*.*
- *checkURI(nodo:Nodo).* Comprueba si la URI del nodo es correcta o no.
- *iniciaRelacion(inicio:Nodo).* Comienza una relación desde el nodo inicio.
- *finRelacion(inicio:Nodo).* Finaliza la creación de una relación en el nodo fin.
- *addRelacion(nodoInicio:Nodo, nodoFin:Nodo). Crea una relación entre dos nodos y la añade a la pantalla.*
- *cambiaHerrmienta(herramienta: String).* Cambia la herramienta que se está utilizando en la aplicación.
- *seleccionaElemento(Elemento: Sprite).* Selecciona como resultado el elemento.
- *getSPARQLquery().* Obtiene la consulta SPARQL asociada al grafo creado.
- *consulta().* Inicia la consulta SPARQL generada y muestra el resultado.
- *modificarConsulta(con:String).* Modifica la consulta SPARQL y lo muestra por pantalla.
- *resultadoConsulta().* Recoge los datos del servidor y transforma los resultados.
- *muestraManual().* Muestra el manual de ayuda de la aplicación.

Figura 4.65: Clase ControladorCensus

4.10.2. Capa lógica o de negocio

Se denomina capa de negocio, pues es aquí donde se establecen todas las reglas que deben cumplirse. En la siguiente figura podemos ver el modelo de diseño aplicado al proyecto.

4.10.2.1. Modelo de Negocio

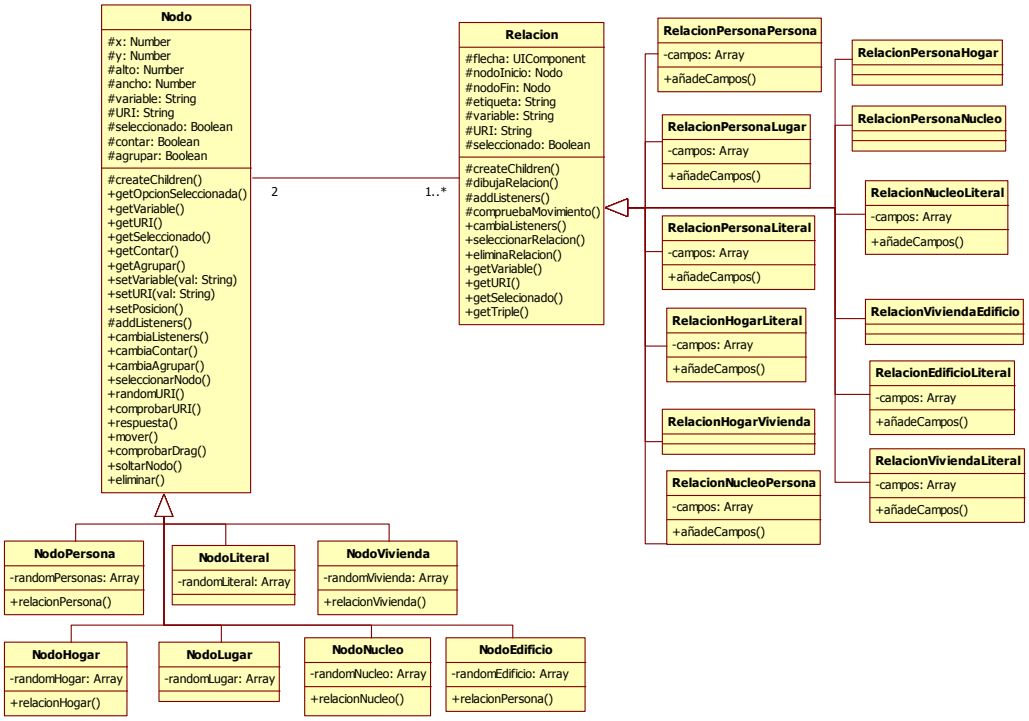

Figura 4.66: Diagrama Clases Negocio

4.10.2.2. Explicación de las clases de diseño

Nombre: Nodo

Descripción: Representa los nodos del grafo. Hereda de la clase panel de flash.

Atributos:

- *x: Number*. Indica la posición X del nodo en la pantalla.
- *y: Number*. Indica la posición Y del nodo en la pantalla.
- *alto: Number*. Indica la altura del nodo.
- *ancho: Number*. Indica el ancho del nodo.
- *variable: String*. Nombre de la variable del nodo.
- *URI: String*. Nombre del identificador URI del nodo.
- *selecionado: Boolean*. Indica si el nodo esta seleccionado o no.
- *contar: Boolean*. Indica si selecciona la cuenta como resultado.
- *agrupar: Boolean*. Indica si se agrupa el resultado por este nodo.

Métodos:

- *createChildren().* Sobrescribimos el método createChildren de panel para configurar nuestro nodo.
- *getOpcionSelecionada().* Nos indica si esta seleccionada la opción de variable o de URI.
- *getvariable().* Devuelve el valor de la variable del nodo.
- *getURI().* Devuelve el identificador URI del nodo.
- *getSelecionado().* Devuelve si el nodo esta seleccionado o no.
- *getContar().*Devuelve si se debe obtener la cuenta (count) como resultado.
- *getAgrupar().*Devuelve si se debe agrupar el resultado por el nodo.
- *setvariable(val:String).* Establece el valor de la variable del nodo.
- *setURI(val:String).* Establece el valor de la URI del nodo.
- *setPosicion(x:Int, y:Int).* Posiciona el nodo en un punto de la pantalla.
- *addListeners().* Añade escuchadores de eventos con el fin de realizar determinadas acciones en respuesta a esos eventos.
- *cambiaListeners().* Cambia los escuchadores de eventos según la herramienta actual de la aplicación (Añadir nodos, Establecer relaciones, Seleccionar).
- *mover().* Permite desplazar el nodo por la pantalla.
- *selecionarNodo().* Cambia el estado de seleccionado y muestra las posibilidades de contar o agrupar.
- *cambiaContar().*Cambia el valor de contar.
- *cambiaAgrupar ().* Cambia el valor de agrupar.
- *randomURI().* Devuelve una URI aleatoria correcta.
- *comprobarURI().* Comprueba si la URI introducida es valida o no.
- *comprobarDrag().* Comprueba si el nodo puede ser objetivo del evento drag and drop.
- *soltarNodo().* Función activada al final de un drag and drop con el fin de establecer una relación.
- *respuesta().* Recoge los datos del servidor cuando se comprueba una URI y devuelve un booleano.
- *eliminar().* Elimina el nodo de la pantalla.

Figura 4.67: Clase Nodo

Nombre: NodoPersona, NodoHogar, NodoNucleo,….
Descripción: Representan a las distintas entidades (nodos) de la pantalla con las que se podrá interactuar.
Atributos: • *random*: Array*. Array con distintas URIs correctas para cada tipo de nodo.
Métodos: • *relacionNodo().* Función iniciada al comienzo de un drag and drop con el fin de establecer una relación.

Figura 4.68: Clases descendientes de Nodo

Nombre: Relación
Descripción: Representa la relación (flecha) entre dos nodos. Desciende directamente de la clase UIComponent.
Atributos: • *flecha: UIComponet*. Gráfico de la flecha. • *nodoInicio: Nodo*. Nodo de inicio de la relación. • *nodoFin: Nodo*. Nodo don finaliza la relación. • *etiqueta: String*. Nombre del campo seleccionado. • *variable: String*. Nombre de la variable de la relación. • *URI*: *String.* Identificador de la relación. • *selecionado: Boolean*. Indica si la relación está seleccionada o no.
Métodos: • *createChildren().* Sobrescribimos el método createChildren de panel para configurar la relación. • *dibujaRelacion().* Dibuja la relación en la pantalla, y la mueve si alguno de sus nodos es movido. • *addListeners().* Añade escuchadores de eventos con el fin de realizar determinadas acciones en respuesta a esos eventos. • *cambiaListeners().* Cambia los escuchadores de eventos según el estado actual de la aplicación (Añadir nodos, Establecer relaciones, Seleccionar). • *compruebaMovimiento().* Comprueba si alguno de sus nodos se está moviendo por la pantalla. • *selecionarRelacion().* Cambia el estado de seleccionado. • *eliminaRelacion().* Elimina la relación del sistema. • *getURI().* Devuelve el identificador de la relación. • *getVariable().* Devuelve la variable de la relación. • *getSelecionado().* Devuelve si el nodo esta seleccionado o no. • *getTriple().* Devuelve el triple formado por la relación y sus nodos.

Figura 4.69: Clase Relación

Nombre: RelacionPersonaPersona, RelacionPersonaNucleo,…
Descripción: Representa cada tipo de las posibles relaciones entre los nodos.
Atributos: • *campos: Array*. Array con los distintos campos posibles de la relación.
Métodos: • *añadeCampos().* Muestra la lista con los distintos campos de la relación.

Figura 4.70: Clases descendientes de Relación

4.10.3. Capa de persistencia

En este proyecto no ha sido necesario desarrollar la capa de persistencia ya que el censo de 2001 en formato RDF se encuentra disponible en el servidor gutenberg.dcs.fi.uva.es de la Universidad de Valladolid. Se accederá a él mediante una conexión TCP usando el protocolo HTTP, realizando las distintas consultas mediante la URL.

Además del censo en formato RDF, se han añadido en el servidor dos archivos XML (randomNodos.xml y datosRelaciones.xml) donde se encuentran almacenados las URIs aleatorias de cada nodo y los datos de los campos de las relaciones.

4.11. Especificación de los Casos de Uso - Diagramas de Secuencia

UC-0001	Ver ayuda
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el Usuario desee obtener ayuda acerca del sistema.
Secuencia Normal	1. El Caso de Uso comienza cuando el usuario selecciona "Ayuda" 2. El caso de uso finaliza cuando el sistema muestra el manual de ayuda.
Poscondición	Se muestra la ayuda

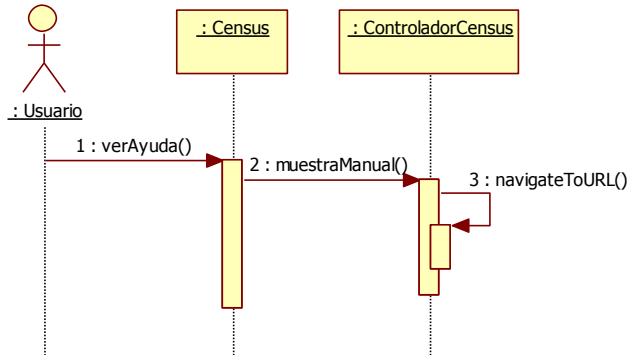

Figura 4.71: Especificación UC Ver Ayuda

UC-0002	Añadir Nodo
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el Usuario desee añadir un nodo entidad a la pantalla.
Precondición	Está seleccionada la herramienta "Añadir Nodos".
Secuencia Normal	1. El Caso de Uso comienza cuando el usuario selecciona uno de los botones de las entidades. 2. El sistema crea un nuevo nodo y lo añade a la pantalla. 3. El usurario puede modificar la variable o completar una URI. 4. El caso de uso finaliza.
Poscondición	El sistema ha creado un nuevo nodo en la pantalla.
Flujos Alternativos	En el paso 4, si el usuario desea: - Añadir una URI aleatoria: Punto de extensión: "URI Aleatoria [UC-0003]" - Comprobar valided de la URI introducida: Punto de extensión: "Comprobar URI [UC-0004]" - Eliminar el nodo: Punto de extensión: "Eliminar Nodo [UC-0005]"

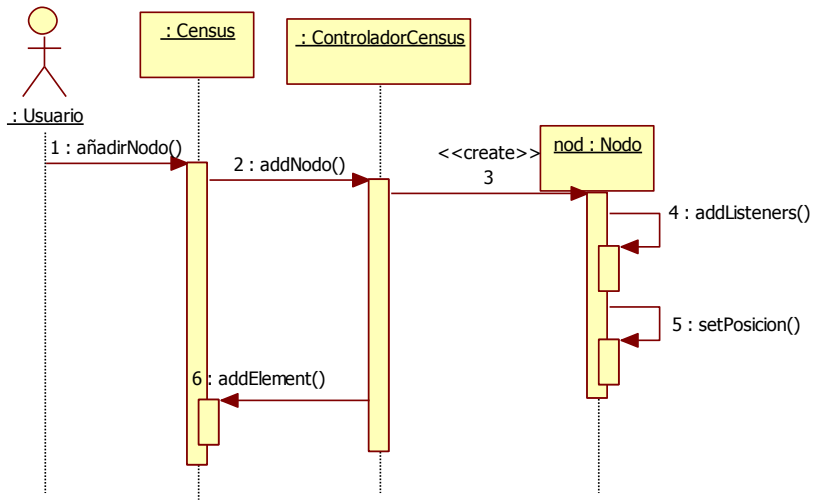

Figura 4.72: Especificación UC Añadir Nodo

Nota: El nodo creado será del tipo seleccionado por el usuario.

UC-0003	**URI Aleatoria**
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el Usuario desee establecer una URI aleatoria correcta en un nodo.
Precondición	Existe al menos un nodo en la pantalla.
Secuencia Normal	1. El Caso de Uso comienza cuando el usuario selecciona “randomURI” en un nodo. 2. El caso de uso finaliza cuando el sistema genera una URI aleatoria y la muestra.
Poscondición	Se asigna una URI aleatoria al nodo.

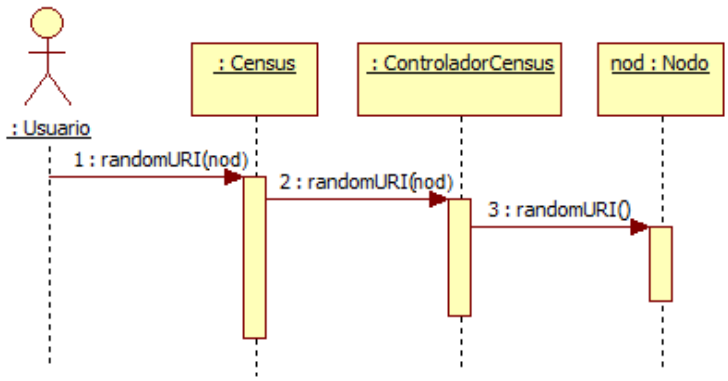

Figura 4.73: Especificación UC URI aleatoria

UC-0004	Comprobar URI
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el Usuario desee comprobar la validad de una URI introducida.
Precondición	Existe al menos un nodo en la pantalla y se ha introducido una URI para éste.
Secuencia Normal	1. El Caso de Uso comienza cuando el usuario selecciona “CheckURI” en un nodo. 2. El sistema realiza la consulta al servidor via http y recibe el resultado. 3. El caso de uso finaliza cuando el sistema muestra si la URI es valida o no.
Poscondición	Se sabe si la URI introducida es valida o no.

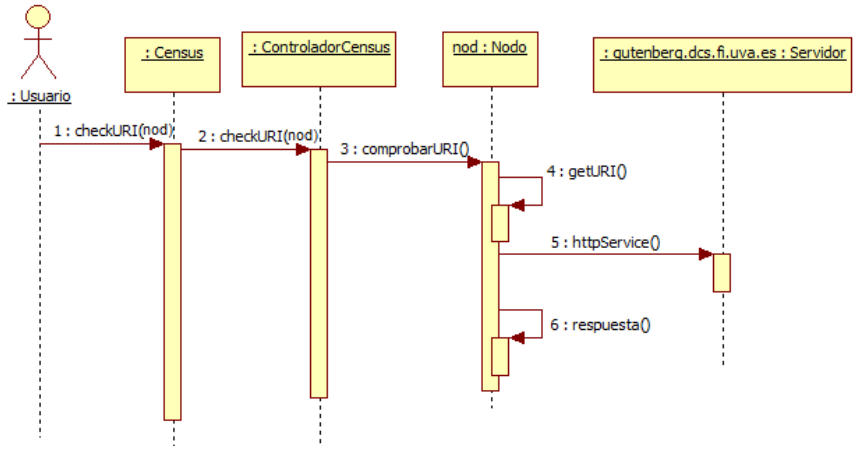

Figura 4.74: Especificación UC Comprobar URI

UC-0005	Eliminar Nodo
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el Usuario desee eliminar un nodo de la pantalla.
Precondición	Existe al menos un nodo en la pantalla.
Secuencia Normal	1. El Caso de Uso comienza cuando el usuario selecciona “Eliminar” en un nodo. 2. El caso de uso finaliza cuando el sistema elimina el nodo y lo borra de la pantalla.
Poscondición	El nodo ha sido eliminado de la pantalla.
Flujos Alternativos	En el paso 2, si el nodo tiene alguna relación establecida: El sistema elimina también las relaciones que salían o entraban en el nodo. - Punto de extensión: “Eliminar Relación [UC-0017]”

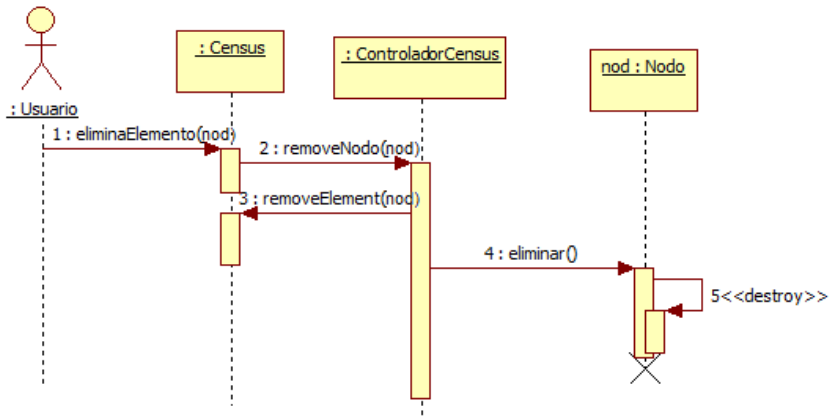

Figura 4.75: Especificación UC Eliminar Nodo

UC-0006	**Añadir Relación**
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el Usuario desee añadir una relación entre dos nodos.
Precondición	Está seleccionada la herramienta “Establecer Relaciones”.
Secuencia Normal	1. El Caso de Uso comienza cuando el Usuario arrastra un nodo hacia otro. 2. El sistema comprueba entre que tipos de nodos se esta estableciendo la relación (compruebaDrag). 3. El sistema crea la relación entre nodos y la añade en pantalla pudiendo el usuario cambiar el su campo. 4. El caso de uso finaliza.
Poscondición	Se crea una relación entre los nodos.
Flujos Alternativos	En el paso 2, si el sistema comprueba que no puede haber relación entre esos nodos el caso de uso finaliza. Después del paso 4, si el usuario desea eliminar la relación: - Punto de extensión: “Eliminar Relación [UC-0017]”

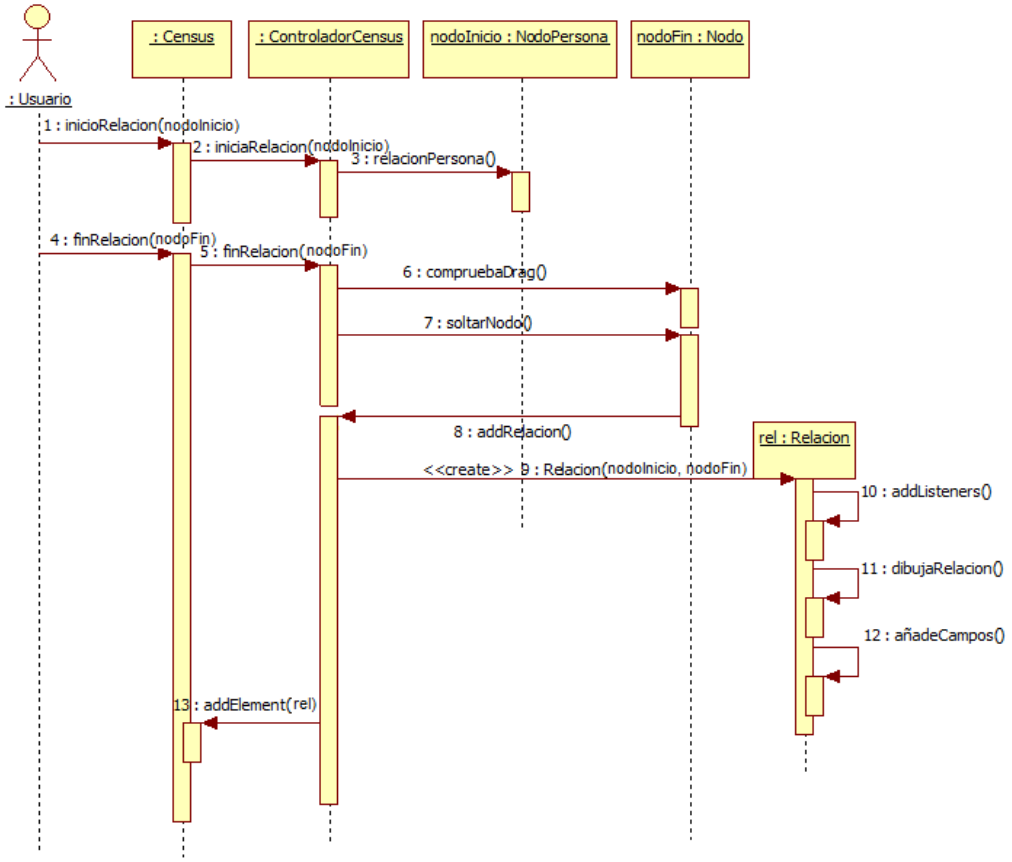

Figura 4.76: Especificación UC Añadir Relación

Nota: El tipo de relación creada dependerá del tipo de nodoInicio y del tipo del nodoFin. En este caso el nodoInicio es un nodo Persona pero podrá ser cualquiera de los que implementen una relación análoga a relaciónPersona().

UC-0007	**Eliminar Relación**
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el Usuario desee eliminar una relación existente entre nodos.
Precondición	Está seleccionada la herramienta “Añadir Nodos” y existe al menos una relación establecida entre nodos.
Secuencia Normal	1. El Caso de Uso comienza cuando selecciona eliminar relación. 2. El sistema elimina la relación entre los nodos. 3. El caso de uso finaliza.
Poscondición	La relación entre los nodos ha sido eliminada.

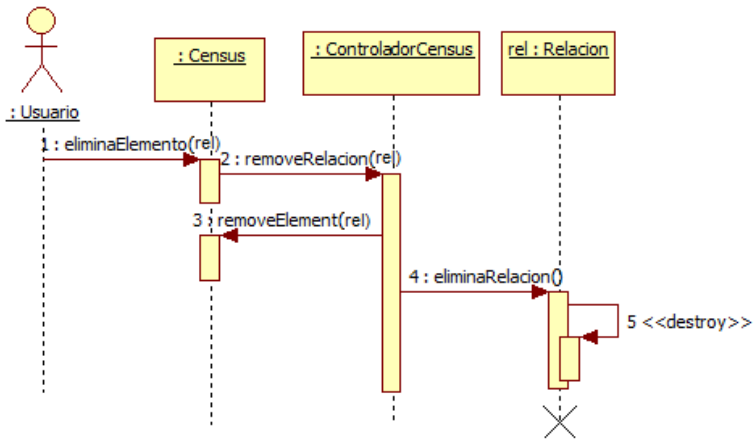

Figura 4.77: Especificación UC Eliminar Relación

UC-0008	**Seleccionar Resultado**
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el Usuario desee seleccionar un elemento para que sea parte del resultado de la consulta.
Precondición	Está seleccionada la herramienta “Seleccionar Resultado” y existen elementos añadidos a la pantalla.
Secuencia Normal	1. El Caso de Uso comienza cuando el usuario selecciona uno de los elementos de la pantalla. 2. El sistema selecciona el elemento y muestra las opciones de contar y agrupar en caso de que el elemento sea un nodo. 3. El Caso de Uso finaliza.
Poscondición	El elemento se encuentra en el estado “seleccionado”.
Flujos Alternativos	En el paso 2, si el elemento ya estaba seleccionado: - El sistema cambia el estado “deseleccionado” del elemento y oculta las opciones de contar y agrupar en caso de que el elemento sea un nodo. En el paso 3, si el elemento seleccionado es un nodo: - El usuario podrá seleccionar las opciones de contar y agrupar. - El Caso de Uso finaliza.

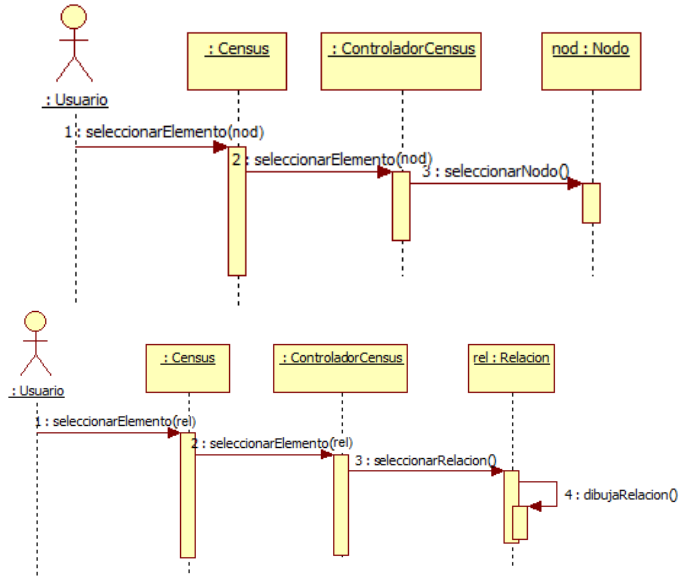

Figura 4.78: Especificación UC Seleccionar Resultado

UC-0009	**Seleccionar Herramienta**
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el Usuario desee cambiar el estado de la aplicación
Secuencia Normal	1. El Caso de Uso comienza cuando el Usuario selecciona una de las herramientas del sistema. 2. El sistema cambia la herramienta seleccionada, cambiando los escuchadores de todos los elementos en la pantalla: - Añadir Nodos: El sistema permitirá añadir nuevos nodos y colocarlos por la pantalla, o eliminar cualquier elemento de la pantalla. - Establecer Relaciones: El sistema permitirá arrastrar los nodos para establecer relaciones. - Seleccionar Resultado: El sistema permitirá seleccionar los distintos elementos como resultado. 3. El Caso de Uso finaliza cuando el sistema actualiza el estado actual.
Poscondición	El sistema se encuentra en el estado seleccionado.

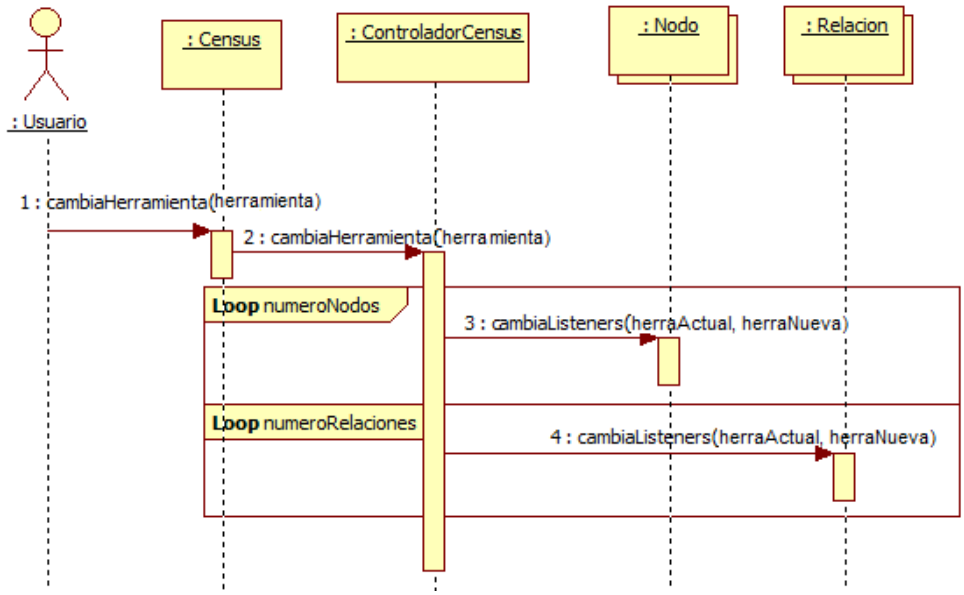

Figura 4.79: Especificación UC Seleccionar Herramienta

UC-0010	**Ver Consulta**
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el Usuario desee ver la consulta SPARQL asociada al grafo creado.
Precondición	Existe un grafo creado y es correcto.
Secuencia Normal	1. El Caso de Uso comienza cuando el Usuario selecciona la pestaña “SPARQL”. 2. El sistema transforma el grafo diseñado a una consulta SPARQL. 3. El caso de uso finaliza cuando el sistema muestra dicha consulta.
Poscondición	La consulta asociada al grafo es mostrada.
Flujos Alternativos	En el paso 2, si el grafo no esta bien diseñado: - El Caso de Uso finaliza cuando el sistema muestra un mensaje de error al trasformar la consulta. Después del paso 3, si el usuario desea modificar la consulta: - Punto de extensión: “Modificar Consulta [UC-0011]”

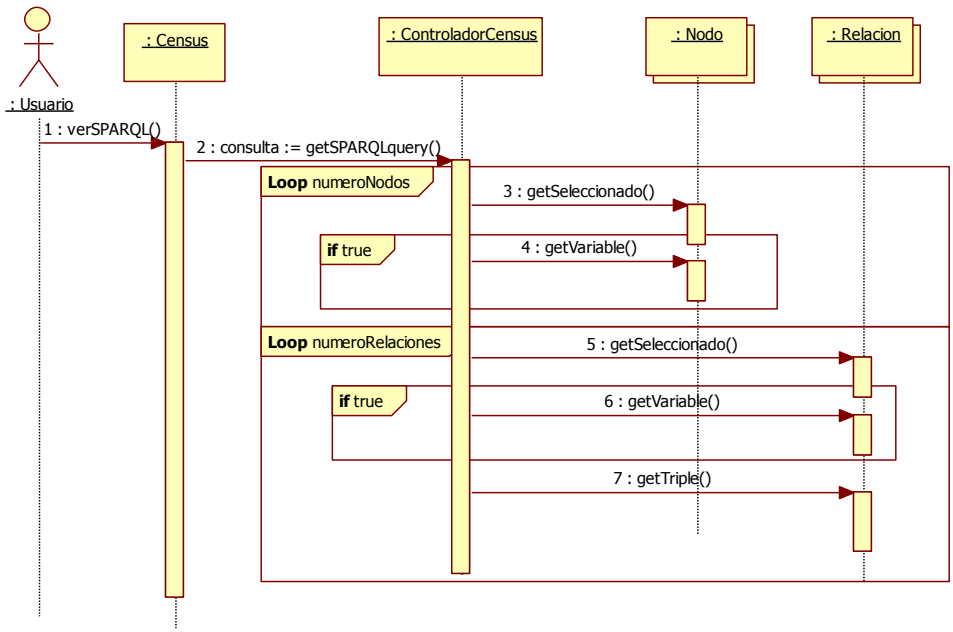

Figura 4.80: Especificación UC Ver Consulta

UC-0011	**Modificar Consulta**
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el Usuario desea modificar la consulta SPARQL generada por el grafo.
Precondición	Se ha realizado el caso de uso “Ver Consulta [UC-0010]”.
Secuencia Normal	1. El Caso de Uso comienza cuando el Usuario escribe en la consulta mostrada por el sistema. 2. El sistema modifica la consulta con lo introducido por el usuario y lo muestra. 3. El caso de uso finaliza
Poscondición	Se muestra la consulta SPARQL modificada por el usuario.

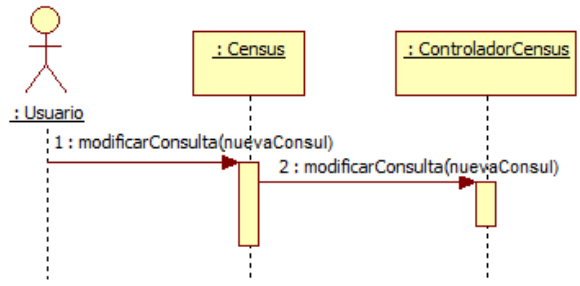

Figura 4.81: Especificación UC Modificar Consulta

UC-0012	**Iniciar Consulta**
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el Usuario desee obtener los resultados de la consulta.
Precondición	Existe un grafo creado.
Secuencia Normal	1. El Caso de Uso comienza cuando el Usuario selecciona “Iniciar consulta”. 2. El sistema obtiene la consulta del grafo creado. 3. El sistema envía la consulta al servidor. 4. El caso de uso finaliza cuando el sistema muestra los resultados obtenidos.
Postcondición	Los resultados de la consulta SPARQL asociada al grafo son mostrados en pantalla.
Flujos Alternativos	En el paso 2, si la consulta a sido modificada anteriormente por el usuario: - El sistema recoge la consulta modificada y el caso de uso continúa.

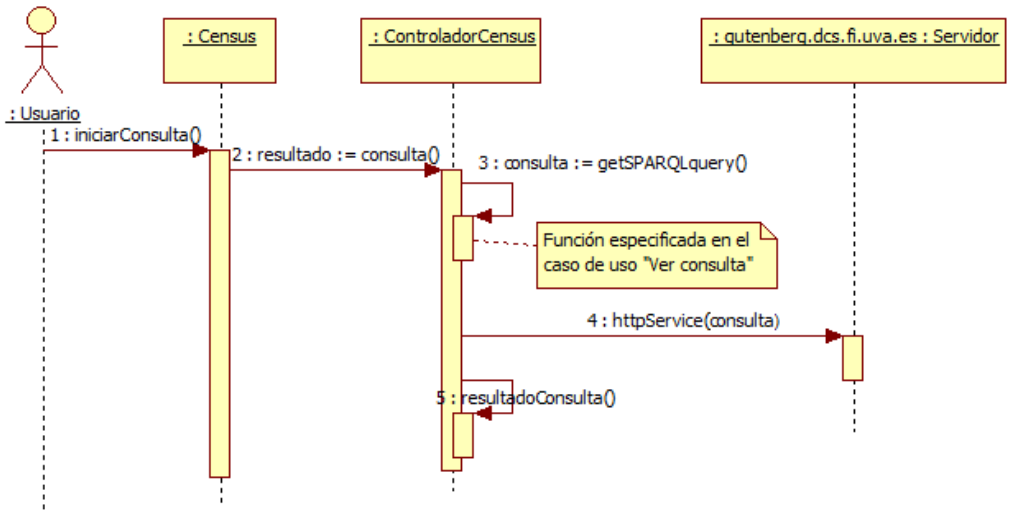

Figura 4.82: Especificación UC Iniciar Consulta

- **Diagrama de secuencia de inicio de la aplicación**

Este diagrama de secuencia incorpora la clase ficticia “InicioAplicación”, cuyo método “ejecutar” representa la ejecución por parte del usuario de la aplicación. Las operaciones que ella inicia son lo que ocurre tras esa ejecución. En concreto, se trata de la recogida de los datos de los archivos XML que se encuentran en el servidor y la inicialización del estado a “Añadir Nodos”.

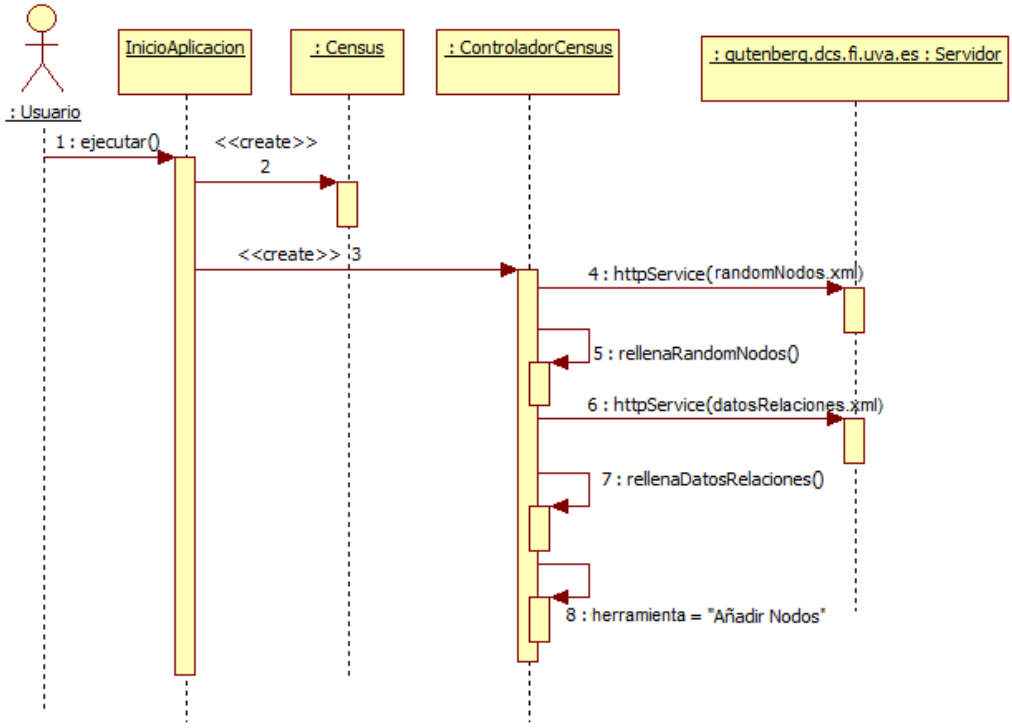

Figura 4.83: DS – Inicio Aplicación

IMPLEMENTACIÓN:

La implementación del sistema consiste en la traducción del diseño realizado en el capítulo anterior a código de un lenguaje de programación.

4.12 Entorno de programación

Para el desarrollo del proyecto se propuso el lenguaje de programación ActionScript 3.0. Este lenguaje está integrado en el entorno de desarrollo de Adobe Flash Builder. Es un lenguaje orientado a objetos (OOP) e ideal para la realización de aplicaciones web animadas como es nuestro proyecto.
Dado que nuestra aplicación estaba orientada al funcionamiento en páginas Web, contar con un lenguaje de programación basado en objetos, podría ser una forma de facilitar la realización del proyecto.

4.12.1 Adobe Flash Builder

Adobe Flash Builder [10] es un entorno de desarrollo integrado escrito en la plataforma Eclipse destinado para el desarrollo de aplicaciones de Internet enriquecidas (RIA) y aplicaciones de escritorio multiplataforma, particularmente para la plataforma de Adobe Flash. El soporte para aplicaciones de escritorio multiplataforma fue añadido en Flex Builder 3 con la introducción de Adobe AIR.

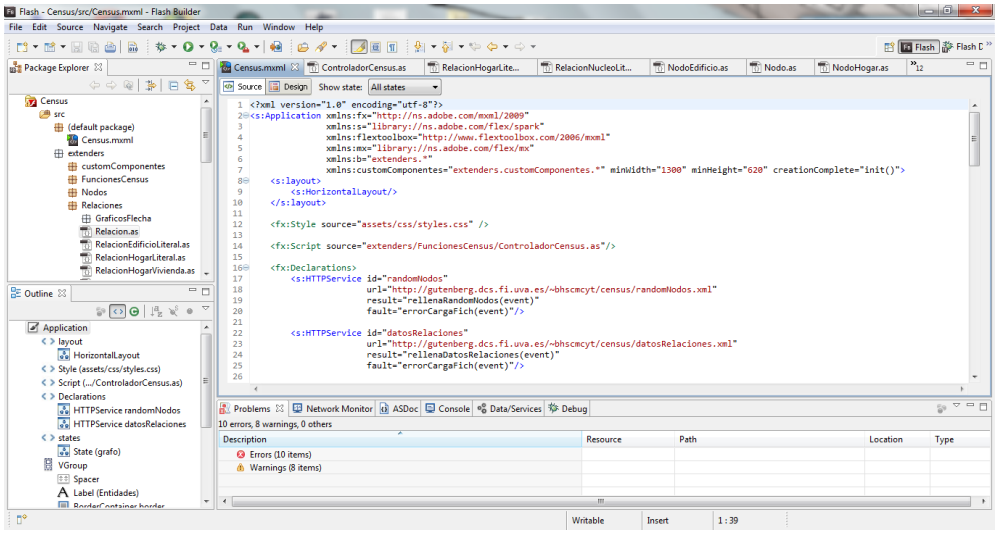

Figura 4.84: Adobe Flash Builder 4.6

Flash se usa con frecuencia en los anuncios, juegos Web y aplicaciones animadas. Anualmente, se ha posicionado como una herramienta importante para las “Aplicaciones de Internet enriquecidas” (RIA).

Permite manipular gráficos vectoriales y mapas de bits para proporcionar animación de texto, de dibujos e imágenes fijas. Soporta streaming bidireccional de audio y vídeo, y se puede capturar la entrada del usuario mediante el mouse, teclado, micrófono y cámara. Flash contiene un lenguaje orientado a objetos denominado ActionScript.

El contenido Flash puede ser mostrado en diversos sistemas informáticos y dispositivos, usando Adobe Flash Player, que está disponible en navegadores comunes de Internet, en algunos teléfonos móviles y en algunos otros dispositivos electrónicos (con Flash Lite).

Los archivos de Flash, tienen generalmente la extensión de archivo SWF, pueden aparecer en una página web para ser vista en un navegador, o pueden ser reproducidos independientemente por un reproductor Flash. Los archivos de Flash aparecen muy a menudo como animaciones en páginas Web y sitios Web multimedia, y más recientemente en Aplicaciones de Internet Enriquecidas.

Flash Builer proporciona un soporte completo para ActionScript 3.0, gracias a un potente editor de código. Además, proporciona un leguaje de etiquetado, Adobe Flex [5], con el cual es posible desarrollar rápidamente las interfaces gráficas de aplicaciones ya que nos permite añadir componentes prefabricaos (botones, etiquetas, contenedores, gráficos, etc.) los cuales podrán ser customizados y extendidos.

4.12.2 ActionScript 3.0

ActionScript [9] es un lenguaje de programación orientado a objetos (OOP) desarrollado por Adobe que permite un mejor control y la reutilización del código en la construcción de complejas aplicaciones Flash. ActionScript es un lenguaje de "código abierto" en el sentido de que sus características se ofrecen de forma gratuita, así como el código del compilador y de la máquina virtual están disponibles.

ActionScript es un lenguaje de script, esto es, no requiere la creación de un programa completo para que la aplicación alcance los objetivos. Se trata de un dialecto de ECMAScript (lo que significa que tiene la misma sintaxis y la semántica del más conocido JavaScript), y se utiliza principalmente para el desarrollo de sitios web y aplicaciones software realizadas en el entorno Adobe Flash. Se utiliza en las páginas Web en forma de archivos SWF embebidos en ésta.

Se diseñó inicialmente para el control de animaciones simples vectoriales 2D realizados en Adobe Flash. Inicialmente se centró en la animación, pero pronto las primeras versiones ofrecieron algunas características interactividad y por lo tanto había una capacidad limitada de scripting. Hoy en día, las versiones posteriores agregaron funcionalidad que permitió la creación de juegos basados en Web, de aplicaciones ricas de Internet con streaming (como vídeo y audio), de algunas aplicaciones de bases de datos, etc.
ActionScript nos permite realizar las siguientes funciones:

- Controlar las propiedades de los objetos: ActionScript puede ser utilizado para examinar o modificar las propiedades de los elementos. Por ejemplo:
  - Cambiar las características de un objeto.

- Reproducir, parar, reducir/aumentar el volumen de un sonido o video.
- Modificar las propiedades repetidamente produciéndo comportamientos únicos como son los movimientos basados en la física y la detección de colisiones.
- Campos de texto que permiten a los usuarios introducir datos como en un formulario.

- Generación de contenido programado: Con ActionScript podemos generar contenido directamente desde bibliotecas. El contenido generado en forma de programa puede servir como: un elemento estático, un elemento interactivo (por ejemplo una nave en un juego espacial), una opción de un menú que se abre cuando la presionan, etc.
- Comunicación con el servidor: ActionScript provee de una amplia variedad de herramientas para enviar y recibir información del servidor. Ejemplos de comunicación con el servidor son: enlace a una página web, libro de visitas, aplicación de Chat, juegos multijugadores a través de la red, transacción de e-comercio, sitio personalizado con nombre de usuario y contraseña, ect.

La versión más extendida actualmente es ActionScript 3.0, que significo una mejora en el manejo de programación orientada a objetos al ajustarse mejor al estándar ECMA-262 y es utilizada en las últimas versiones de Adobe Flash. ActionScript 3.0 mejora su rendimiento en comparación de sus antecesores (ejecuta hasta 10 veces más rápido), además de incluir nuevas características como el uso de expresiones regulares y nuevas formas de empaquetar las clases. ActionScript 3.0 se ha diseñado para facilitar la creación de aplicaciones muy complejas con conjuntos de datos voluminosos y bases de código reutilizables y orientadas a objetos.

ActionScript 3.0 ofrece un modelo de programación robusto que resultará familiar a los desarrolladores con conocimientos básicos sobre programación orientada a objetos. Algunas de las principales funcionalidades de ActionScript 3.0 son:

- Una nueva máquina virtual ActionScript, denominada AVM2, que utiliza un nuevo conjunto de instrucciones de código de bytes y proporciona importantes mejoras de rendimiento. ActionScript 3.0 permite introducir unas mejoras de rendimiento que sólo están disponibles con AVM2.
- Una base de código de compilador más moderna, que se ajusta mejor al estándar ECMAScript (ECMA 262) y que realiza mejores optimizaciones que las versiones anteriores del compilador.
- Una interfaz de programación de aplicaciones (API) [12] ampliada y mejorada, con un control de bajo nivel de los objetos y un auténtico modelo orientado a objetos.
- Un núcleo del lenguaje basado en el próximo borrador de especificación del lenguaje ECMAScript (ECMA-262).
- Una API XML basada en la especificación de ECMAScript para XML (E4X) (ECMA-357 edición 2). E4X es una extensión del lenguaje ECMAScript que añade XML como un tipo de datos nativo del lenguaje. E4X ofrece un conjunto fluido y natural de construcciones del lenguaje para manipular XML, por lo que se optimiza el desarrollo de aplicaciones que manipulan XML, pues reduce drásticamente la cantidad de código necesario.
- Un modelo de eventos basado en la especificación de eventos DOM (modelo de objetos de documento) de nivel 3.

- Excepciones en tiempo de ejecución. ActionScript 3.0 notifica más situaciones de error que las versiones anteriores de ActionScript. Las excepciones en tiempo de ejecución se utilizan en situaciones de error frecuentes y permiten mejorar la depuración y desarrollar aplicaciones para gestionar errores de forma robusta. Los errores en tiempo de ejecución pueden proporcionar trazas de pila con la información del archivo de código fuente y el número de línea. Esto permite identificar rápidamente los errores.

4.13. Ficheros XML

XML [33], siglas en inglés de Extensible Markup Language (lenguaje de marcas extensible), es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Es una simplificación y adaptación del SGML y permite definir la gramática de lenguajes específicos (de la misma manera que HTML es a su vez un lenguaje definido por SGML). Por lo tanto XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. Algunos de estos lenguajes que usan XML para su definición son XHTML, SVG, MathML, XAML.

XML no nació sólo para su aplicación en Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo y casi cualquier cosa imaginable.

XML es una tecnología sencilla que tiene a su alrededor otras que la complementan y la hacen mucho más grande y con unas posibilidades mucho mayores. Tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil.

Para almacenar los datos de las URIs aleatorias y los datos de los campos de las relaciones, se decidió por el uso de ficheros XML como almacenamiento de información. La elección de guardar la información en ficheros XML se ha hecho por varias razones:

- ActionScript 3.0 implementa ECMAScript for XML (E4X). E4X ofrece un conjunto fluido y natural de construcciones del lenguaje tanto para la lectura como para la escritura de ficheros XML (XML con E4X se comporta como un tipo de datos nativo del lenguaje), facilitando así el almacenamiento y recuperación de datos de estos ficheros, y permitiendo el manejo óptimo de los datos de los ficheros. E4X optimiza el desarrollo de aplicaciones que manipulan XML, pues reduce drásticamente la cantidad de código necesario.
- Facilidad de crear una estructura jerárquica para organizar la información de los componentes de un equipo así como sus estadísticas.
- Por las propiedades de XML:
  - Es ampliable: es posible extender XML con la adición de nuevas etiquetas, de modo que se pueda continuar utilizando sin complicación alguna.
  - La portabilidad de XML es consecuencia de que es el propio desarrollador el que define las etiquetas y los atributos. No se necesitan bibliotecas ni servidores de aplicaciones especiales para leer un documento XML. Los

documentos XML son archivos de texto normal, por lo que no requieren un software propietario para interpretarlos, como ocurre con la mayoría de los archivos binarios.

- Estructura sencilla: si un tercero decide usar un documento creado en XML, es sencillo entender su estructura y procesarla. Mejora la compatibilidad entre aplicaciones.
- El contenido es independiente de la presentación: Sus clientes se concentrarán en usar HTML y CSS para definir el diseño y la presentación, lo cual no se verá afectado por cambios en la información, que se almacena por separado en un archivo XML.

XML proporciona a los programadores la capacidad de ofrecer datos estructurados desde muchas aplicaciones al sistema local con el fin de trabajar localmente con ellos.

4.13.1 Estructura de los ficheros XML.

Como ya se ha mencionado, se utilizarán dos ficheros XML para almacenar las URIs aleatorias (randomNodos.xml) y los datos de los campos de las relaciones (datosRelaciones.xml). La estructura de los ficheros XML será la siguiente:

- **Fichero randomNodos.xml**: en este fichero se han almacenado los datos de URIs validas (100 URIs por entidad) de los distintos tipos de entidades que se pueden encontrar en la aplicación. Su estructura es la siguiente:

```
<randomNodos>
        <personas>
                <persona>0000690003</persona >
                <persona >0015290004</persona >
                ….
        </personas>
        <hogares>
                <hogar>000029 </ hogar >
                < hogar >003812</hogar>
                …..
        </hogares>
        <viviendas>
                < vivienda >000001</ vivienda >
                < vivienda >000069</ vivienda >
                …..
        </viviendas>
        …..
</randomNodos>
```

Modificando este fichero podremos añadir o eliminar nuevas URIs validas aleatorias para las entidades de la aplicación.

- **Fichero datosRelaciones.xml**: en este fichero se han almacenado los datos de los campos de las relaciones establecidas entre dos nodos. Existe un apartado para cada tipo de relación. Su estructura es la siguiente:

```
<datosRelaciones>
        <relacionPersonaPersona>
                <relacion label="Padre" data="PAD"/>
                <relacion label="Madre" data="MAD"/>
                ….
        </relacionPersonaPersona>
        < relacionNucleoPersona>
                <relacion label="Hombre del Nucleo" data="HOM"/>
                <relacion label="Mujer del Nucleo" data="MUJ"/>
        </ relacionNucleoPersona>
        <relacionPersonaLiteral>
                <relacion label="Año de llegada a España" data="ANYOE"/>
                <relacion label="Año de llegada a la C. Autónoma" data="ANYOC"/>
                …..
        </relacionPersonaLiteral>
        …..
        <relacionNucleoLiteral>
                <relacion label="Combinación de nacionalidades" data="COMBNACIN"/>
                <relacion label="Número de hijos" data="NHIJO"/>
                …..
        </relacionNucleoLiteral>
</datosRelaciones>
```

Modificando este fichero podremos añadir o eliminar nuevos campos en las relaciones entre los distintos tipos de nodos.

4.14. Modelo de Implementación.

En este apartado se define el Modelo de Implementación del sistema. Muestra cómo se traduce el Modelo de Diseño en los distintos componentes ejecutables de la aplicación a desarrollar.

El Modelo de Implementación especifica los distintos componentes creados para desarrollar el Modelo de Diseño. Esta versión del Modelo de Implementación se corresponde con la última versión de la aplicación.

A continuación se muestra el diagrama de paquetes que forma el Modelo de Implementación del sistema.

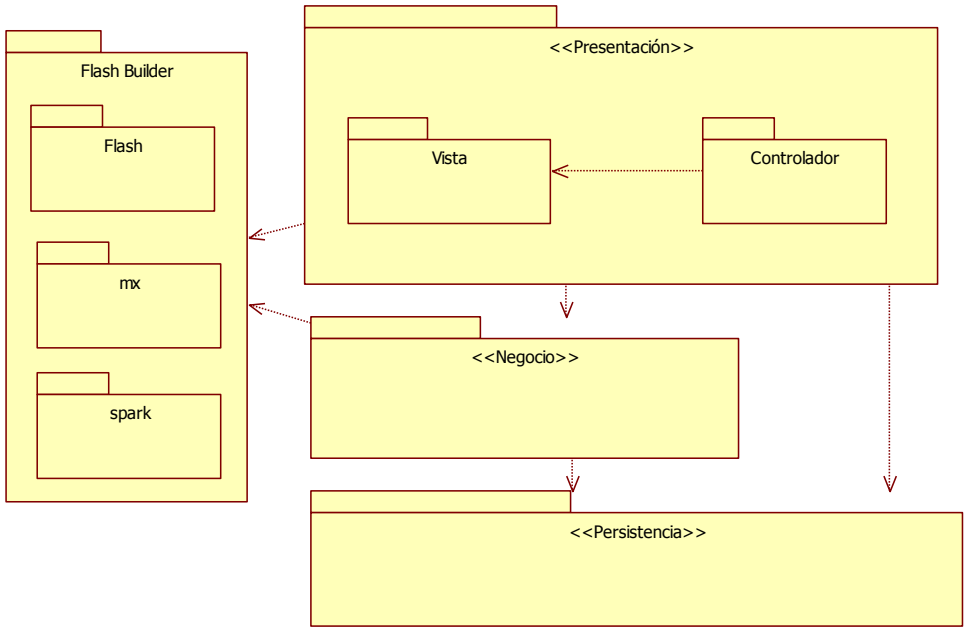

Figura 4.85: Modelo de Implementación

El paquete Flash Builder contiene las librerías Flash, mx y spark las cuales nos han permitido:

- Representar componentes gráficos como pueden ser botones, imágenes, comboBox, paneles, formas geométricas (flecha), etc.
- Capturar y manejar los distintos tipos de eventos (click de ratón, cambio de pestaña, drag and drop, finalización de un servicio http, etc.)
- Realizar peticiones de servicio http, para la realización de las consultas en el servidor gutenberg.dcs.fi.uva.es

4.14.1 Paquete Presentación.

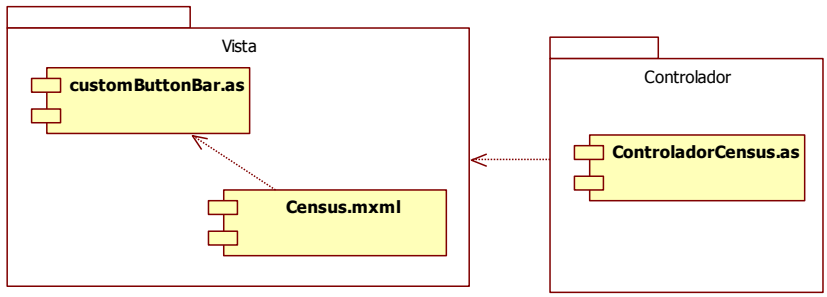

Figura 4.86: Paquete Presentación

La clase custtomButtonBar.as del paquete Vista, es una barra de botones personalizada. Ésta fue implementada a partir de la base obtenida en el blog Flex'n Stuff [16] de Ged Nickson donde muestra como crear una barra de botones personalizada.

4.14.2 Paquete Negocio.

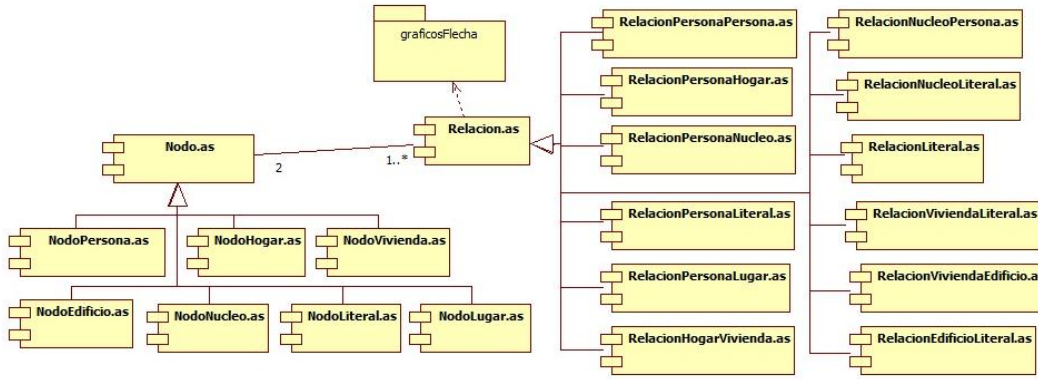

Figura 4.87: Paquete Negocio

La clase Nodo.as y sus descendientes, representan los nodos en la pantalla extendiendo la clase panel de flash. Para implementar estas clases se utilizo de punto de partida la clase SuperPanel.as [32] implementada por Wietse Veenstra en su página web.

Por otro lado se utilizó en paquete graficosFlecha, el cual se compone de un conjunto de librerías que nos permiten dibujar flechas en la pantalla fácilmente. Este paquete fue obtenido de la página web www.doesnotcompute.com de Noel Billig en el articulo “GraphicsUtil. A Utility Class for Drawing Arrows” [22].

Capítulo 5
PRUEBAS

5. PRUEBAS

5.1. Introducción

Uno de los últimos pasos realizados tras terminar la implementación del código fuente, fue la exhaustiva realización de pruebas para asegurarme del correcto funcionamiento de la aplicación. El objetivo de esta fase no es convencerse de que el programa funciona bien, sino ejercitarlo con la peor intención a fin de encontrarle fallos. Para ello he llevado a cabo una serie de casos de prueba diseñados para determinar si los requisitos establecidos durante el diseño del proyecto se han cumplido total o parcialmente. La finalidad de las pruebas es detectar los posibles errores para su corrección.

El desarrollo de las pruebas se ha llevado a cabo desde dos perspectivas diferentes:

- Las **pruebas de caja blanca** son aquellas que se realizan sobre funciones internas de un módulo. Se comprueba así la lógica interna de un programa.

- En las **pruebas de caja negra** no conocemos la implementación del código, sólo la interfaz. Se centran en lo que se espera de un módulo, por ello se denominan pruebas funcionales, y el probador se limita a suministrar datos de entrada y a estudiar la salida.

5.2. Pruebas de caja blanca

Estas pruebas también llamadas estructurales o de caja transparente, se han ido realizando a medida que desarrollaba el código, de forma que cada método, clase y la colaboración entre todas ellas han sido probadas.

A medida que probaba el código he tratado de recorrer todos los posibles caminos de ejecución; he testado las operaciones lógico-aritméticas; he revisado la definición y uso de las variables y finalmente he comprobado los bucles.

5.3. Pruebas de caja negra

Este tipo de pruebas está enfocado a comprobar que los requisitos funcionales se han cumplido. Es decir, la prueba de caja negra permite obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. No son una alternativa a las pruebas de caja blanca sino que se han de realizar las dos por separado y de forma complementaria, con el fin de detectar diferentes tipos de errores.

Para organizar los casos de prueba he realizado su organización en la siguiente tabla:

Prueba	Resultado Obtenido	Valoración
Descripción de la prueba: interacciones, caracterización de los datos de entrada, observaciones a hacer sobre la interfaz de usuario, etc.…	Resultado observado al ejecutar la prueba.	Valoración del resultado de la prueba: - Correcto - Incorrecto - Corregido

Figura 5.1: Ejemplo Tabla de Pruebas

Para realizar estas pruebas se comprobó uno a uno si todos los requisitos funcionales habían sido cubiertos y en segundo lugar han sido probados cada caso de uso:

Ver Ayuda:

Prueba	Resultado Obtenido	Valoración
El usuario pincha en el botón "Ayuda"	El sistema muestra el manual de ayuda de la aplicación en una nueva pestaña.	Correcto

Figura 5.2: Pruebas Ver Ayuda

Añadir Nodo:

Prueba	Resultado Obtenido	Valoración
El usuario pincha en los botones de las entidades de la aplicación.	El sistema crea un nuevo nodo del tipo pulsado y lo añade en la pantalla.	Correcto
El usuario crea una gran cantidad de nodos pulsando en los botones.	El sistema crea los nodos y los añade a la pantalla siempre dentro de los límites.	Correcto
El usuario introduce datos en el campo de variable del nodo.	El sistema muestra los datos en el campo variable y selecciona la opción "Variable"	Correcto
El usuario introduce datos en el campo de URI del nodo.	El sistema muestra los datos en el campo URI y selecciona la opción "URI"	Correcto

Figura 5.3: Pruebas Añadir Nodo

URI Aleatorio:

Prueba	Resultado Obtenido	Valoración
El usuario pincha en el botón "randomURI" de un nodo.	El sistema muestra en el campo URI del nodo una URI correcta.	Correcto

Figura 5.4: Pruebas URI Aleatorio

Comprobar URI:

Prueba	Resultado Obtenido	Valoración
El usuario introduce una URI correcta en el nodo y pincha en el botón "checkURI".	El sistema muestra la imagen de URI correcta.	Correcto
El usuario introduce una URI incorrecta en el nodo y pincha en el botón "checkURI".	El sistema muestra la imagen de URI incorrecta.	Correcto
El usuario pulsa el botón "checkURI" sin introducir ninguna URI.	El sistema muestra la imagen de URI incorrecta.	Corregido. El sistema devolvía un error de consulta mal formulada y no un booleano.
El usuario modifica la URI del nodo.	El sistema muestra la imagen de URI sin comprobar.	Correcto

Figura 5.5: Pruebas Comprobar URI

Eliminar Nodo:

Prueba	Resultado Obtenido	Valoración
El Usuario pulsa el botón de eliminar un nodo.	El sistema elimina el nodo de la pantalla y todas sus relaciones entrantes y salientes.	Correcto

Figura 5.6: Pruebas Eliminar Nodo

Añadir Relación:

Prueba	Resultado Obtenido	Valoración
El usuario arrastra un nodo hacia otro con el que puede establecer una relación.	El sistema crea una nueva relación entre los dos nodos.	Correcto
El usuario arrastra un nodo hacia otro con el que puede establecer una relación.	El sistema no realiza ninguna acción.	Correcto
El usuario selecciona uno de los campos de una relación.	El sistema muestra dicho campo en el comboBox.	Correcto

Figura 5.7: Pruebas Añadir Relación

Eliminar Relación:

Prueba	Resultado Obtenido	Valoración
El usuario pulsa en eliminar una relación.	El sistema pide confirmación del usuario.	Correcto
El usuario confirma la acción de eliminar relación.	El sistema elimina la relación solicitada de la pantalla.	Correcto
El usuario rechaza la acción de eliminar la relación.	El sistema no realiza ninguna acción, por lo que la relación sigue en pantalla.	Correcto

Figura 5.8: Pruebas Eliminar Relación

Seleccionar Resultado:

Prueba	Resultado Obtenido	Valoración
El usuario pulsa en un nodo para seleccionarlo.	El sistema cambia el estado del nodo ha seleccionado, muestra las opciones que se desean seleccionar como resultado y selecciona la variable.	Correcto
El usuario pulsa en un nodo para deseleccionarlo.	El sistema cambia el estado del nodo a seleccionado y oculta las opciones que se desean seleccionar como resultado.	Correcto
El usuario pulsa en una relación para seleccionarla.	El sistema cambia el estado de la relación a seleccionado y selecciona el campo variable.	Correcto
El usuario pulsa en una relación para deseleccionarlo.	El sistema cambia el estado de la relación a no seleccionado.	Correcto
El usuario selecciona las distintas opciones de un nodo seleccionado.	El sistema selecciona la opción escogida. Cuando se solicita la consulta aparece correctamente lo seleccionado.	Correcto

Figura 5.9: Pruebas Seleccionar Resultado

Seleccionar Herramienta:

Prueba	Resultado Obtenido	Valoración
El usuario selecciona la opción "Añadir nodos".	El sistema permitirá añadir nuevos nodos y desplazar los ya existentes por la pantalla.	Correcto
El usuario selecciona la opción "Establecer Relaciones".	El sistema permitirá añadir nuevas relaciones entre nodos.	Correcto
El usuario selecciona la opción "Seleccionar Resultado".	El sistema permitirá seleccionar como resultado los distintos elementos.	Correcto

Figura 5.10: Pruebas Seleccionar Herramienta

Ver Consulta:

Prueba	Resultado Obtenido	Valoración
El usuario pulsa en la pestaña "SPARQL" y ha formado un grafo de consulta correcto.	El sistema muestra la consulta formada por dicho grafo.	Correcto
El usuario pulsa en la pestaña "SPARQL" y ha formado un grafo de consulta incompleto.	El sistema muestra una advertencia sobre la situación del grafo.	Correcto

Figura 5.11: Pruebas Ver Consulta

Modificar Consulta:

Prueba	Resultado Obtenido	Valoración
El usuario modifica la consulta mostrada por el sistema.	El sistema muestra la consulta modificada.	Correcto

Figura 5.12: Pruebas Modificar Consulta

Iniciar Consulta:

Prueba	Resultado Obtenido	Valoración
El usuario pulsa la opción "Iniciar Consulta" con una consulta correcta.	El sistema muestra el resultado en una nueva pestaña.	Correcto
El usuario pulsa la opción "Iniciar Consulta" con una consulta incorrecta.	El sistema muestra un mensaje de que la consulta no se ha podido realizar.	Correcto
El usuario modifica la consulta pulsa la opción "Iniciar Consulta".	El sistema muestra el resultado de la consulta modificada.	Correcto

Figura 5.13: Pruebas Iniciar Consulta

Las pruebas anteriores han sido realizadas sobre la aplicación final, es por ello que el resultado de todas ellas sea el esperado. Sin embargo esto no fue así durante el desarrollo del proyecto. Los resultados incorrectos fueron numerosos, causados principalmente a errores de programación, de diseño, o de planteamiento.

5.4. Pruebas propias de una aplicación Web

Una aplicación Web tiene aspectos comunes con cualquier otro tipo de aplicación, como podría ser una aplicación de escritorio, pero también tiene peculiaridades propias. Es por ello que se le debe dedicar un apartado específico de pruebas.

El contenido de una aplicación Web puede albergar errores, desde simples fallos ortográficos, hasta la violación de las leyes de propiedad intelectual. Es por ello que se debe tener cuidado en la presentación y contenido de la aplicación.

La facilidad de uso es la capacidad de que cualquier usuario pueda realizar las funciones que desarrolla la aplicación sin ningún tipo de dificultad. Es decir, que la navegabilidad a través de nuestras páginas sea sencilla y que el usuario pueda acceder a la selección que desee en cualquier momento.

En mi caso, para llevar a cabo esta prueba, conté con la ayuda de compañeros. Se trata de una aplicación sencilla de usar, pero para realizar una prueba exhaustiva, proporcioné el manual de usuario y a continuación les pedí que realizasen varias pruebas. En la gran mayoría de los casos el resultado fue satisfactorio.

5.4.1 Pruebas de Interfaz de Usuario

Las pruebas realizadas se han agrupado y seleccionado, obteniendo un subconjunto de lo marcado por la normativa ISO 9126-1998 para la evaluación de usabilidad.

- **Correlación entre el sistema y el mundo real**

Se evalúa la adaptación del lenguaje, convenciones y estándares utilizados en la actividad diaria de los usuarios para que sean plasmados en la aplicación de la manera más correcta posible.

Prueba	Test OK	Comentarios
¿Son familiares y descriptivos los iconos utilizados?	Sí	Se han utilizado iconos que se parezcan a lo que representan.
¿Están ordenadas las opciones de menú de forma lógica, proporcionando al usuario los nombres y las distintas tareas disponibles?	Sí	La barra de herramientas sigue la distribución de la creación de un grafo de consulta.

Sí existe una secuencia natural de opciones, ¿se ha utilizado en el sistema?	Sí	La barra de herramientas sigue la distribución de la creación de un grafo de consulta.
¿Se emplea en los comandos y acciones la jerga del usuario con mayor prioridad que la jerga informática?	Sí	
¿Utiliza el sistema automáticamente los separadores (miles, millones,…) para valores superiores a 999?	No	No es necesario ya que no existirán cifras grandes.
¿Se corresponde el uso de los colores en el sistema con el uso común de los códigos de colores?	No	
En los paneles de entrada de datos, ¿se utiliza terminología familiar para el usuario en la descripción de las mismas?	Si	Se utiliza terminología próxima al usuario (contar, agrupar, etc.)
¿Ofrece la interfaz gráfica algún tipo de indicador que muestre de forma obvia la acción (o acciones) que deben ejecutarse a continuación?	No	El sistema no es lineal.

Figura 5.14: Pruebas Correlación entre sistema y mundo real

- **Control y libertad del Usuario**

Se comprueba la flexibilidad de la aplicación en su ejecución, de modo que el usuario pueda moverse a través de la misma con cierto grado de seguridad, autonomía y libertad.

Prueba	Test OK	Comentarios
¿Se pide confirmación al usuario cuando se va a ejecutar algún tipo de comando con consecuencias peligrosas y/o irreversibles?	Parcial	A la hora de eliminar relaciones si se pide confirmación para eliminar nodos no.
¿Existe la función deshacer a nivel de acción, entrada de datos y de grupo de acciones?	Parcial	Solo en la modificación de la consulta se pueden deshacer los cambios (Ctrl +Z)
¿Se permite la función de "cortar" y "pegar" para introducir datos nuevos o modificar datos ya existentes?	Sí	

Cuando se solapan varios paneles, ¿el intercambio entre estas resulta fácil de realizar para el usuario?	Parcial	Podrá pulsar en un panel y este se seleccionará.
¿Se permite la edición del contenido de los campos de entrada de datos?	Sí	
¿Pueden deshacer fácilmente sus acciones los usuarios?	Parcial	Solo pueden deshacer cambios en la consulta SPARQL con Ctrl +Z.

Figura 5.15: Pruebas Control y libertad del Usuario

- **Estándares y consistencia**

Se comprueba que no exista ningún tipo de ambigüedad en el uso de la nomenclatura de los datos e información utilizada que pueda provocar confusión al usuario. Asimismo se examina también la homogeneidad en la presentación de la información y la forma de acceso e introducción de los datos en toda la aplicación.

Prueba	Test OK	Comentarios
¿Se ha evitado en el diseño de pantallas el uso de palabras con todas las letras en mayúscula?	Sí	
¿Están correctamente etiquetados todos los iconos?	Sí	
¿Disponen de título todos los elementos?	Sí	
¿Están todas las etiquetas de las distintas opciones siempre centrados o justificados a la izquierda?	Sí	
¿Se ha evitado el uso de colores de alto contraste y de colores extremos?	Parcial	Suelen ser colores grisáceos excepto los iconos.
¿Se usan los colores llamativos para atraer la atención del usuario?	Parcial	Se usa una distinción de colores, pero no excesivamente llamativos.
¿Es constante la manera de realizar la entrada de datos a lo largo de todos los paneles?	Sí	

Figura 5.16: Pruebas Estándares y consistencia

- **Ayuda al usuario para reconocer, diagnosticar y recuperar la aplicación de errores producidos**

Se verifica la eficacia y suficiencia de los mensajes de error generados por el sistema y como estos permiten al usuario identificar clara y correctamente el error producido y, por tanto, poder actuar en consecuencia para solucionarlo.

Prueba	Test OK	Comentarios
¿Se utiliza algún sonido como señal de error?	No	
¿Son claros y sin ambigüedades los mensajes de error?	Sí	
¿Se ha evitado el uso de signos de exclamación en los mensajes de error?	Sí	
¿Se ha evitado el uso de palabras violentas u hostiles en los mensajes de error?	Sí	
¿Indican o sugieren los mensajes de error cual puede ser la causa del problema?	Sí	
¿Se indica en los mensajes de error que acciones debe realizar el usuario para corregir el error?	Sí	
¿Se utiliza algún método para prevenir errores?	Sí	Se pueden comprobar si son válidas las URI introducidas, y cambios en la imagen del ratón

Figura 5.17: Pruebas reconocer, diagnosticar y recuperar errores

- **Estética y Diseño**

Se evalúa el diseño del interfaz, enfocado a la objetividad y evitando la aparición de información excesiva y/o redundante que no sea de utilidad para el usuario tanto en la interpretación de la información presentada como en la ejecución de acciones a realizar.

Prueba	Test OK	Comentarios
¿Todos los iconos son claramente distinguibles según su significado?	Sí	Las imágenes son suficientemente claras.

¿Se ha resaltado a cada icono del fondo?	Sí	
¿Dispone cada uno de los elementos de entrada de datos de un título o encabezado que las identifique claramente?	Sí	
¿Las etiquetas de los campos utilizan términos familiares y descriptivos?	Sí	

Figura 5.18: Pruebas estética y diseño

- **Ayuda y documentación**

Se recoge la calidad, claridad de contenidos y disponibilidad de la documentación y el soporte de ayuda.

Prueba	Test OK	Comentarios
¿Las instrucciones siguen la secuencia de acciones realizada por el usuario?	Sí	
¿La función de ayuda es visible; p.ej.: una tecla con la palabra HELP o AYUDA?	Sí	Está en la esquina superior derecha.
Para la presentación, ¿está bien diseñada la disposición visual de los objetos en las distintas pantallas?	Sí	
¿El usuario puede cambiar fácilmente entre la ayuda y su trabajo actual?	Sí	Se abre en una nueva pestaña.
¿Resulta fácil el acceso y retorno a y desde el sistema de ayuda?	Sí	Es un cambio de pestaña.
¿Los usuarios pueden continuar su trabajo después de acceder al sistema de ayuda?	Sí	Es un cambio de pestaña.

Figura 5.19: Pruebas Ayuda y documentación

- **Interacción agradable y respetuosa con el usuario**

Se examina el diseño y la estética de la interfaz, primando la utilización de colores suaves y discretos así como la sencillez de las pantallas.

Prueba	Test OK	Comentarios
¿Se ha evitado un detalle excesivo en el diseño de los iconos?	Sí	Son suficientemente descriptivos, pero no sobrecargados
¿Se ha utilizado el color de forma discreta?	Sí	
¿La cantidad de ventanas a gestionar se ha reducido al mínimo posible?	Sí	
¿El color ha sido utilizado de manera específica para captar la atención, comunicar a la organización, indicar cambios de estado y establecer relaciones?	Sí	

Figura 5.19: Pruebas interacción agradable y respetuosa

Capítulo 6
MANUALES DE USUARIO

6. MANUALES DE USUARIO

6.1. Manual de Instalación

6.1.1 Aplicaciones Necesarias

Para poder llevar a cabo una correcta instalación y ejecución de la Aplicación Web, será obligatorio disponer de unas aplicaciones en la máquina reservada para ese fin y que ésta sea accesible vía Internet.

Cabe resaltar la necesidad de tener abierto el puerto 80 del servidor ya que a través de este será por donde se gestionen las peticiones http para la conexión a la aplicación. En este servidor se situarán los archivos de la aplicación, los cuales se podrán encontrar en el CD-Rom adjunto en la carpeta de ejecutables/instalación.

Para el funcionamiento correcto de la aplicación en las maquinas de los clientes será necesario tener instalado el reproductor de Flash "Adobe Flash Player 11". Adobe Flash Player [11] es una aplicación de tiempo de ejecución multiplataforma basada en explorador que ofrece una visualización inmejorable de aplicaciones, contenido y videos atractivos en pantallas y exploradores.

6.1.1.1 Instalar Adobe Flash Player

Para la instalación de la última versión del reproductor Adobe Flash Player, se deberán seguir los siguientes pasos:

1. Acceda a la página web http://get.adobe.com/es/flashplayer/ y pulse en descargar.
   - La página nos mostrará la opción de descarga adecuada para el sistema operativo y navegador con el que estamos accediendo, si se desea otra versión pulse la opción "¿Tiene un sistema operativo o navegador diferentes?" e indique el sistema operativo y navegador deseados.
2. Una vez descargado el archivo de instalación, ejecútelo realizando doble click sobre él para iniciar la instalación.
3. Realice la secuencia de pasos que se le van indicando hasta completar la instalación.
   - Es posible que durante este paso el navegador nos indique si deseamos permitir la instalación, en este caso pulsamos el botón "Permitir".

6.2. Manual de Usuario

6.2.1 Inicio de la aplicación

Al tratarse de una aplicación Web, su inicio es común al de todas las aplicaciones de este tipo. No hay más que abrir una ventana de un navegador Web que utilice el protocolo http, típicamente Explorer, Firefox o Chrome y escribir la dirección correspondiente donde se encuentra alojada, en este caso:

http://jair.lab.fi.uva.es/~ivaloma/Census/Census.html

La aplicación comienza mostrando la pantalla de inicio y otorga el control al usuario.

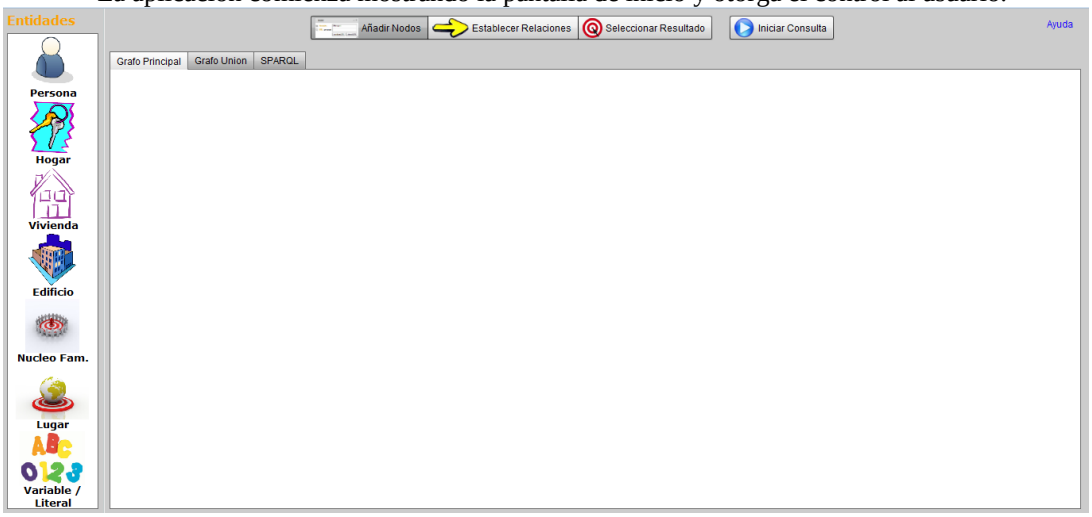

Figura 6.1: Pantalla de Inicio

6.2.2 Usabilidad

Toda la aplicación está desarrollada para un único tipo de usuario. Es decir, cualquier usuario tendrá accesibilidad a toda la funcionalidad de la aplicación, por tanto no haré distinciones durante la explicación.

Esta interfaz gráfica permitirá al usuario crear diferentes grafos de consulta sobre el censo español de 2001, los cuales serán trasformados al lenguaje de consulta SPARQL permitiéndonos así obtener el resultado de dicha consulta.

6.2.2.1 Creación del Grafo de Consulta

Para la creación de un grafo de consulta correcto es necesario seguir los siguientes pasos: Añadir los nodos del grafo, establecer las relaciones entre los nodos añadidos anteriormente y por último seleccionar al menos uno de los elementos que queremos que formen parte del resultado.

Para ello disponemos de una barra de herramientas en la parte central-superior, la cual nos permitirá realizar una u otra acción según la herramienta seleccionada en dicha barra.

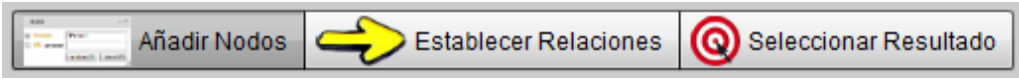

Figura 6.2: Barra de selección de Herramienta.

6.2.2.1.1 Añadir Nodos

Como ya se ha mencionado el primer paso para crear un grafo de consulta es añadir los distintos nodos que formarán el grafo. Para ello habrá que seleccionar la herramienta “**Añadir Nodos**” (1), estar situados en la pestaña “**Grafo Principal**” y pulsar en el tipo de **Entidad** (2) que deseamos añadir al grafo.

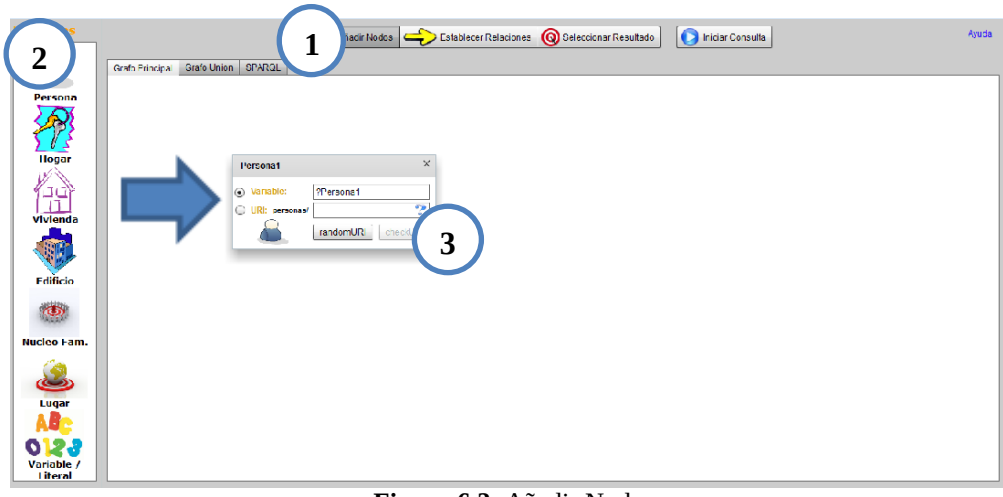

Figura 6.3: Añadir Nodo

Una vez añadido el nodo, podremos desplazarlo por la pantalla arrastrándolo por su cabecera. Además también se podrá establecer el nombre de la variable (debe comenzar siempre por ?) o establecer un identificador (URI) del nodo.

Figura 6.4: Nodo Variable o URI

Cada nodo entidad tiene un tipo de identificador URI:

- **Persona**: es una combinación de 10 dígitos (los 6 primeros identifican el número de hogar y los 4 últimos a la persona dentro del hogar).
- **Hogar**: combinación de 6 dígitos que identifican el número de hogar.
- **Vivienda**: combinación de 6 dígitos que identifican el número de vivienda.
- **Edificio**: combinación de 6 dígitos que identifican el número de edificio.
- **Núcleo Familiar**: es una combinación de 8 dígitos (los 6 primeros identifican el número de hogar y los 2 últimos al número de núcleo dentro del hogar).
- **Lugares y Literales**: el identificador de estas entidades dependerá del predicado establecido entre los nodos, para ver los posibles valores que pueden asignarse como identificador de estas entidades consultar el "Apéndice C".

Se dispone también de dos botones los cuales nos permitirán generar un identificador (URI) aleatorio correcto o revisar si un identificador introducido es válido.

Figura 6.5: URI aleatoria y comprobar URI

Se recomienda comprobar todas los identificadores URI introducidos con el fin de ver si son válidos o no, para así generar un grafo de consulta correcto.

Se podrá eliminar cualquiera de las entidades añadidas pulsando el botón cerrar que tiene cada nodo en su cabecera (**Nota**: esto también eliminará todas las relaciones entrantes o salientes que tenga dicho nodo).

6.2.2.1.2 Establecer Relaciones

Una vez añadidos los nodos necesarios a la pantalla (si es necesario añadir más se podrá repetir el paso anterior) hay que establecer las distintas relaciones entre las entidades añadidas. Para ello habrá que seleccionar la herramienta “**Establecer Relaciones**” (1) y después **arrastrar** uno de los nodos (inicio 2) añadidos hacia otro nodo (fin 3) con el que queramos establecer una relación (4).

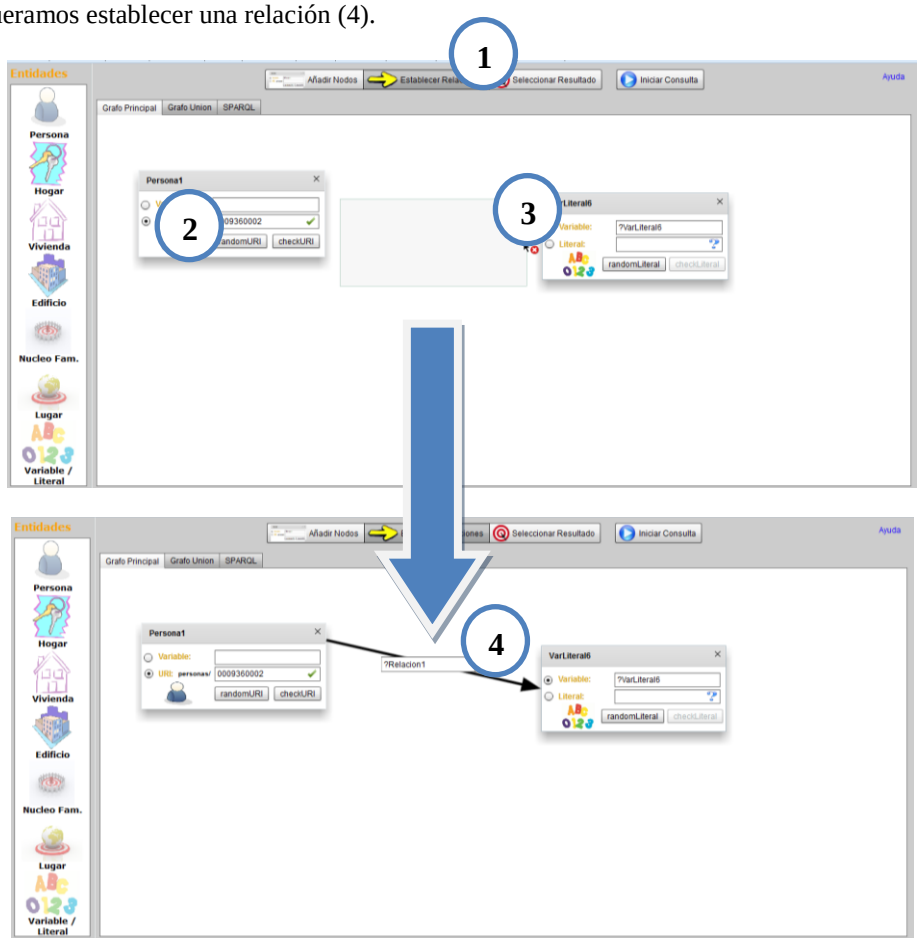

Figura 6.6: Establecer Relaciones.

No todos los nodos (entidades) pueden establecer relaciones entre sí. Las posibles relaciones entre entidades son las siguientes:

- **Entre Persona y**: Persona u Hogar o Núcleo Familiar o Lugar o Literal.
- **Entre Hogar y**: Vivienda o Literal.
- **Entre Vivienda y**: Edificio o Literal.
- **Entre Edificio y**: Literal.
- **Entre Núcleo Familiar y**: Persona o Literal.
- **Los Lugares y Literales**: no podrán iniciar ninguna relación, serán siempre nodos fin en una relación.

Nota: Para facilitar la creación de las relaciones al arrastrar un nodo hacia otro nodo, el ratón se transforma en mientras no se arrastre el ratón hasta un nodo con el que se pueda realizar una relación entre ambos, momento en el cual desaparecerá la del ratón.

Muchas de estas relaciones tendrán campos los cuales se podrán seleccionar mediante el comboBox de la relación o bien dejar la relación como una variable (campo ?Relación).

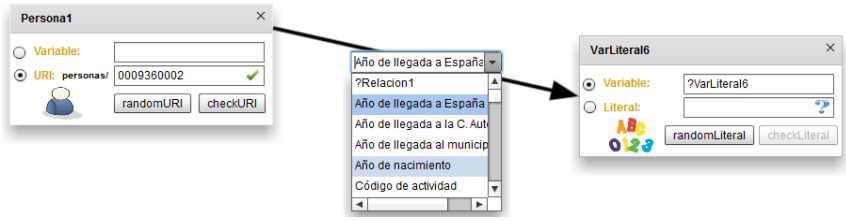

Figura 6.7: Establecer campo de la relación.

Nota: Para generar un grafo correcto todos sus nodos deberán estar interconectados, es decir, no puede existir nodos aislados y tampoco pueden existir ciclos en el grafo.

Para **eliminar** una de las relaciones se deberá pulsar encima de la flecha de la relación cuando el ratón cambie a . Se pedirá confirmación para la eliminación de dicha relación.

6.2.2.1.3 Seleccionar Resultado

Como paso final para generar el grafo de consulta, se necesita seleccionar los elementos que deseamos que formen parte del resultado final. Para realizar esto, se selecciona la herramienta “**Seleccionar Resultado**” y después se **pulsa en la cabecera** de los nodos (2) o en la **flecha** de las relaciones (3) que se deseen como resultado final.

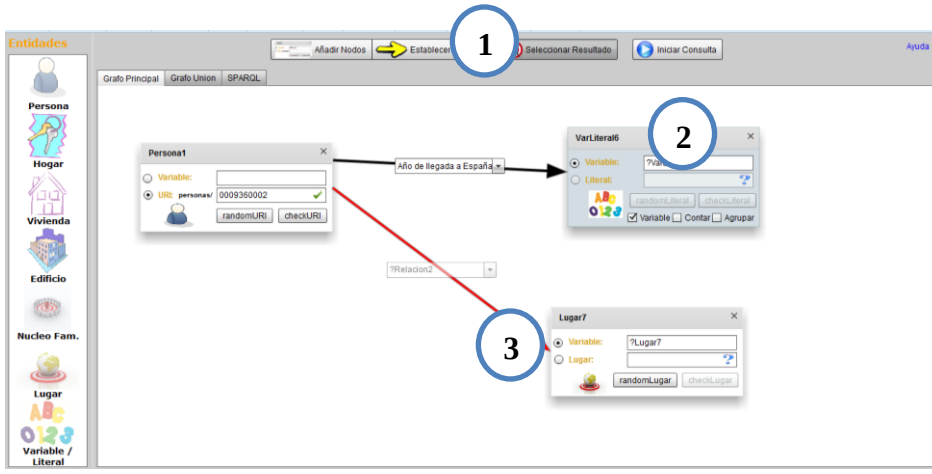

Figura 6.8: Seleccionar Resultado

Una vez seleccionado uno de los nodos se seleccionará su variable y aparecerán las opciones que nos permitirán elegir la forma en que deseamos los resultados.

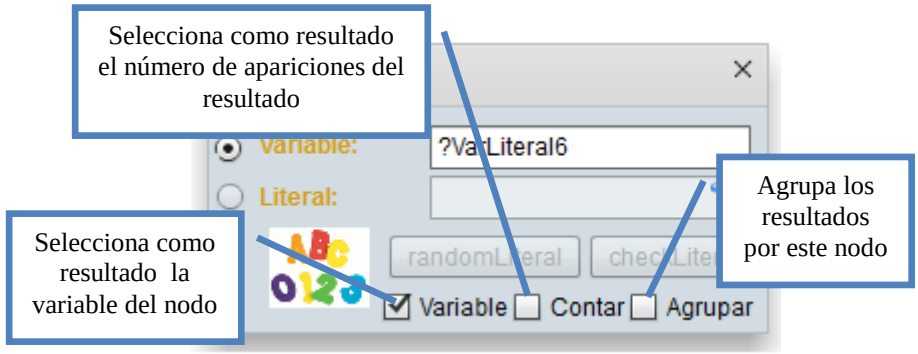

Figura 6.9: Opciones de selección del nodo

En el caso de que la selección sea una relación se seleccionará su variable y esta se mostrará en rojo en la pantalla.

Figura 6.10: Relación Seleccionada

Nota: Si se vuelve a pulsar en un elemento seleccionado este se deseleccionará y volverá a su estado normal.

6.2.2.1.4 Grafo Union

Si se desea realizar una consulta sobre dos grafos disjuntos, se podrá realizar ésta creando un grafo de consulta en la pestaña “Grafo Principal” (1) y otro grafo en la pestaña “Grafo Union” (2) siguiendo el procedimiento explicado anteriormente.

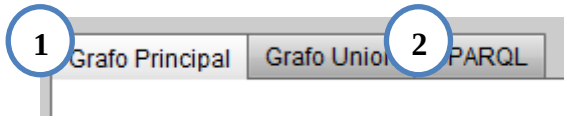

Figura 6.11: Grafo Union

Si se desea crear una consulta UNION entre dos grafos que no sean disjuntos se deberá repetir el nombre de las variables que coincidan entre ambos grafos tanto en el grafo Principal como en el grafo Union.

6.2.2.2 Consulta SPARQL y Resultados

Una vez creado un grafo de consulta correcto se podrá ver la consulta SPARQL asociada a dicho grafo y/o lanzar dicha consulta para obtener los resultados. Para ver la **consulta SPARQL** asociada al grafo creado solo hay que seleccionar la pestaña “SPARQL” (1).

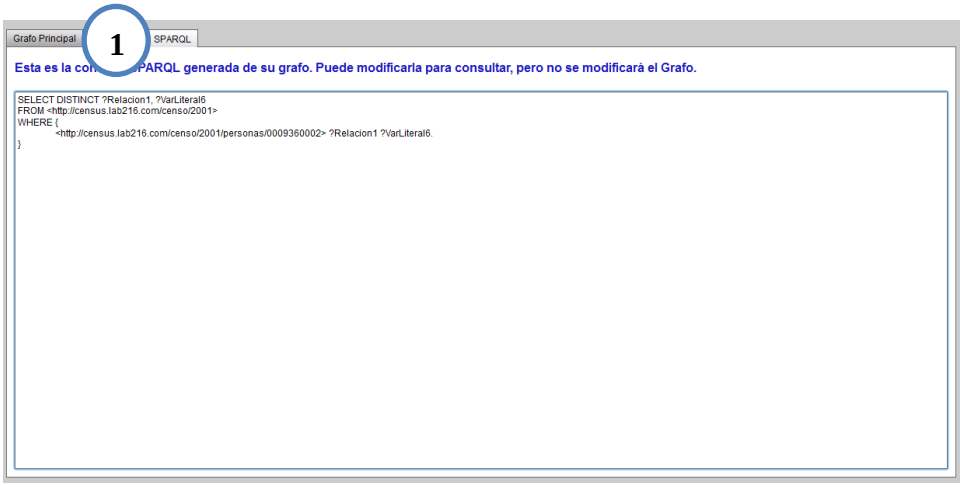

Figura 6.12: Consulta SPARQL asociada al grafo

Esta consulta puede ser modificada por el usuario. No obstante los cambios realizados en la consulta no se verán reflejados en el grafo.

Si se desean obtener los **resultados** de dicha consulta, se deberá pulsar el botón “Iniciar Consulta” (1), situado en la parte superior, y el sistema mostrara los resultados asociados a la consulta en una nueva pestaña (2)

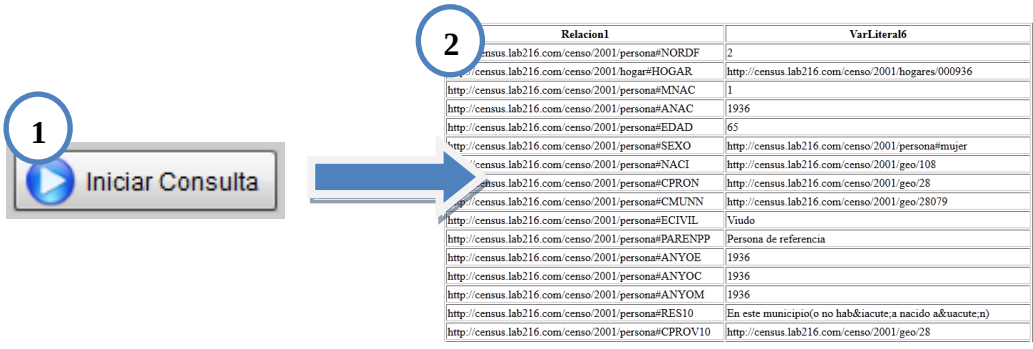

Figura 6.13: Resultados consulta.

No es necesario visionar la consulta para obtener los resultados (puede ser lanzada directamente desde la pestaña del grafo). En caso de que la consulta o el grafo generados no sean correctos el sistema mostrará un mensaje de alerta indicando cual es la posible causa de dicho error.

6.2.2.3 Ejemplos de Consultas

A continuación se muestra el grafo de consulta generado para distintas consultas de ejemplo.

- **CONSULTA 1: Tipo de estudios y número de personas realizándolos**

El grafo de consulta sería el siguiente:

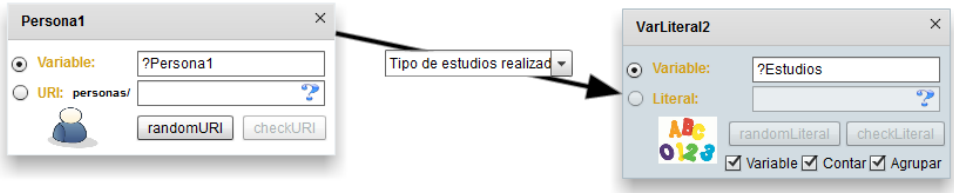

Figura 6.14: Grafo Consulta ejemplo 1.

Como resultado se selecciona la variable (devuelve el nombre de los tipos de estudio), contar (devuelve el número de veces que se repite el resultado) y lo agrupamos para que un mismo resultado solo aparezca una vez.

- **CONSULTA 2: Número de nacimientos por provincia en un determinado año**

El grafo de consulta sería el siguiente:

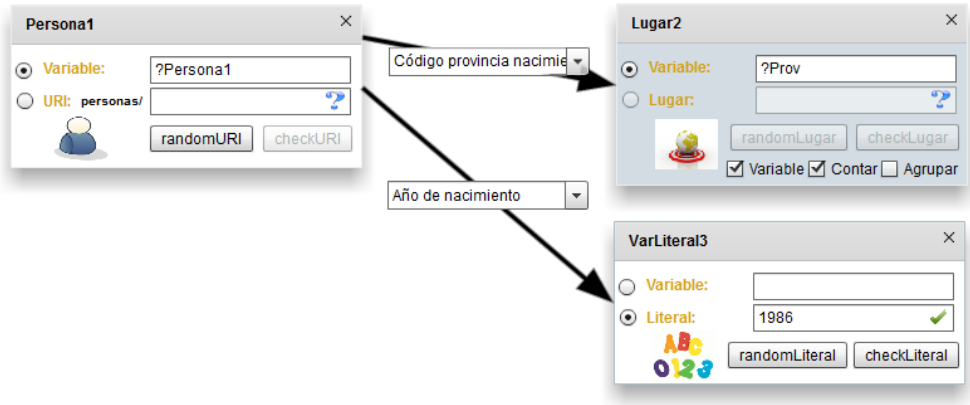

Figura 6.15: Grafo Consulta ejemplo 2.

Como resultado se selecciona la variable (devuelve el nombre de las provincias) y contar (nos devuelve en número). Se ha seleccionado los nacidos en el año 1986 como ejemplo.

- **CONSULTA 3: Toda la información disponible de un determinado Hogar**

El grafo de consulta sería el siguiente:

Figura 6.16: Grafo Consulta ejemplo 3.

Como resultado se selecciona la variable del nodo (devuelve los datos) y la relación con el fin de ver el campo al que pertenecen los datos. En este caso particular, se ha seleccionado los datos del hogar 000499.

- **CONSULTA 4: Viviendas en Valladolid con pocas zonas verdes**

El grafo de consulta sería el siguiente:

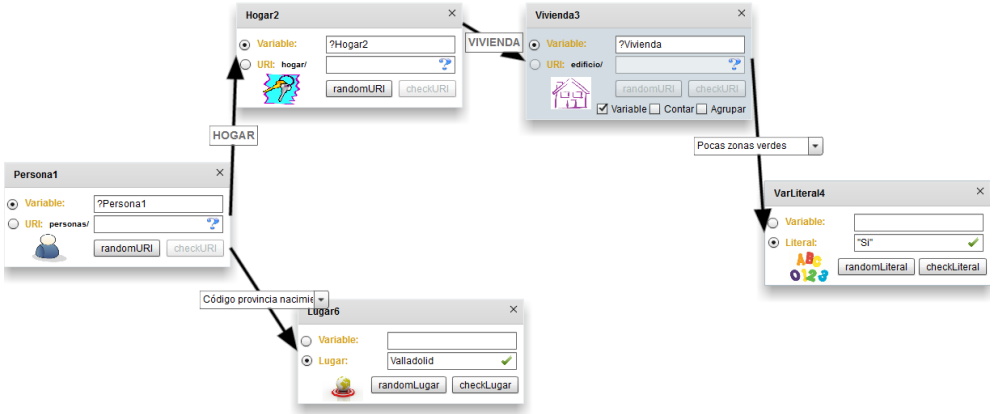

Figura 6.17: Grafo Consulta ejemplo 4.

Como resultado se selecciona la variable para que nos indique el identificador de las viviendas.

- **CONSULTA 5: Vallisoletanos y Palentinos**

En este caso hay que crear dos grafos (Principal y Union) ya que se trata de dos consultas disjuntas. Por lo que los grafos de consulta serán:

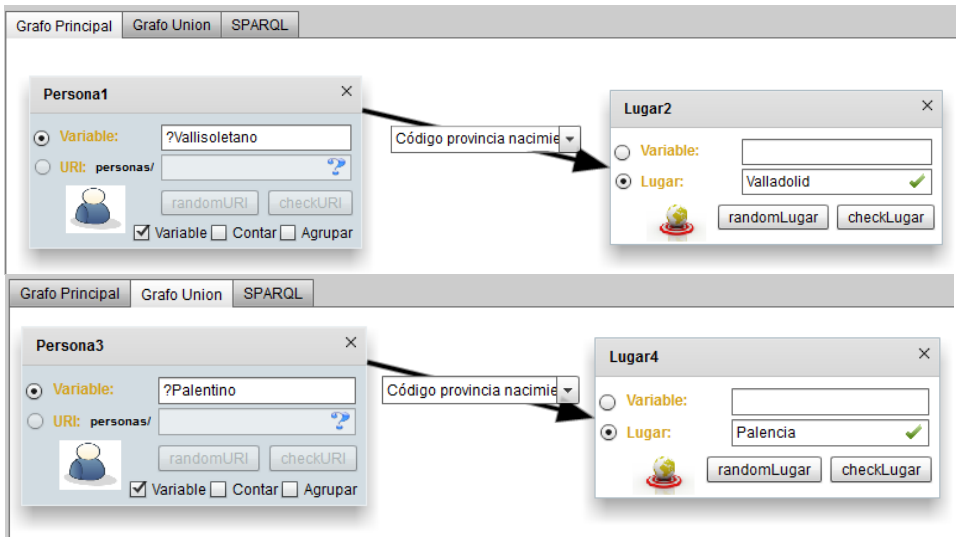

Figura 6.18: Grafo Consulta ejemplo 5.

Debemos seleccionar en cada grafo los nodos persona como resultado. Si deseamos que nos muestre los vallisoletanos y palentinos juntos, habrá que establecer la misma variable para la entidad persona del grafo principal y el grafo unión (ej: ?ValliPalen) y tan solo seleccionar como resultado la entidad del grafo principal.

- **CONSULTA 6: Número de Españoles por edad**

  El grafo de consulta sería el siguiente:

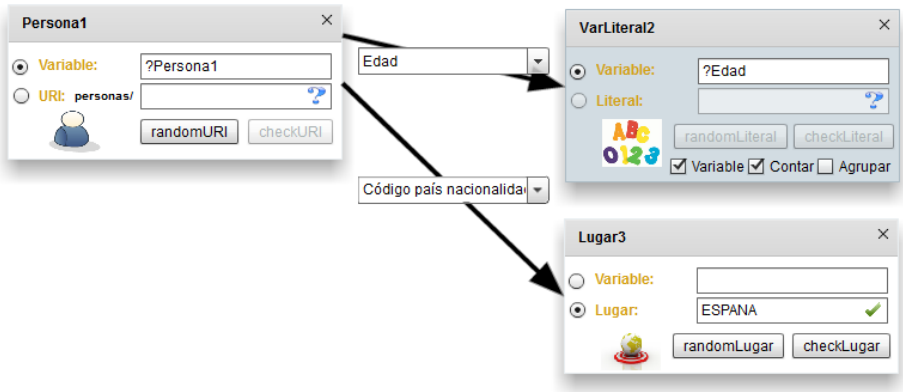

Figura 6.19: Grafo Consulta ejemplo 6.

Se selecciona como resultado la edad y la cuenta de veces que aparece (número de personas con esa edad).

Capítulo 7
CONCLUSIONES

7. CONCLUSIONES

En este capítulo se expondrán las conclusiones derivadas de la realización del presente Trabajo Fin de Grado, las principales dificultades encontradas, los objetivos alcanzados tras el mismo, los conocimientos adquiridos y las posibles mejoras propuestas.

7.1. Dificultades encontradas

A continuación se detallan las principales dificultades que han surgido durante la elaboración y desarrollo del presente Trabajo Fin de Grado:

- Era una de las primeras veces que me afrontaba a un proyecto de gran envergadura, por lo que el desarrollo de las diversas fases del proyecto ha sido costoso, pero se ha superado a un nivel aceptable.
- El dominio del tema del Trabajo fin de grado (web de datos, RDF, proyecto Linked Open Data) era totalmente desconocido para mi. Afortunadamente, con la asistencia al curso "Introducción a la Web de datos" impartido por los tutores de este proyecto, además de su ayuda a la hora de resolverme cualquier duda sobre el tema, el aprendizaje resulto sencillo.
- Además, la programación en ActionScript en el entorno de Adobe Flash Builder me era desconocida. Por suerte, la documentación sobre estos temas es extensa y pude adquirir conocimientos en estas áreas de un modo relativamente sencillo.
- Adobe Flash es un Software de código no libre, lo que me llevó a hacer diferentes suposiciones sobre su funcionamiento interno para el correcto desarrollo de la fase de diseño.
- Otra dificultad encontrada fue las políticas de seguridad de Adobe Flash, ya que está sujeto a las *security sandbox restrictions* de Flash Player, lo cual no permite que una aplicación de un dominio pueda cargar datos de otro dominio si este último no le ha dado los permisos correspondientes. Para solucionar esto, se tuvo que crear un archivo crossdomain.xml en el servidor gutenberg.dcs.fi.uva.es (de donde obtiene los datos necesarios la interfaz de consulta) para permitir el acceso a los datos desde otros dominios.

7.2. Objetivos alcanzados

Con el desarrollo del presente Trabajo Fin de Grado, se han conseguido la gran mayoría de los objetivos propuestos al inicio y que a continuación se detallan:

- Se ha simulado un proceso de trabajo profesional siguiendo todos los pasos de planificación, análisis, diseño e implementación, tratando de cumplir con todos los requisitos establecidos en un período de tiempo establecido. He usado todos los conocimientos adquiridos a lo largo de mi carrera universitaria para la elaboración del Trabajo Fin de Grado, además de los nuevos conocimientos adquiridos sobre el desarrollo de aplicaciones con Adobe Flash Builder, centrándome en el lenguaje de

programación ActionScript, y de los nuevos conocimientos adquiridos sobre la filosofía Link Open Data y la web de datos.

- Se ha desarrollado una interfaz gráfica que permite construir un grafo de consulta SPARQL sobre el Censo español de 2001 en formato RDF, pudiendo consultar la representación SPARQL de dicho grafo. Además, este sistema actúa de intermediario con el sistema que representa la información censal en RDF para ejecutar y devolver el resultado de las consultas.
- Se ha realizado dicho sistema en el entorno de programación de Adobe Flash utilizando el lenguaje de programación Flex y ActionScript.
- Se ha intentado demostrar cómo la gran cantidad de datos generados por las administraciones, que son prácticamente inservibles para terceros, pueden ser convertidos en formatos abiertos, que generar valor no sólo a terceros, sino a la propia administración en tanto que son interoperables,. Dando así un paso muy importante en el fomento del desarrollo de la filosofía de un gobierno abierto.
- He puesto en práctica mis habilidades para el trabajo, y las he mejorado en la medida de lo posible mediante la elaboración de un proyecto completo.
- He aprendido más sobre el concepto de reutilización gracias a la utilización de diferentes paquetes elaborados por otros usuarios, e intentando crear un sistema fácilmente reutilizable y por lo tanto ampliable.

7.3. Conocimientos Adquiridos

En este apartado me gustaría comentar los conocimientos que me ha aportado la realización de este proyecto.

En primer lugar este Trabajo Fin de Grado me ha enseñado a enfrentarme, de nuevo, a una aplicación de cierta envergadura. He tenido que realizar las tareas de planificación, análisis y desarrollo, así como la implementación y las pruebas, dando como resultado el proyecto que aquí se describe. Comprendiendo la dificultad que puede llegar a tener el elaborar un sistema de gran envergadura y el por qué de la poca fiabilidad en cuanto al tiempo de entrega de éstos por parte de la industria.

En segundo lugar la investigación necesaria para llevar a cabo cualquier tarea me ha aportado una serie de conocimientos teóricos de los que carecía. He adquirido una numerosa terminología, filosofías de trabajo, nuevas tecnologías, futuras líneas de desarrollo… indispensable de cara a mi incorporación al mercado laboral.

En tercer lugar cabe destacar los conocimientos adquiridos sobre la web de datos y la filosofía Linked Open Data y de sus respectivas ventajas e inconvenientes, y de cómo su fin último es poder construir aplicaciones más ricas que aprovechen la gran cantidad de datos expuestos. Una de estas posibles aplicaciones es esta interfaz de consulta del censo 2001 en formato RDF cuyo objetivo es proporcionar datos públicos más accesibles a la ciudadanía en un formato reutilizable, generando así valor, transparencia e interoperabilidad. Además, se ha comprobado como la apertura y disponibilidad de estos datos puede crear nuevas oportunidades de negocio al permitir a terceros crear nuevos servicios de valor añadido utilizando los datos públicos de forma integrada.

Y en cuarto y último lugar, la realización del proyecto ha supuesto la adquisición de una gran cantidad de conocimientos prácticos. El aprendizaje de la tecnología Adobe Flash Builder y de su lenguaje de programación asociado ActionScript. Para ello he necesitado conocer su filosofía, los conceptos que engloban, ver ejemplos de utilización, interactuar con la herramienta y finalmente ponerla en práctica. En mi opinión considero el entorno de trabajo Adobe Flash Builder cómo una buena herramienta de trabajo en cuanto a la creación de aplicaciones Web interactivas.

7.4. Posibles Mejoras

Todo proyecto es mejorable, por ello se ha pensado en una serie de ampliaciones que podrían hacerse sobre esta interfaz web de consulta del Censo de 2001. La interfaz podría mejorarse con un diseño mucho más profesional, el código podría sufrir modificaciones buscando una mayor modularidad o la inclusión de diversos patrones, pero las mejoras que me dispongo a exponer están dirigidas a aumentar la funcionalidad de la misma.

Algunas de las funciones que se pueden añadir son:

- Permitir la realización de consultas ASK y CONSTRUCT.
- Permitir que una modificación manual en la consulta SPARQL generada sea reflejada en el grafo de consulta.
- Facilitar una metáfora para implementar las consultas con el modificador OPTIONAL.
- Posibilidad de añadir filtros (de idioma, de tipo de datos, que sea una URI o un literal, aritméticos, etc.) en la creación de la consulta.
- Integración de los resultados en la propia interfaz.
- Posibilidad de creación de gráficas con los resultados de la consulta.

Posibles mejoras sobre el aspecto:

- Mejora de los gráficos, del aspecto de los elementos y de las diferentes animaciones empleadas, ya que en este caso son bastante básicos por falta de tiempo.

Sabemos por lo tanto que es posible que este proyecto pueda dar pie a otros nuevos. Por esta razón, a la hora de escribir el código he procurado utilizar nombres significativos para las variables y métodos, así como comentarios que ayuden a dar a una visión más clara del trabajo realizado a futuros desarrolladores.

Capítulo 8
BIBLIOGRAFÍA

8. BIBLIOGRAFÍA

8.1. Fuentes Bibliográficas

[1] I. Sommerville. "Ingeniería del Software". 7ª ed. Addison-Wesley, 2005.

[2] Ivar Jacobson, Grady Booch, James Rumbaugh. "El Proceso Unificado de Desarrollo Software". Addison Wesley, 1999.

[3] J. Arlow & I. Neustadt. "UML 2". Ed. Anaya Multimedia, 2006

[4] Javier D. Fernández, Miguel A. Martínez-Prieto, Claudio Gutiérrez. "Publishing Open Statistical Data: the Spanish Census". http://dataweb.infor.uva.es/wp-content/uploads/2011/06/dgo11.pdf 12th Annual International Conference on Digital Government Research (dg.o 2011), 2011.

[5] Jeff Tapper, Matthew Boles, James Talbot. "*Adobe Flex*". Ed. Anaya Mulrimedia, 2008. ISBN: 978-84-415-2319-2007

[6] LARMAN, Craig. "UML Y PATRONES. Una introducción al análisis y diseño orientado a objetos y al proceso unificado". 2ª Ed. Madrid: Pearson Educación, 2003.

[7] Pressman, Roger S. "Ingenieria del Software. Un enfoque práctico" (6ª edicion). Ed: Mc Graw Hill 2005

[8] Project Management Institute [PMI]. "A Guide to the Project Management Body of Knowledge". 2004.

8.2. Referencias Web

[9] ActionScript 3.0 [Última visita: 23/08/2012]:
http://es.wikipedia.org/wiki/ActionScript

[10] Adobe Flash Builder [Última visita: 23/08/2012]:
http://www.adobe.com/products/flash-builder.html

[11] Adobe Flash Player [Última visita: 23/08/2012]:
http://www.adobe.com/es/products/flashplayer.html

[12] API ActionScript 3.0 [Última visita: 23/08/2012]:
http://help.adobe.com/en_US/FlashPlatform/reference/actionscript/3/index.html

[13] Asunción Gómez Pérez; "Apúntate al reto de los Datos Enlazados en la Web"; 25/05/2010 [Última visita: 23/08/2012]:
http://www.madrimasd.org/informacionIdi/analisis/analisis/analisis.asp?id=44078#nota2

[14] DBpedia [Última visita: 23/08/2012]:
http://es.dbpedia.org/

[15] DISCO [Última visita: 23/08/2012]:
http://www4.wiwiss.fu-berlin.de/bizer/ng4j/disco/

[16] Ged Nickson, Blog Flex'n Stuff [Última visita: 23/08/2012]:
http://gednickson.wordpress.com/2010/01/23/placing-icons-inside-a-togglebuttonbar/

[17] gFacet [Última visita: 23/08/2012]:
http://www.visualdataweb.org/gfacet.php

[18] Instituto Nacional de Estadística (INE) [Última visita: 23/08/2012]:
http://www.ine.es/

[19] iSPARQL [Última visita: 23/08/2012]:
http://dbpedia.org/isparql/

[20] Javier D. Fernández y Miguel A. Martínez Prieto; "Aprendiendo a nadar en el diluvio de datos", 08/03/2012 [Última visita: 23/08/2012]:
http://dataweb.infor.uva.es/2012/02/27/nuevo-curso-aprendiendo-a-nadar-en-el-diluvio-de-datos/

[21] Linked Data Web [Última visita: 23/08/2012]:
http://linkeddata.org/

[22] Noel Billig, "GraphicsUtil. A Utility Class for Drawing Arrows" [Última visita: 23/08/2012]:
http://www.dncompute.com/blog/2008/07/17/graphicsutil-a-utility-class-for-drawing-arrows.html

[23] Prototipo no funcional de interfaz gráfica [Última visita: 23/08/2012]:
http://gutenberg.dcs.fi.uva.es/~bhscmcyt/SPARQLInterfaz/Census.html

[24] Proyecto Census 2001 RDF [Última visita: 23/08/2012]:
http://census.lab216.com/_index.php?p=index

[25] RDF (Resource Description Framework) [Última visita: 23/08/2012]:
http://www.w3.org/TR/rdf-primer/

[26] SPARQL, Lenguaje de consulta para RDF [Última visita: 23/08/2012]:
http://www.w3.org/TR/rdf-sparql-query/

[27] SPARQL GUI [Última visita: 23/08/2012]:
http://www.dotnetrdf.org/content.asp?pageID=SparqlGUI

[28] Tabulator [Última visita: 23/08/2012]:
http://www.w3.org/2005/ajar/tab

[29] Twinkle [Última visita: 23/08/2012]:
http://www.ldodds.com/projects/twinkle/

[30] UPEDU [Última visita: 23/08/2012]:
http://www.upedu.org/

[31] W3C [Última visita: 23/08/2012]:
http://www.w3.org/

[32] Wietse Veenstra, SuperPanel.as [Última visita: 23/08/2012]:
http://www.wietseveenstra.nl/files/flex/SuperPanel/v1_5/srcview/source/nl/wv/extenders/panel/SuperPanel.as.html

[33] XML [Última visita: 23/08/2012]:
http://www.w3.org/XML/

APÉNDICE

APÉNDICE

A) Software utilizado para el desarrollo del producto

Edición de textos:
- Microsoft Word 2010

Herramientas de Planificación y Gestión de tiempo:
- Microsoft Proyect 2010

Modelado UML:
- StarUML 5.02

Programación Flex/ActionScript:
- Adobe Flash Builder 4.6

Visualización de películas SWF:
- Adobe Flash Player 11

Pruebas de Navegador:
- Microsoft Internet Explorer
- Mozilla Firefox
- Google Chrome

B) Hardware utilizado para el desarrollo del producto

Para la elaboración de este proyecto se han empleado los siguientes recursos hardware:

- Ordenador de sobremesa
- Procesador Intel P4 1.8 GHz.
- 1024 MB de RAM.
- 80 GB HDD.

- Ordenador portátil HP
- Procesador Intel Core i7 1.80 GHz.
- 4 GB de RAM.
- 500 GB HDD.

C) VALORES VÁLIDOS DE LOS URI

Cada nodo entidad tiene un tipo de identificador:

- **Persona**: es una combinación de 10 dígitos (los 6 primeros identifican el número de hogar y los 4 últimos a la persona dentro del hogar).
- **Hogar**: combinación de 6 dígitos que identifican el número de hogar.
- **Vivienda**: combinación de 6 dígitos que identifican el número de vivienda.
- **Edificio**: combinación de 6 dígitos que identifican el número de edificio.
- **Núcleo Familiar**: es una combinación de 8 dígitos (los 6 primeros identifican el número de hogar y los 2 últimos al número de núcleo dentro del hogar).
- **Lugares y Literales**: el identificador de estas entidades dependerá del predicado establecido entre los nodos.

En los siguientes apartados se mostrarán los posibles valores de URI que pueden tener las entidades lugar y literal dependiendo del predicado establecido entre los nodos que componen la relación.

C.1. URIs para Lugares

En este caso solo existen relaciones entre Personas y Lugares por lo que los campos posibles y los posibles valores, dependiendo del campo, de lugar son los siguientes:

- **Código país nacionalidad: NACI →** Ver diccionario de países [Países].
- **Código provincia nacimiento: CPRON →** Ver diccionario de provincias [Provincias].
- **Código o tamaño municipio (o país) de nacimiento: CMUNN →** Ver diccionario de municipios [Municipios].
- **Código provincia anterior de residencia: CLPRO →** Análogo a CPRON.
- **Código o tamaño municipio (o país) anterior de residencia: CLMUNP →** Análogo a CMUNN.
- **Código provincia hace 10 años: CPROV10 →** Análogo a CPRON.
- **Código o tamaño municipio (o país) hace 10 años: CMUN10 →** Análogo a CMUNN.
- **Código provincia de trabajo o estudio: CLPROV →** Análogo a CPRON.
- **Código o tamaño municipio (o país) de trabajo o estudio: CLMUNI →** Análogo a CMUNN.
- **Código provincia 2ª vivienda: CPROSV →** Análogo a CPRON.
- **Código o tamaño municipio (o país) 2ª vivienda: CMUNSV →** Análogo a CMUNN.

C.2. URIs para Literales

En este caso los campos posibles y los posibles valores, dependiendo del campo, de los literales dependerán del tipo de relación establecida, es decir, del tipo de entidad (persona, hogar, vivienda, etc.) de la cual queremos saber un dato. Por ello se distinguirán en los siguientes apartados:

C.2.1 Relación Persona-Literal

Los campos posibles y los posibles valores, dependiendo del campo, del literal en este tipo de relación son los siguientes (las cadenas de caracteres deberán ir entre comillas ""):

- **Año de llegada a España: ANYOE** → 1990, 1991,……., 2001
- **Año de llegada a la C. Autónoma: ANYOC** → 1990, 1991,……., 2001
- **Año de llegada al municipio: ANYOM** → 1990, 1991,……., 2001
- **Año de nacimiento: ANAC** → 1990, 1991,……., 2001
- **Código de actividad: COACTI** → Ver diccionario de actividades [Actividades].
- **Código de ocupación: COCUP** → Ver diccionario de ocupaciones [Ocupación].
- **Condición dentro del núcleo: CNUCF** →
  - "No pertenece a un núcleo familiar"
  - "Pertenece a un núcleo familiar como hijo/a"
  - "Pertenece a un núcleo familiar como padre/madre o pareja"
- **Condición socioeconómica: CSE** →
  - "Contramaestres y capataces de establecimientos no agrarios."
  - "Directores y gerentes de establecimientos no agrarios, altos funcionarios de la administración pública, comunidades autónomas y corporaciones locales"
  - "Directores y jefes de expotaciones agrarias"
  - "Empresarios agrarios con asalariados."
  - "Empresarios agrarios sin asalariados."
  - "Empresarios no agrarios con asalariados"
  - "Empresarios no agrarios sin asalariados"
  - "Miembros de cooperativas agrarias"
  - "Miembros de cooperativas no agrarias."
  - "No clasificables por condicion socioeconómica"
  - "Operarios cualificados y especializados de establecimientos no agrarios"
  - "Operarios sin especialización de establecimientos no agrarios."
  - "Profesionales de las fuerzas armadas."
  - "Profesionales en ocupaciones exclusivas de la administración pública"

  - "Profesionales, técnicos y asimilados que ejercen su actividad por cuenta ajena"
  - "Profesionales, técnicos y asimilados que ejercen su actividad por cuenta propia, con o sin asalariados"
  - "Resto de trabajadores de explotaciones agrarias"
  - "Resto del personal administrativo y comercial"
  - "Resto del personal de los servicios"

- **Conocimiento de la lengua propia: CONLEN →**
  - "Lo entiende y lo sabe hablar y leer, pero no escribir"
  - "Lo entiende y lo sabe hablar, leer, y escribir"
  - "No lo entiende"
  - "Sólo lo entiende"
  - "Sólo lo entiende y lo sabe hablar"
  - "Sólo lo entiende y lo sabe leer"

- **Convivencia con pareja e hijos: CONPARHIJ →**

- **Edad: EDAD →** 0,1,…., 101

- **Estado civil: ECIVIL →**
  - "Soltero"
  - "Casado"
  - "Viudo"
  - "Separado"
  - "Divorciado"

- **Estudios en curso 1: ESCUR1 →**
  - "Enseñanzas iniciales para adultos"
  - "Programas de Garantía Social"
  - "ESO, Educación Secundaria para adultos"
  - "Bachillerato, BUP, COU"
  - "Enseñanzas Artísticas de Grado Elemental o Medio"
  - "Formación Profesional de Grado Medio o equivalente"
  - "Formación Profesional de Grado Superior , FPII o equivalente"
  - "Diplomatura universitaria, Arquitectura o Ingeniería Técnica o equivalente"
  - "Licenciatura universitaria, Arquitectura, Ingeniería o equivalente"
  - "Estudio de posgrado, máster, MIR o análogo"
  - "Doctorado"
  - "Escuela Oficial de Idiomas"
  - "Curso del INEM, Escuela Taller"
  - "Curso de formación promovido por la empresa"
  - "Otros cursos(Informática, preparación de oposiciones..)"

- **Estudios en curso 2: ESCUR2 →** Análogo a ESCUR1.

- **Estudios en curso 3: ESCUR3 →** Análogo a ESCUR1.

- **Forma de convivencia en personas de 16 a 64 años: FCONADU →**
  - "Solos"
  - "Sin pareja ni hijos y con alguno de sus padres"
  - "Con su pareja, sin hijos ni padres"
  - "Con su pareja y 1 hijo, sin padres"
  - "Con su pareja y 2 hijos, sin padres"
  - "Con su pareja y tres hijos, sin padres"
  - "Con su pareja y 4 ó más hijos, sin padres"
  - "Con su pareja y/o algún hijo y alguno de sus padres"
  - "Sin pareja ni hijos y sin padres"

- **Forma de convivencia en personas de 65 años ó más: FCONANC →**
  - "En un establecimiento colectivo"
  - "Solos"
  - "Sólo con su pareja"
  - "Con una hija"
  - "Con un hijo"
  - "Con 2 ó más hijos"
  - "Otra forma"

- **Lugar de trabajo o estudio: LUGTE →**
  - "Domicilio propio"
  - "Varios municipios"
  - "Mismo municipio al de residencia"
  - "Distinto municipio"
  - "En otro país"

- **Medio de desplazamiento 1: MDESP1 →**
  - "Coche particular o furgoneta, conduciendo"
  - "Coche particular o furgoneta, de pasajero"
  - "Autobús"
  - "Metro"
  - "Moto"
  - "Andando"
  - "RENFE"
  - "Otros trenes"
  - "Bicicleta"
  - "Otros medios"

- **Medio de desplazamiento 2: MDESP2 →** Análogo a MDESP1.

- **Mes de nacimiento: MNAC →** 1, 2, 3,…., 12

- **Nivel de estudios: ESREAL →**
  - "Sin estudios"
  - "Primer grado"
  - "ESO, EGB, Bachillerato Elemental"
  - "Bachillerato Superior"
  - "FP Grado Superior"
  - "Diplomatura"
  - "Licenciatura"
  - "Doctorado"

- **Nº de la persona dentro del hogar: NORDF →** 1, 2, 3, 4, …..

- **Nº de viajes diarios: NVIAJE →**
  - "Ninguno"
  - "Uno"
  - "Dos o más diarios"

- **Número de familia dentro del hogar: FAM →** 0, 1, 2

- **Parentesco con la persona de referencia: PARENPP →**
  - "Persona de referencia"
  - "Cónyuge"
  - "Pareja"
  - "Hijo/a de la persona de referencia y/o de su pareja"
  - "Yerno / Nuera"
  - "Hermano/a"
  - "Hermano de la pareja"
  - "Pareja del hermano"
  - "Padre/Madre"
  - "Suegro/a"
  - "Otro pariente con apellido común"
  - "Otro pariente sin apellido común"
  - "No emparentado"

- **Residencia hace 10 años: RES10 →**
  - "En este municipio(o no había nacido aún)"
  - "En otro municipio o país"

- **Sexo: SEXO →**
  - varon
  - mujer

- **Situación la semana pasada (respuesta primera): SITU1 →**
  - "Ocupados"
  - "Parados buscando el primer empleo"
  - "Parados que han trabajado antes"
  - "Estudiantes"
  - "Pensionistas de invalidez"

  - "Pensionistas de viudedad u orfandad"
  - "Pensionistas de jubilación"
  - "Realizando o compartiendo las tareas del hogar"
  - "Otra situación"

- **Situación la semana pasada (segunda): SITU2 →** Análogo a SITU1.

- **Situación la semana pasada (tercera): SITU3 →** Análogo a SITU1.

- **Situación preferente: SITU →** Análogo a SITU1.

- **Situación profesional: SITP →**
  - "Empresario o profesional que emplea personal"
  - "Empresario o profesional que no emplea personal"
  - "Trabajador por cuenta ajena con carácter fijo o indefinido"
  - "Trabajador por cuenta ajena con carácter eventual, temporal.."
  - "Otra situación (Ayuda familiar)"
  - "Otra situación (Miembro de cooperativas)"

- **Tiempo de desplazamiento: TDESP →**
  - "Menos de 10 minutos"
  - "Entre 10 y 20 minutos"
  - "Entre 20 y 30 minutos"
  - "Entre 30 y 45 minutos"
  - "Entre 45 minutos y 1 hora"
  - "Entre 1 hora y hora y media"
  - "Más de hora y media"

- **Tiempo usual de trabajo: TTRAB →** 1, 2, 3, 4, ….., 99

- **Tipo de estudios realizados: TESTUD →**
  - "Derecho"
  - "Magisterio, Educación infantil.."
  - "Ciencias Sociales"
  - "Artes y Humanidades"
  - "Informática"
  - "Ingenierías"
  - "Formación Técnica e Industrias"
  - "Ciencias"
  - "Arquitectura o Construcción"
  - "Agricultura, Ganadería, Pesca; Veterinaria"
  - "Salud, Servicios Sociales"
  - "Otros Servicios"

- **Tipo de hijo: TIHI →**
  - "El padre emparejado"
  - "El padre no emparejado"

- "El padre y la madre emparejados entre si"
- "La madre emparejada"
- "La madre emparejada con otra persona y el padre no emparejado"
- "La madre no emparejada"
- "Ni el padre ni la madre están emparejados"

C.2.2 Relación Hogar-Literal

Los campos posibles y los posibles valores, dependiendo del campo, del literal en este tipo de relación son los siguientes (las cadenas de caracteres deberán ir entre comillas ""):

- **Edad de la generación más antigua: EDADGA →** 18, 19, 20, …..
- **Edad de la generación más reciente: EDADGR →** 0, 1, 2, 3, ……
- **Estructura (definición de 1991): ESTHOG91 →**
  - "De un varón de 15 a 64 años"
  - "De una mujer de 15 a 64 años"
  - "De un varón de 65 y más años"
  - "De una mujer de 65 y más años"
  - "De dos personas de 15 a 64 años"
  - "De dos personas, una al menos de 64 años"
  - "De 1 varón con 1 o más niños menores 15 años"
  - "De 1 mujer con 1 o más niños menores 15 años"
  - "De 2 adultos con 1 niño menor 15 años"
  - "De 2 adultos con 2 niños menores 15 años"
  - "De 2 adultos con 3 niños menores 15 años"
  - "De 2 adultos con 4 o más niños menores 15 años"
  - "De 3 o más adultos con 1 o más niños menores 15 años"
  - "Otro tipo de hogar"
- **Estructura (definición nueva): ESTHOG →**
  - "Una mujer de 16 a 64"
  - "Un hombre de 16 a 64"
  - "Una mujer de 65 o mas"
  - "Un hombre de 65 o mas"
  - "Una mujer con uno o mas menores"
  - "Un hombre con uno o mas menores"
  - "Dos adultos de 16 a 64, sin menores"
  - "Dos adultos, uno al menos de 65 o mas, sin menores"
  - "Dos adultos y un menor"
  - "Dos adultos y dos menores"
  - "Dos adultos y tres o mas menores"
  - "Dos adultos de 35 o más, uno de 16 a 35, sin menores"
  - "Dos adultos de 35 o más, uno de 16 a 35 y un menor"
  - "Dos adultos de 35 o más, uno de 16 a 35 y dos o mas menores"
  - "Otro hogar de tres adultos, con o sin menores"

  - "Dos adultos de 35 o más, dos de 16 a 35, sin menores"
  - "Dos adultos de 35 o más, dos de 16 a 35 y un menor"
  - "Dos adultos de 35 o más, dos de 16 a 35 y dos o mas menores"
  - "Otro hogar de cuatro adultos, con o sin menores"
  - "Cinco o mas adultos, con o sin menores"

- **Extranjeras: EXTAS →** 0, 1, 2, 3, 4, 5, 6, 7
- **Extranjeros: EXTOS →** 0, 1, 2, 3, 4, 5, 6, 7
- **Hombres menores de 15: HM15 →** 0, 1, 2, 3, …
- **Hombres de 15 a 24 años: H1524 →** 0, 1, 2, 3,…
- **Hombres de 15 años: H15 →** 0, 1, 2, 3,…
- **Hombres de 25 a 34 años: H2534 →** 0, 1, 2, 3,…
- **Hombres de 35 a 64 años: H3564 →** 0, 1, 2, 3,…
- **Hombres de 65 a 84 años: H6584 →** 0, 1, 2, 3,…
- **Hombres de 85 ó más: H85 →** 0, 1, 2, 3,…
- **Mujeres menores de 15: MM15 →** 0, 1, 2, 3,…
- **Mujeres de 15 a 24 años: M1524 →** 0, 1, 2, 3,…
- **Mujeres de 15 años: M15 →** 0, 1, 2, 3,…
- **Mujeres de 25 a 34 años: M2534 →** 0, 1, 2, 3,…
- **Mujeres de 35 a 64 años: M3564 →** 0, 1, 2, 3,…
- **Mujeres de 65 a 84 años: M6584 →** 0, 1, 2, 3,…
- **Mujeres de 85 o más: M85 →** 0, 1, 2, 3,…
- **Nº de ocupados del hogar: NOCU →** 0, 1, 2, 3,…
- **Nº de parados del hogar: NPARA →** 0, 1, 2, 3,…
- **Nº de personas en familia: NPFAM →** 0, 1, 2, 3,…
- **Nº de personas en núcleo tipo 1: NNUCF1 →** 0, 1, 2, 3,…
- **Nº de personas en núcleo tipo 2: NNUCF2 →** 0, 1, 2, 3,…

- **Nº de personas en núcleo tipo 3: NNUCF3 →** 0, 1, 2, 3,…
- **Nº de personas en núcleo tipo 4: NNUCF4 →** 0, 1, 2, 3,…
- **Número de familias: NFAM →** 0, 1, 2, 3,…
- **Número de generaciones: NGENER →** 0, 1, 2, 3,…
- **Número de núcleos: NNUCH →** 0, 1, 2, 3,…
- **Tamaño del hogar: NMIEM →** 0, 1, 2, 3,…
- **Tipo de hogar: TIPOHOG →**
  - "Hogares unipersonales"
  - "No forman familia"
  - "Una familia, sin otras personas: Madre con hijos"
  - "Una familia, sin otras personas: Padre con hijos"
  - "Una familia, sin otras personas: Pareja con hijos"
  - "Una familia, sin otras personas: Pareja sin hijos"
  - "Una familia, sin otras personas: Sin núcleo"
  - "Una familia, con otras personas no emparentadas: Madre con hijos"
  - "Una familia, con otras personas no emparentadas: Padre con hijos"
  - "Una familia, con otras personas no emparentadas: Pareja con hijos"
  - "Una familia, con otras personas no emparentadas: Pareja sin hijos"
  - "Una familia, con otras personas no emparentadas: Sin núcleo"
  - "Dos o más familias con otras personas no emparentadas"
  - "Dos o más familias sin otras personas"
  - "Dos o más núcleos con otras personas emparentadas"
  - "Dos o más núcleos con otras personas emparentadas"
  - "Dos o más núcleos sin otras personas emparentadas"
  - "Un núcleo con otras personas emparentadas: Madre con hijos"
  - "Un núcleo con otras personas emparentadas: Padre con hijos"
  - "Un núcleo con otras personas emparentadas: Pareja con hijos"
  - "Un núcleo con otras personas emparentadas: Pareja sin hijos"

C.2.3 Relación Vivienda-Literal

Los campos posibles y los posibles valores, dependiendo del campo, del literal en este tipo de relación son los siguientes (las cadenas de caracteres deberán ir entre comillas ""):

- **Altura (en nº de plantas): ALTURA →** 0, 1, 2, 3,…
- **Año llegada a la vivienda: ANYORES →** 1990, 1991,……., 2001
- **Calefacción: CALE →**
  - "Con calefacción colectiva"
  - "Con calefacción individual"

- "Sin instalación pero con aparatos que permiten calentar alguna habitación"
- "Sin ningún medio"

- **Clase de colectivo: CCOLF →**
  - "Hoteles, pensiones, albergues.."
  - "Colegios mayores, residencias de estudiantes"
  - "Residencias de trabajadores"
  - "Internados, academias y escuelas militares, seminarios.."
  - "Hospitales generales y especiales de corta estancia"
  - "Hospitales psquiátricos"
  - "Hospitales de larga estancia"
  - "Asilos o residencias de ancianos"
  - "Instituciones para personas con discapacidades"
  - "Albergues para marginados sociales"
  - "Otras instituciones de asistencia social a la infancia, juventud.."
  - "Instituciones religiosas(monasterios, abadías..)"
  - "Establecimientos militares(cuarteles..)"
  - "Instituciones penitenciarias (cárceles, reformatorios..)"
  - "Otro tipo de colectivo"

- **Clase de vivienda: CVIVIF →**
  - "Convencionales"
  - "Alojamientos"
  - "Vacías"
  - "Otro tipo"
  - "Viviendas colectivas"

- **Combustible para calefacción: COMBU →**
  - "Gas"
  - "Electricidad"
  - "Petróleo o derivados"
  - "Madera"
  - "Carbón o derivados"
  - "Otros"

- **Contaminación: CONTA →** "Si" o "No"

- **Delincuencia en la zona: DELIN →** "Si" o "No"

- **Días al año de uso de 2ª vivienda: DIANYO →** 0, 1, 2, 3,…, 365

- **Disp. vehículos a motor (no motos): VEHIC →**
  - "No disponen"
  - "1 solo vehículo"
  - "2 vehículos"
  - "3 o más"
- **Disponibilidad segunda vivienda: SVIV →** "Si" o "No"

- **Falta de servicios: SERVI →** "Si" o "No"

- **Localización segunda vivienda: LOCSV →**
  - "En el mismo municipio"
  - "En otro municipio"
  - "En otro país"

- **Malas comunicaciones: COM →** "Si" o "No"

- **Nº de habitaciones: NHAB →** 0, 1, 2, 3,…

- **Poca limpieza en las calles: LIMP →** "Si" o "No"

- **Pocas zonas verdes: VERDE →** "Si" o "No"

- **Refrigeración: REF →** "Si" o "No"

- **Régimen de tenencia: TENEN →**
  - "Cedida gratis o a bajo precio por otro hogar, la empresa.."
  - "En alquiler"
  - "En propiedad por compra, totalmente pagada"
  - "En propiedad por compra, con pagos pendientes(hipotecas…)"
  - "En propiedad por herencia o donación"
  - "Otra forma"

- **Ruidos exteriores: RUIDEX →** "Si" o "No"

- **Superficie útil: SUT →** 15, 16, 17, 18,….

C.2.4 Relación Edificio-Literal

Los campos posibles y los posibles valores, dependiendo del campo, del literal en este tipo de relación son los siguientes (las cadenas de caracteres deberán ir entre comillas ""):

- **Accesibilidad del edificio: ACCESIB →** "Si" o "No"

- **Agua caliente central: ACAL →** "Si" o "No"

- **Agua corriente: AGUA →**
  - "Abastecimiento público"
  - "Abastecimiento privado"
  - "No tiene agua corriente"

- **Año de construcción: CONST →**
  - "Antes de 1900"
  - "1900-1920"
  - "1921-1940"
  - "1941-1950"
  - "1951-1960"

  - "1961-1970"
  - "1971-1980"
  - "1981-1990"
  - "1991-2001"

- **Año exacto: ANYO →** 1991, 1992, 1993,…,2001

- **Clase de propietario: PROP →**
  - "Una persona"
  - "La comunidad"
  - "Una sociedad"
  - "Un organismo público"

- **Estado del edificio: ESTADO →**
  - "Ruinoso"
  - "Malo"
  - "Deficiente"
  - "Bueno"

- **Evacuación de aguas residuales: RESID →**
  - "Alcantarillado"
  - "Otro tipo"
  - "No tiene"

- **Garaje: GARAJE →** "Si" o "No"

- **Gas: GAS →** "Si" o "No"

- **Nº de plantas sobre rasante: PLANTAS →** 1, 2, 3, 4, 5,…

- **Nº de plantas bajo rasante: PLANTAB →** 0, 1, 2, 3,…

- **Nº de plazas de garaje: PLAZAS →** 1, 2, 3, 4, 5,…

- **Portería: PORTE →**
  - "Sólo automático"
  - "Sólo encargado"
  - "Ambos"
  - "No tiene"

- **Tendido telefónico: TELEF →** "Si" o "No"

- **Tipo de edificio: TIPOED →**
  - "Edificio sólo con una vivienda familiar"
  - "Edificio sólo con varias viviendas familiares"
  - "Edificios principalmente con viviendas familiares compartido con locales"
  - "Edificios principalmente con vivienda colectiva:hotel, albergue, pensión"

- "Edificios principalmente con vivienda colectiva:convento, cuartel, prisión"
- "Edificios principalmente con vivienda colectiva:instituciones de enseñanza, internados de enseñanzas medias, academias militares"
- "Edificios principalmente con vivienda colectiva:hospitales en general, instituciones para discapacitados, marginados.."
- "Edificios principalmente con locales compartidos con alguna vivienda"

C.2.5 Relación Núcleo Familiar-Literal

Los campos posibles y los posibles valores, dependiendo del campo, del literal en este tipo de relación son los siguientes (las cadenas de caracteres deberán ir entre comillas ""):

- **Combinación de nacionalidades: COMBNACIN →**
  - "Todos españoles"
  - "Españoles y extranjeros de la Unión Europea"
  - "Españoles y extranjeros, alguno no de la Unión Europea"
  - "Extranjeros de la misma nacionalidad, de la Unión Europea"
  - "Extranjeros de la misma nacionalidad, no de la Unión Europea"
  - "Extranjeros de dos o más nacionalidades, todas de la Unión Europea"
  - "Extranjeros de dos o más nacionalidades, alguna no de la Unión Europea"

- **Indicador de familia numerosa: FAMNUM →** "Si" o "No"

- **Número de hijos: NHIJO →** 0, 1, 2, 3,…

- **Número de hijos entre 0 y 4 años: NHIJNUC04 →** 0, 1, 2, 3,…

- **Número de hijos entre 5 y 9 años: NHIJNUC59 →** 0, 1, 2, 3,…

- **Número de hijos entre 10 y 14 años: NHIJNUC1014 →** 0, 1, 2, 3,…

- **Número de hijos de 15 años: NHIJNUC15 →** 0, 1, 2, 3,…

- **Número de hijos entre 15 y 19 años: NHIJNUC1519 →** 0, 1, 2, 3,…

- **Número de hijos entre 20 y 24 años: NHIJNUC2024 →** 0, 1, 2, 3,…

- **Número de hijos entre 25 y 29 años: NHIJNUC2529 →** 0, 1, 2, 3,…

- **Número de hijos de 30 y más años: NHIJNUC30 →** 0, 1, 2, 3,…

- **Número de ocupados: OCNUC →** 0, 1, 2, 3,…

- **Número de parados: PANUC →** 0, 1, 2, 3,…

- **Tamaño del núcleo: TAMNUCF →** 0, 1, 2, 3,…

- **Tipo de núcleo: TNUCF →**
  - “Pareja sin hijos”
  - “Pareja con hijos”
  - “Padre con hijos”
  - “Madre con hijos”
  - “No existe núcleo”

- **Tipo de pareja(de hecho o de derecho): TIPOPARECIV →**
  - “Pareja de derecho”
  - “Pareja de hecho(ambos solteros)”
  - “Otro tipo de pareja de hecho”

- **Tipo de pareja(mismo sexo, distinto sexo): TIPOPARSEX →**
  - “Pareja de distinto sexo”
  - “Pareja del mismo sexo, femenino”
  - “Pareja del mismo sexo, masculino”