Universidad Internacional de La Rioja (UNIR)

Escuela de Ingeniería

Máster universitario en Dirección e Ingeniería de Sitios Web

RAMA INVESTIGACIÓN

[Herramienta para la creación de páginas web con contenido semántico en base a un modelo conceptual]

Trabajo Fin de Máster

Presentado por: Sanabria Villarroel, Cristian Marcelo

Director/a: Cobo Martín, Manuel Jesús

Ciudad: Cochabamba-Bolivia
Fecha: 22/09/2015

RESUMEN

El presente proyecto describe la creación de una herramienta de software que permita crear páginas web que están dotadas de contenido semántico mediante la utilización y combinación de tecnologías actuales dando a los usuarios que no estén familiarizados con las tecnologías de representación de información semántica la facilidad de abstraer el proceso de inclusión de esta información mediante la utilización un modelo conceptual más fácil de comprender que pueda ser además portable y manejable. Este sistema se constituirá en un aporte importante al conjunto de herramientas existentes relacionadas con web Semántica con características específicas y bastante útiles ya que si bien existen varios sistemas y lenguajes para publicar datos semánticos en la web, estos sistemas trabajan de un modo asilado y no están integrados como parte de los procesos actuales de creación de contenido en la web.

Palabras Clave: web semántica, generación de código, modelado conceptual, metadatos

ABSTRACT

This project describes the creation of a software tool that allows to create web pages that are equipped with semantic content by utilizing and combining current technologies, giving users who are unfamiliar with technologies of representation of semantic information a system to abstract the process of inclusion of this information using a conceptual model that is easiest to understand.

This system will constitute an important contribution to the set of existing tools related to Semantic Web. Although there are various systems and languages to publish semantic data on the web, a common drawback of these systems is that they work in an asylum mode and are not integrated as part of the current process of content creation for the web.

Keywords: semantic content, semantic web, code generation, conceptual modelling, metadata

CONTENIDO

Tabla de Figuras

1 INTRODUCCIÓN

La web semántica se ha convertido en un campo bastante dinámico en el desarrollo web. Una de las ideas básicas dentro de este concepto es el de poder dotar de significado al contenido que se publica en internet para que posteriormente este significado en forma de información pueda ser procesado de manera automática dándonos así la posibilidad de implementar diversas aplicaciones prácticas que se han descrito en muchas publicaciones al respecto [1], [2].

Los sitios web actuales están basados principalmente en documentos HTML siendo este un lenguaje versátil en cuanto a posibilidades de agregar diferentes tipos de contenido a las páginas web (videos, imágenes, audio, archivos, etc.) pero que ofrece pocas posibilidades para categorizar el texto más allá de las etiquetas estructurales. Si bien existen algunas técnicas y tecnologías promovidas por el w3c para dotar de información semántica a nuestros sitios web, son pocas las herramientas que nos permiten utilizar dichas tecnologías al momento de crear sitios y aplicaciones web.

La información en la web desde sus inicios ha sido predominantemente orientada a ser consumida por humanos, Sin embargo en los últimos años este escenario ha ido cambiando de manera radical

Con el auge de la web Semántica y la inmensa cantidad de información que se ha ido generando en Internet, se ha hecho necesario el tener mecanismos que permitan a maquinas entender y consumir toda esta información para poder dar un cambio radical a la forma en la que se encuentra información, ya que el uso de búsquedas por patrones de texto se ha hecho insuficiente en la actualidad.

El crecimiento de la web Semántica ha promovido una gran comunidad de investigadores, fuentes de información y proyectos relacionados con este tema. Dentro de estos proyectos podríamos mencionar por ejemplo el concepto de datos enlazados (Linked Data) que se basa en la idea de publicar información estructurada mediante tecnologías específicas, pero estas tecnologías no se encuentran integradas con la publicación de contenidos realizada por humanos por lo que podríamos decir que es un campo que aun esta requerido de herramientas.

1.1 JUSTIFICACIÓN DEL PROYECTO

Actualmente, en el contexto de datos enlazados existen muchas herramientas que nos permiten crear y publicar datos semánticos en la web, entre las que destacan herramientas de creación de ontologías como “Protegé”, así como software específicos para el diseño de conocimiento.

Sin embargo estas herramientas si bien son muy adecuadas, son un tanto específicas y no están pensadas para ser integradas en las actividades tradicionales de publicación de contenidos web y nos referimos más específicamente a la creación de páginas web.

Teniendo en cuenta este escenario, podríamos decir que un punto pendiente para el conjunto de herramientas de web Semántica es el de proveer herramientas que puedan ser fácilmente utilizadas en ámbitos no académicos y además por desarrolladores que no estén inmersos en el campo de la web Semántica y que no dominen los conceptos subyacentes sino más bien hacer transparente el proceso de creación y publicación de esta información

1.2 OBJETIVOS DEL PROYECTO

El objetivo principal consiste en la realización de un sistema que permita generar páginas web con información Semántica insertada en el código HTML de estas páginas de manera que esta información Semántica pueda ser extraída posteriormente por herramientas automatizadas.

Lo que diferencia este proyecto de otras herramientas para incluir anotaciones o metadatos en HTML es que el proceso de creación tanto del contenido de la página como los datos semánticos tendrá como inicio un modelo conceptual mediante el cual los usuarios del sistema van a poder definir qué información desean expresar como contenido de la página y como esta información va a estar integrada en la estructura misma de la página.

Es importante recalcar que el modelo conceptual al que hace referencia este proyecto no es una Ontología sino más bien un modelo más superficial que pueda ser fácilmente entendible por un usuario que no tiene experiencia previa en el diseño de información estructurada (conocimiento).

1.3 SITUACIÓN ACTUAL Y HERRAMIENTAS SIMILARES

En la actualidad existen varias herramientas que generan etiquetado semántico mediante anotaciones y que estan orientadas a generar documentos web con informacion de este tipo. Entre los proyectos más conocidos podemos mencionar a las siguientes aplicaciones:

1.3.1 Annotea

Annotea [3] es un proyecto se software libre que cuenta con el estado de LEAD (Live Early Adoption and Demonstration) dentro del W3C.

Este software permite mejorara la colaboración mediante el uso de anotaciones web basadas en metadatos compartidos, enteniendo por anotaciones a diversos elementos como comentarios, notas, explicaciones y otro tipo de marcas externas que pueden ser agregadas a cualquier documento web sin necesidad de modificar la estructura original del documento [3].

Proporciona además un marco extensible basado en metadatos RDF para publicación de páginas web ofreciendo simultaneamente una sencilla interfaz de usuario para anotaciones y marcaciones [3].

1.3.2 CREAM

Cream [4] es un framework que permite crear metadatos, cuenta con dos modalidades de edición el modo de anotaciones que sirve para la creación de metadatos para páginas existentes y el modo de autoría que permite a los autores crear metadatos casi gratuitamente uniéndolos en el contenido de la página.

Otra de las características de esta herramienta es que brinda la posibilidad de crear metadatos relacionales, por ejemplo, metadatos que instancian definiciones interrelacionadas de clases en una ontología de dominio en lugar de un esquema rígido [4].

1.3.3 RDFaCE

RDFaCE [5] es un editor de contenido RDFa basado en TinyMCE. Soporta diferentes vistas para la edición de contenido semántico y usa API's semánticos existentes para facilitar la anotación y edición de contenidos RDFa.

RDFaCE ofrece cuatro vistas para la edición de contenido semántico:

- WYSIWYG View. What-You-See-Is-What-You-Get
- WYSIWYM View. What-You-See-Is-What-You-Mean
- RDF Triple View.
- Source Code View.

Este software es un proyecto de la universidad de Leipzig y cuenta además con una distribución en forma de plugin para el CMS Wordpress.

1.3.4 Otros

Debemos mencionar también al mismo HTML5 [6] que cuenta con etiquetas estructurales semánticas que no existían en las versiones anteriores.

Dentro de los proyectos comerciales tenemos a SemanticUI [7] que nos permite crear interfaces de usuario web en base a la combinación de elementos HTML con valores específicos en clases para dar un significado a estos elementos.

Si bien todas herramientas son proyectos relativamente maduros, el ámbito de aplicación varía en gran medida de uno a otro, desde muy específicos hasta demasiado complejos para ser aplicados dentro de un proceso de desarrollo de software.

1.4 PROPUESTA DE SOLUCIÓN

Con este proyecto se pretende crear una alternativa mas novedosa en comparación a las herramientas existentes actualmente para crear o incluir informacion semántica dentro de páginas web.

Esta herramienta consiste en un sistema de software orientado a la web que estará enfocado en la facilidad de uso por parte de los usuarios pero a la vez será lo suficientemente flexible como para que usuarios con conocimiento mas avanzado en RDFa y HTML puedan personalizar los resultados de manera deseada.

Ademas el ambito de acción de este sistema nos permite pensar en las posibilidades de ampliar la funcionalidad ofrecida en las primeras versiones de manera que con el tiempo se pueda constituir en un sistema complejo y que abarque diferentes escenarios.

2 ASPECTOS TEÓRICOS Y ESTADO DEL ARTE

2.1 LENGUAJES DE ETIQUETADO

2.1.1 HTML5

HTML5 es la última evolución de la norma que define HTML. El término representa dos conceptos diferentes [6], [8].

- primero se trata de una nueva versión del lenguaje HTML, con nuevos elementos, atributos y comportamientos [8].
- segundo, un conjunto más amplio de tecnologías que permite a los sitios Web y a las aplicaciones ser más diversas y de gran alcance. Este conjunto se le llama HTML5 y amigos, a menudo reducido a sólo a HTML5 [8].

HTML5 está diseñado para ser utilizable por todos los desarrolladores de Open Web, existen varias tecnologías de HTML5 que se pueden clasificar en los siguientes grupos [6], [8]:

- Semántica: Permite describir con mayor precisión cuál es su contenido.
- Conectividad: Permite comunicarse con el servidor de formas nuevas e innovadoras.
- Fuera de línea y almacenamiento: Permite a páginas web almacenar datos, localmente, en el lado del cliente y operar fuera de línea de manera más eficiente.
- Multimedia: Nos otorga un excelente soporte para utilizar contenido multimedia como lo son audio y video nativamente.
- Gráficos y efectos 2D/3D: Proporcionar una amplia gama de nuevas características que se ocupan de los gráficos en la web como lo son el lienzo 2D, WebGL, SVG, etc.
- Rendimiento e Integración: Proporcionar una mayor optimización de la velocidad y un mejor uso del hardware.

- Acceso al dispositivo: Proporciona Apis para el uso de varios componentes internos de entrada y salida de nuestro dispositivo.

2.1.2 XML

Es un formato de texto simple y muy flexible derivado del lenguaje SGML (ISO 8879). Fue originalmente diseñado para afrontar el reto de la publicación electrónica a gran escala, XML juega un papel importante en el intercambio de una gran variedad de datos en internet y en otros ámbitos [9].

2.2 CONTENIDO SEMÁNTICO

2.2.1 Etiquetado semántico

La web semántica es una web de datos mas que de documentos, para este proposito es necesario crear nuestros sitios web de manera que sean entendibles por humanos y computadoras.

Existen diferentes tecnicas para etiquetar semánticamente las páginas web. Siendo las mas populares los Microformatos, Microdata y RDFa.

Aunque dichas herramientas son similares en cuanto a objetivos, son diferentes con respecto a su implementación, en este proyecto de investigacion se realizará una explicación de estas tecnologias pero se dara prioridad a RDFa como la herramienta principal para desarrollar el proyecto.

2.2.2 Microformatos

Los microformatos son la forma mas sencilla de publicar abiertamente diferentes tipos de informacion estructurada en la web como por ejemplo contactos, eventos, recetas, etc. [10]

Diseñados para humanos primero y para maquinas en segundo lugar, los microformatos son un conjunto de formatos de datos simples y abiertos basados en estandares existentes adoptados ampliamente. En lugar de desechar lo que funciona hoy en dia, los microformatos

tienen la intencion de resolver problemas sencillos en primer lugar mediante la adaptación a comportamientos actuales y patrones de uso [10].

La idea detrás del concepto de los microformatos es bastante simple y es debido a esto que ha tenido cierta popularidad. Esto Consiste en la utlización de los atributos de las etiquetas HTML existenes “class”, “rel” y “rev” asignando ciertos valores previamente definidos a estas propiedades podriamos indicar que el contenido dentro de las etiquetas que tienen esos atributos tienen un significado [10].

En el siguiente ejemplo:

```
<div class="vcard">
   <span class="given-name">Juan Perez</span>
   <span class="org">www.perez.org</span>
   <ul class="adr">
      <li class="street-address">Av. Siempre Viva 2016</li>
      <li class="locality">Springfield</li>
      <li class="region">Illinois</li>
   </ul>
</div>
```

Mediante el uso del microformato hCard estamos especificando Un Contacto llamado “Juan Perez” y sus propiedades :

Nombre: Juan Perez

Organización: www.perez.org

Direccion: Av. Siempre Viva 2016, Springfield – Illinois

Al igual que hCard existen diferentes tipos de microformatos para diferentes elementos.

2.2.3 Microdata

MIcrodata [11] es un mecanismo que permite incluir datos legibles por maquinas en documentos HTML de una manera facil de escribir, con un modelo de extraccion no ambiguo ademas es compatible con otros formatos de datos como RDF y JSON.

Microdata consiste en un conjunto de pares clave/valor. Los grupos son llamados ‘items’ y cada par clave/valor es una propiedad. Items y propiedades son representados por elementos regulares.

En cuanto a complejidad, Microdata va un poco mas alla que los microformatos permitiendo el uso de vocabularios que pueden extender la semántica.

Los atributos globales que son la base de microdata son los siguientes:

- Itemscope: Crea un elemento e indica que los decendientes de este elemento tienen informacion sobre este.
- Itemtype: la URL de un vocabulario válido que describe el item y la información de su contexto.
- Itemid: identificador unico del elemento.
- Itemprop: Indica que el contenido del tag es una propiedad del elemento padre , el nombre de esta propiedad viene definido en el contexto del vocabulario del elemento.
- Itemref: es usado para relacionar elemento externos al elemento “itemscope”.

En el siguiente ejemplo podemos ver la forma de uso de Microdata:

```
<div id="juan" itemscope itemtype="http://schema.org/Contact">
   <span itemprop="given-name">Juan Perez</span>
   <span itemprop="org">www.perez.org</span>
   <ul itemprop="addressr">
      <li itemprop="street-address">Av. Siempre Viva 2016</li>
      <li itemprop="locality">Springfield</li>
      <li itemprop="region">Illinois</li>
   </ul>
</div>
<div id="hermano" itemscope itemtype="http://schema.org/Contact" itemref="juan">
  <span itemprop="given-name">Carlos Perez</span>,

</div>
```

Podemos apreciar que la estructura es similar a Microformatos, en el caso de Microdata hacemos uso de namespaces para definir el vocabulario de posibles atributos para nuestros elementos. Además una característica adicional es que mediante la propiedad “itemref” podemos ya relacionar elementos dentro de la página más allá de la relación elemento/atributo lo cual ya nos va dando mayor potencialidad de expresión del contenido.

Si bien Microdata es parte de W3C aún no es una recomendación por lo tanto no se puede decir que sea una tecnología que se considere terminada o con soporte.

2.2.4 RDFa

RDFa (Resource Description Framework in Attributes) [12] es una tecnología creada por el W3C que nos permite añadir información Semántica al contenido XHTML de una página web reutilizando atributos existentes de XHTML y aplicándolos a otras partes.

Usando unos pocos atributos HTML simples, autores de páginas HTML pueden etiquetar datos legibles por humanos con indicadores legibles por máquinas para que los navegadores puedan interpretar. Una página web puede incluir etiquetado para elementos tan simple como el título de un artículo o tan complejo como la red social completa de un usuario.

Mediante el uso de RDFa es posible también añadir tripletas RDF a los documentos web lo que nos da la habilidad de incluir varios vocabularios dentro del etiquetado XHTML.

RDFa hace uso de los siguientes atributos específicos:

- about: usado para representar el tema.
- property: usado para representar el valor.
- resource: usado para representar el objeto.
- datatype: usado para representar el tipo de datos del recurso (resource).
- typeof: representa el tipo del recurso.

A continuación se presenta un ejemplo de inclusión de RDFa en una página web XHTML:

```
<div vocab="http://rdf.data-vocabulary.org/#" typeof="Review">
        <span property="itemreviewed">Pizza Amorita</span>
        Reviewed by
        <span property="reviewer">Pablo Soto</span> on
        <span property="dtreviewed" content="2009-01-06">Jan 6</span>.
        <span property="summary">Pizza muy bien preparada</span>
        <span property="description">Esta pizza representa una de las comidas más tradicionales de
        la cocina italiana en la category de comida internacional</span>
        Rating:
        <span property="rating">4.5</span>
</div>
```

2.2.4.1 Ventajas de RDFa

Las Principales ventajas de RDFa son [12]:

- RDFa hace el código legible para humanos y máquinas. El etiquetado semántico añade estructura a un sitio web haciendo uso de atributos que ya existen en XHTML, atributos propios de RDFa y tripletas RDF.
- Diferentes vocabularios pueden ser implementados como parte de la estructura del sitio web. Los desarrolladores pueden implementar sus propios vocabularios o hacer uso de los varios vocabularios externos.
- RDFa es fácil de implementar en una página web y al ser un subconjunto de RDF es posible extraer tripletas RDF de un sitio etiquetado semánticamente con RDFa.
- RDFa ha sido creado y cuenta con el soporte del W3C que es la principal organización de la World Wide Web. Este hecho brinda a RDFa el estado de tecnología estándar.
- RDFa hace uso de la filosofía DRY (Don't Repeat Yourself) que reduce la probabilidad de introducir inconsistencias. Cada pieza de código es presentada con información completa solo una vez.

2.2.4.2 Desventajas de RDFa

Las principales desventajas de RDFa son [12]:

- El contenido de la página web debe ser escrito con XHTML 1.1 o versiones posteriores. Asimismo, de acuerdo con W3C no es posible implementar RDFa en HTML porque no es un lenguaje extensible como XHTML.
- A diferencia de RDF en RDFa los desarrolladores deben sustituir las URIs completas con URIs más reducidas (CURIEs).

2.2.4.3 RDFa Lite 1.1

RDFa Lite [13] es un subconjunto mínimo de RDFa , el Marco de Descripción de Recursos en los atributos , que consiste en un par de atributos que pueden ser usados para expresar datos legibles por máquinas en documentos web como HTML , SVG y XML . Si bien no es una solución completa para las tareas de marcado de datos avanzados, funciona para la mayoría

de las necesidades del día a día y puede ser aprendido por la mayoría de los autores de sitios Web en un día.

RDFa Lite 1.1 consta de cinco atributos simples; vocab, typeof, property, resource, y prefix. RDFa 1.1 Lite es compatible con la versión completa de RDFa 1.1. Esto significa que si un autor encuentra que RDFa Lite no es lo suficientemente potente, la transición a la versión completa de RDFa es sólo una cuestión de añadir los atributos de la versión complete y más potente de RDFa en el marcado RDFa Lite existente.

2.2.4.3.1 Vocab

Por lo general, cuando hablamos de una cosa, utilizamos un vocabulario especial para hablar de ello. Así que, si usted quiere hablar de personas, el vocabulario que utilizaría especificaría términos como nombre y número de teléfono. Cuando queremos marcar cosas en la web, tenemos que hacer algo muy similar, que es especificar qué vocabulario que vamos a utilizar. [13]

Aquí está un ejemplo simple que especifica un vocabulario que tenemos la intención de utilizar para el marcado de las cosas en el párrafo:

```
<p vocab="http://schema.org/">
   My name is Manu Sporny and you can give me a ring via 1-800-555-0199.
</p>
```

Figura 1: RDFa Lite uso de vocab

En este ejemplo hemos especificado que vamos a utilizar el vocabulario que se puede encontrar en http://schema.org/. Se trata de un vocabulario que ha sido publicado por las principales compañías de motores de búsqueda para hablar de las cosas comunes en la web que los motores de búsqueda se preocupan - cosas como Gente, Lugares, Reseñas, recetas y eventos. [13]

2.2.4.3.2 Typeof

Una vez que hemos determinado el vocabulario, tenemos que especificar el tipo del objeto del que estamos hablando. [13] En este caso particular, estamos hablando de una persona, esto puede ser marcado hasta este modo:

```
<p vocab="http://schema.org/" typeof="Person">
   My name is Manu Sporny and you can give me a ring via 1-800-555-0199.
</p>
```

Figura 2: RDFa Lite uso de typeof

2.2.4.3.3 Property

Ahora todo lo que tenemos que hacer es especificar qué propiedades de esa "persona" queremos señalar al motor de búsqueda. En el siguiente ejemplo, marcamos los atributos nombre, teléfono y pagina web de la persona. Tanto el texto como las direcciones URL se pueden marcar con RDFa Lite. [13] En el siguiente ejemplo, prestar especial atención a los tipos de datos que se están señalando al motor de búsqueda, que se resalta en azul:

```
<p vocab="http://schema.org/" typeof="Person">
   My name is
   <span property="name">Manu Sporny</span>
   and you can give me a ring via
   <span property="telephone">1-800-555-0199</span>
   or visit
   <a property="url" href="http://manu.sporny.org/">my homepage</a>.
</p>
```

Figura 3: RDFa Lite uso de property

Ahora, cuando alguien escribe en " número de teléfono para Manu Sporny" en un motor de búsqueda, el motor de búsqueda pueden responder de forma más fiable la pregunta directamente, o señalar la persona que busca a una página Web más relevante. [13]

2.2.4.3.4 Resource

Si se desea que autores Web sean capaces de hablar de cada cosa en una página, es necesario crear un identificador para cada una de las cosas. Al igual que creamos identificadores de partes de una página utilizando el atributo id en HTML, se pueden crear identificadores para las cosas que se describen en una página utilizando el atributo de "resource" [13]:

```
<p vocab="http://schema.org/" resource="#manu" typeof="Person">
   My name is
   <span property="name">Manu Sporny</span>
   and you can give me a ring via
   <span property="telephone">1-800-555-0199</span>.
   <img property="image" src="http://manu.sporny.org/images/manu.png" />
</p>
```

Figura 4: RDFa Lite uso de resource

Si asumimos que el etiquetado mostrado arriba puede ser encontrado en *http://example.org/people* , entonces el identificador para este objeto es la dirección más el valor del atributo “resource”. Por lo tanto el identificar para este objeto dentro de la página web seria *http://example.org/people#manu*. Este identificador es también útil si se desea referenciar a este objeto desde otra página web. [13]

2.2.4.3.5 Prefix

En algunos casos, un vocabulario no puede tener todos los términos que un autor necesita para describir lo que desea. La última característica en RDFa Lite 1.1 que algunos autores podrían necesitar es la capacidad de especificar más de un vocabulario. [13] Por ejemplo, si estamos describiendo una persona y tenemos que especificar que tienen un animal favorito, podríamos hacer algo como lo siguiente:

```
<p vocab="http://schema.org/" prefix="ov: http://open.vocab.org/terms/" resource="#manu" typeof="Person">
   My name is
   <span property="name">Manu Sporny</span>
   and you can give me a ring via
   <span property="telephone">1-800-555-0199</span>.
   <img property="image" src="http://manu.sporny.org/images/manu.png" />
   My favorite animal is the <span property="ov:preferredAnimal">Liger</span>.
</p>
```

Figura 5: RDFa Lite uso de prefix

El ejemplo asigna un prefijo (prefix) para el Vocabulario abierto (ov) y utiliza ese prefijo para especificar el término ‘preferredAnimal’ (animal favorito) de este vocabulario. Puesto que schema.org no tiene una forma clara de expresar un “animal favorito”, el autor en este ejemplo depende de este vocabulario alternativo para conseguir el objetivo. [13]

RDFa 1.1 Lite también pre - define una serie de prefijos útiles y populares, como cc, foaf y esquema. Esto asegura que incluso si los autores olvidan declarar estos prefijos populares, sus datos estructurados continuarán trabajando. [13]

2.2.5 Comparación de tecnologías de etiquetado semántico

Característica	RDFa 1.1	Microdata 1.0	Microformats 1.0
Complejidad relativa	Alta	Media	Baja
Modelo de datos	Grafo	Arbol	Árbol
elemento opcionalmente identificado por URI	Si	Si	No
propiedades de los elementos especificados por URI	Si	Si	No
Múltiples objetos por página	Si	Si	Si
Objetos superpuestos	Si	Si	No
Propiedades en texto plano	Si	Si	Si
Propiedades IRI	Si	Si*	No
Propiedades literales con Tipo	Si	No	No
Propiedades literales XML	Si	No	No
Etiquetado de lenguaje	Si	Si	Inconsistente
Sobre escritura de texto y contenido IRI	Si	No	Solo Texto
facilidad de mapeo a RDF	Si	Problemático	No
Lenguajes a los que va dirigido	(XHTML1, HTML4, HTML5, XHTML5, XML, SVG, ePub, OpenDocument)	(HTML5, XHTML5)	(XHTML1, HTML4, HTML5, XHTML5)
Nuevos atributos	about, datatype, profile, prefix, property, resource, typeof, vocab	itemid, itemprop, itemref, itemscope, itemtype	0

atributos re-utilizados	content, href, rel, rev, src	content, src, href, data, datetime	class, title, rel, href
Múltiples tipos IRI por objeto	Si	Solo RDF	No
Múltiples instrucciones por elemento	Si	No	Si
Términos de vocabulario de "ámbito local"	Si, vía vocab	Si, vía itemscope	No
encadenamiento de elementos	Si	Basic	No
Translación	No	Si	Si, vía include pattern
IRIs compactas	Si	No	No
reasignación de prefijos	Si	No	No
Vocabulario	Si	No	No
soporte para el elemento "time" de HTML5	NO aun	Si	No
Diferentes atributos para diferentes tipos de propiedades	Si	No	Si
	property for text, rel/rev for URLs, resource/content for overrides		class para texto y rel para IRLs
Transformación a JSON	Si (RDFa API)	Si (Parser and Microdata DOM API)	No
DOM API	Si	Si	No
Parser Unificado	Si	Si	No
Objetos superpuestos	Si	Si	No
Propiedades en texto plano	Si	Si	Si

Tabla 1: Comparación de tecnologías de etiquetado semántico

2.3 MODELADO CONCEPTUAL

En terminos generales el modelado conceptual es el proceso de abstraer un modelo del mundo real. El creador del modelo identifica un problema o situación que es suceptible de simulación y luego determina los aspectos del mundo real que se van a incluir y cuales se van a excluir del modelo y ademas el nivel de detalle [14].

Un modelo conceptual captura ideas en un dominio de conocimiento especifico y nos sirve para representar diferentes conceptos o situaciones del mundo real, a menudo el modelado conceptual es confundido con el modelado de clases o modelado de entidades y relaciones [14].

Si bien tanto un modelo de clases como un modelo relacional son tambien modelos conceptuales, el modelado conceptual no está limitado e estos dos tipos, y no está restringido a un lenguaje y notación especificos con UML [14].

Los modelos conceptuales son utilizados en una diversidad de campos de estudio como las ciencias, ingenierias, ciencias sociales, filosofia, estadistica, etc [14].

El valor de un modelo radica en la exactitud dentro de un determinado contexto y el nivel de utilidad que brinda a un proposito ya sea el diseño o la expliación [14].

En el presente proyecto se esta haciendo uso de un modelo propio sin ninguna notación generica sino mas bien mediante el uso de elementos que respresentan.

2.3.1 Modelo Conceptual de una página web

HTML es un lenguaje de etiquetas cuya filosofía se basa en la inclusión de elementos (tags) de la página dentro de otros elementos en una estructura de árbol de contenido.

Esta estructura de árbol hace que sea sencillo poder representar un documento HTML y sus elementos mediante un modelo conceptual que sirva para tener una visión grafica de como luce un documento, esto podría ser útil en diversos escenarios como por ejemplo el proyecto que se explica en este documento.

Un documento HTML básico y correctamente formado tiene la siguiente estructura:

```
<html>
        <head>
                <title></title>
        </head>
        <body>
        .... elementos
        </body>
</html>
```

Esta estructura puede ser representada por el siguiente modelo:

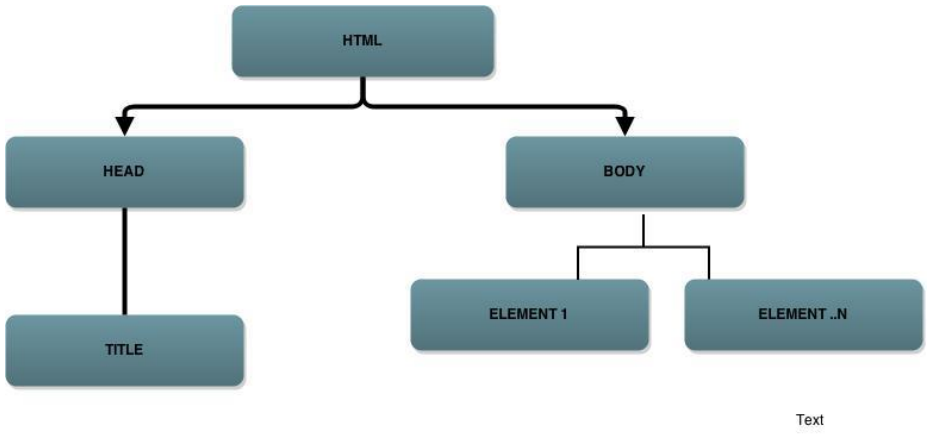


Figura 6: Modelo que representa una página básica HTML

Este esquema de modelo es el que se ha de utilizar en la herramienta desarrollada, sin embargo este modelo propuesto es una base que puede ser extendido y con mucha más información de acuerdo se necesite

3 DISEÑO DE LA INVESTIGACIÓN

Los diseños cuasi experimentales son los principales instrumentos de trabajo dentro del ámbito de la investigación aplicada.

El “Cuasi-Experimento” [15] también conocido como experimento de campo, o experimento “in-situ”, es un tipo de diseño experimental en el cual el investigador tiene influencia y control limitados en la selección de los participantes de un estudio.

Específicamente, en los Cuasi-Experimentos, el investigador no tiene la habilidad de asignar los participantes al azar y/o asegurarse de que el ejemplo elegido es tan homogéneo como se desea. Adicionalmente, en numerosas investigaciones, incluyendo aquellas que se llevan a cabo en investigación de sistemas de información, la aleatorización podría no ser factible, dejando al investigador con asignaciones de grupo pre-asignadas [15].

En Consecuencia, la habilidad de controlar totalmente todas las variables de estudio podría ser limitada. Sin embargo, los cuasi-experimentos continúan proporcionando información fructífera para el avance de la investigación [15].

He considerado apropiado este tipo de diseño de investigación ya que mi proyecto es un TFM de investigación que consiste en el desarrollo de una herramienta constituyéndose en una investigación aplicada además tomando en cuenta las características de las actividades a realizar la metodología es la más cercana en comparación a otros diseños.

4 PLANIFICACIÓN

4.1 IDENTIFICACIÓN DE ACTIVIDADES DEL PROYECTO

En la siguiente tabla se describen las cuatro principales y más generales fases del proyecto en general estas fases a su vez serán subdivididas y presentadas en puntos posteriores del presente documento

Actividad	Descripción	Duración estimada (semanas)
Identificación de requerimientos del proyecto	Fase inicial en la que se identifican los requisitos tanto técnicos como del proyecto que servirán de base estimación del trabajo necesario	1-2
Fase de Diseño y Desarrollo del Sistema	Fase técnica que incluye el proceso de desarrollo de	4-5

	software en si acorde a una metodología elegida	
Pruebas y Validación	Etapa de verificación en la que se validan los requerimientos y además los alcances del proyecto	1-2
Documentación	Proceso de generación de todos los documentos que sean necesarios para la correcta utilización del software creado.	1

Tabla1: Etapas del proyecto

4.2 METODOLOGÍA

Tomando en cuenta las características del proyecto se ha elegido una metodología iterativa e incremental de desarrollo ya que se adecua perfectamente a las características y contexto dentro del cual se va a ejecutar este proyecto.

Existen varios métodos de esta naturaleza que son conocidos como metodologías agiles, la idea principal de estos métodos consiste en realizar lapsos cortos de tiempo en los cuales se desarrolla funcionalidad especifica de forma incremental de manera que cada ciclo conocido generalmente como “sprint” va añadiendo funcionalidad al sistema final. [16]

Le definición de los Sprint (ciclos necesarios) depende de varios factores como la división de funcionalidades, recursos disponibles, tiempo y grado de dificultad del desarrollo.

4.3 ESTIMACIÓN DE RECURSOS NECESARIOS

4.3.1 Estimación basada en casos de uso

En esta técnica se utilizan los casos de usos identificados para calcular el esfuerzo necesario para desarrollar cada uno, a cada caso de uso se asigna una complejidad basada en transacciones, entendidas como interacción entre el usuario y el sistema mientras que a los

actores se les asigna una complejidad basada en su tipo es decir si son interfaces con usuarios u otros sistemas también se utilizan factores de entorno y complejidad técnica. [17]

4.3.2 Estimación del proyecto en base a puntos de función

Factor de peso de los actores sin ajustar (UAW)

Tipo de Interacción	Peso
Simple (a través de un API)	1
Medio (a través de un protocolo)	2
Complejo (a través de la interfaz gráfica)	3

Factor de peso de casos de uso sin ajustar

Tipo de caso de uso	Descripción	Factor de peso
Simple	El caso de uso contiene de 1 a 3 transacciones	5
Medio	El caso de uso contiene de 4 a 7 transacciones	10
Complejo	El caso de uso contiene más de 8 transacciones	15

Calculo de UUCP

UUCP: Puntos de casos de uso a utilizar

UAW: Factor de peso de los actores sin ajustar

UUCW: Factor de peso de los casos de uso sin ajustar

La fórmula para el cálculo de UUCP es:

UUCP = UAW + UUCW

Estimación Inicial por casos de uso del proyecto

Caso de Uso	Complejidad del autor (UAW)	Número de transacciones (UUCW)	Valor calculado (UAW+UUCW)
Crear Modelo	3*1	3*5	18
Generar Pagina	3*1	2*5	13
Guardar Modelo	3*1	4*10	43
Abrir Proyecto	3*1	1*5	8
Configuración Global	3*1	1*5	8
Crear elemento semántico	3*1	2*5	13

Total UUCP = 103

Calculo del Factor de complejidad técnica TCF

El factor de complejidad técnica se calcula con la siguiente formula:

TCF = 0.6 + 0.0.1* ∑ (Peso * Valor asignado)

Factor	Descripción	Peso	Valor asignado	Comentario
T1	Sistema distribuido	2	0	El sistema es centralizado
T2	Objetivos de performance y tiempo de respuesta	1	1	El tiempo de respuesta no es un requerimiento critico

T3	Eficiencia del usuario final	1	4	Se requiere cierto nivel de capacitación hacia el usuario
T4	Procesamiento interno complejo	1	4	Se tienen algoritmos de conversión y control relativamente complejos
T5	El código debe ser reutilizable	1	0	No requerido inicialmente
T6	Facilidad de instalación	0.5	1	Escasos requisitos de facilidad de instalación
T7	Facilidad de uso	0.5	3	Normal
T8	Portabilidad	2	0	No se requiere que el sistema sea portable
T9	Facilidad de cambio	1	2	Es deseable que se tenga cierta facilidad de cambio
T10	Concurrencia	1	0	No hay concurrencia
T11	Incluye objetivos especiales de seguridad	1	2	Seguridad normal a baja
T12	Provee acceso directo a terceras partes	1	4	Los usuarios web tiene acceso directo
T13	Se requieren facilidades	1	3	Relativamente necesario

	especiales de entrenamiento a usuarios			

TCF = 0.6 + 0.01 *(0+1+4+4+0+0.5+1.5+0+2+0+2+4+3)

TCF = 0.82

Factor de Ambiente

El factor de ambiente se calcula con la siguiente formula:

$$EF = 1.4 – 0.03*\sum (Peso^{i} * Valor\ Asignado)$$

Factor	Descripción	Peso	Valor asignado	Comentario
E1	Familiaridad con el modelo de proyecto utilizado	1.5	4	Si tiene experiencia con el modelo del proyecto utilizado
E2	Experiencia en la aplicación	0.5	2	Se trata de una aplicación nueva
E3	Experiencia en orientación a objetos	1	5	Mucha experiencia
E4	Capacidad de analista líder	0.5	1	No se tiene a una persona dedicada
E5	Motivación	1	5	Buena motivación
E6	Estabilidad de los requerimientos	2	2	Se esperan cambios a lo

				largo del proyecto
E7	Personal – part time	-1	4	Tiempo parcial de dedicación
E8	Dificultad del lenguaje de programación	-1	3	C#

EF = 1.4 – 0.03 * (1.5*4+0.5*2+1*5+0.5*1+1*5+2*2-1*4-1*3)

EF = 0.96

Calculo de puntos de función ajustados y esfuerzo total del proyecto

UCP = UUCP * TCF * EF

UCP = 103*0.82*0.96= 81.08

E = Esfuerzo = 5 horas/hombre por cada UCP

E = 81.08 * 5 = 405.4 horas

4.4 CRONOGRAMA

Tomando en cuenta la estimación de recursos y la planificacion se tendra un cronograma tentativo que engloba todas las etapas durante la realizacion del proyecto.

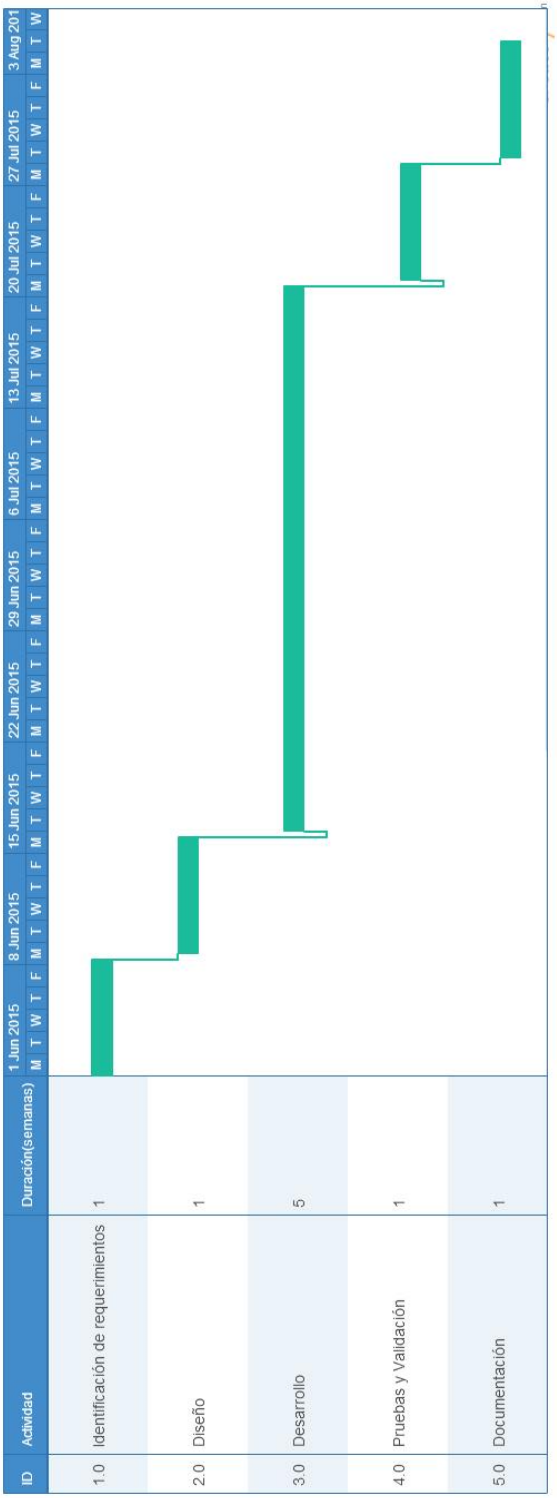


Figura 7: Cronograma del proyecto

5 ANÁLISIS DEL SISTEMA

5.1 DEFINICIÓN DEL ALCANCE DEL SISTEMA

El sistema consiste en una aplicación web que permitirá crear páginas web con información Semántica a partir de un modelo diseñado mediante la interfaz de usuario, el sistema permitirá además guardar y abrir proyectos creados anteriormente. Entre los principales componentes funcionales de este software están

- Interfaz de usuario para el diseño de un modelo y configuración de elementos para este modelo
- Generación automática de código en base al modelo realizado
- Funcionalidad para guardar proyectos y abrir proyectos existentes.

5.2 DEFINICIÓN DE USUARIOS DEL SISTEMA

Se contemplan los siguientes usuarios dentro del sistema:

- Usuario Standard .- Un usuario standard es aquel que puede ejecutar la funcionalidad principal de la aplicación es decir crear modelos, guardar datos y generar páginas web.

  Este tipo de usuario a su vez puede ser clasificado desde un punto de vista conceptual en dos grupos en base a la funcionalidad a utilizar (especialidad):

  - Diseñador de Logica (Logic Designer).- El usuario esta encargado de crear el modelo conceptual de la pagina web utilizando los elementos previamente creados (elementos, vocabularios, etc.)

  - Diseñador de Cotenidos (Content Designer).- usuario con conocimientos de RDFa Lite que esta encargado de crear los elementos semánticos y vocabularios que van a ser utilizados en el Diseñador por un usuario “Logic Designer”

- Usuario Administrador.- Un usuario administrador puede realizar todas las funciones de un usuario standard y ademas tiene habilitadas ciertas funcionalidades adicionales como la administracion de usuarios.

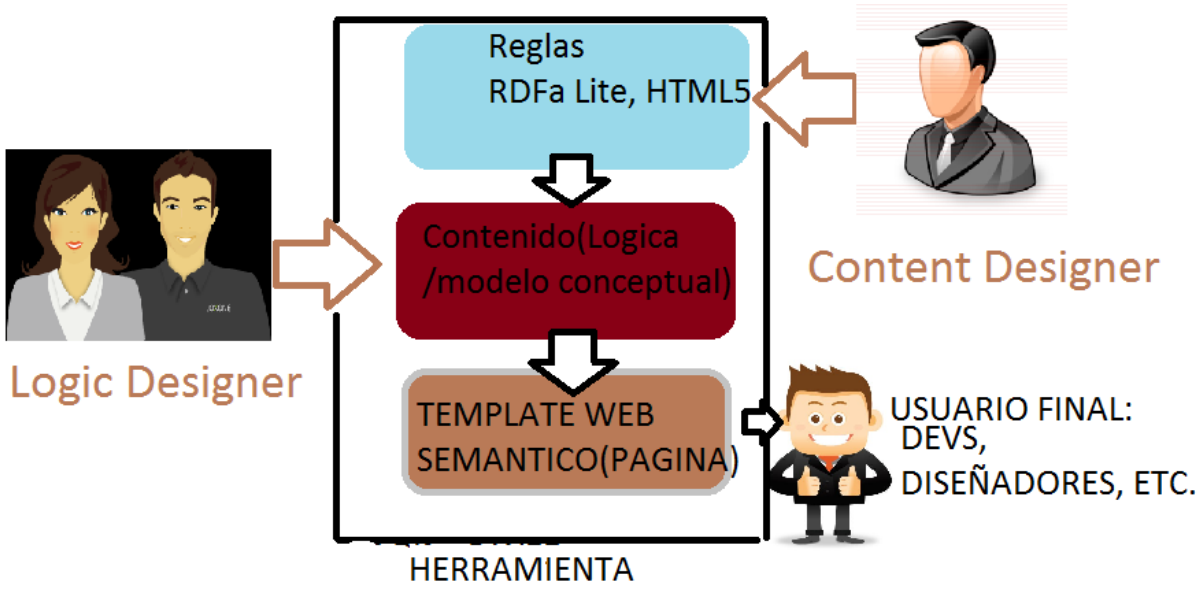


Figura 8: Ámbito del Sistema

5.3 ESPECIFICACIÓN DE REQUISITOS

5.3.1 Requerimientos Funcionales

A continuación se detallan los requerimientos funcionales del sistema a desarrollar:

- El sistema proveerá un editor gráfico que permitirá diseñar el modelo de la página web incluyendo elementos tanto de HTML como elementos conceptuales previamente definidos en el sistema
- El sistema proveerá un mecanismo para definir elementos conceptuales que previamente podrán ser incluidos en el modelo de la página web por ejemplo: Contacto, Persona, Libro, etc.
- El sistema permitirá guardar un proyecto que incluya tanto el modelo, elementos personalizados y demás información necesaria para poder pausar el trabajo y continuarlo
- El sistema tendrá la funcionalidad para generar la página web (contenido HTML+RDFa) en base el modelo conceptual creado
- Para el uso de objetos y atributos RDFa el sistema permitirá definir los vocabularios que el usuario desee incluir en el contenido de la página web
- Se tendrá la opción para poder editar el contenido de una página web si se desea

- Debe existir información de ayuda sobre la funcionalidad y términos utilizados en la interfaz de usuario del sistema

5.3.2 Requerimientos No Funcionales

A continuación se especifican los requerimientos no funcionales del sistema:

- El sistema debe funcionar en un entorno cliente/servidor y debe poder utilizarse mediante internet o intranet
- El sistema proveerá una interfaz web para ser accedido usando distintos navegadores
- El sistema debe utilizar SQL server como plataforma de base de datos e 'Internet Information Services' como servidor de aplicaciones
- El sistema debe ejecutarse en un entorno Windows.

5.4 IDENTIFICACIÓN DE ESCENARIOS Y CASOS DE USO

Caso de Uso	Crear Modelo
Actor	Usuario – Logic Designer
Descripción	Un usuario crear un modelo conceptual de la página mediante la interfaz gráfica de la aplicación, este procedimiento está basado en "drag and drop" (arrastrar y soltar elementos)
Precondición	Elementos semánticos/otros deben haber sido creados previamente
Pos Condición	Después de haber creado un modelo se podrá genera una página y guardar el proyecto

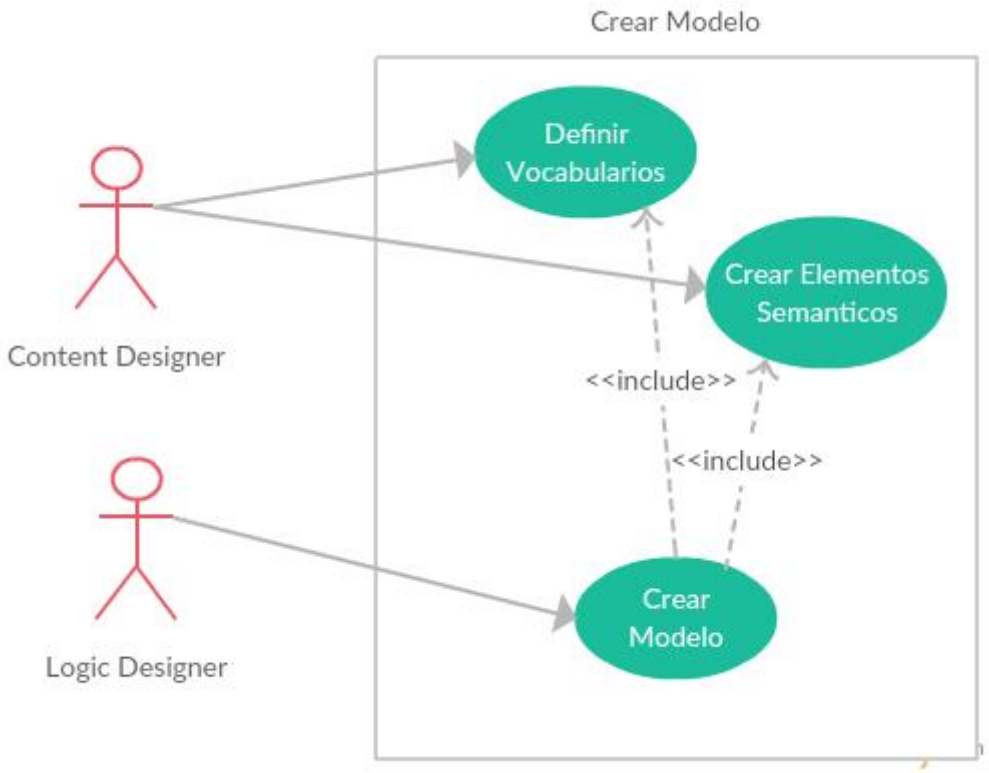


Figura 9: Caso de Uso – Crear Modelo

Caso de Uso	Generar Pagina HTML
Actor	Usuario
Descripción	Una vez creado un modelo será posible generar una página HTML con el contenido especificado en el modelo (elementos)
Precondición	Se debe de haber creado un modelo previamente
Pos Condición	Después de haber generado la página HTML (markup) será posible exportar el contenido y editar el resultado

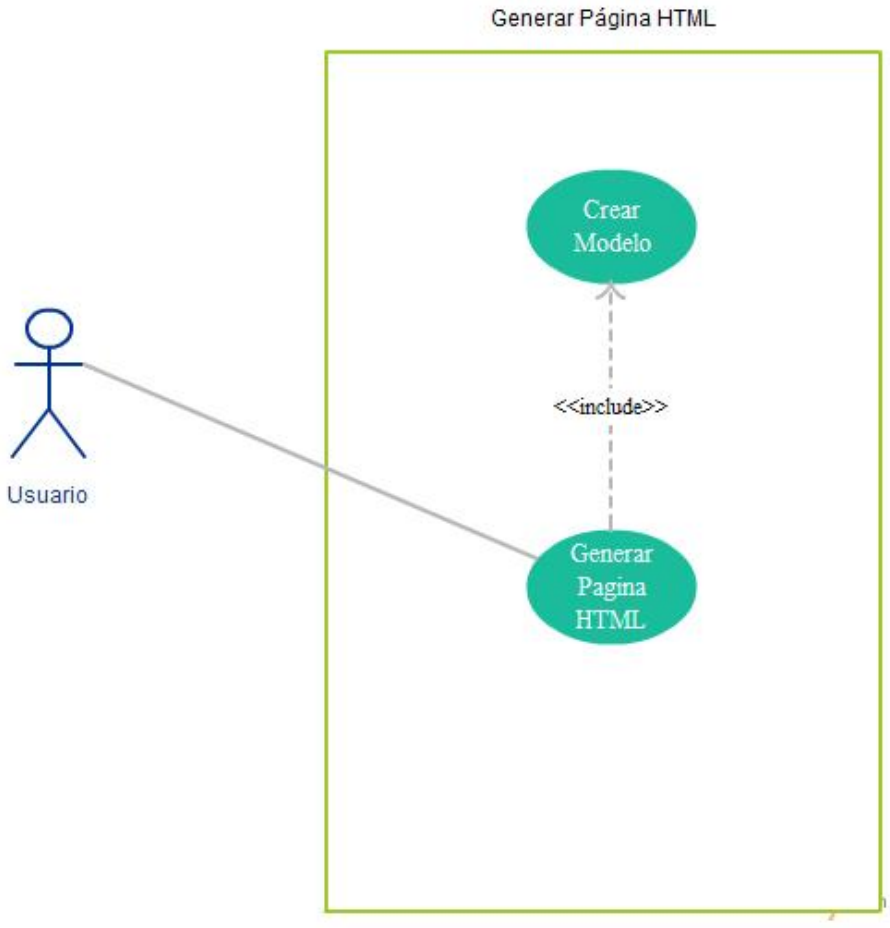


Figura 10: Caso de Uso – Generar Página

Caso de Uso	Guardar Proyecto
Actor	Usuario
Descripción	Una vez creado un modelo será posible guardar el modelo y creado y otras configuraciones
Precondición	
Pos Condición	Una vez guardado el proyecto podrá ser abierto posteriormente.

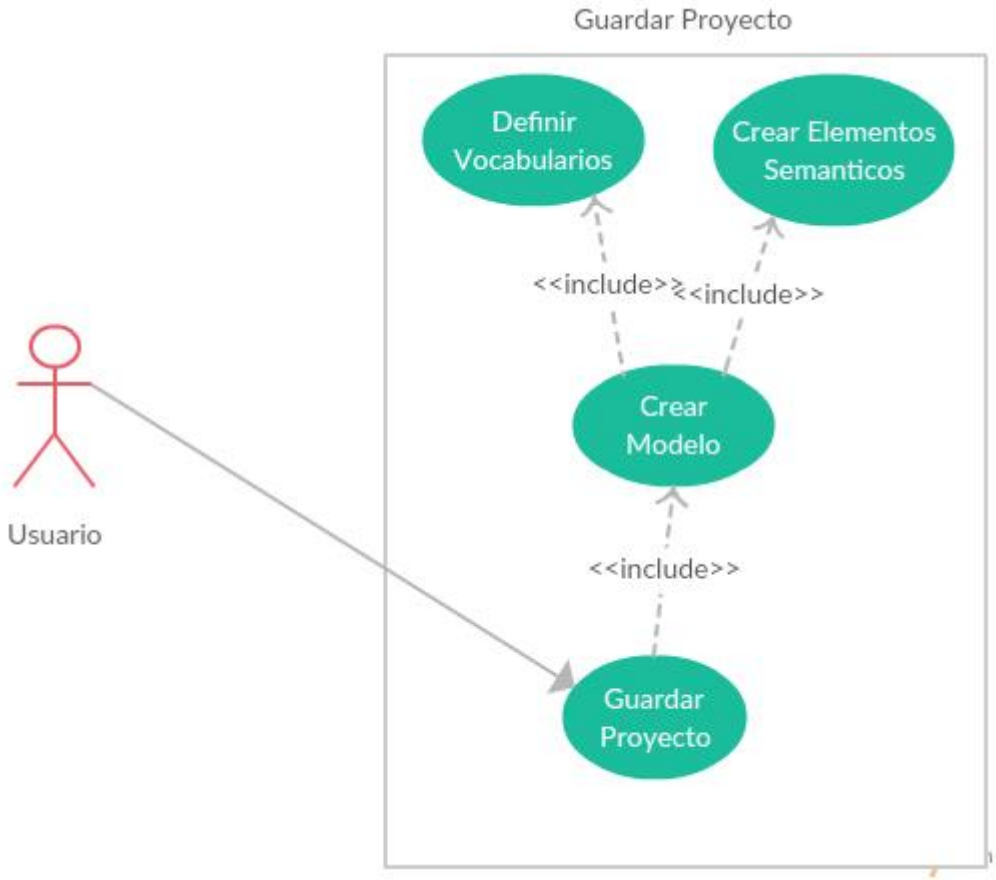


Figura 11: Caso de Uso – Guardar proyecto

Caso de Uso	Abrir Proyecto
Actor	Usuario
Descripción	Sera posible abrir un proyecto guardado previamente en la aplicación
Precondición	Debe existir un proyecto guardado anteriormente
Pos Condición	

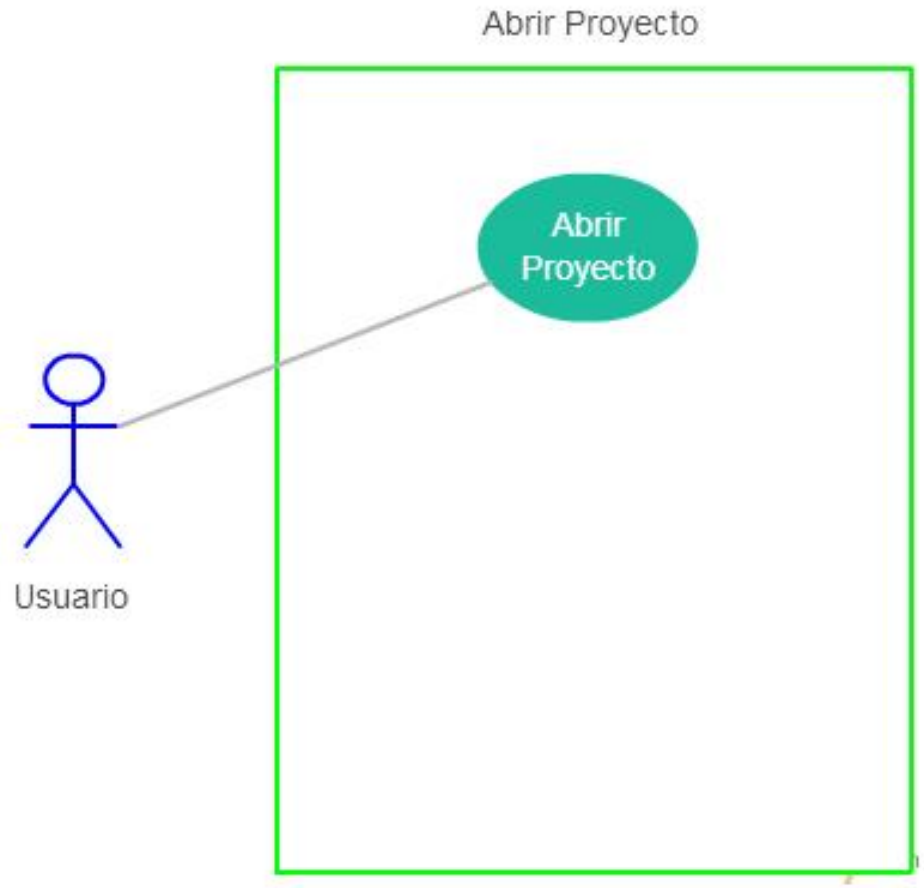


Figura 12: Caso de Uso - Abrir Proyecto

Caso de Uso	Administrar usuarios
Actor	Administrador
Descripción	Un usuario administrador tendrá habilitada la sección para administración de usuarios donde este podrá crear/editar y eliminar usuarios del sistema
Precondición	Usuario debe ser administrador
Pos Condición	Nuevos usuarios creados, usuarios existentes modificados, usuarios eliminados

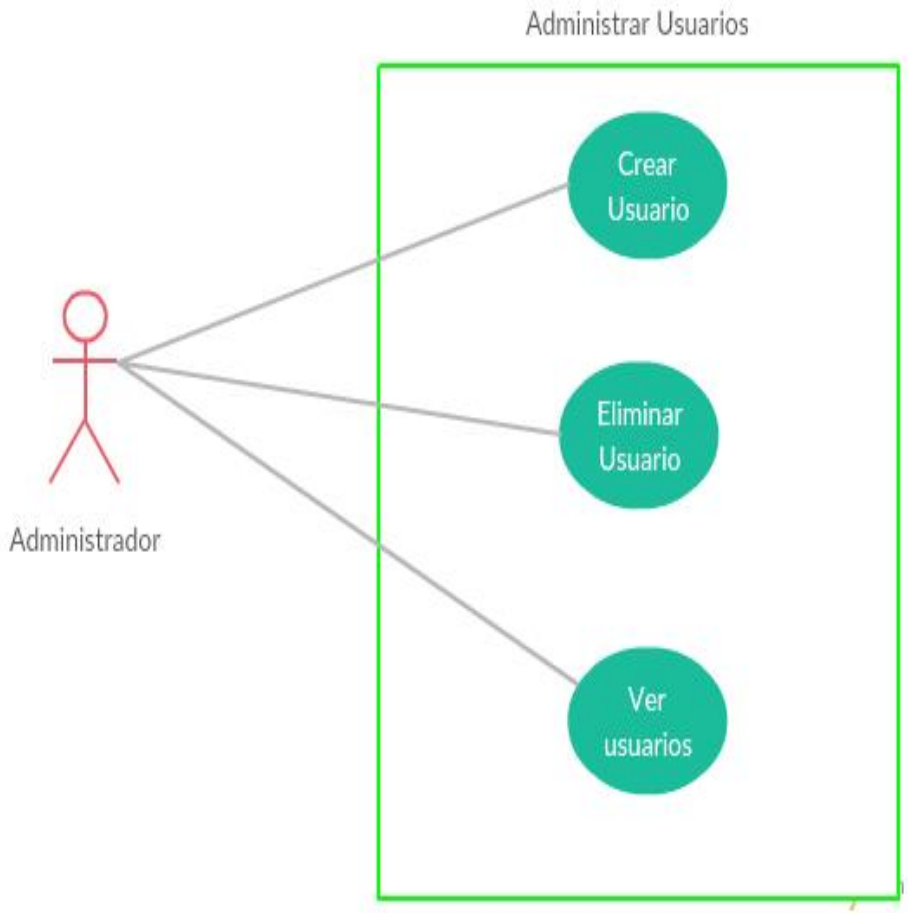


Figura 13: Caso de Uso – Administración de usuarios

Caso de Uso	Crear Elemento Semántico
Actor	Usuario – Content Designer
Descripción	Un elemento semántico es un componente que puede ser agregado al contenido de la página este elemento está compuesto de html+RDFa
Precondición	
Pos Condición	Todos los elementos semánticos creados estarán disponibles para posteriores modelos en la aplicación

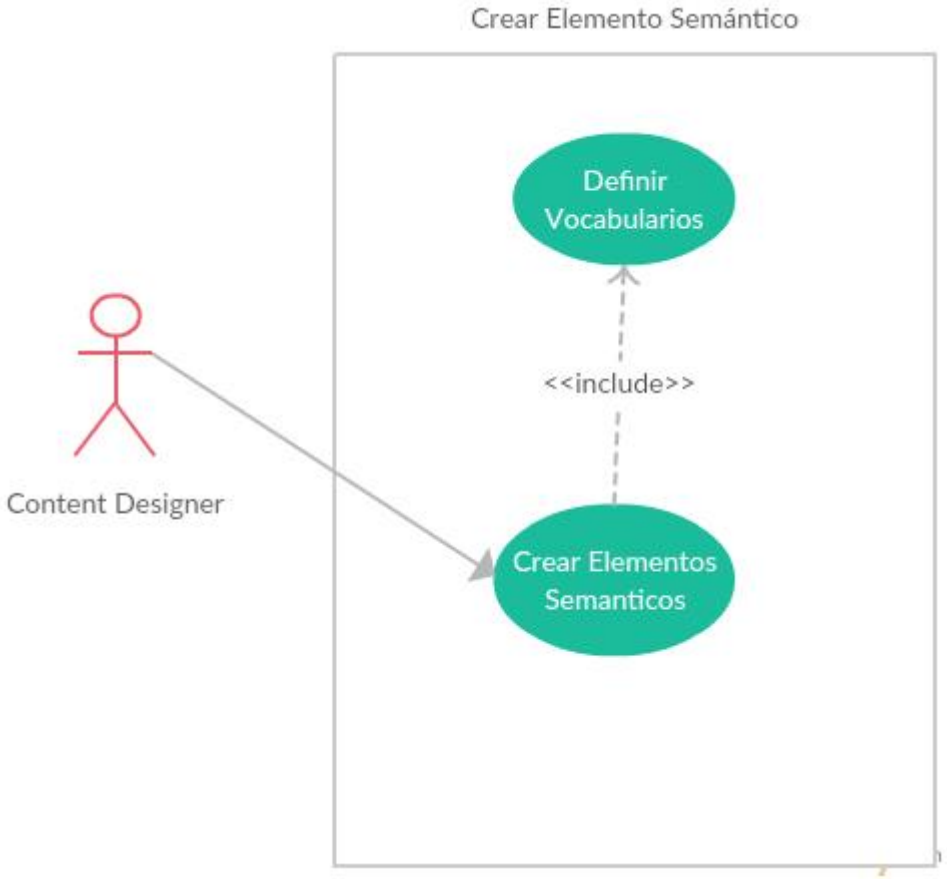


Figura 14: Caso de Uso - Crear elemento semántico

Caso de Uso	Definir Vocabularios
Actor	Usuario – Content Designer
Descripción	Un usuario necesita crear vocabularios y prefijos asociados para incluirlos en el modelo
Precondición	
Pos Condición	Vocabularios disponibles para ser agregados a la página web.

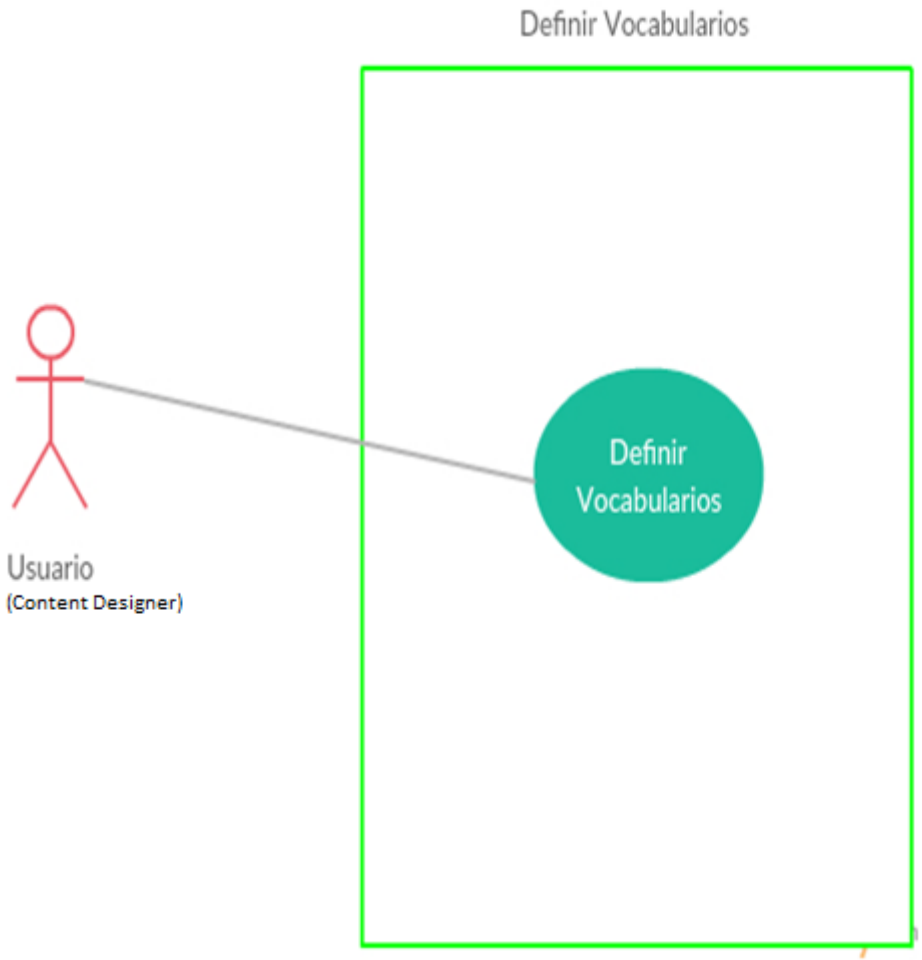


Figura 15: Caso de Uso - Definir Vocabularios

5.5 DIAGRAMA DE ACTIVIDADES

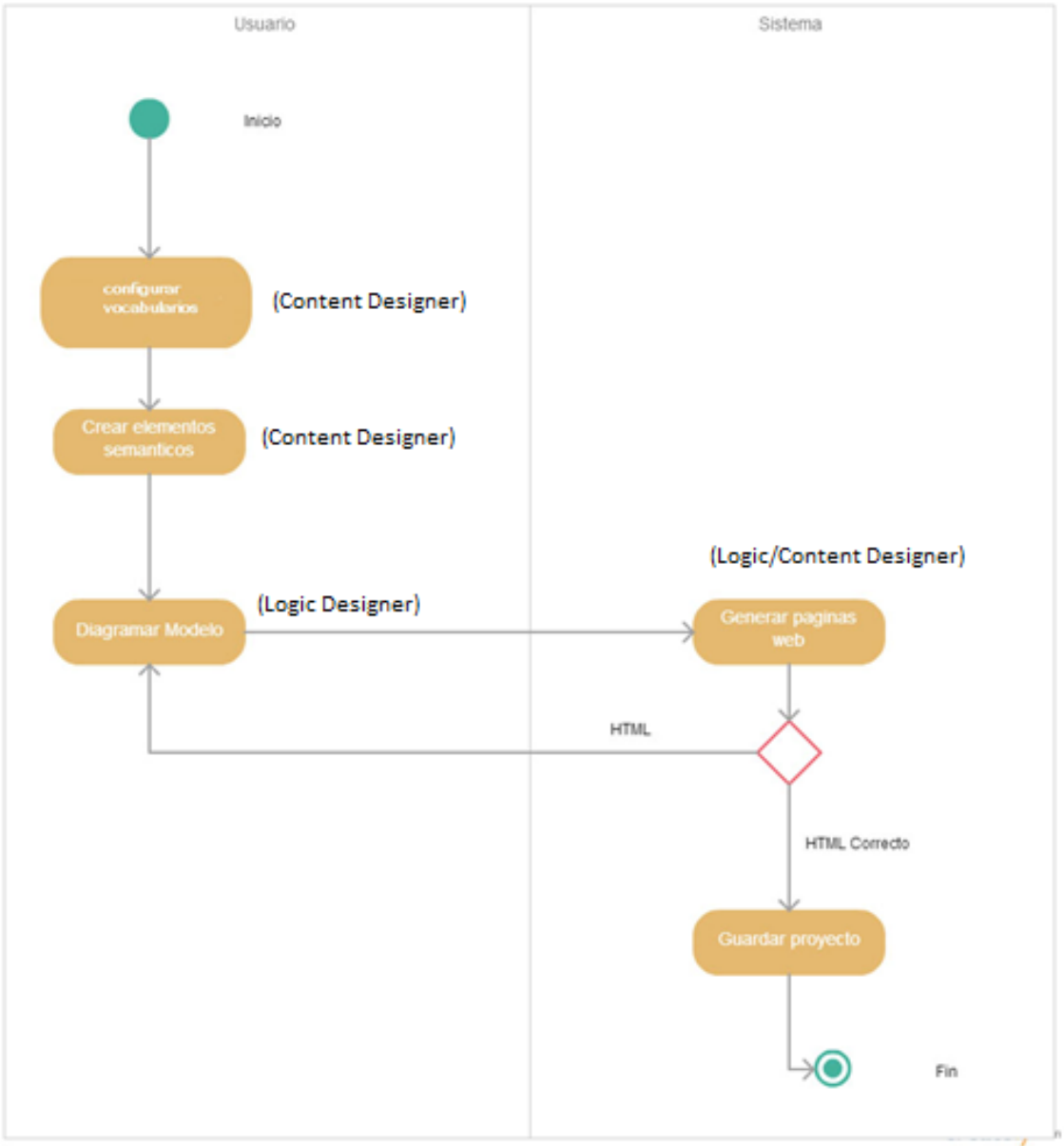


Figura 16: Diagrama de Actividades

6 DISEÑO DEL SISTEMA

6.1 MODELO DE NEGOCIO

El diagrama de clases a continuación muestra los objetos de negocio en alto nivel que componen e interactúan en la aplicación y la relación entre estos.

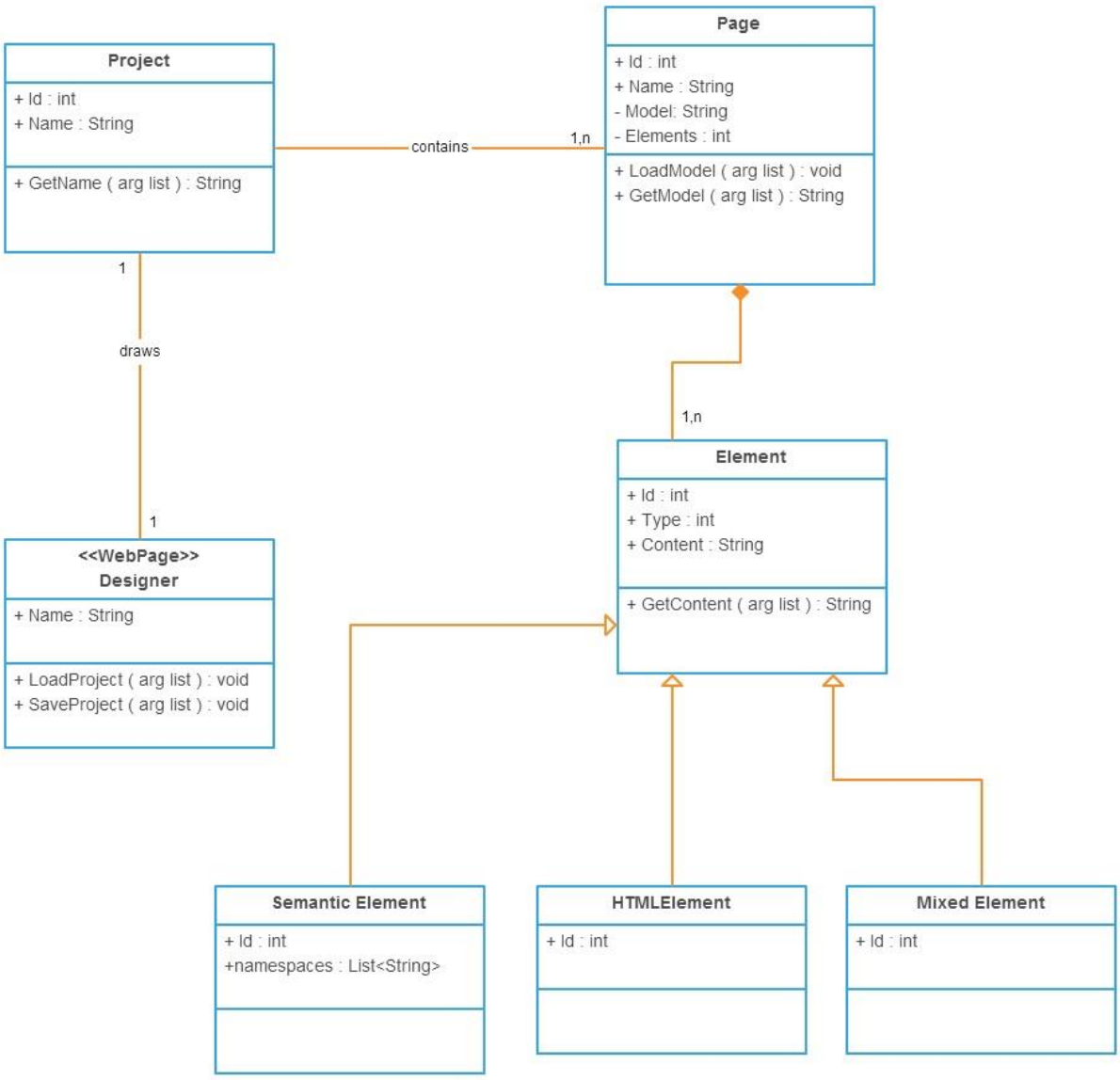


Figura 17: Diagrama de Clases

6.2 DISEÑO DE LA ARQUITECTURA DEL SISTEMA

6.2.1 Patrón arquitectónico – Arquetipo

El patrón arquitectónico o arquetipo define los aspectos fundamentales sobre la estructura del sistema, convirtiéndose en el modelo más abstracto o de más alto nivel del sistema en cuestión.

El patrón elegido para este sistema es un modelo de capas que a su vez es una especialidad del modelo cliente/servidor

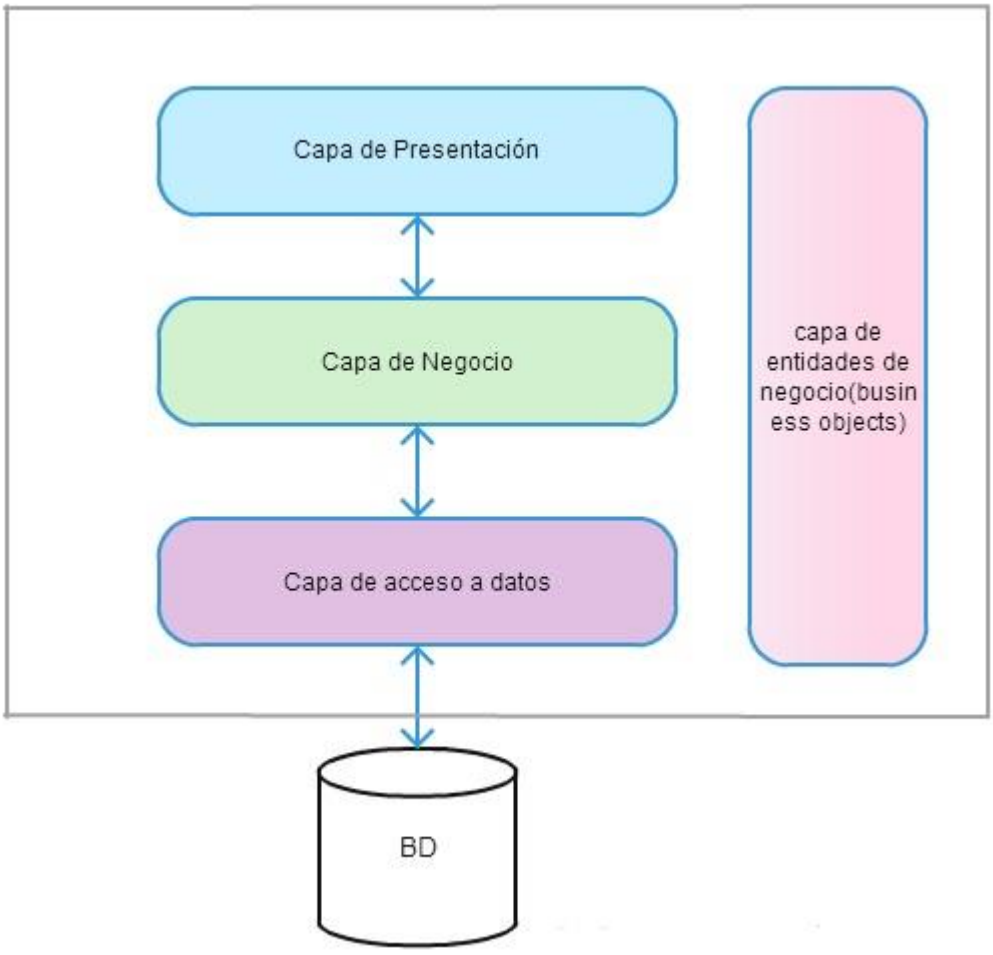


Figura 18: Arquitectura Multicapa

- Capa de Presentación.- Capa que tiene agrupa todos los componentes que ofrecen la interacción con el usuario recepción y comunicación de la información, estas pueden ser Interfaz de usuario, Servicios

- Capa de Negocio.- Capa de sirve de comunicación entre la capa de presentación y la capa de acceso a datos mediante la definición de reglas y objetos comunes a todo el proyecto (Lógica de negocio)
- Capa de Acceso a Datos.- recibe solicitudes de consulta y almacenamiento desde la capa de negocio , la principal función es abstraer el proceso de interacción con la base de datos
- Capa de Entidades de negocio.- capa transversal en la aplicación encargada de almacenar las clases/objetos comunes

6.2.2 Diagrama de Componentes

El diagrama de componentes muestra la división del Sistema en elementos más pequeños y además las relaciones entre estos componentes.

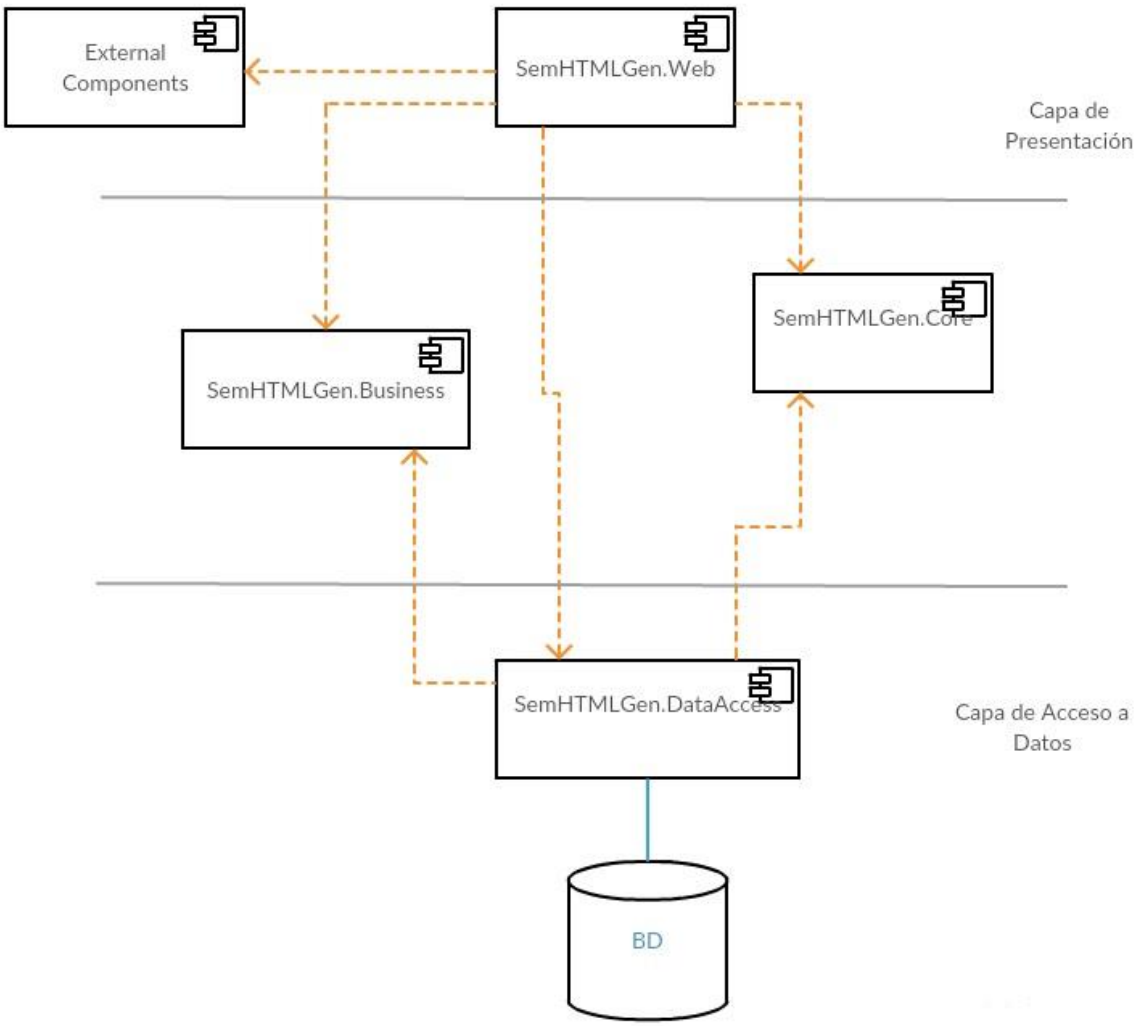


Figura 19: Diagrama de Componentes

- App.Web.- Aplicación web que contiene la interfaz de usuario y está compuesto por páginas web con formato .aspx además de librerías del lado del cliente
- App.Business.- Librería de clases de contiene las entidades de negocio (clases) que son comunes para toda la solución.
- App.Services.-proyecto de aloja los servicios web que serán invocados mediante JavaScript
- App.DataAccess.- Proyecto que contiene el código y clases que realizan el mapeo de las entidades de negocio y las operaciones directas sobre la base de datos (Operaciones CRUD)

6.3 DIAGRAMA DE PAQUETES

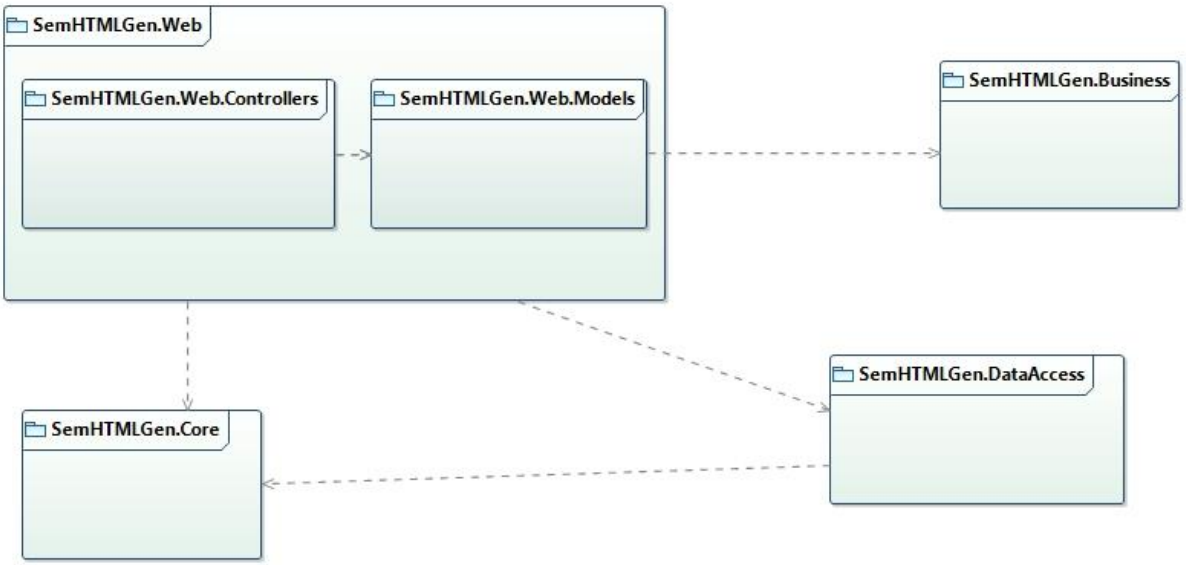


Figura 20: Diagrama de paquetes

6.4 DIAGRAMA DE CLASES DEL SISTEMA

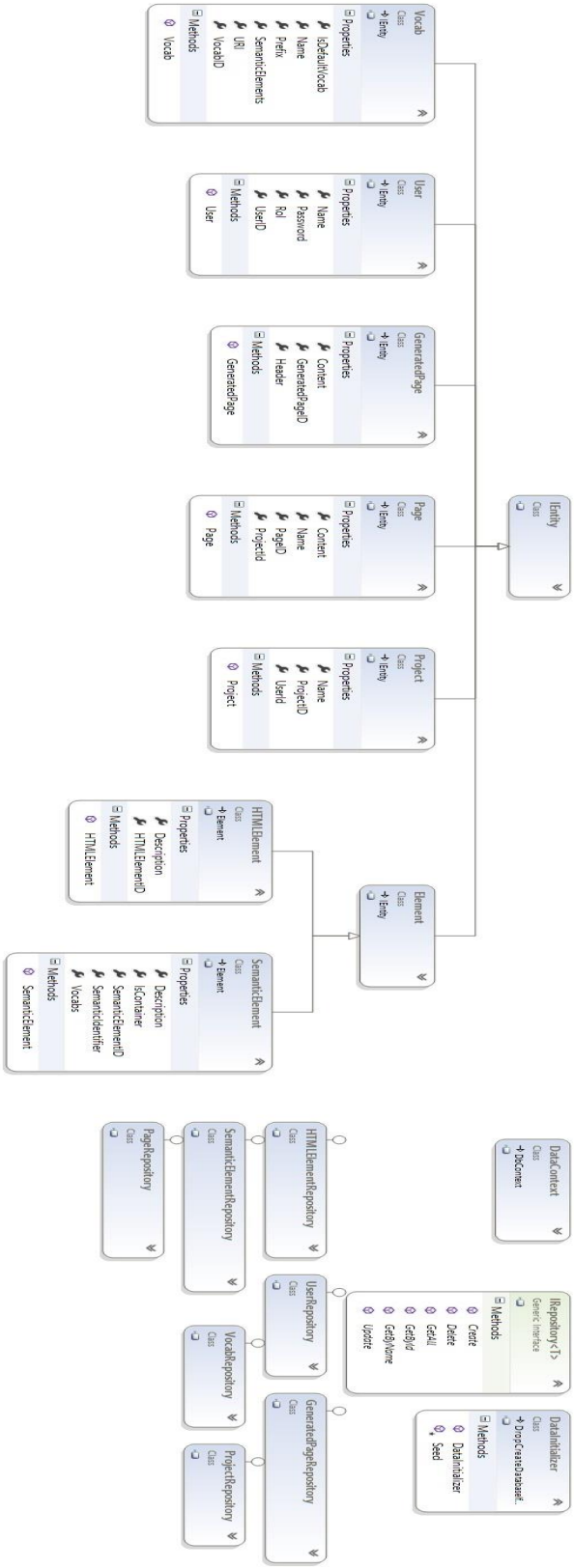


Figura 21: Diagrama de clases de sistema

6.5 MODELADO DINÁMICO (DIAGRAMAS DE SECUENCIA)

- Crear Modelo

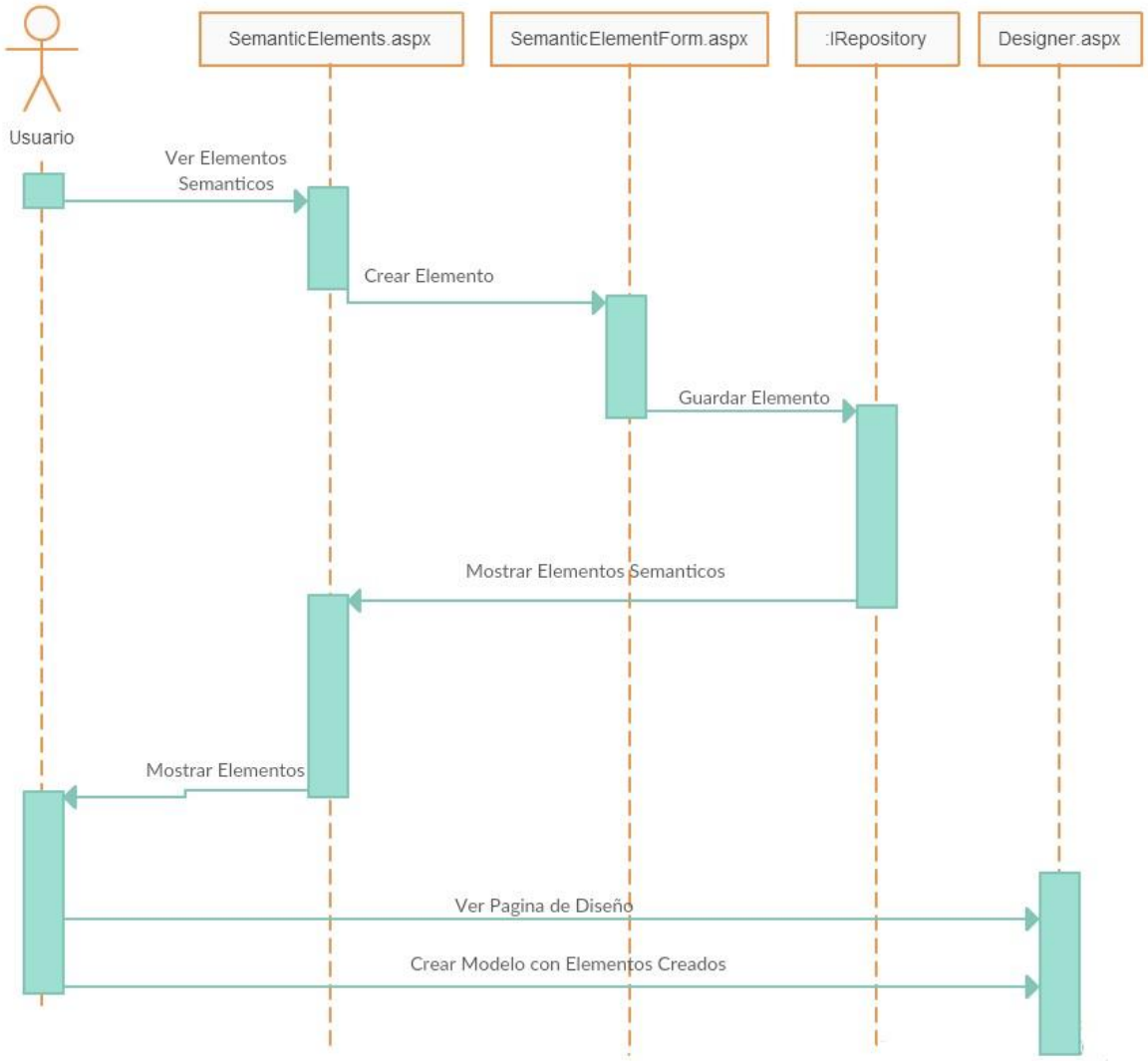


Figura 22: Secuencia Crear Modelo

- Generar Página HTML

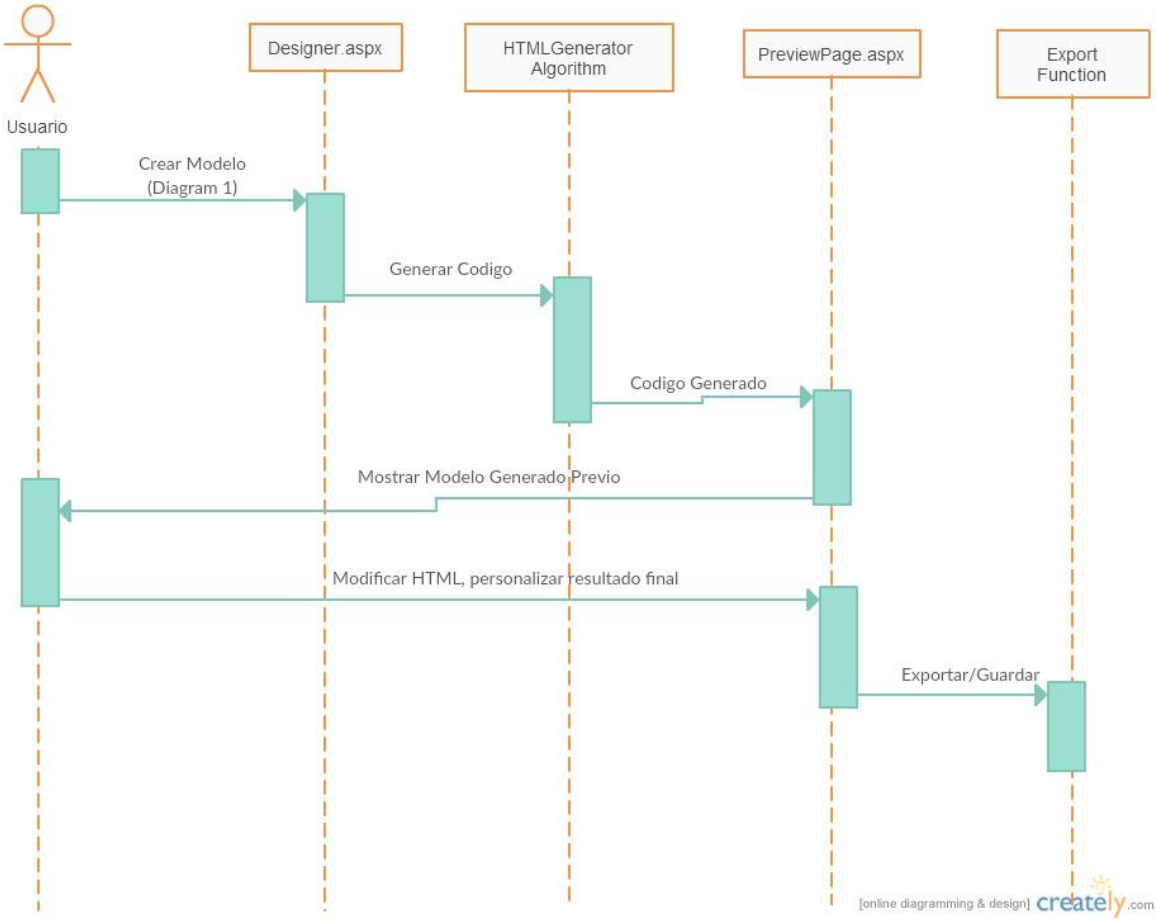


Figura 23: Generar Pagina HTML

- Guardar Proyecto

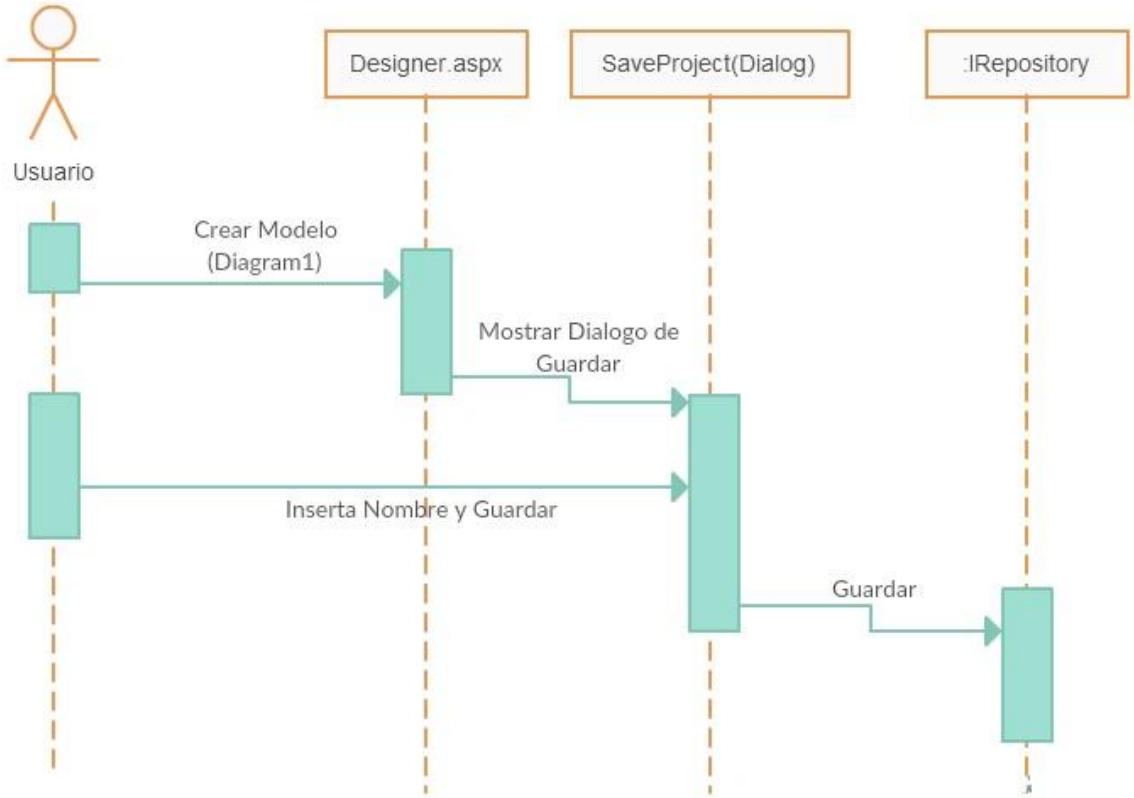


Figura 24: Guardar proyecto

- Abrir Proyecto

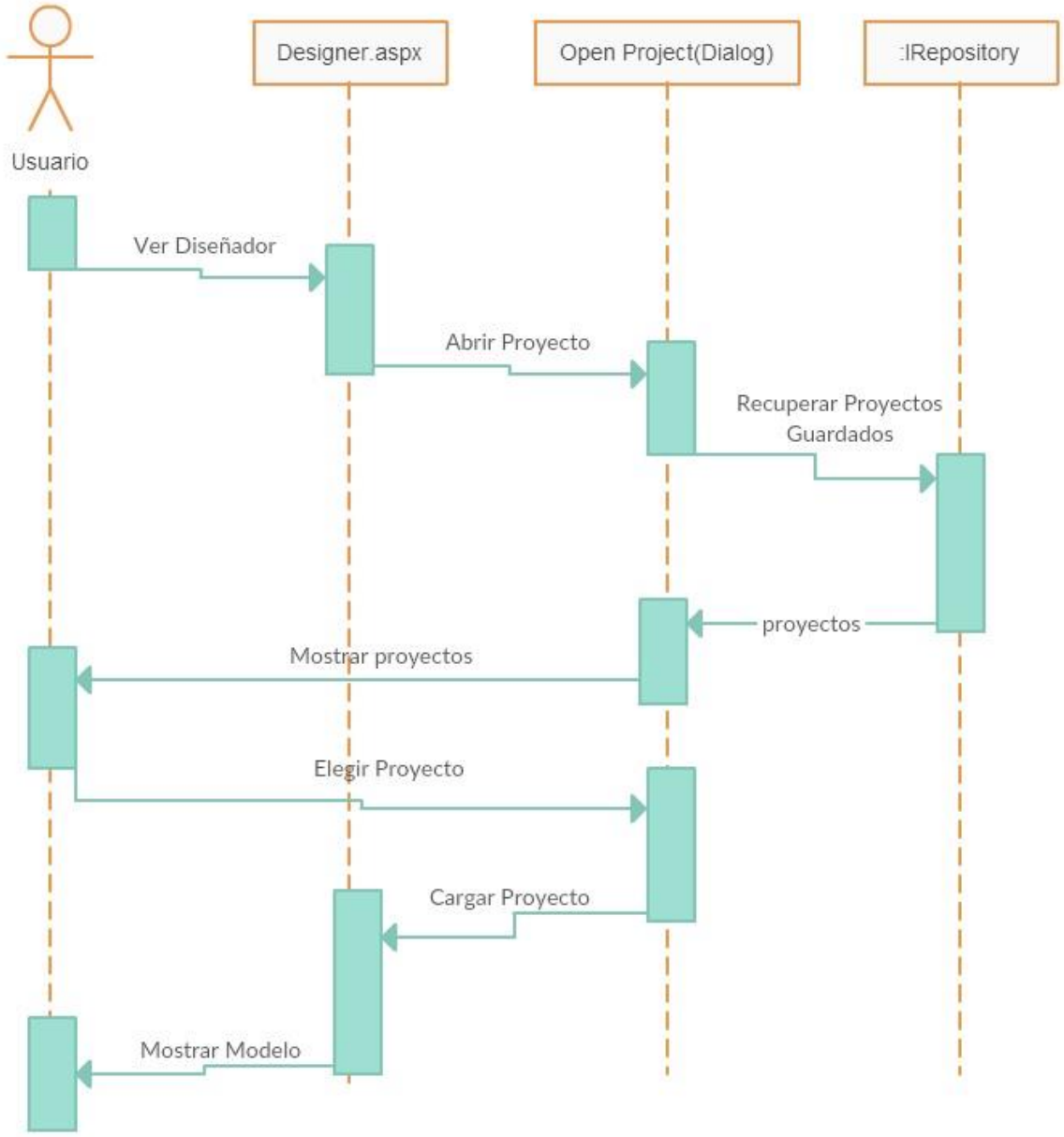


Figura 25: Abrir Proyecto

6.6 DISEÑO DE LA BASE DE DATOS

Se explicara todo el proceso de diseño de la base de datos del sistema

6.6.1 Identificación de Entidades

Se han Identificado las siguientes entidades para el sistema

- User.- Usuario del sistema
- Project.- Proyecto que contiene una o varias páginas web.
- Page.- Entidad que representa a una página web
- HTMLElement.- Entidad que representa un elemento HTML común
- SemanticElement.- Entidad que representa un elemento con datos estructurados
- GeneratedPage.- Pagina web generada en base a un modelo
- Vocab.- entidad que almacena la información de un Vocabulario RDFa Lite
- _MigrationHistory.- Historial de modificaciones en la Base de datos que se genera automáticamente.

6.6.2 Modelo Relacional

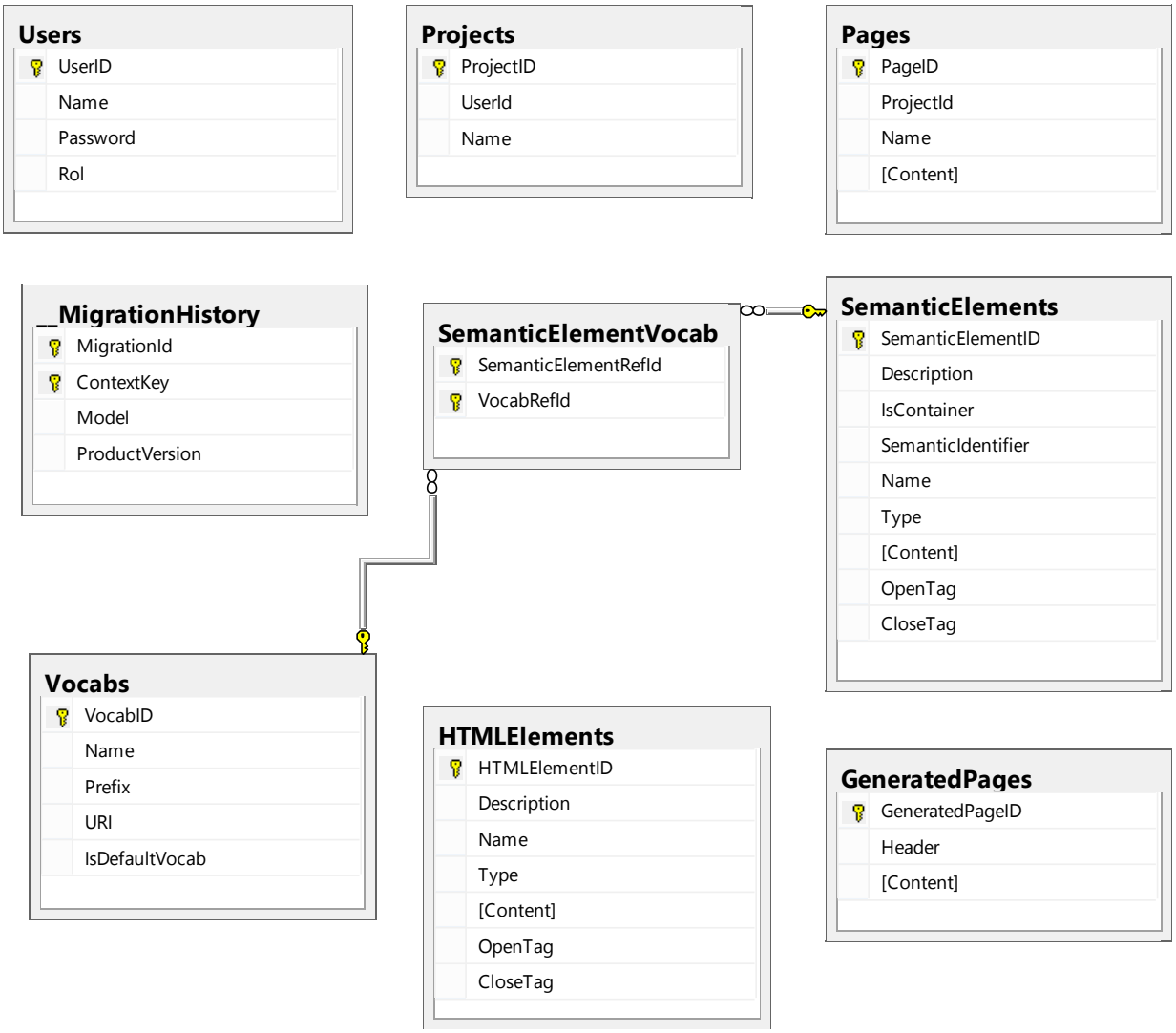


Figura 26: Modelo relacional de la Base de Datos

7 CONVERSIÓN DEL MODELO CONCEPTUAL A HTML

7.1 PROCESO DE CONVERSIÓN

Existiendo los mecanismos para incrustar información semántica en una página web y también un método para poder generar código HTML en base a un modelo conceptual se hace sencillo pensar en un sistema que pueda unir ambas tareas de la siguiente forma

- Fase 1.- Obtención de un modelo conceptual mediante una Interfaz de usuario, este modelo conceptual podrá incluir mucha más información para generar el contenido HTML por ejemplo atributos, valores, etc.

- Fase 2.- Inclusión de información semántica en el modelo de la página mediante abstracciones de objetos/atributos por ejemplo. Contactos, Personas, etc. También a través de la interfaz de usuario

- Fase 3.- Generación de HTML en base al modelo incluyendo además la información semántica proporcionada utilizando RDFa para incluir esta información en el código final haciendo este proceso totalmente transparente para el usuario final.

Sera necesario también realizar una conversión intermedia del modelo conceptual a un formato estándar que pueda ser procesado por un algoritmo para posteriormente realizar la conversión a HTML.

Para esto tenemos dos opciones:

- XML/XSLT .- una vez convertido el modelo a un documento XML podemos utilizar hojas de estilo XSLT para realizar la conversión a HTML
- JSON.- Podemos realizar la conversión del modelo conceptual a JSON que una notación mucho más ligera y sencilla sin embargo se tendrá que realizar la programación para convertir JSON a HTML o utilizar alguna herramienta de terceros (no oficial) para realizar la conversión.

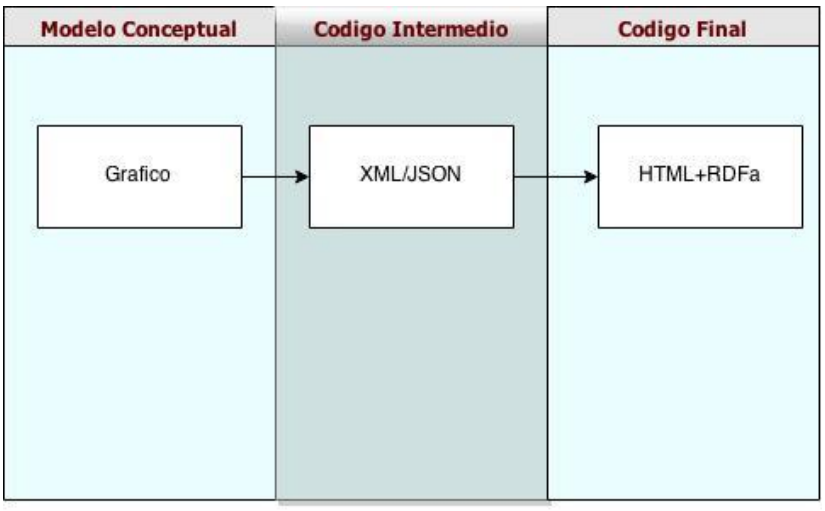


Figura 27: Proceso de conversión de un modelo conceptual a HTML

7.2 FORMATO INTERMEDIO JSON

El formato intermedio en JSON a utilizar es el siguiente:

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "id": "http://jsonschema.net",
  "type": "object",
  "properties": {
    "shapes": {
      "id": "http://jsonschema.net/shapes",
      "type": "array",
      "items": {
        "id": "http://jsonschema.net/shapes/2",
        "type": "object",
        "properties": {
          "id": {
            "id": "http://jsonschema.net/shapes/2/id",
            "type": "string"
          },
          "hover": {
            "id": "http://jsonschema.net/shapes/2/hover",
            "type": "object",
            "properties": {}
          },
          "cursor": {
            "id": "http://jsonschema.net/shapes/2/cursor",
            "type": "string"
          },
          "content": {
            "id": "http://jsonschema.net/shapes/2/content",
            "type": "object",
            "properties": {
              "align": {
                "id": "http://jsonschema.net/shapes/2/content/align",
                "type": "string"
              }
            }
          },
          "selectable": {
            "id": "http://jsonschema.net/shapes/2/selectable",
```

```
    "type": "boolean"
  },
  "serializable": {
    "id": "http://jsonschema.net/shapes/2/serializable",
    "type": "boolean"
  },
  "enable": {
    "id": "http://jsonschema.net/shapes/2/enable",
    "type": "boolean"
  },
  "type": {
    "id": "http://jsonschema.net/shapes/2/type",
    "type": "string"
  },
  "path": {
    "id": "http://jsonschema.net/shapes/2/path",
    "type": "string"
  },
  "autoSize": {
    "id": "http://jsonschema.net/shapes/2/autoSize",
    "type": "boolean"
  },
  "visual": {
    "id": "http://jsonschema.net/shapes/2/visual",
    "type": "null"
  },
  "x": {
    "id": "http://jsonschema.net/shapes/2/x",
    "type": "number"
  },
  "y": {
    "id": "http://jsonschema.net/shapes/2/y",
    "type": "integer"
  },
  "minWidth": {
    "id": "http://jsonschema.net/shapes/2/minWidth",
    "type": "integer"
  },
  "minHeight": {
    "id": "http://jsonschema.net/shapes/2/minHeight",
```

```
    "type": "integer"
  },
  "width": {
    "id": "http://jsonschema.net/shapes/2/width",
    "type": "integer"
  },
  "height": {
    "id": "http://jsonschema.net/shapes/2/height",
    "type": "integer"
  },
  "editable": {
    "id": "http://jsonschema.net/shapes/2/editable",
    "type": "object",
    "properties": {
      "connect": {
        "id": "http://jsonschema.net/shapes/2/editable/connect",
        "type": "boolean"
      },
      "tools": {
        "id": "http://jsonschema.net/shapes/2/editable/tools",
        "type": "array",
        "items": {}
      }
    }
  },
  "connectors": {
    "id": "http://jsonschema.net/shapes/2/connectors",
    "type": "array",
    "items": {
      "id": "http://jsonschema.net/shapes/2/connectors/4",
      "type": "object",
      "properties": {
        "name": {
          "id": "http://jsonschema.net/shapes/2/connectors/4/name",
          "type": "string"
        },
        "Description": {
          "id": "http://jsonschema.net/shapes/2/connectors/4/Description",
          "type": "string"
        }
```

```
        }
       }
      },
      "rotation": {
       "id": "http://jsonschema.net/shapes/2/rotation",
       "type": "object",
       "properties": {
        "angle": {
         "id": "http://jsonschema.net/shapes/2/rotation/angle",
         "type": "integer"
        }
       }
      },
      "elementName": {
       "id": "http://jsonschema.net/shapes/2/elementName",
       "type": "string"
      },
      "caption": {
       "id": "http://jsonschema.net/shapes/2/caption",
       "type": "string"
      }
     },
     "required": [
      "id",
      "hover",
      "cursor",
      "content",
      "selectable",
      "serializable",
      "enable",
      "type",
      "path",
      "autoSize",
      "visual",
      "x",
      "y",
      "minWidth",
      "minHeight",
      "width",
      "height",
```

```
        "editable",
        "connectors",
        "rotation",
        "elementName",
        "caption"
      ]
    },
    "required": [
      "2"
    ]
  },
  "connections": {
    "id": "http://jsonschema.net/connections",
    "type": "array",
    "items": {
      "id": "http://jsonschema.net/connections/1",
      "type": "object",
      "properties": {
        "id": {
          "id": "http://jsonschema.net/connections/1/id",
          "type": "string"
        },
        "hover": {
          "id": "http://jsonschema.net/connections/1/hover",
          "type": "object",
          "properties": {
            "stroke": {
              "id": "http://jsonschema.net/connections/1/hover/stroke",
              "type": "object",
              "properties": {}
            }
          }
        },
        "cursor": {
          "id": "http://jsonschema.net/connections/1/cursor",
          "type": "string"
        },
        "content": {
          "id": "http://jsonschema.net/connections/1/content",
          "type": "object",
```

```
    "properties": {
      "align": {
        "id": "http://jsonschema.net/connections/1/content/align",
        "type": "string"
      },
      "color": {
        "id": "http://jsonschema.net/connections/1/content/color",
        "type": "string"
      }
    }
  },
  "selectable": {
    "id": "http://jsonschema.net/connections/1/selectable",
    "type": "boolean"
  },
  "serializable": {
    "id": "http://jsonschema.net/connections/1/serializable",
    "type": "boolean"
  },
  "enable": {
    "id": "http://jsonschema.net/connections/1/enable",
    "type": "boolean"
  },
  "startCap": {
    "id": "http://jsonschema.net/connections/1/startCap",
    "type": "string"
  },
  "endCap": {
    "id": "http://jsonschema.net/connections/1/endCap",
    "type": "string"
  },
  "points": {
    "id": "http://jsonschema.net/connections/1/points",
    "type": "array",
    "items": {}
  },
  "stroke": {
    "id": "http://jsonschema.net/connections/1/stroke",
    "type": "object",
    "properties": {
```

```
      "width": {
        "id": "http://jsonschema.net/connections/1/stroke/width",
        "type": "integer"
      },
      "color": {
        "id": "http://jsonschema.net/connections/1/stroke/color",
        "type": "string"
      }
    }
  },
  "selection": {
    "id": "http://jsonschema.net/connections/1/selection",
    "type": "object",
    "properties": {
      "handles": {
        "id": "http://jsonschema.net/connections/1/selection/handles",
        "type": "object",
        "properties": {
          "width": {
            "id": "http://jsonschema.net/connections/1/selection/handles/width",
            "type": "integer"
          },
          "height": {
            "id": "http://jsonschema.net/connections/1/selection/handles/height",
            "type": "integer"
          },
          "fill": {
            "id": "http://jsonschema.net/connections/1/selection/handles/fill",
            "type": "object",
            "properties": {
              "color": {
                "id": "http://jsonschema.net/connections/1/selection/handles/fill/color",
                "type": "string"
              }
            }
          },
          "stroke": {
            "id": "http://jsonschema.net/connections/1/selection/handles/stroke",
            "type": "object",
            "properties": {
```

```
            "color": {
              "id": "http://jsonschema.net/connections/1/selection/handles/stroke/color",
              "type": "string"
            }
          }
        }
      }
    }
  }
},
"editable": {
  "id": "http://jsonschema.net/connections/1/editable",
  "type": "object",
  "properties": {
    "tools": {
      "id": "http://jsonschema.net/connections/1/editable/tools",
      "type": "array",
      "items": {}
    }
  }
},
"toX": {
  "id": "http://jsonschema.net/connections/1/toX",
  "type": "integer"
},
"toY": {
  "id": "http://jsonschema.net/connections/1/toY",
  "type": "integer"
},
"to": {
  "id": "http://jsonschema.net/connections/1/to",
  "type": "object",
  "properties": {
    "shapeId": {
      "id": "http://jsonschema.net/connections/1/to/shapeId",
      "type": "string"
    }
  }
},
"from": {
```

```
            "id": "http://jsonschema.net/connections/1/from",
            "type": "object",
            "properties": {
              "shapeId": {
                "id": "http://jsonschema.net/connections/1/from/shapeId",
                "type": "string"
              },
              "connector": {
                "id": "http://jsonschema.net/connections/1/from/connector",
                "type": "string"
              }
            }
          }
        }
      }
    }
  },
  "required": [
    "shapes",
    "connections"
  ]
}
```

Dentro de este esquema las dos clases más importantes a tomar en cuenta son:

Shape: representa a un elemento grafico del modelo que posteriormente será mapeado a un elemento de tipo “Element” que ha sido definido en la fase de diseño del sistema, esta clase tiene la siguiente estructura:

```
public class Element : IEntity
{
    public String Name { get; set; }
    public String Type { get; set; }

    public String Content { get; set; }
    public string OpenTag { get; set; }
    public string CloseTag { get; set; }
}
```

Figura 28: Clase Element

7.3 ALGORITMO DE CONVERSIÓN DE FORMATO INTERMEDIO A HTML

A continuación se describe el algoritmo de conversión de la entrada del usuario en el código resultante:

- Entradas:
  - Identificador del elemento inicial: el elemento “raíz”, es el elemento padre de todos los demás elementos que componen la página web, dado que el algoritmo es totalmente recursivo este parámetro marca el punto de inicio
  - “m” figuras {m > 0}: objetos de tipo “Shape”, son los elementos que componen el contenido de la página web, estos son elementos que están mapeados a elementos semánticos dentro del sistema.
  - “n” relaciones {n > 0}: objetos de tipo “Connection”, son las conexiones entre los diferentes elementos que componen la página web según el modelo.

- Salida: texto en formato HTML
- Pasos:
  - Paso 1: definir una cadena de texto inicial vacía/resultado.
  - Paso 2: en base al parámetro identificador obtener el objeto al que pertenece ese identificador (mapeo de Objeto “Shape” del modelo JSON a “Element” en C#).
  - Paso 3: agregar la etiqueta de apertura del elemento.
  - Paso 4: agregar el contenido(html+rdfa) del elemento a la cadena
  - Paso 5: obtener los objetos “hijo” del elemento
    Paso 5.1: Para cada objeto “hijo” llamar el mismo método recursivamente enviando como parámetros el id de cada hijo y las lista de “Shape” y “Connection”.
  - Paso 6: devolver la cadena de texto con el contenido apilado recursivamente

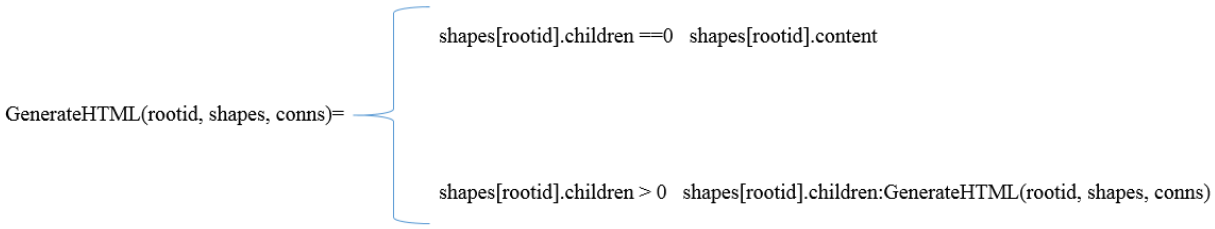


Figura 29: Algoritmo Recursivo de generación de HTML

7.3.1 Pseudocódigo e implementación

```
ALGORITMO GenerarHTML
1: ENTRADA: id del elemento padre, figuras y conexiones (rootid, shapes, connections)
2: SALIDA: cadena de texto HTML
3: VARIABLES: objetoPadre, htmlResult
4: figuraRaiz ← ObjetnerFiguraRaiz(rootid)
5: objetoPadre ← ObjetnerObjetoPadre(figuraRaiz) //mapping
6: htmlResult += objetoPadre.obtenerContenido
7: INICIO
8:        PARA i=0 HASTA objetoPadre.Hijos.cuenta CON INCREMENTO +1
9:                htmlResult += GenerarHTML(objetoPadre.Hijos[i].id, shapes, connections)
10:       FIN_PARA
11:FIN
```

```
public string Generate(string rootId, List<Shape> shapes, List<Connection> connections)
{
   string htmlContent = "";
   Shape sshp = GetElement(rootId, shapes);
   SemanticElement semElem = repo.GetById(Convert.ToInt32(sshp.elementName));

   htmlContent += semElem.OpenTag;

   if (semElem != null)
   {
      htmlContent += semElem.Content;
   }

   List<Shape> children = GetChildren(rootId, shapes, connections);

   foreach (Shape child in children)
   {
      htmlContent += Generate(child.id, shapes, connections);
   }

   htmlContent += semElem.CloseTag;

   return htmlContent;
}
```

8 IMPLEMENTACIÓN DEL SISTEMA

8.1 EVALUACIÓN Y SELECCIÓN DE TECNOLOGIAS A UTILIZAR

Se ha decidido el uso de tecnologías Microsoft para desarrollar el sistema, la principal razón es la compatibilidad y rendimiento en entornos Windows además de la robustez de las herramientas existentes.

8.1.1 Plataforma

.NET framework [18] es la plataforma para desarrollo de aplicaciones para Windows desarrollada por Microsoft.

Este entorno de ejecución nos permite la creación, compilación y generación de aplicaciones de escritorio y web así como también servicios web ofreciendo a los programadores un contexto coherente de programación incluso en diferentes lenguajes como C# y Visual Basic.

Entre las principales ventajas de esta tecnología podemos mencionar la robustez, amplia documentación y compatibilidad con el sistema operativo Windows lo que permite aprovechar al máximo el uso de recursos.

Existen también una gran cantidad de herramientas y librerías de terceros compatibles con el framework .NET ampliando mucho más nuestras posibilidades de desarrollo.

8.1.2 Servidor

8.1.2.1 Servidor web

Internet Information Services (IIS) [19] es un conjunto de servicios incluidos en el sistema operativo Windows cuya principal función es la de un Servidor Web.

Originalmente venia incluida en versiones específicas de Windows como Windows Server, pero en las últimas versiones de este sistema operativo ya viene pre instalado como un componente que se puede habilitar.

IIS ha sufrido varias mejoras en los últimos años las cuales lo han convertido en un servidor web muy potente y de alto rendimiento entre sus principales características están la de poseer un arquitectura modular, un modelo de ejecución que aumenta la seguridad y permite optimizar el rendimiento. Originalmente soportaba solo aplicaciones web desarrolladas con la plataforma asp.net pero ahora es posible usarlo con otros lenguajes de programación como PHP.

8.1.2.2 Servidor de Base de datos

SQL Server [20] es un sistema de creación, administración y análisis de bases de datos relacionales propiedad de la compañía Microsoft.

Aunque puede ser utilizado con diversos lenguajes de programación es mayormente utilizado en proyectos desarrollados sobre la plataforma .NET debido a su natural compatibilidad al ser parte de la misma compañía.

Es un sistema de gestión de Bases de datos muy robusto y potente que provee además de varias herramientas adicionales para el desarrollo y análisis de bases de datos una variación propia de lenguaje de consultas conocido como T-SQL que enriquece al SQL con características de lenguajes de programación como variables, funciones predefinidas y estructuras propias de consulta.

8.1.2.3 Tecnología de programación en el servidor

ASP.NET [21] es el framework para el desarrollo de aplicaciones web de Microsoft para la plataforma .NET.

ASP.NET incluye un modelo de desarrollo web unificado que incluye los servicios necesarios para crear sitios y aplicaciones web utilizando también otros lenguajes propiedad de Microsoft como C# y Visual Basic.NET.

Actualmente ASP.NET brinda dos enfoques de programación web que son el basado en formularios Web Forms y MVC basado en el patrón Modelo-Vista-Controlador. Si bien ambos enfoques son diferentes en cuanto a la filosofía de programación en el fondo comparten las características comunes de la plataforma asp.net como son seguridad, entorno de ejecución, lenguaje intermedio y las herramientas de desarrollo comunes.

8.1.2.4 Entity Framework (EF)

Entity Framework (EF) [22] es un asignador objeto-relacional (ORM) que permite a los desarrolladores de .NET trabajar con datos relacionales usando objetos específicos del dominio. Elimina la necesidad de la mayor parte del código de acceso a datos que los desarrolladores suelen tener que escribir.

Entity Framework permite crear un modelo escribiendo código o usando cuadros y líneas en EF Designer. Ambos enfoques se pueden usar con una base de datos existentes o para crear una nueva base de datos.

Existen tres enfoques o estrategias de desarrollo que se pueden seguir cuando se utiliza EF como tecnología de acceso a datos, estos enfoques son:

- Database First.- El Modelo de negocio o clases del negocio de crean a partir de una base de datos existente.
- Model First.- Se crea un modelo conceptual del negocio y a partir de este modelo se generan las clases del dominio a utilizar en la aplicación
- Code First.- Disponible A partir de la versión EF 4.1, es un enfoque simplificado que permite mapear clases POCO a una base de datos usando convenciones y anotaciones en el código.

8.1.3 Cliente

8.1.3.1 Lenguaje de programación en el cliente

JavaScript [23] es el lenguaje de programación en el lado del cliente por excelencia, es un lenguaje interpretado basado en prototipos con funciones de primera clase que soporta los estilos de programación funcional, Orientada a objetos e imperativa.

Aunque originalmente y de manera más común es usado como lenguaje de scripts para páginas web que se ejecutan en un navegador es también usado en entornos NO-WEB como Node.JS.

8.1.3.2 KendoUI

Kendo UI [24] es un framework HTML5 basado en Jquery para la construcción de aplicaciones web. El framework contiene varios controles de interfaz de usuario, un marco de visualización, un framework para móviles auto descriptivo , y todas las herramientas necesarias para el desarrollo de aplicaciones web HTML5, como fuentes de datos, plantillas, componentes de arrastrar-soltar y mucho más.

8.1.3.3 JQuery

JQuery es una librería para JavaScript rápida, pequeña y rica en funciones [25].

Esta librería hace cosas como el recorrido y manipulación de documentos HTML, manejo de eventos, animación y Ajax mucho más simple con un API fácil de usar que es compatible con una gran cantidad de navegadores. Con una combinación de versatilidad y extensibilidad JQuery ha cambiado le forma en que millones de personas escriben JavaScript [25].

8.2 INTERFAZ DE USUARIO

La interfaz de usuario consiste en una Interfaz Web con características de Rich Internet Applications (RIA) para brindar al usuario una adecuada experiencia de usabilidad.

Login

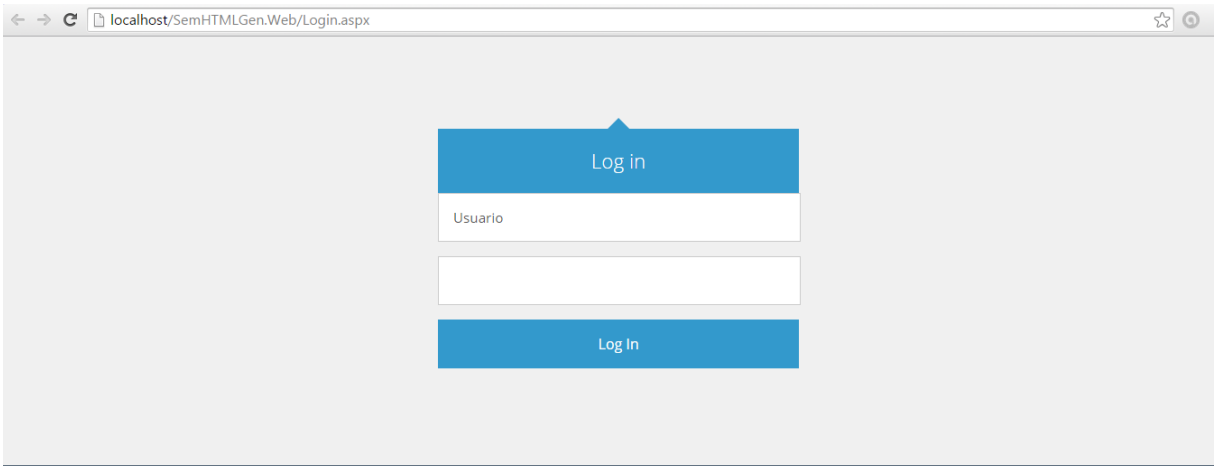


Figura 30: pantalla de login

Pantalla de Diseño

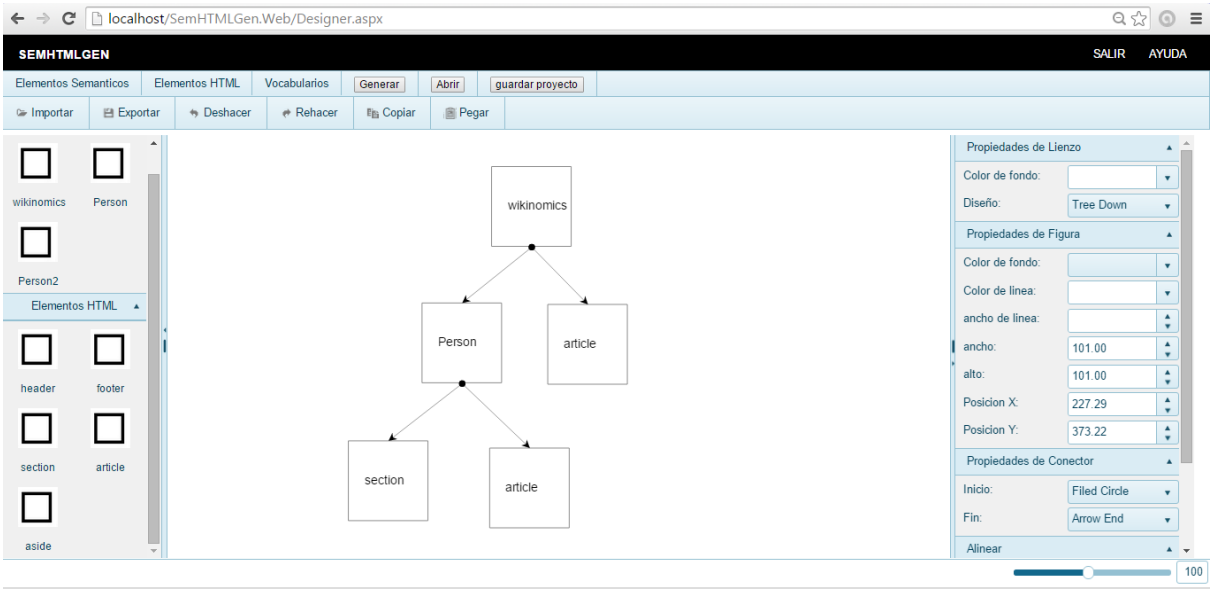


Figura 31: Pantalla de diseño

Pantalla de Visualización Y Edición de página generada

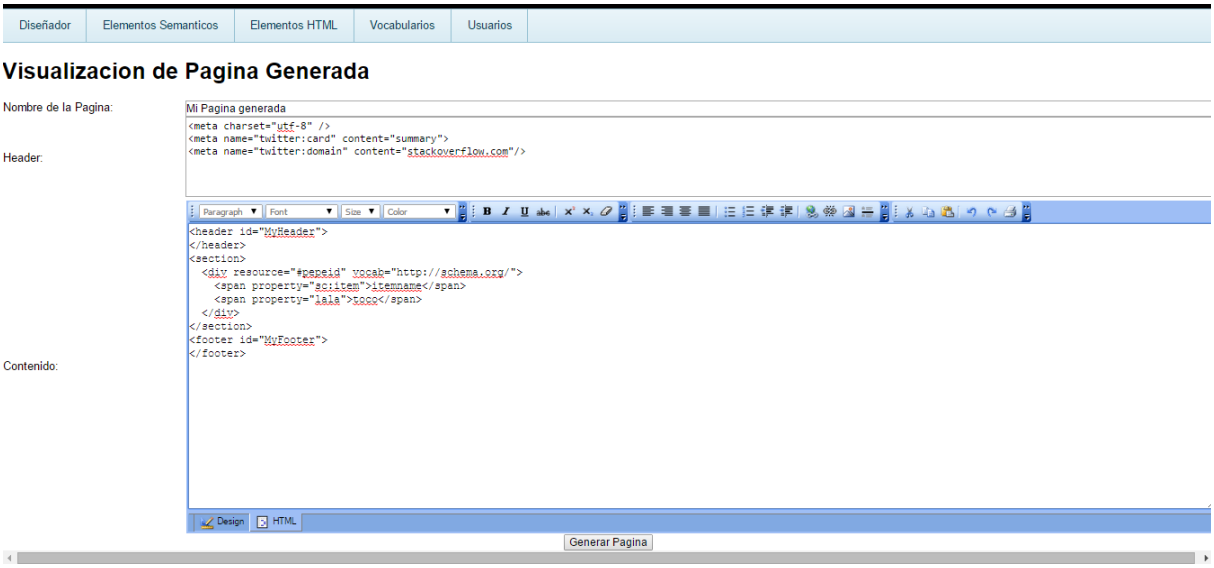


Figura 32: Pantalla de Edición del HTML Generado

Pantalla Maestro/Detalle

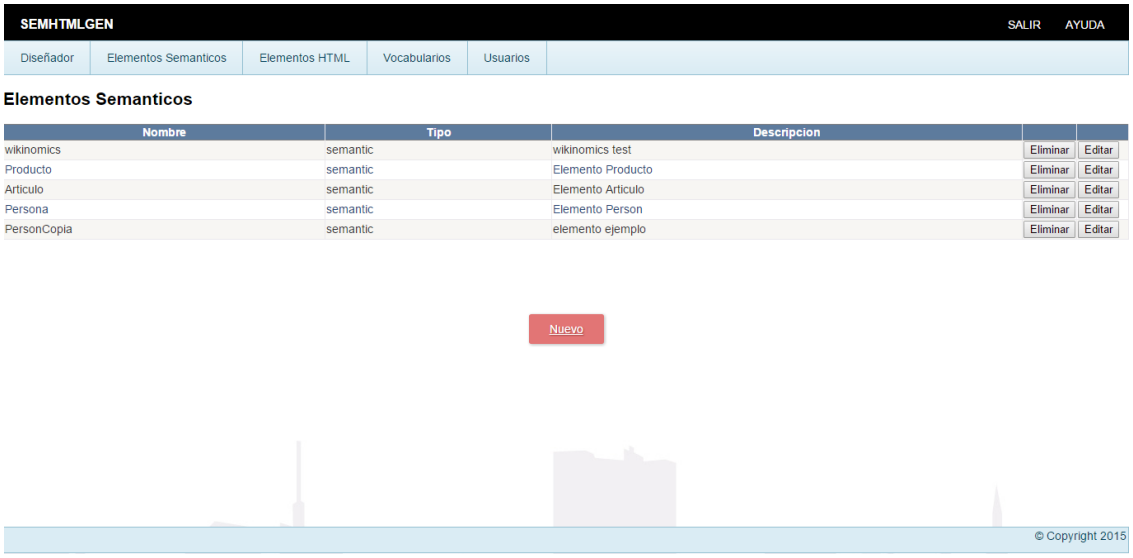


Figura 33: Témplate de registros

Formularios

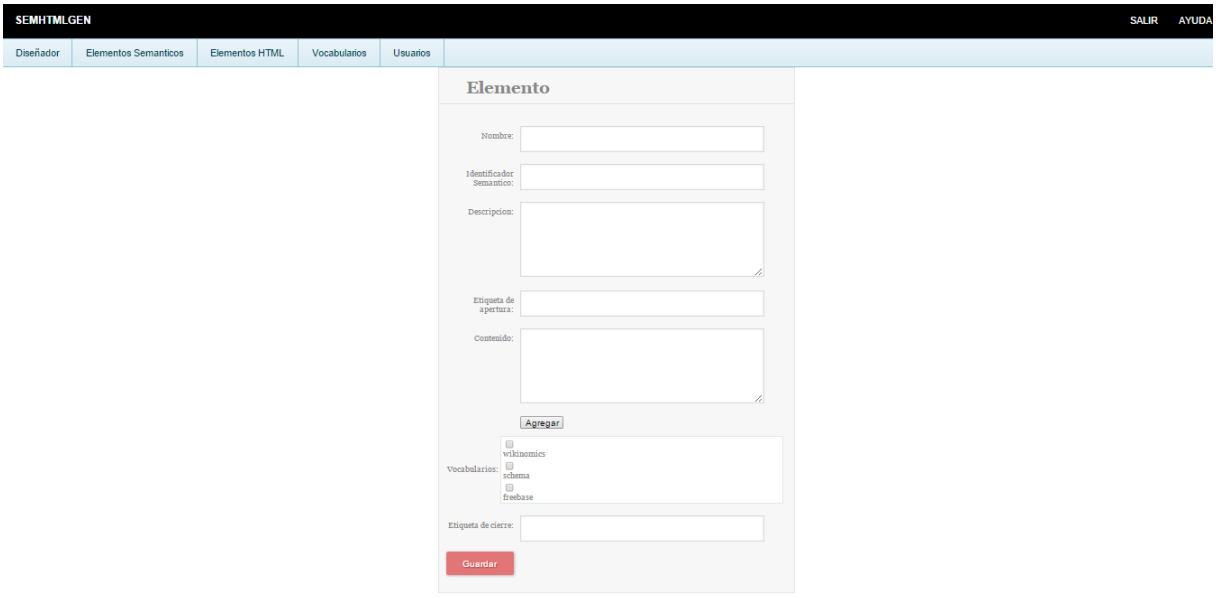


Figura 34: Formulario

8.3 ESTRUCTURA DE LA SOLUCIÓN

El código fuente está estructurado en una Solución de Visual Studio que contiene 4 diferentes proyectos que corresponden a cada componente del diseño arquitectónico, se incluye además un proyecto que contiene las pruebas unitarias. Se puede apreciar la estructura en la siguiente imagen:

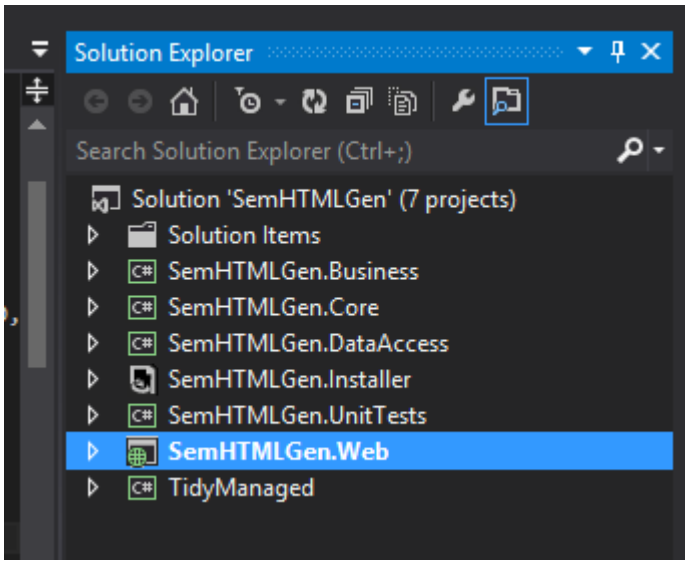


Figura 35: Estructura de la Solución (código fuente)

8.4 PATRONES DE DISEÑO

8.4.1 Page Controller

Este patrón es una variación del conocido patrón MVC, es adecuado cuando se desea construir páginas web dinámicamente pero la navegación entre estas páginas es mayormente estática [26].

La clase controladora (Page Controller) recibe una solicitud de página mediante una URL, extrae los datos relevantes, invoca cualquier actualización necesaria al modelo y redirecciona la solicitud a la vista. La vista a su vez depende del modelo para recuperar los datos que deben ser mostrados [26].

Definir un controlador separado aisla el modelo de los detalles relativos a la solicitud web. Por ejemplo manejo de sesiones, o el uso de QueryStrings o campos ocultos para pasar parámetros a la página. En esta forma básica se crea un controlador para cada enlace en la aplicación web, esto mantiene los controladores en un estado simple y manejable porque solo deben ocuparse de una acción a la vez [26].

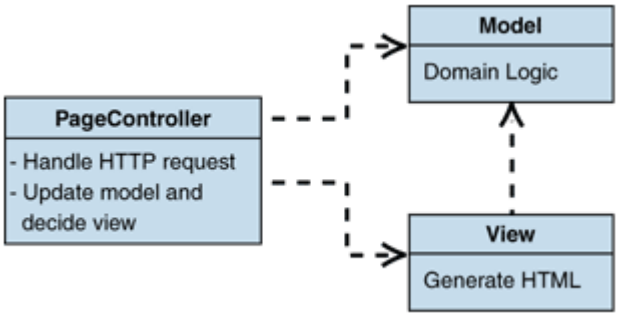


Figura 36: Patrón Page Controller

Crear un controlador separado para cada página o acción puede causar duplicación de código. Por lo tanto es necesario crear una clase base para incorporar funciones comunes como la validación de parámetros [26].

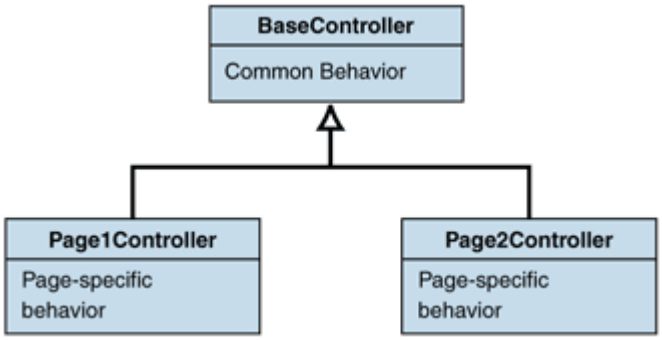


Figura 37: Variación de Page Controller

El patrón Page Controller es una necesidad tan común que la mayoría de los frameworks para desarrollo de aplicaciones web proveen una implementación por defecto. La mayoría de los frameworks incorporan el patrón "Page Controller" en forma de una página de servidor (por ejemplo ASP, JSP y PHP). Las páginas de servidor actualmente combinan funciones de vista y controlador y no proveen la separación deseada entre el código de presentación y el código de control [26].

Desafortunadamente, algunos de los frameworks hacen muy fácil mezclar el código relacionado con la vista con el código relacionado con el controlador y hacen difícil separar apropiadamente la lógica de control. Como resultado, el enfoque "Page Controller" ha desarrollado una mala reputación con muchos desarrolladores. Ahora, muchos desarrolladores asocian el patrón con un mal diseño y el patrón "Front Controller" con un buen diseño, lo cual no es correcto y se debe a una mala implementación del patrón [26].

Los conceptos del patrón “Page Controller” están implementados en ASP.NET por defecto. Las páginas dentro del framework ASP.NET implementan estos conceptos de manera que el mecanismo subyacente de capturar un evento en el cliente, trasmitirlo el servidor, y llamar al método apropiado es automático e invisible al desarrollador. El controlador de página es extensible en el sentido de que expone en puntos específicos en el ciclo de vida de una página ASP.NET [26].

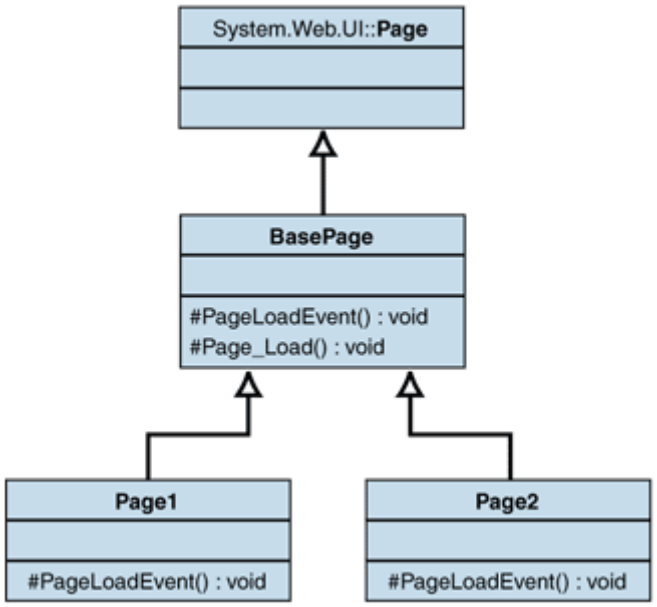


Figura 38: Patrón Page Controller en ASP.NET

8.4.2 Repository

En muchas aplicaciones, la logica de negocio accede a los datos de forma directa lo que puede resultar en diversos problemas como codigo duplicado, mayor probabilidad de errores y principalmente la imposibilidad de de probar la logica de negocio de manera separada de las dependencias externas [26].

Utilizando el patron Repository (Repositorio) se tiene una mediacion entre la capa de acceso a datos y las capas de negocio de la aplicación de manera que es posible tener un elemento “puente” que encapsula diferentes implementaciones de acceso a datos a la vez que nos

ofrece un conjunto de servicios estandar para la modificaciones de las entidades de negocio (datos) [26].

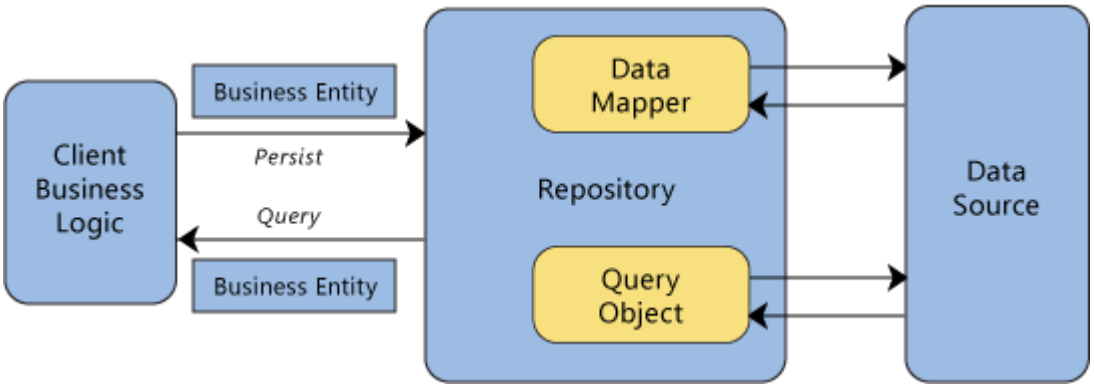


Figura 39: Patrón Repository

9 PRUEBAS

9.1 PRUEBAS UNITARIAS

9.1.1 Herramientas para realización de pruebas unitarias.

- Visual Studio Unit Testing Framework

El entorno de desarrollo de Visual Studio viene integrado con un conjunto de herramientas y librerías para realizar Pruebas Unitarias de nuestro código si necesidad de instalar software adicional o crear proyectos de software separados de la solución existente [27].

- HTMLAgilityPack

Es una biblioteca de código .NET que permite analizar "fuera de la web " archivos y textos HTML. El analizador es muy tolerante con HTML mal formado que podría considerarse bastante común en la páginas actuales. El modelo de objetos es muy similar a lo que propone la biblioteca System.Xml, pero para documentos HTML (o corrientes).

Entre los muchos usos de esta biblioteca tenemos la opción de usarla como una herramienta para validar contenidos HTML así como XML.

9.1.2 Detalle de Pruebas

Unit Test 1	
Prueba	Generación de HTML válido en base a JSON
Código	**SemHTMLGen.UnitTests.UnitTests.TestGenerateValidHTML**
Descripción	Prueba para validar que el HTML generado en base a un modelo es válido y bien formado
Resultado Esperado	en base a una cadena JSON un HTML bien formado
Precondiciones/input	Modelo Intermedio JSON correspondiente a un modelo

Unit Test 2	
Prueba	Generación de JSON en base a objetos conceptuales
Código	**SemHTMLGen.UnitTests.UnitTests.TestGenerateValidJSON**
Descripción	Prueba para validar el algoritmo de generación de código intermedio JSON en base a un conjunto de objetos (figuras/relaciones) que representan a un modelo conceptual creado
Resultado Esperado	Descripción en formato JSON de un modelo valido
Precondiciones/input	Lista de objetos (figuras y relaciones)

Unit Test 3	
Prueba	Generación de HTML en base a JSON Invalido
Código	**SemHTMLGen.UnitTests.UnitTests.TestGenerateInvalidJSON**
Descripción	Prueba para validar el comportamiento del proceso de generación de HTML en base a un texto JSON invalido o que no representa a un modelo coherente
Resultado Esperado	Excepción o mensaje de error indicando que el texto intermedio JSON no es válido
Precondiciones/input	Modelo Intermedio JSON inválido

Unit Test 4	
Prueba	Creación de un Elemento Semántico
Código	**SemHTMLGen.UnitTests.UnitTests.TestCreateSemanticElement**
Descripción	La creación de un elementos semántico en base a datos válidos, este elemento será después utilizado para la creación del modelo conceptual
Resultado Esperado	Un Elemento Semántico creado correctamente
Precondiciones/input	Objeto “SemanticElement” con datos válidos

Unit Test 5	
Prueba	Creación de un Elemento Semántico Invalido
Código	**SemHTMLGen.UnitTests.UnitTests.TestCreateInvalidElement**
Descripción	La creación de un elementos semántico en base a datos inválidos.
Resultado Esperado	Excepción o mensaje de error indicando que los datos del objeto “SemanticElement” que se pretende crear no son correctos
Precondiciones/input	Objeto “SemanticElement” con datos inválidos

Unit Test 6	
Prueba	Modificación de un Elemento HTML
Código	**SemHTMLGen.UnitTests.UnitTests.TestUpdateHTMLElement**
Descripción	Prueba la modificación de un elemento existente de tipo HTML que también es usado para crear el modelo conceptual
Resultado Esperado	Elemento modificado correctamente
Precondiciones/input	Objeto “HTMLElement” existente o previamente creado

Unit Test 7	
Prueba	Modificación Fallida de un Elemento HTML
Código	**SemHTMLGen.UnitTests.UnitTests.TestFailUpdateHTMLElement**
Descripción	Prueba la modificación de un elemento existente de tipo HTML incluyendo valores inválidos
Resultado Esperado	Excepción o mensaje de error indicando que los datos proporcionados para actualizar el objeto no son válidos
Precondiciones/input	Objeto “HTMLElement” existente o previamente creado, datos inválidos

Unit Test 8	
Prueba	Guardar Proyecto
Código	**SemHTMLGen.UnitTests.UnitTests.TestSaveProject**
Descripción	Test para guardar un proyecto que contiene un objeto de tipo “Page” que contiene un código JSON intermedio que puede visualizarse en la interfaz de usuario
Resultado Esperado	Proyecto y código JSON guardado en la base de datos
Precondiciones/input	Texto JSON que representa a un modelo creado

9.2 PRUEBAS DE USABILIDAD

La usabilidad es una característica muy importante dentro de cualquier aplicación de software ya que pueden determinar la aceptación por parte de los usuarios y por lo tanto el éxito de esta de cara a su utilización.

Estas pruebas han sido efecutadas utilizado individuos de prueba con conocimiento de html y programación/creacion de sitios web asi como un reducido conocimiento de web semántica.

9.2.1 Pruebas con usuarios reales

Los indivuduos seleccionados pertenecen al grupo objetivo al que va dirigido esta herramienta.

Usuario #1				
Descripción:	licenciado en informática , programador de aplicaciones			
Nivel de conocimiento técnico	alto			
Edad	30			
¿Con que frecuencia utiliza la computadora?				
diaria				
¿Qué actividades realiza la mayor parte del tiempo con la computadora?				
es mi herramienta de trabajo/programación				
¿Ha usado antes un software similar al de esta prueba?				
No				
¿Cuál es su principal criterio de evaluación de un programa/software?				
La facilidad de uso y de aprendizaje				
¿Posee conocimientos de programación?				
Si				
¿Posee conocimiento de creación de HTML?				
Si				
Facilidad de Uso				
	Siempre	Casi Siempre	Ocasionalmente	Nunca
¿La Navegación es adecuada?	X			
¿Cuenta con ayudad para todas las funciones del sistema en sus respectivas pantallas?			X	
Le resulta sencillo/intuitivo el uso de la aplicación		X		
Funcionalidad				
	Siempre	Casi Siempre	Ocasionalmente	Nunca

¿Cada tarea dentro del sistema se realiza como Ud. Espera?		X		
¿El tiempo de respuesta es adecuado?	X			
¿Los mensajes de error son útiles?		X		
¿La ayuda del sistema es adecuada para cada función?			X	
Interfaz de Usuario				
	Muy Adecuado	Adecuado	Poco Adecuado	Nada Adecuado
Combinación de colores	X			
Textos y tamaño de letra		X		
Interfaz intuitiva y amigable	X			
Diseño de pantallas claro y atractivo		X		
adecuación al entorno de trabajo		X		
¿Cree que la interfaz está acorde al sistema?	X			

Usuario #2				
Descripción:	Diseñador Gráfico, Front-End Developer			
Nivel de conocimiento técnico	Medio			
Edad	26			
¿Con que frecuencia utiliza la computadora?				
diaria				
¿Qué actividades realiza la mayor parte del tiempo con la computadora?				
Diseño de páginas web y navegar en internet				
¿Ha usado antes un software similar al de esta prueba?				
No				
¿Cuál es su principal criterio de evaluación de un programa/software?				
Eficiencia y facilidad de aprendizaje				
¿Posee conocimientos de programación?				
No				
¿Posee conocimiento de creación de HTML?				
Si				
Facilidad de Uso				
	Siempre	Casi Siempre	Ocasionalmente	Nunca
¿La Navegación es adecuada?		X		

¿Cuenta con ayudad para todas las funciones del sistema en sus respectivas pantallas?			X	
Le resulta sencillo/intuitivo el uso de la aplicación	X			
Funcionalidad				
	Siempre	Casi Siempre	Ocasionalmente	Nunca
¿Cada tarea dentro del sistema se realiza como Ud. Espera?		X		
¿El tiempo de respuesta es adecuado?		X		
¿Los mensajes de error son útiles?		X		
¿La ayuda del sistema es adecuada para cada función?		X		
Interfaz de Usuario				
	Muy Adecuado	Adecuado	Poco Adecuado	Nada Adecuado
Combinación de colores			X	
Textos y tamaño de letra		X		
Interfaz intuitiva y amigable		X		
Diseño de pantallas claro y atractivo		X		
adecuación al entorno de trabajo	X			
¿Cree que la interfaz está acorde al sistema?	X			

Usuario #3	
Descripción:	Técnico en Informática, desarrollador web
Nivel de conocimiento técnico	alto
Edad	29
¿Con que frecuencia utiliza la computadora?	
Todos los dias	
¿Qué actividades realiza la mayor parte del tiempo con la computadora?	
Creación de aplicaciones web	
¿Ha usado antes un software similar al de esta prueba?	
No	
¿Cuál es su principal criterio de evaluación de un programa/software?	
La eficiencia	
¿Posee conocimientos de programación?	
Si	
¿Posee conocimiento de creación de HTML?	
Si	

Facilidad de Uso				
	Siempre	Casi Siempre	Ocasionalmente	Nunca
¿La Navegación es adecuada?		X		
¿Cuenta con ayudad para todas las funciones del sistema en sus respectivas pantallas?			X	
Le resulta sencillo/intuitivo el uso de la aplicación		X		
Funcionalidad				
	Siempre	Casi Siempre	Ocasionalmente	Nunca
¿Cada tarea dentro del sistema se realiza como Ud. Espera?		X		
¿El tiempo de respuesta es adecuado?		X		
¿Los mensajes de error son útiles?		X		
¿La ayuda del sistema es adecuada para cada función?			X	
Interfaz de Usuario				
	Muy Adecuado	Adecuado	Poco Adecuado	Nada Adecuado
Combinación de colores	X			
Textos y tamaño de letra		X		
Interfaz intuitiva y amigable	X			
Diseño de pantallas claro y atractivo	X			
adecuación al entorno de trabajo		X		
¿Cree que la interfaz está acorde al sistema?		X		

9.2.2 Observaciones

Usuario#1	
Aspecto Observado	**Detalle**
El usuario comienza a realizar de manera rápida las tareas	Relativamente rápido, el usuario se ha adaptado a la interfaz
Tiempo en realizar cada tarea	esperado
Errores Cometidos	
Graves	ninguno
Leves	2
Preguntas realizadas por el usuario	1

Usuario#2	
Aspecto Observado	**Detalle**
El usuario comienza a realizar de manera rápida las tareas	No muy segura de las tareas que debe realizar, falta de conocimiento del ámbito del sistema
Tiempo en realizar cada tarea	esperado
Errores Cometidos	
Graves	1
Leves	3
Preguntas realizadas por el usuario	5

Usuario#3	
Aspecto Observado	**Detalle**
El usuario comienza a realizar de manera rápida las tareas	Normal con una pregunta inicial se ha adaptado a manejar el sistema
Tiempo en realizar cada tarea	esperado
Errores Cometidos	
Graves	ninguno
Leves	4
Preguntas realizadas por el usuario	2

10 CONCLUSIONES Y TRABAJO FUTURO

En el presente trabajo se ha desarrollado una herramienta, que nos permite generar HTML con contenido semántico, haciendo uso de tecnologias que han sido creadas con el objetivo de promover la inclusión de datos semánticos en la web y cuyas potencialidades no están siendo usadas tan ampliamente como se podría.

Este proyecto ha permitido demostrar que un software de este tipo, dadas sus características, se constituiría en un alternativa muy útil y diferente al conjunto de herramientas para la Web Semántica que existen actualmente.

Dentro de las aportaciones más importantes podriamos destacar que se ha creado un sistema, que puede ser utilizado como un nuevo enfoque en el proceso de desarrollo actual de páginas web, sin necesidad de crear de forma separada nuestros datos a publicar de nuestras aplicaciones en sí y que además, será también una gran ayuda para que gente que no tiene un conocimiento profundo de Web Semántica pueda iniciarse en este campo.

Si bien es cierto que en esta primera versión del prototipo resultante existen aspectos que serán mejorados en un trabajo futuro, los resultados han sido satisfactorios desde el punto de vista de que se han cumplido los objetivos planteados desde el inicio del proyecto.

Es importante asimismo mencionar, que desde la concepción del desarrollo, este sistema ha sido pensado para ser el inicio de una aplicación mucho más grande y compleja que permita, entre otras cosas, la utilización del conjunto completo de extensiones y atributos RDFa y no estar limitados a su versión reducida “RDFa Lite”.

Entre los principales puntos planificados como trabajo futuro podemos resaltar a los siguientes:

- Utilización del conjunto completo de atributos y extensiones RDFa, permitiendonos la inclusión de información semántica mucho mas compleja y estructurada e incluso ontologías en el contenido de las páginas generadas.
- Implementación de herramientas para la edición visual de las páginas mediante CSS.
- Permitir la edición colaborativa, dando a varios usuarios la posibilidad de trabajar de manera síncrona sobre diferentes tareas.
- Creación de proyectos que contengan varias páginas web, ampliando el alcance del sistema hacia la creación de sitios web completos.

11 REFERENCIAS

[1] L. Feigenbaum, I Herman, T, Hongsermeier, E. Neumann, E. Stephens, “The Semantic Web in Action”, *Scientific American, 297(6), pp. 90-97,* Dec 2007.

[2] T. Berners-Lee, J. Hendler, O. Lassila, "The Semantic Web", *Scientific American*, pp. 29-37, May 2001.

[3] M.R. Koivunen, “Annotea Project”, w3.org/2001/Annotea, para 1, Oct. 31, 2005. [Online]. Available: http://www.w3.org/2001/Annotea. [Accessed: Aug. 1, 2015]

[4] S. Handschuh and S. Staab, “CREAM: CREAting Metadata for the Semantic Web,” *Comput. Networks*, vol. 42, no. 5 SPEC., pp. 579–598, 2003.

[5] A. Khalili, S. Auer, and D. Hladky, “The RDFa content editor - From WYSIWYG to WYSIWYM,” in *Proceedings - International Computer Software and Applications Conference*, 2012, pp. 531–540.

[6] M. Pilgrim, *HTML5: Up and Running*. 2010.

[7] Learnsemantic.com, 'Learn Semantic | Semantic UI', 2015. [Online]. Available: http://learnsemantic.com/. [Accessed: 09- Sep- 2015].

[8] P. Lubbers, B. Albers, and F. Salim, “Overview of HTML5,” in Pro HTML5 Programming, 2010, pp. 1–23

[9] World Wide Web Consortium, “XML essentials,” *XML Technology*, 2010. [Online]. Available: http://www.w3.org/standards/xml/core.

[10] J. Allsopp, *Microformats: Empowering your markup for web 2.0*. 2007.

[11] I. Hickson, “HTML Microdata,” *W3C Working Draft*, 2012. .

[12] W3C, “RDFa 1.1 Primer,” *W3C Working Group Note 07*. 2012.

[13] B. Adida, I. Herman, M. Sporny, and M. Birbeck, “RDFa 1.1 Primer Rich Structured Data Markup for Web Documents,” *… Web Consortium, Note NOTE-rdfa-primer- …*, no. June, pp. 1–19, 2012.

[14] K. Kotiadis, S. Robinson, "Conceptual Modelling: knowledge Acquisition and Model Abstraction." *Operational Research and Management Sciences Group, Warwick Business School, University of Warwick. Coventry, CV4 7 AL, UK*

[15] *P.D.* Leedy, J. E. Ormrod, "Practical research: Planning and design (9th Ed.)." *Upper Saddle River, NJ: Prentice Hall*, 2010

[16] V. Szalvay, “An introduction to agile software development,” *Danube Technol.*, 2004.

[17] M. Braz and S. Vergilio, “Software Effort Estimation Based on Use Cases,” in *30th Annual International Computer Software and Applications Conference (COMPSAC’06)*, 2006, vol. 1, pp. 221–228.

[18] T. Thai and H. Q. Lam, *NET Framework Essentials*. 2001.

[19] A. Zambelli, “IIS Smooth Streaming Technical Overview,” *Microsoft Corp.*, no. March, 2009.

[20] B. Evjen, S. Hanselman, and D. Rader, *Professional ASP.NET 4 in C# and VB*. 2010.

[21] T. Dykstra and R. Anderson, “Getting Started with Entity Framework 6 Code First using MVC 5.”

[22] B. I. Design and W. Development, *Learning jQuery*. 2007.

[23] E. Gamma, R. Helm, R. E. Johnson, and J. Vlissides, *Design patterns: elements of reusable object-oriented software*, vol. 206. 1995.

[24] V. Studio, *Software Testing using Visual Studio 2010*. 2010.

12 APÉNDICES

12.1 MANUALES

12.1.1 Manual de instalación

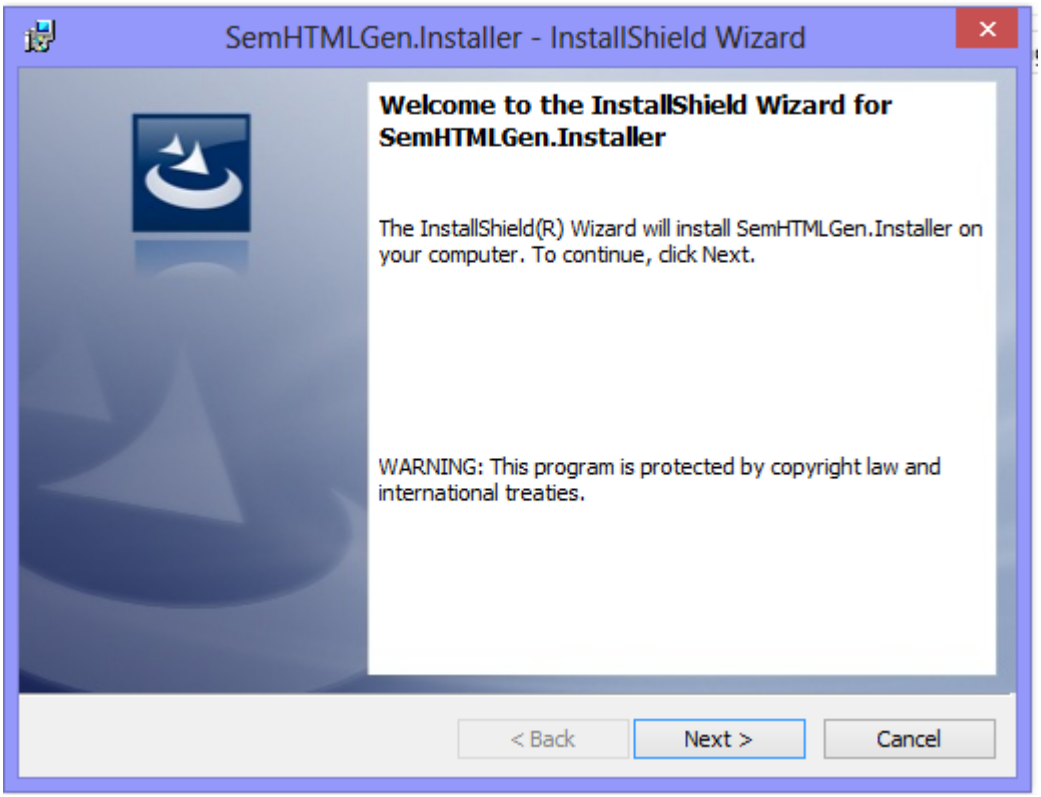
12.1.1.1 Prerequisitos

- Sistema Operativo Windows 7, 8, 8,1
- SQL Server 2008, 2008 R2, 2012
- Internet Information Services 7, 8
- Microsoft .NET framework 4+

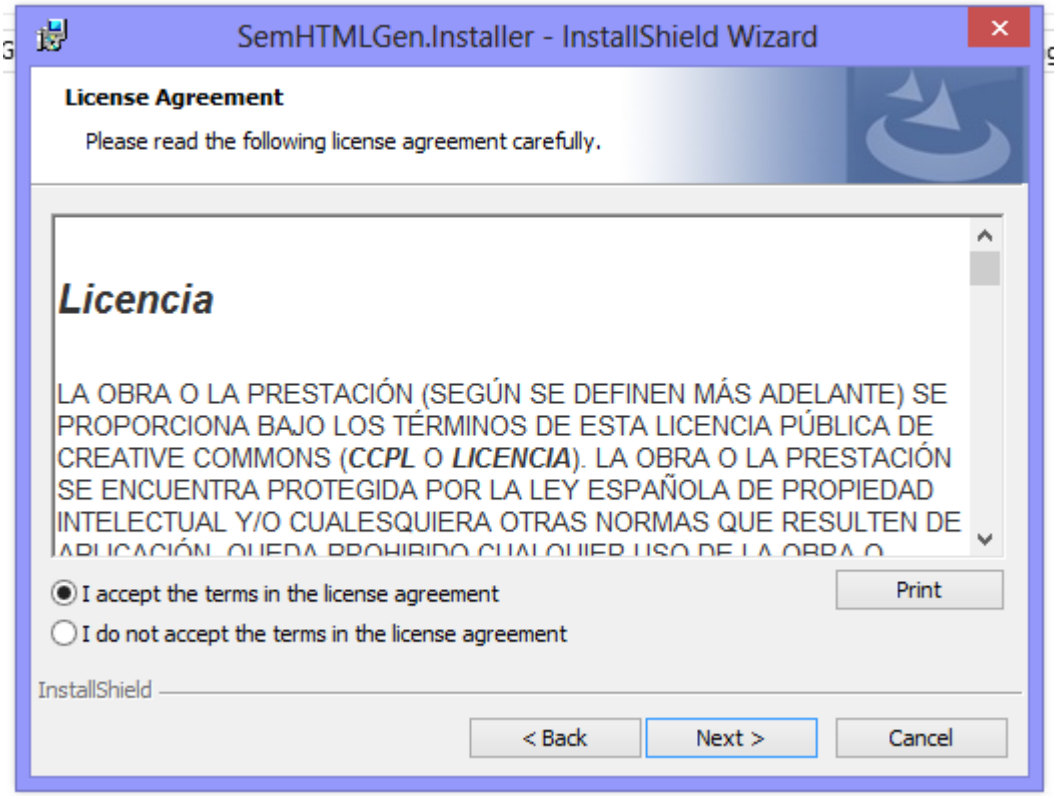
12.1.1.2 Instalación

La instalación de realiza mediante un archivo instalador que fue creado para facilitar la configuración del sitio web.

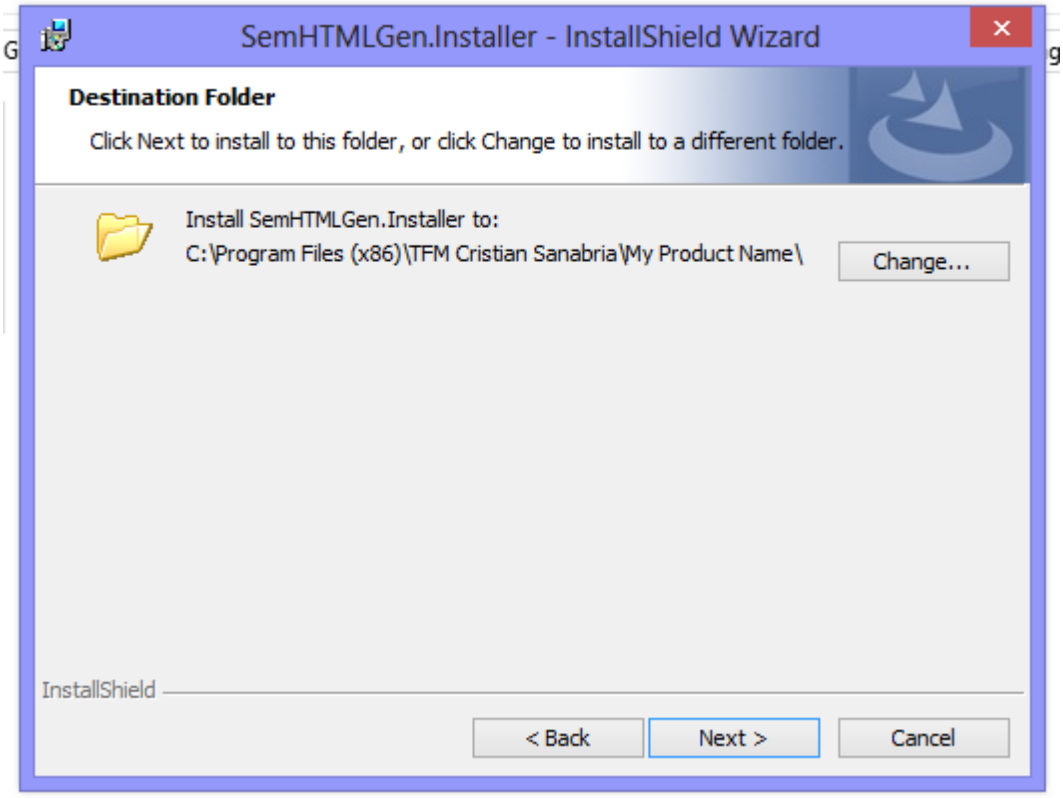
Se debe ejecutar el archivo ejecutable con permisos de administrador y seguir los siguientes pasos a continuación:



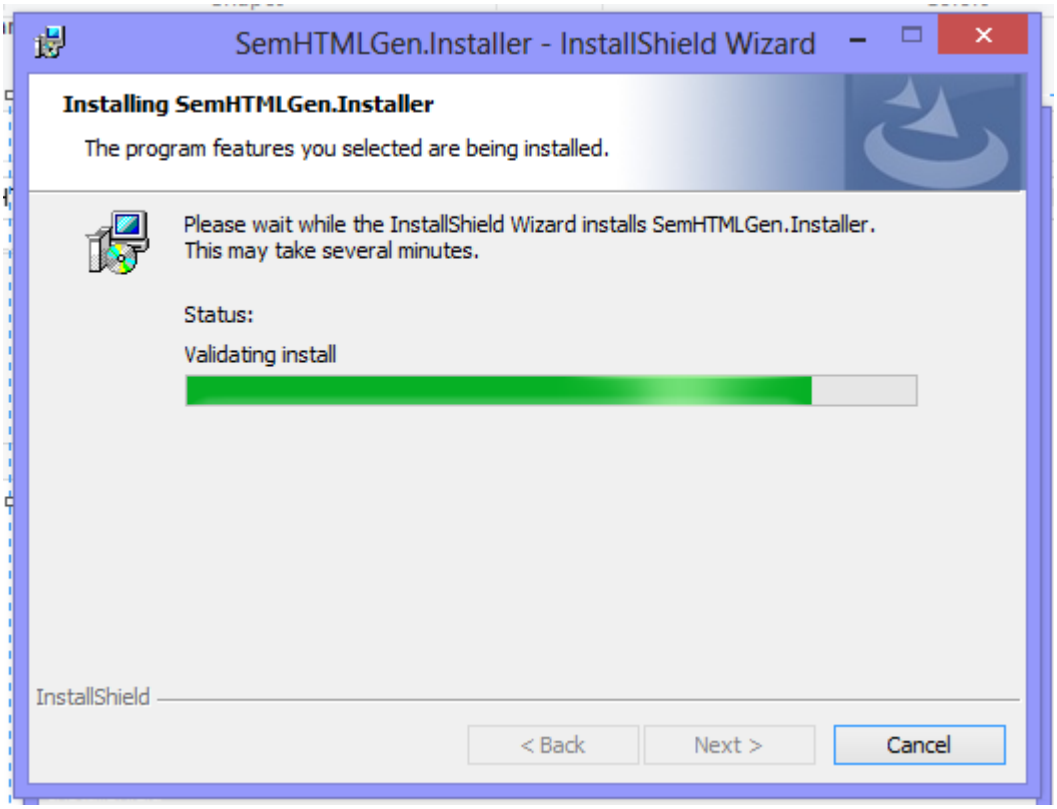
Aceptar la Licencia del software, Creative Commons
http://creativecommons.org/licenses/by/4.0/legalcode

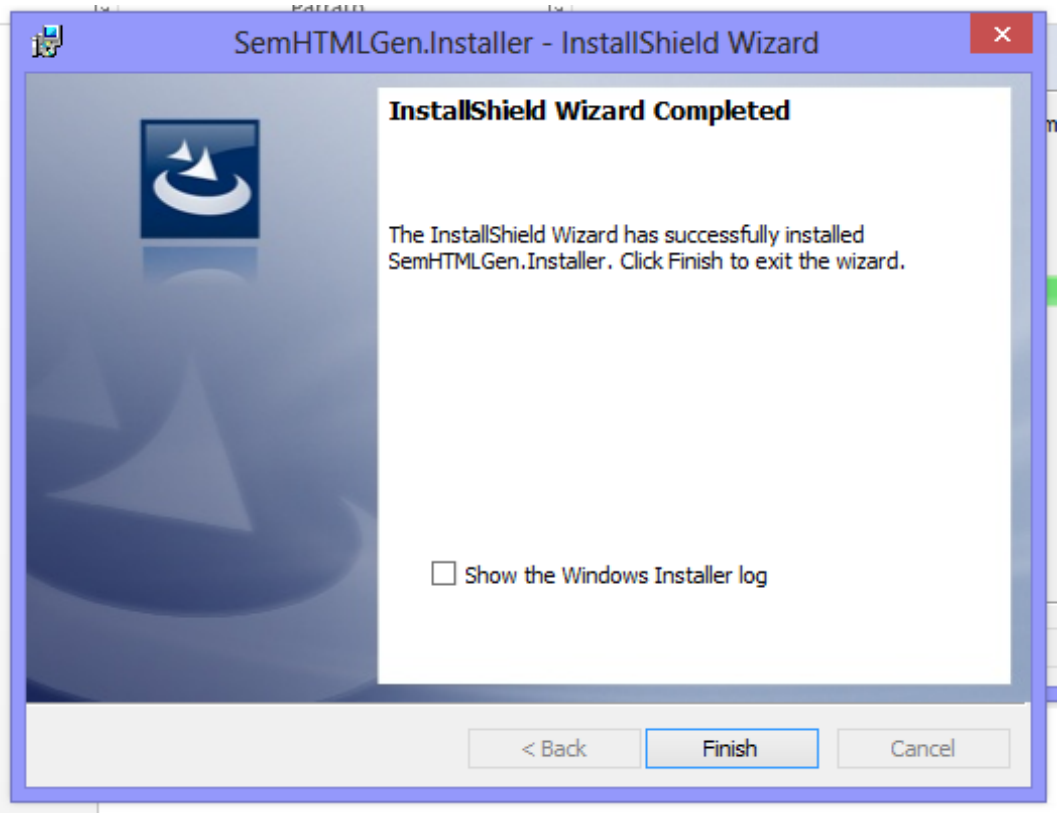


Seleccionamos la ubicación donde se van a copiar los archivos:



Presionamos el boton siguiente y el instalador procedera a copiar los archivos de la aplicación web y a crear el directorio virtual (sitio web en IIS)





Una vez terminada la instalación verificamos que la aplicación web ha sido correctamente creada

12.1.1.3 Creación y Configuración de la Base de Datos

Junto con el instalador se provee un archivo ejecutable Database.bat que realizara la creación de la base de datos en SQL server. Es necesario ejecutar este archivo.

12.1.1.4 Configuración Personalizada de la Base de Datos

En caso de no utilizar el archivo .BAT para la creación de la base de datos, se puede crear la misma de manera personalizada utilizando SQL Server Management Studio o algún script personalizado.

Si se ha elegido esta opción es necesario modificar la cadena de conexión en el archivo Web.Config de la aplicación que se encuentra en la carpeta raíz de la aplicación web como se muestra en la siguiente imagen:

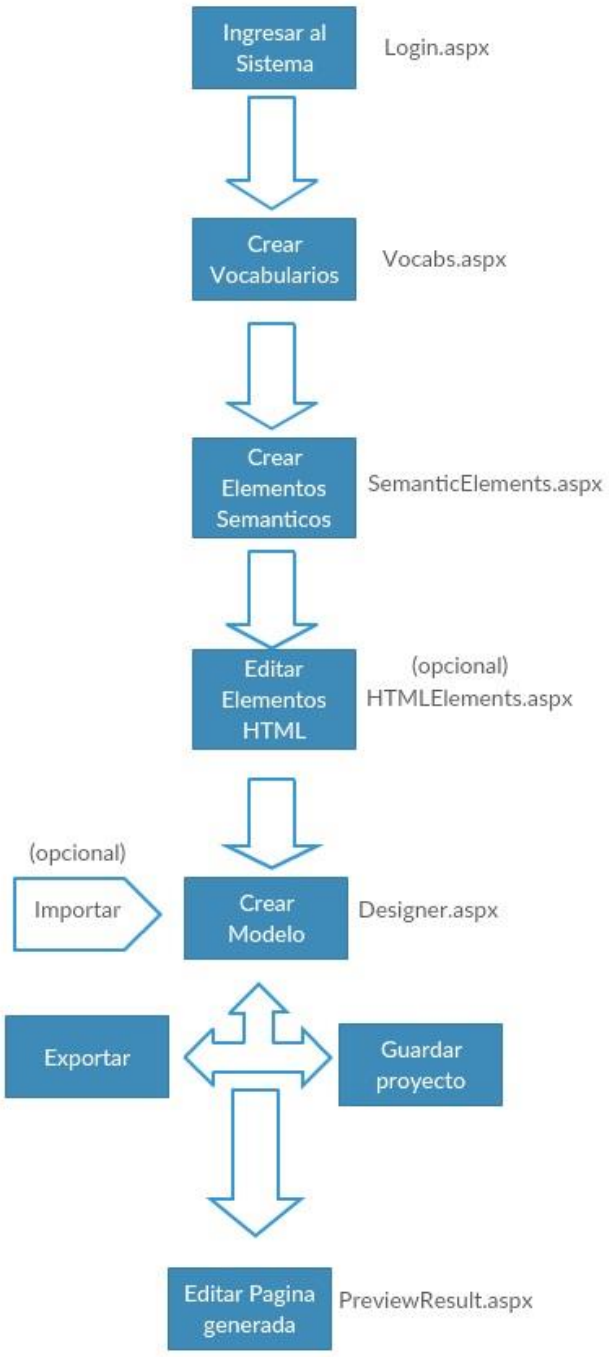
```
<connectionStrings>
  <add name="DefaultConnection" providerName="System.Data.SqlClient"
       connectionString="Data Source=VAIO\MSSQLSERVER2012;Initial Catalog=SemHTMLGen;Integrated Security=True" />
</connectionStrings>
<system.web>
```

12.1.2 Manual de usuario

12.1.2.1 Prerrequisitos

Es necesario para una correcta utilización del sistema tener conocimiento de HTML y es deseable tener también un conocimiento al menos básico de RDF para estar familiarizado con ciertos términos dentro de la aplicación

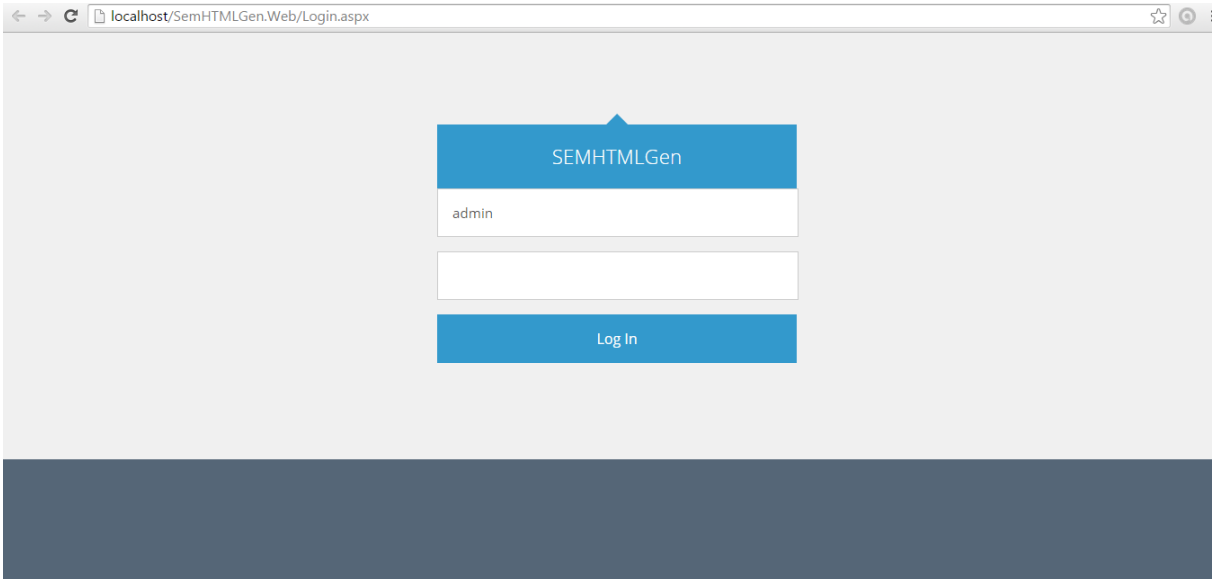
12.1.2.2 Flujo de Trabajo



12.1.2.3 Casos de Uso

Acceso

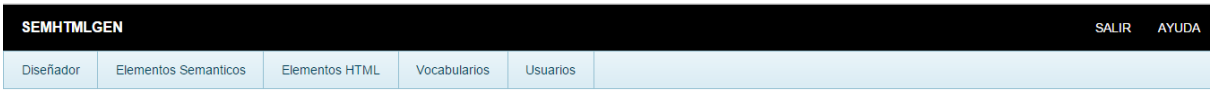
Al iniciar la aplicación se mostrará la pantalla de Login, para ingresar podemos utilizar credenciales de un usuario ya creado.



Inicialmente y con propósitos de prueba existen dos usuarios previamente creados en la Base de Datos al momento de la instalación/primera ejecución del sistema

- Usuario: admin, Contraseña: admin, Rol: Administrador
- Usuario: user, Contraseña: user, Rol: usuario

Menú Principal de Navegación



En el menú principal de navegación tenemos enlaces a las diferentes pantallas de la aplicación, estas son las siguientes:

- Diseñador: Es la Pantalla principal donde se puede realizar la creación del modelo conceptual de una página web mediante el diseñador gráfico utilizando elementos previamente creados (Semánticos y HTML)

- Elementos Semánticos: Los elementos semánticos son porciones de HTML+RDFa Lite que podrán ser agregados al modelo para crear el contenido, en esta pantalla se pueden crear/editar y eliminar estos elementos.

- Elementos HTML: Los Elementos HTML son elementos pre configurados en el sistema que pueden ser usados junto con los elementos semánticos. Estos elementos son etiquetas semánticas ya existentes en HTML5 como header, footer, aside, nav, etc.

- Vocabularios: los Vocabularios son espacios de nombres que definen los posibles términos a ser usados en el contenido semántico RDFa Lite. En esta sección de pueden crear y editar estos vocabularios.

- Usuarios: Esta sección estará visible solo para usuarios “Administradores” y brindara acceso a la sección de administración de usuarios del sistema.

- Enlace Salir: enlace para cerrar la sesión actual y retornar a la pantalla de Login

- Enlace Ayuda: enlace a la ayuda del sistema

Diseñador

El Menú Principal de la pantalla de diseño se compone de todos los elementos del menú de navegación normal y adicionalmente los siguientes elementos:

1.- Botón “Generar”: una vez que se ha creado el modelo conceptual de la página web este botón realiza la función de generar la página HTML y re-direccionar a una página posterior donde se podrá personalizar el HTML inicial generado.

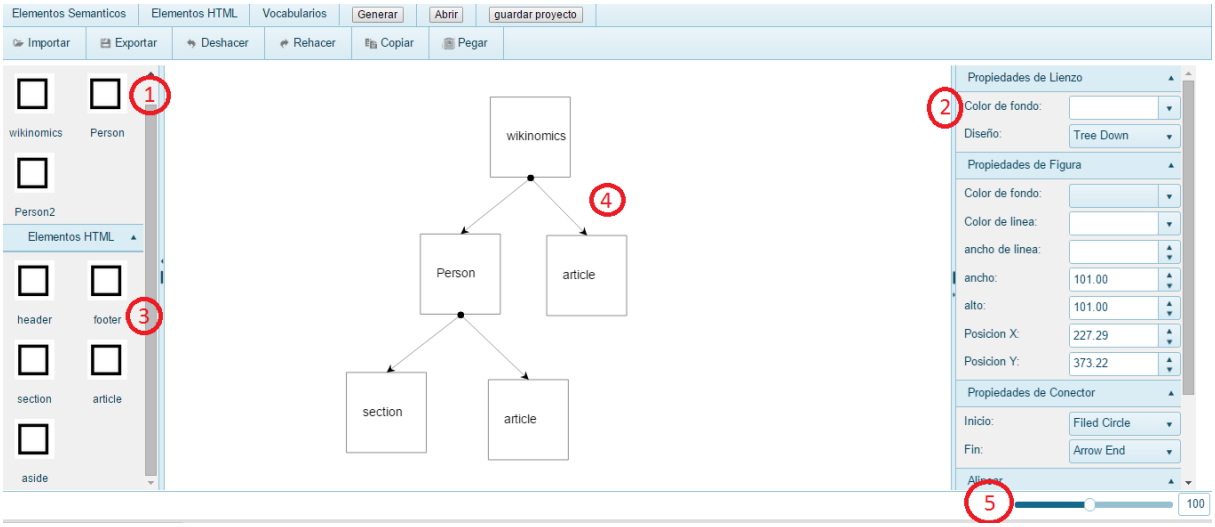
2.- Botón “Abrir”: Al presionar este botón se abrirá un dialogo para poder cargar en el sistema un modelo existente previamente guardado

3.- Botón “Guardar Proyecto”: Guarda el modelo actual creado en la base de datos para poder ser abierto posteriormente.

4.- Botones de Edición.- Estos botones permiten ejecutar operaciones standard de edición sobre el modelo (deshacer, rehacer, copiar, pegar, etc.)

Paneles de Diseño

La Página de diseño está compuesta por los siguientes paneles/secciones



1.- Elementos Semánticos: en este panel se listan todos los elementos semánticos que han sido creados previamente (ver sección Elementos Semánticos) cada uno de estos elementos puede ser arrastrado al panel central

2.- Panel de Propiedades Graficas: acá se muestran diferentes opciones para personalizar la apariencia visual del modelo y sus elementos, propiedades de figuras, colores, posición de elementos, etc.

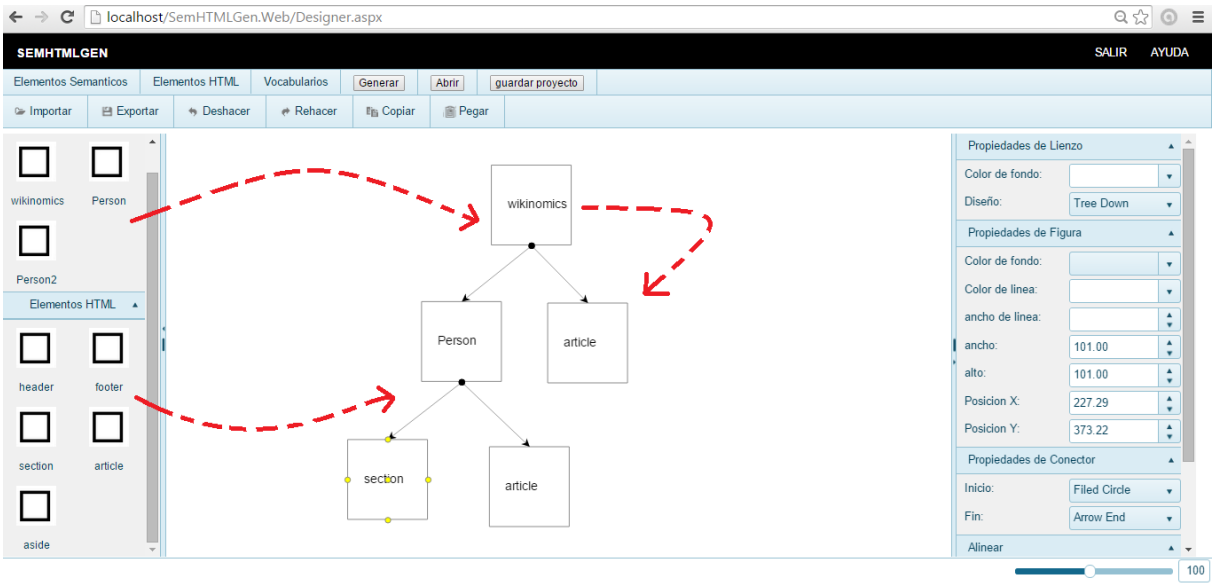
3.- Elementos HTML: Similar a los elementos semánticos estos elementos previamente creados pueden ser arrastrados el panel central (ver sección Elementos HTML)

4.- Panel Central: Visualiza el modelo conceptual creado a este panel es donde se arrastran los elementos de los paneles 1 y 3 además puede ser manipulado por la rueda (wheel) del mouse

5.- Zoom Slider: Zoom para el modelo (panel central)

Manipulación Grafica del Modelo

Para incluir elementos en el modelo simplemente basta con arrastrar y soltar los mismo en el área blanca (Drag and Drop), para crear relaciones hacer click en el elemento elegido y crear la flecha hacia el elemento deseado



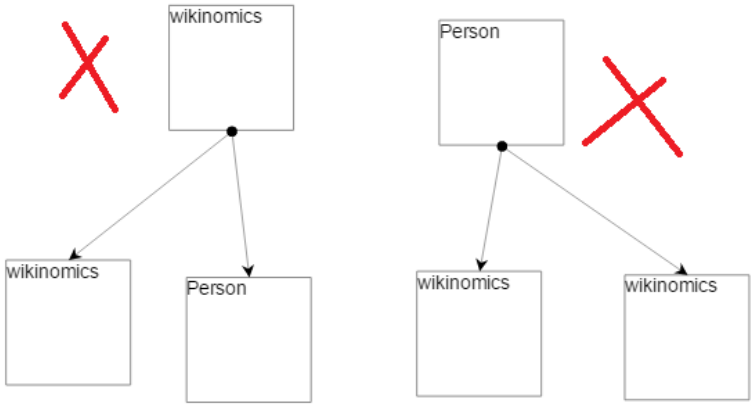
Consistencia/Estructura del Modelo Conceptual

Debido a la flexibilidad que brinda el diseñador gráfico del sistema no es posible controlar todos los casos de creación de un gráfico por lo tanto es importante tomar en cuenta al momento de crear el modelo las siguientes consideraciones:

- Siempre debe existir un único elemento padre por ejem, “div”, “section”, etc. generalmente una página web contiene un elemento (wrapper/div) que agrupa a todo el contenido dentro de ‘body’, este elemento vendría a ser nuestro elemento padre.
- La línea de relación siempre debe ir desde el padre a los hijos y no viceversa ya que no tendría sentido que un elemento hijo tenga como hijo a su padre.
- La Lógica general de creación de un modelo de una página web es un “Árbol de Elementos”

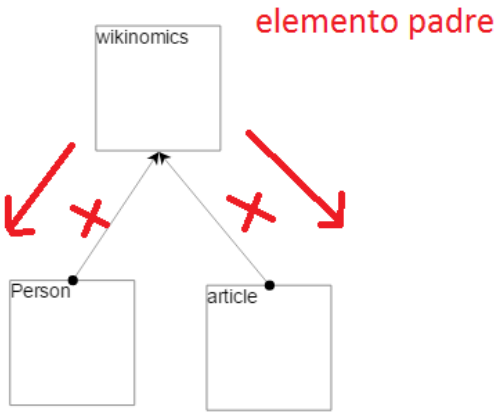
A continuación se presentan algunos ejemplos de modelos incorrectos:

Ejemplo Incorrecto 1:



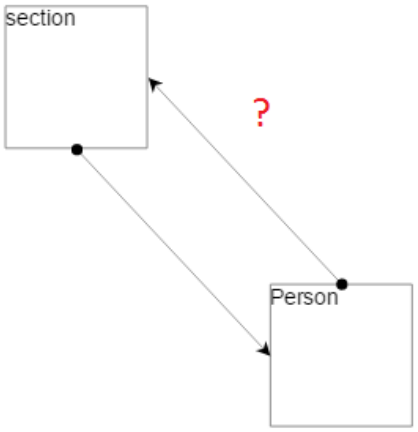
Descripción: Incorrecto porque siempre debe existir solo un elemento padre al que están relacionados los elemento hijos.

Ejemplo Incorrecto 2:



Descripción: El sentido de las líneas/flechas siempre debe ser desde el elemento padre hasta el elemento hijo NO viceversa como se muestra en la imagen.

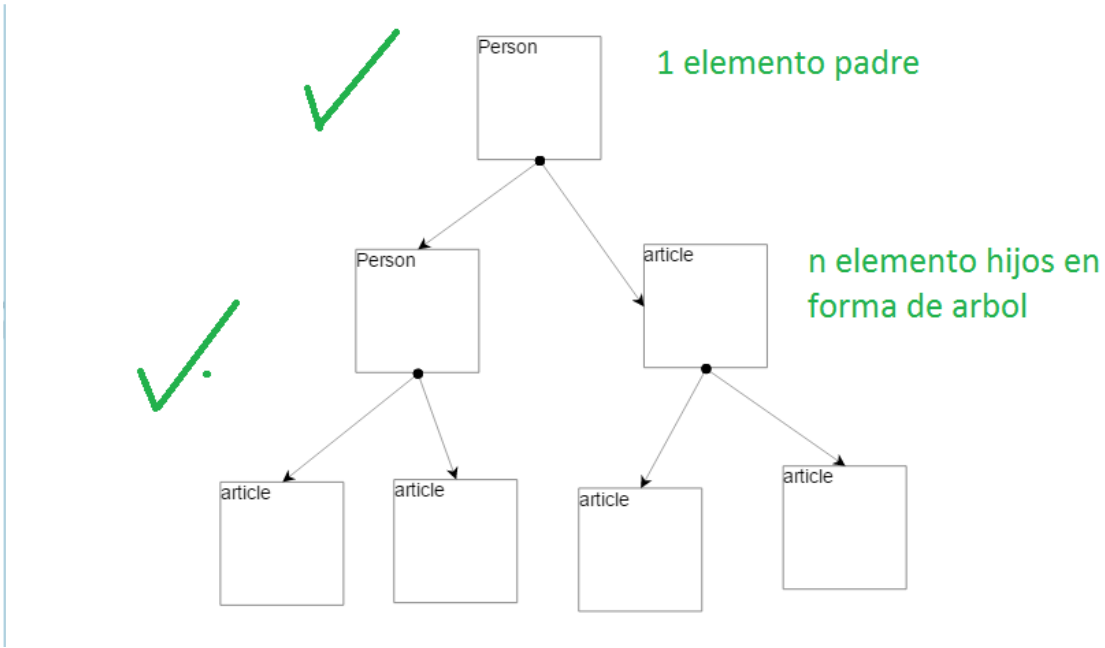
Ejemplo Incorrecto 3:



Descripción: Las Dependencias circulares como se muestran en la imagen no están soportadas por el algoritmo de generación ya que no tiene sentido lógico que un elemento que sea hijo de su padre en HTML.

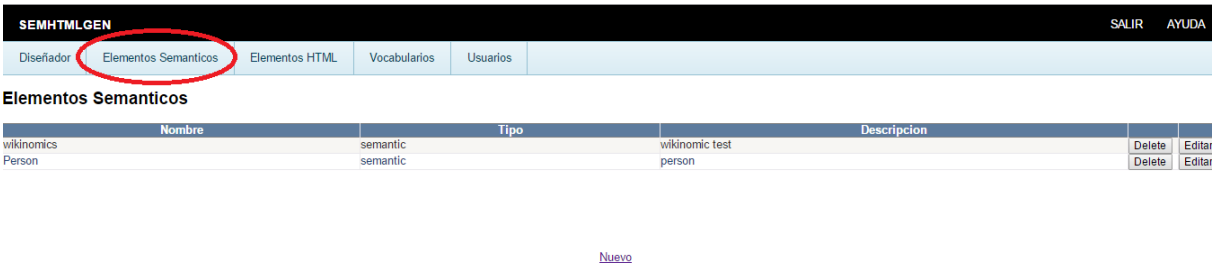
Ejemplo Correcto

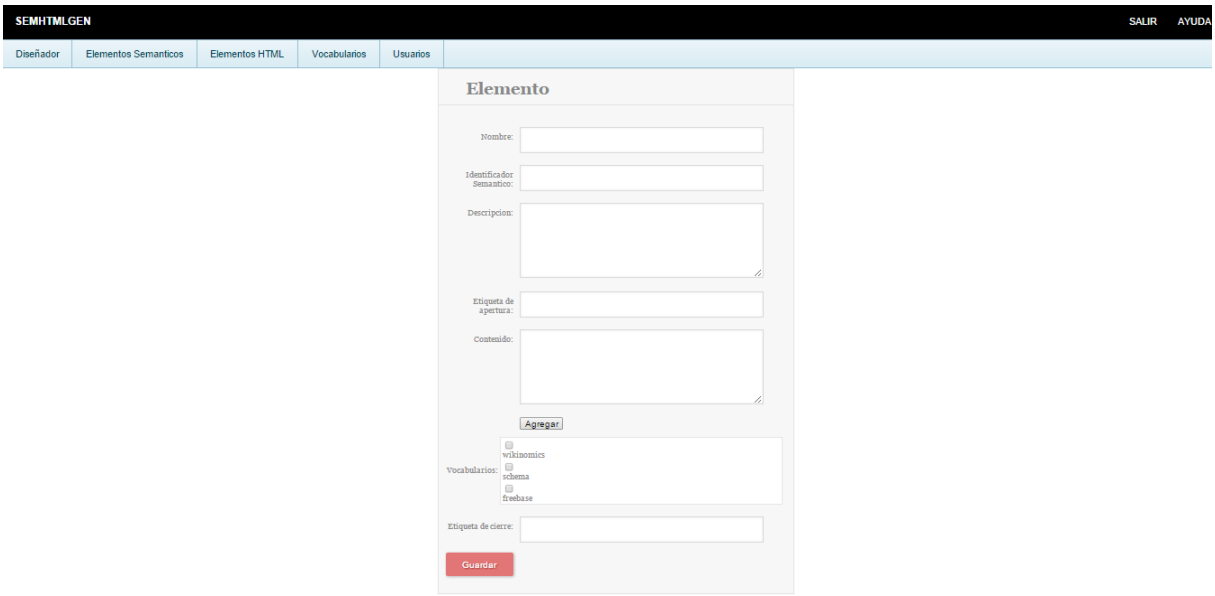
Un modelo correctamente creado siempre debe seguir la siguiente estructura lógica, sin importar la profundidad ni cantidad de elementos.



Administración de Elementos Semánticos

Los elementos Semánticos son porciones de código HTML+RDFa Lite que serán utilizado en la pantalla de diseño.



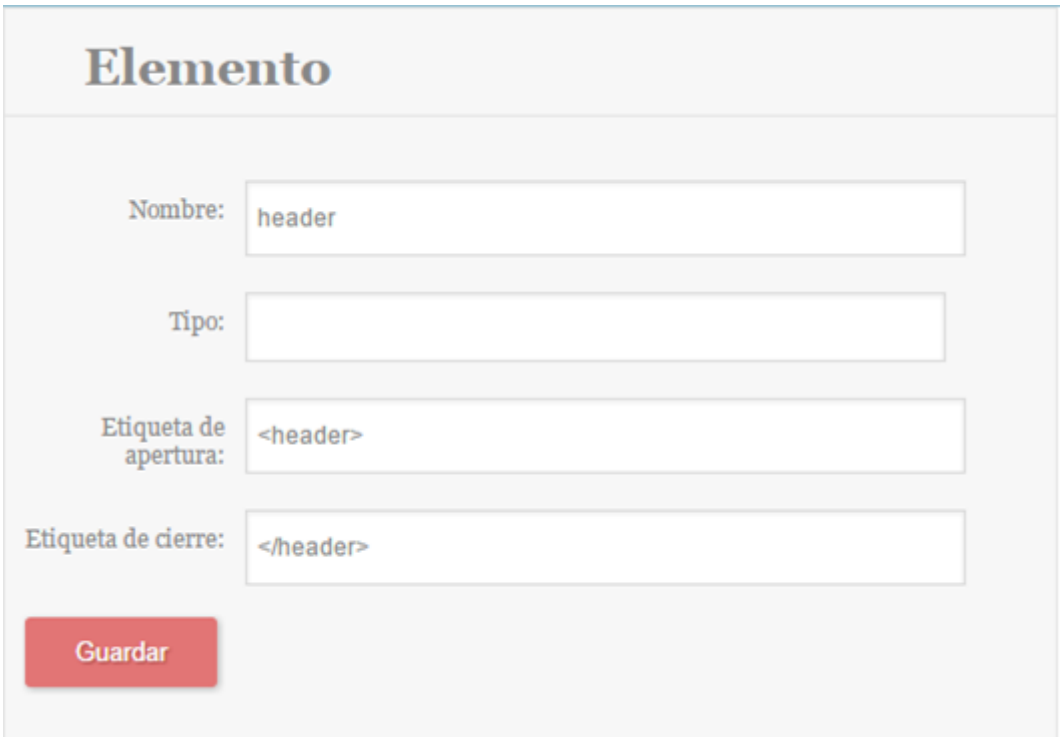


En el Formulario de creación de elementos semánticos se tienen los siguientes campos:

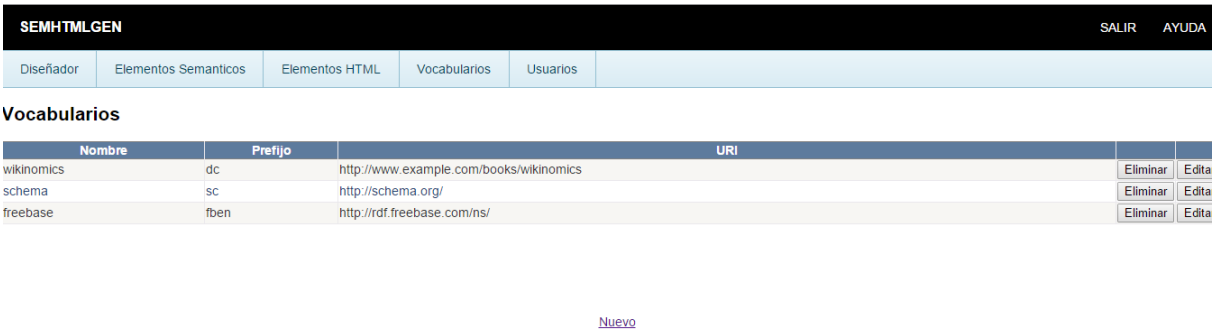
- Nombre: nombre del elemento
- Identificador: valor a ser asignado como identificador RDFa Liste dentro de la página web, este valor se incluirá con el formato #id
- Descripción: descripción del elemento para su fácil entendimiento
- Etiqueta de Apertura: etiqueta de apertura del HTML que va a contener el elemento
- Contenido: contenido RDFa+HTML del elemento (no incluye elemento de apertura ni cierre)
- Vocabularios: vocabularios a ser incluidos en la etiqueta de apertura del elemento
- Etiqueta de cierre: etiqueta de cierre del elemento.

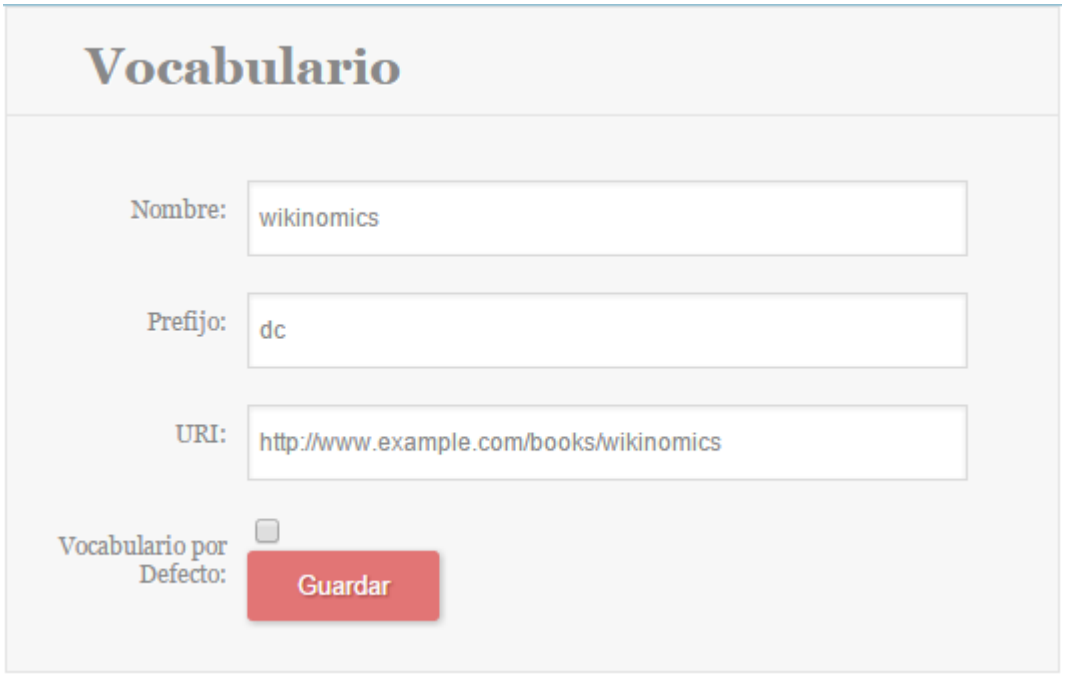
Administración de Elementos HTML

Los Elementos HTML son elementos en el sistema que corresponden a los elementos semánticos existentes en HTML5 header, footer, aside, article, nav, etc.



Administración de Vocabularios

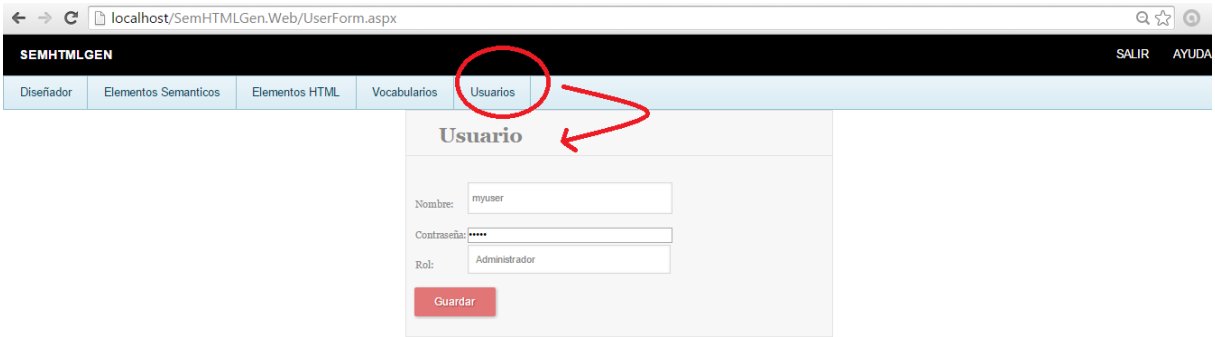




- Nombre: nombre del vocabulario usado para ser fácilmente recordado por el usuario este valor no es utilizado al momento de generar el código HTML
- Prefijo (prefix): prefijo a ser utilizado si se tiene más de un vocabulario en un elemento semántico
- URI: Uri del vocabulario

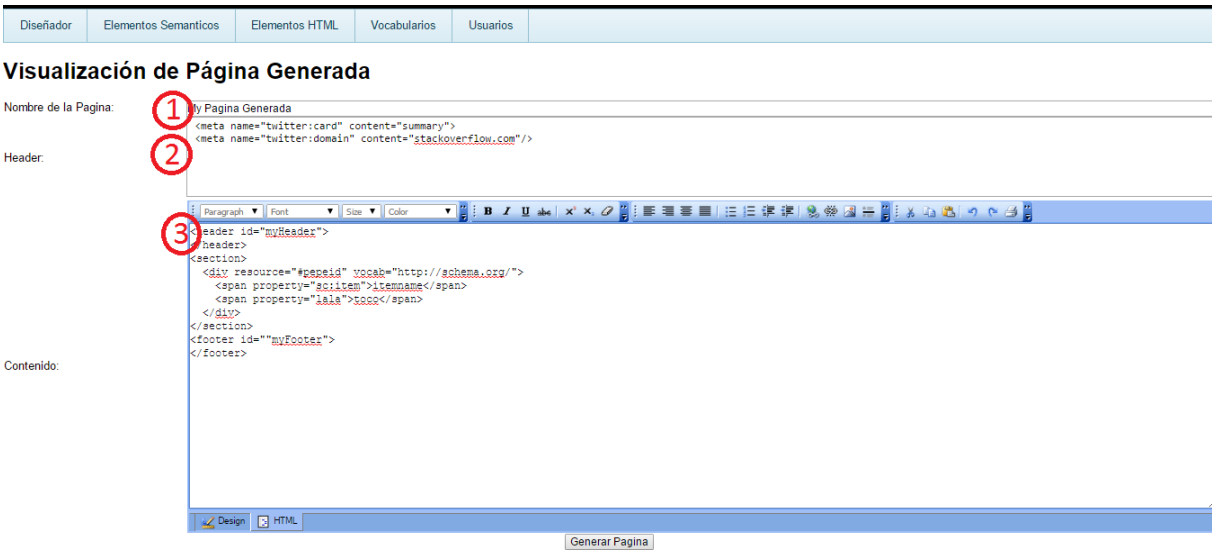
Administración de Usuarios

Un usuario con el rol de “Administrador” tendrá habilitado el menú “usuarios” para poder crear y editar usuarios existentes en el sistema



Generación de la Página Web

Una vez creado el modelo conceptual con los elementos semánticos previamente configurados al presionar el botón “Generar” en la página de diseño se realizara la generación del modelo y el sistema realizara una redirección a la siguiente página:



En la pantalla de “Visualización de la Página generada” se cuenta con un editor de HTML (gráfico y código) para poder editar el HTML inicialmente generado acá se podrán agregar todos aquellos detalles adicionales que se desean para la página generada final.

1.- Nombre de la Página

2.- Header: código que se incluirá en la etiqueta <head> de la página

3.- Contenido: El contenido generado en base al modelo conceptual, este puede ser editado

Exportar/Importar Modelos

El sistema cuenta con la funcionalidad de exportar/Importar Modelos que no son más que archivos en formato JSON que representan a un modelo conceptual con información sobre los elementos gráficos, contenido, posiciones, etc.

12.1.3 Glosario de términos

- Modelo Conceptual: Dentro del ámbito de este proyecto un modelo conceptual se refiere a la representación gráfica jerárquica del contenido de una página web, refiriéndose principalmente a sus elementos.

- Elemento Semántico: Un elemento semántico es una porción de código HTML con contenido semántico en RDFa Lite que puede ser incluido en el modelo conceptual de la página web mediante el diseñador gráfico del sistema.

- Elemento HTML: Un elemento HTML es una poción de código HTML (No semántico) que puede ser incluido en el modelo una página web mediante el diseñador gráfico del sistema.

- Código Intermedio: El modelo intermedio se refiere al código en formato JSON usado internamente por el sistema en base al modelo conceptual para posteriormente convertir este código intermedio a Código HTML.

- Framework: Conjunto de pautas de diseño y librerías que nos permiten desarrollar un sistema utilizando un lenguaje de programación.

- Metadatos: Son información que describe a los datos usados dentro de la aplicación son comúnmente conocidos como “datos sobre los datos”.

- Lenguaje de marcado (Markup): es un mecanismo para codificar un documento que mediante un conjunto de etiquetas agrega información sobre el texto existente por ejemplo HTML.

- JSON: (Java Script Object Notation) es un formato ligero para el intercambio de datos muy popular en la actualidad especialmente en el intercambio de datos en servicios web REST.

- CMS(Content Management System): En español Sistema de Gestión De contenidos