Universidad Carlos III de Madrid

Departamento de Informática

PROYECTO FIN DE CARRERA

Ingeniería Técnica en Informática de Gestión

ASISTENTE PARA LA ADAPTACIÓN DE FORMULARIOS WEB A LA TECNOLOGÍA VOICEXML

Autor: Álvaro Casado Rodríguez

Tutor: Jorge Blasco Alís

Leganés, octubre de 2012

Título: Asistente para la adaptación de formularios web a la tecnología VoiceXML.

Autor: Álvaro Casado Rodríguez.

Director: Jorge Blasco Alís.

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día 05de Octubre de 2012 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de _____

VOCAL

SECRETARIO PRESIDENTE

AGRADECIMIENTOS

A mis padres, Tomás y Mercedes, por darme una gran educación, ésta me ha servido para formarme como persona, también por apoyarme en todos mis proyectos a lo largo de mi vida y estar siempre en los buenos y malos momentos.

A mis hermanos, Tomás y Alejandro, por "soportarme" durante todos estos años, animarme en todo momento y estar ahí siempre para todo lo que he necesitado.

A mi familia por parte paterna y materna que siempre me apoyan desde la distancia.

Al gran número de compañeros de la universidad, por realizar más ameno este camino en la universidad. Debo hacer mención especial a mis amigos Carlos, Jaime y Javi por unos años inigualables, que se pueden calificar como sublimes, con anécdotas, situaciones e historias de todo tipo, siempre positivas y de las que siempre he aprendido.Que esta amistad dure por muchos años.

A mi tutor Jorge Blasco, por animarme y ayudarme en todo momento, su facilidad a la hora de trabajar ha sido clave para el desarrollo de este proyecto. También a mi primer tutor David que fue con el que di los primeros pasos de este proyecto.

Por último a todos los profesores de la carrera, que me han transmitido a mi y al resto de compañeros, todas las ganas por continuar formándome en el mundo de la Informática.

Gracias a todos.

RESUMEN

Este proyecto fin de carrera consiste en el desarrollo de una herramienta que ayude a transformar un Formulario HTML en un Formulario VoiceXML, esto se conseguirá a través de un pequeño asistente que irá guiando al usuario paso a paso, para que configure el formulario como desee.

El Formulario VoiceXML ofrece una comunicación multimodal con el ordenador. El formulario se podrá rellenar de la manera tradicional, mediante teclado y ratón, o bien con la tecnología VoiceXML, basada en que el usuario indique los datos del formulario por voz al sistema.

El resultado de esta herramienta será un formulario adaptado al control por voz, algunos de sus usuarios podrían ser deficientes visuales o invidentes, con ello aumentará la accesibilidad a la navegación por la red con la voz.

Palabras clave: VoiceXML, HTML, Interacción Oral, Comunicación Multimodal, Asistente, Formulario.

ABSTRACT

This final degree project consists of the development of a tool that helps to transform an HTML form in a VoiceXML form; this will be accomplished through a simple assistant that will guide the user step by step to configure the form to his liking.

A VoiceXML form provides a multimodal communication with the computer.You can fill out the form in the traditional way or with VoiceXML technology, based on the user fills the form fields by voice.

The result of this tool will be aform adapted to voice control, some users may be visually impaired or blind, and this will increase the accessibility to the use of Web surfing by voice.

Keywords: VoiceXML, HTML, Oral Interaction, Multimodal Communication, Assistant, Form.

ÍNDICE

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

Capítulo 1:

Introducción

CAPÍTULO1: INTRODUCCIÓN

En este capítulo se muestra la motivación y objetivos principalesdel proyecto final de carrera, definiendo brevemente que es un sistema de comunicación multimodal.

También, se definen los objetivos principales del proyecto así como los subobjetivosy también se detalla la estructura de este documento, explicando sin entrar en detalle los capítulos de los que se conforma este documento.

1.1. MOTIVACIÓN

En el actual siglo XXI se está produciendo un auge en el uso de las tecnologías, desde el punto de vista del desarrollo y puesta en escena de muchas de ellas, hasta el descubrimiento de otras, que intentan hacernos el día a día más fácil y llevadero.

Cada vez más, el ser humano tiene contacto con todo tipo de máquinas y sistemas de carácter electrónico, desde ordenadores personales hasta dispositivos móviles, pasando por todo tipo de máquinas que se usan diariamente, unas veces para ocio y otras para el uso en un entorno laboral o académico.

Éste proyecto tratasobre la comunicación multimodal, que en una breve definición se puededecir que es la forma de comunicarse por parte de un humano con una máquina y viceversa, por dos o más métodos o maneras diferentes.

La motivación principal de este proyecto es la de realizar una herramienta de uso sencillo qué facilite la comunicación entre una persona discapacitada, en particular una persona deficiente visual y una máquina, al navegar por una página

de internet.

Cada vez el uso de Internet y de las nuevas tecnologías en distintos dispositivos es alto y está más expandidopor usuarios de todo tipo, por ello las tecnologías que existen actualmente y las nuevas se tienen que habituar a las necesidades o problemas que pueda tener un usuario, como en este caso.

Otra de las motivaciones para este proyecto fin de carreraha sido el más que probable uso pedagógico de esta tecnología para niños y ancianos, los cuales puedan empezar a conocer el mundo de Internet lo más sencillamente posible, cómo se navega y para qué sirve, la comunicación multimodal ayudará y las tecnologías de la manera más fácil.

Durante el transcurso de los 3 cursos académicos de la Ingeniería Técnica, al cursar asignaturas que se basaban en aprender y poner en práctica algunos conceptos importantes de un sitio web, como son la usabilidad, accesibilidad y funcionalidad, ha sido de gran ayuda para elegir este proyecto y para llevarlo a cabo de la mejor manera posible.

1.2. OBJETIVOS

El objetivo fundamental de este proyecto fin de carrera es el de realizar una herramienta que sirva de asistente para el usuario con la finalidad de convertir un sitio web de uso normal, en un sitio web con una característica especial, que sea multimodal, es decir, que el usuario pueda interactuar con la máquina por diferentes medios, como se hace habitualmente teclado + ratón y también con el uso de la voz.

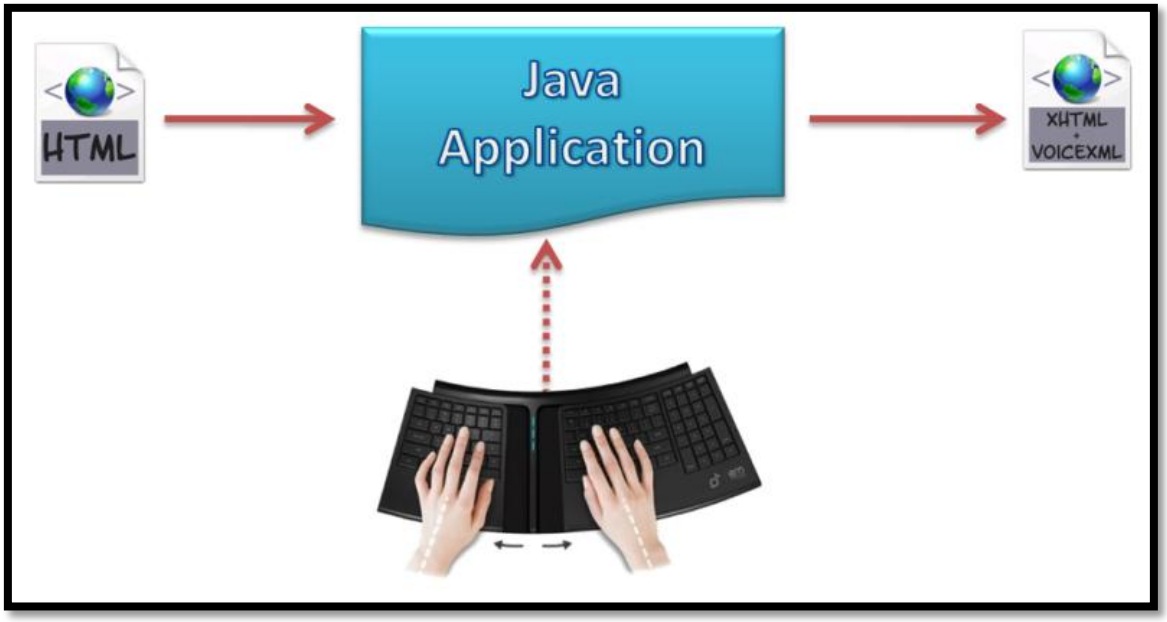

Ilustración 1 - Visión General de la aplicación.

Esta herramienta tiene que ser fácil de utilizar para el usuario y que no requiera un gran aprendizaje de los lenguajes de programación para poder hacer uso de ello, también se debe entender su funcionamiento de manera clara y concisa.

- Interactuar con el usuario, a partir de un asistente para construir finalmente el sitio web adaptado a la comunicación multimodal.
- Obtener del sitito web original información para construir el nuevo sitio web con la capacidad de ser multimodal.
- Obtener un sitio web accesible y navegable para personas con problemas en la visión.
- Minimizar el volumen de información que debe introducir el usuario por teclado para que el sistema pueda generar el nuevo sitio web.

1.3. FASES DEL DESARROLLO

Para realizar este proyecto se ha utilizado el modelo clásico de Ciclo de Vida de un Proyecto Softwareconocido como "waterfallmodel" o en cascada, que fuepropuesto por Winston W.Royce, este científico informático promulgó este modelo en 1970 [1].

Las fases de este modelo para un proyecto software son: *Análisis de requisitos, Diseño del Sistema, Diseño del Programa, Codificación, Pruebas, Implantación y Mantenimiento. Ver en [Ilustración2]*.

Para este proyecto software se han simplificado algunas de las fases del modelo propuesto por el científico informático de WinstonW.Royce, y alguna de ellas han sido suprimidas. Las fases propuestas para este proyecto fin de carrera son las siguientes*[Tabla 1]*:

Tabla 1 - Esquema Fases del Proyecto.

Documentación	Análisis de Requisitos.
	Diseño del Sistema y Programa.
	Desarrollo del código.
	Pruebas.

En las siguientes líneas se definen en profundidad cada una de las fases de este proyecto.

- **Análisis de Requisitos**

En esta primera fase se analizan las necesidades o requisitos de los usuarios finales del software, estos requisitos servirán para conocer los objetivos principales del proyecto.

Todos estos requisitos se recogerán individualmente en una tabla con una breve especificación de cada uno de ellos.

También en esta fase se realizará un pequeño proceso de análisis de la nueva tecnología a usar, para disponer del conocimiento suficiente para poder diseñar el sistema.

- **Diseño del Sistema y Programa**

En esta etapa se realizarán dos labores fundamentales para el buen desarrollo del proyecto.Primero se descompone y organiza el sistema en elementos que puedan elaborarse por separado, en nuestro caso sólo habrá un efectivo para hacerlo, asique apenas se podrá trabajar en paralelo.

Esta primera fase se realizará un diseño a alto nivel del sistema y de sus elementos y partes a partir de los objetivos que se marcaron en el análisis, sin incluir detalles técnicos de ninguna clase, ya que será el siguiente paso a realizar.

La segunda parte de esta fase será dependiente de la primera de la fase, ya que después de haber analizado con un detalle de alto nivel, hay que diseñar este sistema, con elementos más técnicos y sobre todo basados en el lenguaje de programación que se utilizan.

El objetivo fundamental de esta segunda fase, es la de confeccionar un documento lo más detallado posible, para que en la siguiente etapa, que será la de Implementación, sea lo más sencillo usando las especificaciones de esta fase.

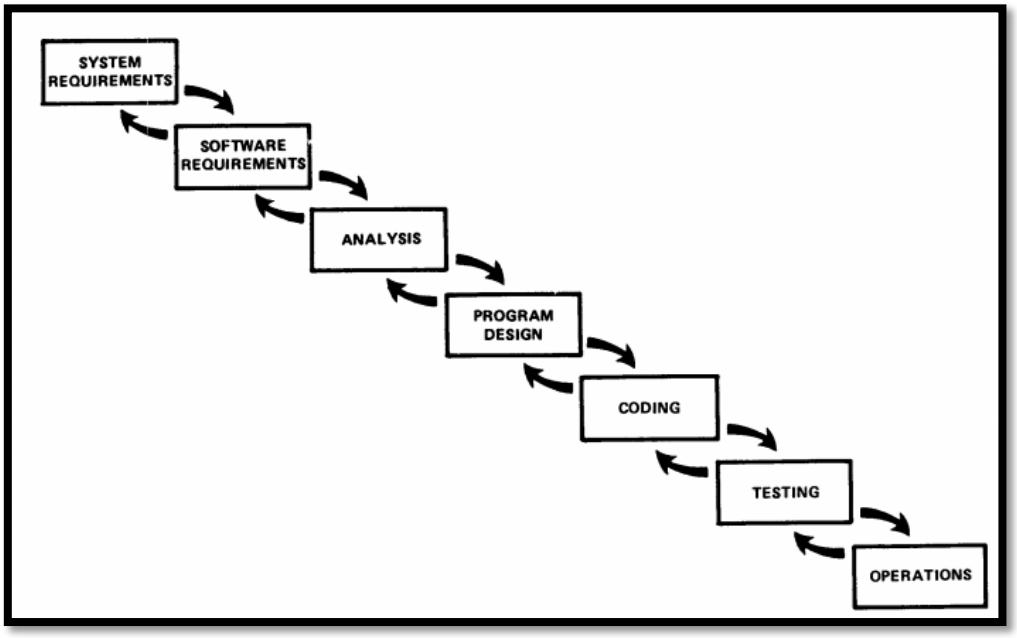

Ilustración 2 - Ciclo de Vida de Software Winston Royce[1].

- **Desarrollo del código**

Es la fase en donde se implementa el código fuente, a partir de los detalles técnicos que se indicaron en la fase anterior. En esta fase es lo más común que se realicen prototipos, versiones de demostración y también pequeñas pruebas para subsanar errores.

También se usan librerías del lenguaje de programación seleccionado y código libre para confeccionar esta importante fase de implementación, todos estos recursos harán más sencilla la labor del programador.

- **Pruebas**

Esta es una de las fases del proyecto más importantes, se trata de testear el código fuente programado anteriormente, las pruebas tienen que ser minuciosas para cerciorarse de que los requisitos que se especificaron en la fase

correspondiente, lógicamente que también el programa tiene un funcionamiento correcto sin excepciones.

En el caso de que se detectaran algunos errores en esta fase, se tiene que volver a realizar cambios en el código volviendo a la fase de desarrollo del código.

- Documentación

Esta será la última fase del proyecto, en la que se escribirá todo lo que ha acontecido durante el proyecto, para que en futuras ocasiones esta documentación pueda ser utilizada y entendida por otros Ingenieros.

Esta fase al estar la última no se tiene que hacer en último lugar, es importante ir actualizando los cambios semanalmente para que se quede recogido todos los análisis del sistema y todo lo que pueda ser relevante para el proyecto.

Se ha optado por acoplar las fases de Diseño del Sistema y Diseño de Programa en una sola fase, se ha decidido suprimir la fase de Implantación y la fase de Mantenimiento, ya que la aplicación no abarcará estas fases en ningún momento del ciclo de vida. Se ha decidido incluir una nueva fase que sea la de documentación, que será continuadurante todo el proyecto.

1.4. ESTRUCTURA DE LA MEMORIA

A continuación se define la estructura de la memoria, y se detallanbrevemente el contenido de los capítulos y la finalidad de cada uno de ellos.

- **Capítulo 1: INTRODUCCIÓN**

Primer epígrafe en el que se definen los objetivos del Proyecto Fin de Carrera y la finalidad del mismo. También la motivación del mismo y un breve inciso en la Comunicación Multimodal.

- **Capítulo 2: ESTADO DEL ARTE**

Este segundo capítulo de la memoria del proyecto fin de carrera se centra describir en profundidad el carácter funcional y técnicodela tecnología a utilizar, que será VoiceXML, también se definirán conceptos clave que serán muy frecuentes durante todas las etapas del proyecto.

- **Capítulo 3: ANÁLISIS**

En este capítulose explica la fase de análisis, que se realizará para conocer las necesidades del software,que se desarrollará en fases posteriores.

Con las herramientas de Casos de Uso y también de Requisitos Funcionales y No Funcionales se dará una visión clara de este capítulo.

Se incorpora también un pequeño Plan de Pruebas para la comprobación de que se cumplen todos y cada uno de los Requisitos Funcionales.

- **Capítulo 4: DISEÑO DEL SISTEMA**

Este capítulo se encarga de dar a conocer el diseño del sistema, en organizar y descomponer las distintas partes del sistema, que se pueden realizar por separado, esta fase es muy importante para realizar una buena implementación.

Para la representación gráfica del diseño del sistema se ha optado por el Diagrama de Clases y al Diagrama de Secuencia, ambos diagramas del lenguaje UML.

- **Capítulo 5: IMPLEMENTACIÓN Y PRUEBAS**

En este epígrafe se muestran los métodos más relevantes del código fuente, divididos por la funcionalidad que aportan al sistema.

También incluye un control de las pruebas definidas en el plan de pruebas en el capítulo de Análisis.

- **Capítulo 6: GESTIÓN DEL PROYECTO**

En este capítulo se detalla la planificación del proyecto, entanto temporal como de costes y también la planificación presupuestaria de todo el proyecto.

En relación al apartado de la planificación se realiza un estudio de seguimiento del proyecto, en él se puede ver el presupuesto y coste económico del proyecto, también se puede ver coste humano y temporal del mismo.

- **Capítulo 7: CONCLUSIONES Y LINEAS FUTURAS**

En estaúltima sección se resumen las conclusiones del Proyecto,los objetivos que se han podido completar con éxito y los que no.

Para finalizar este capítulo, se expondrán algunas posibles mejoras y ampliaciones de la aplicación y poder acrecentar sus funcionalidades en trabajos futuros.

Justo al término de estos capítulos principales,se puede encontrar la información detallada sobre el glosario de términos, donde se desglosan los términos más comunes usados en esta memoria.

En la lista de acrónimos, se describen las definiciones sobre las siglas más importantes.

En la lista de Referencias, se indican cuales de ellas han sido utilizadas y en que momento de la memoria. La bibliografía se ha incluido dentro de este punto, en ella se detallan todas las fuentes que han sido necesarias y utilizadas para la confección del presente documento de memoria del proyecto fin de carrera.

Capítulo 2:

Estado del Arte

CAPÍTULO 2: ESTADO DEL ARTE

El segundo capítulo de la memoria se dedica a dar a conocer una visión global del estado del arte sobre la tecnologíaVoiceXML, también se muestran algunos ejemplos de código para entender de mejor manera este lenguaje.

2.1. INTRODUCCIÓN

Para realizar este estudio sobre la tecnología VoiceXML, se ha realizado una labor de investigación, consultando los estándares de World Wide Web Consortium (W3C) [2], y también de los sub-lenguajes asociados que sonmás importantes.

Como soporte se han consultado otras fuentes bibliográficas, que a continuación se detallan. A posteriori se realizará el desarrollo de esta aplicación de este proyecto basada en esta tecnología.

VoiceXML – Es el acrónimo de Voice Extensible MarkupLanguaje que significa Lenguaje de Marcado basado en la voz, este lenguaje se basa en el conocido lenguaje de etiquetas XML.Este lenguaje está compuesto a su vez, por varios sub-lenguajes y módulos que hacen que sea un lenguaje sólido y compacto.

En este estudio previo se abordarán, tanto la gestión del flujo del dialogo, como la creación de gramáticas y su síntesis, y final se revisaran los elementos acerca de la síntesis del habla.

Algunas aplicaciones prácticas de este lenguaje son:

- El uso en páginas web: para poder acceder a un contenido a través de la voz y también realizar distintas operaciones.

- Contestador Automático: a partir de unas preguntas realizadas por el contestador y las respuestas que del usuario, el sistema irá ofreciendo distinto contenido.

Esta tecnología es desconocida para muchos desarrolladores y usuarios, aún se pueden encontrar problemas abiertos y por resolver con esta tecnología.

La tecnología VoiceXML permite desarrollar aplicaciones que tienen la capacidad de:

- Reconocer tonos de marcado de un teléfono (DTMF).
- Reconocer voz por parte del habla de un usuario.
- Reproducción de ficheros de audio y "streams".
- Síntesis de voz (Text-to-speech).

Algunos de las funciones que VoiceXML no puede controlar son:

- Establecer multiconferencias con uno o más interlocutores.
- Generar llamadas.
- Encaminar y dirigir llamadas.

Para comprender un poco mejor la funcionalidad básica de este lenguaje, se puede apreciar el siguiente gráfico *[Ilustración 3]*:

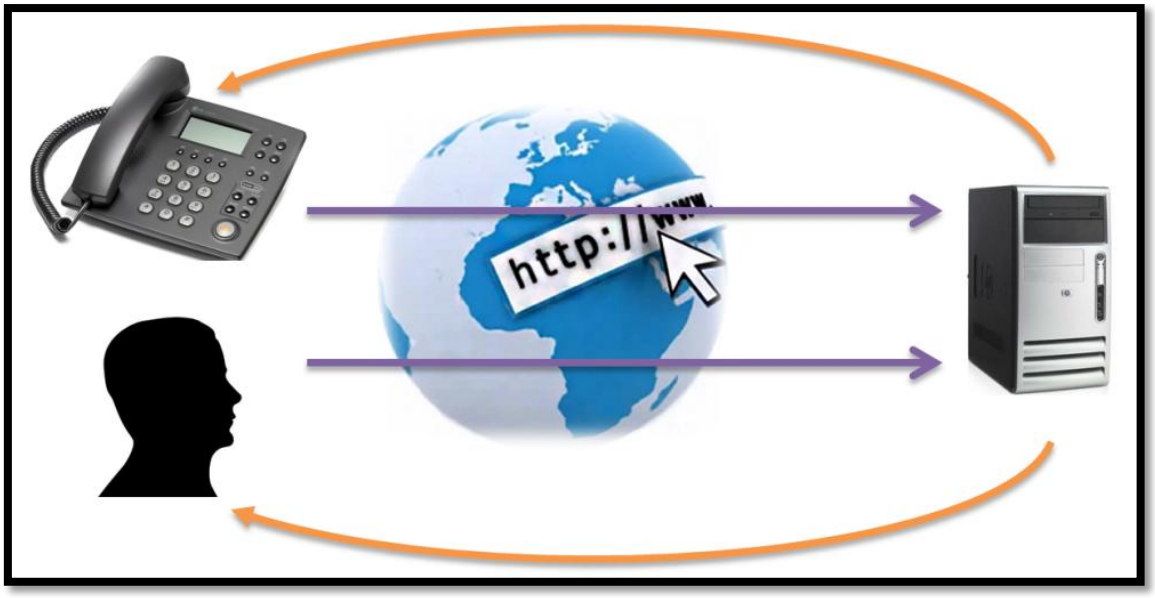

Ilustración 3 - Visión General Sistema VoiceXML.

Se puedeapreciar que los agentes externos al sistema serán tanto un teléfono como un humano que usa un software con VoiceXML, la comunicación pasa a través de Internet por diferentes protocolos, y llega al sistema, en la parte derecha, que será el motor que interactuará con los diferentes agentes.

La comunicación en este ejemplo podría ser por varios canales, por el auricular del teléfono usando la voz para interactuar con el sistema, por el auricular para escuchar y el teclado para interactuar con el sistema y el humano que con ayuda de su ordenador personal recibirá la información del sistema por los altavoces y a través del micrófono podrá interactuar con el sistema.

2.2. CONTROL Y GESTIÓN DEL FLUJO DEL DIÁLOGO

Según la RAE, el dialogo es la "plática o conversación entre dos o más personas, que alternativamente manifiestan sus ideas o afectos" [3].

En esta sección se trata de estilizar nuestro código de los documentos VoiceXML, relacionar nuestro documento con un tipo de dialogo que se pueda dar entre el ordenador y el usuario, puede haber varios sub-diálogos dentro de un diálogo, esta construcción del diálogo ayudará y guiará la interacción entre ordenador y usuario.

2.2.1. ETIQUETAS BÁSICAS SOBRE DIÁLOGOS EN VOICEXML

VoiceXML ofrece 2 tipos de diálogos "`form`" y "`menú`", mientras que el tipo de diálogo menú es sólo para tipos especiales de aplicaciones VoiceXML, es mucho más corriente el uso del dialogo "`form`" para aplicaciones más sencillas.

Este estudio se basa en el tipo de diálogo "`form`", se implementa con la etiqueta "`<form>`" que tendrá en su interior etiquetas como "`<field>`", "`<record>`" y "`<subdialog>`" que a continuación se puede ver con más detalle.

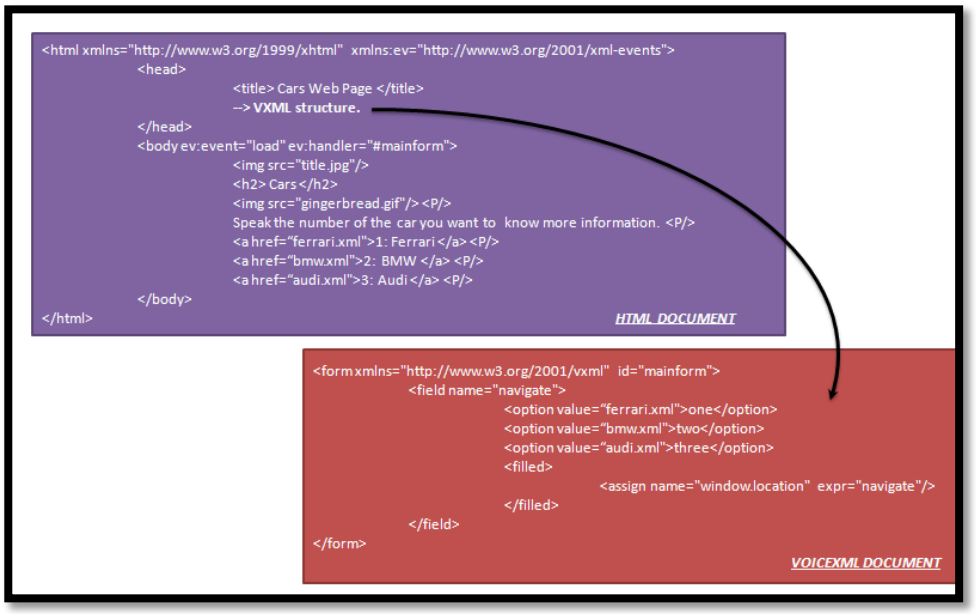

Ilustración 4 - Visión General del código de una aplicación VoiceXML.

2.2.1.1. <FORM>

El uso de esta etiqueta está reservado para la construcción de diálogossencillos, que se pueden incluir en páginas web sencillas, este elemento es el elemento raíz del que luego dependerá otros para el control y seguimiento del diálogo.

Los atributos de los que dispone son los siguientes:

- **Id:** que indicará la cadena de caracteres con la que se podrá identificar a este diálogo "`form`".
- **Scope:** indicará el ámbito del diálogo, puede tomar los valores dialogo (lo más común) o documento que indicará que es un formulario de gramáticas.

2.2.1.2. <FIELD>

Trabaja como subetiqueta del diálogo "`form`", este elemento declara una entrada en el formulario, con la entrada que da el usuario, el sistema intenta ver si corresponde con alguna de las opciones de la gramática que se ha definido previamente.

El valor de la entrada se guarda en una variable que puede ser usada dentro del documento. Algunos de sus atributos más importantes son:

- **Name:** atributo obligatorio usado para definir el nombre de la variable. Éste atributo debe ser único y no puede ser repetido en el mismo "`<form>`".
- **Type:** atributo obligatorio, usado para definir el tipo de datos que contendrá este "`<field>`". Algunos de los valores que puede tomar son: "`boolean`" *(true o false),* "`date`" *(tiempo),* "`currency`" *(moneda),* "`phone`" *(teléfono)….*
- **Expr:** atributo opcional, usado para definir el valor inicial de la variable.

A continuación se pueden ver unos ejemplos de esta etiqueta, en este primer ejemplo se puede apreciar como se le pide al usuario que diga un número y posteriormente la aplicación se lo reproduce.

```
<form id="Ejemplo_de_Captacion_de_Numeros">
<field name ="Selección Numero" type="number">
<prompt>Por favor diga claramente un número.</prompt>
<filled>
<prompt>Usted ha seleccionado el numero
<valueexpr="Selección_Numero"/>
</prompt>
</filled>
</field>
</form>
```

Tabla 2– Ejemplo<field>1.

En el siguiente ejemplo, se debe elegir un elemento de unos cuantos dados se utiliza el elemento "<enumerate>".

```
<form id="Ejemplo_de_eleccion_de_una_lista">
<fieldname ="Eleccion Color">
<prompt>Por favor indique alguno de los siguientes colores para
su ordenador.</prompt>
<option>Rojo</option>
<option> Azul</option>
<option> Negro</option>
<option>Blanco</option>
<filled>
<prompt>Usted ha seleccionado el color <valueexpr="
Eleccion_Color"/></prompt>
</filled>
</field>
</form>
```

Tabla 3 - Ejemplo <field> 2.

2.2.1.3. <FILLED>

La etiqueta "<filled>" es la encargada de permitir al desarrollador especificar las acciones a tomar cuando se produce una coincidencia entre el "input" recibido por la gramática y lo que hay dentro dela misma.

Es decir, depende donde se emplace esta etiqueta se puede tener un flujo del dialogo u otro, lo que mejor le convenga para el desarrollador.

2.2.1.4. <RECORD>

Esta etiqueta se usa para recoger datos sobre del usuario mediante un archivo de audio, y todo lo grabado es graba en una variable. Es posible utilizar la grabación en posteriores secciones del documento VoiceXML ya que se graba en el sistema. Algunos de los atributos de esta etiqueta son:

- **Name:** atributo requerido usado para definir el nombre de la variable. El Name debe ser único y no puede ser repetido en el mismo "`<form>`".
- **Expr:** atributo opcional, usado para definir el valor inicial de la variable.
- **Beep:** atributo opcional, si el valor del atributo es true, un tono se inserta antes de que empiece la grabación, el valor por defecto es false.
- **Maxtime:** es opcional, define el tiempo máximo de grabación.

A continuación se puede apreciar un ejemplo, en el cual se utiliza la etiqueta "`<record>`".

```
<form>
<record beep="true" name="Felicitacion" maxtime="60s">
<prompt>En cuanto suene el tono puede empezar a grabar su Felicitación Navideña</prompt>
<noimput>No se ha escuchar nada, por favor vuelva a intentarlo una vez mas </noimput>
<fieldname="Expresion_Final">
<prompt>Su felicitación sido la siguiente: <valueexpr="Felicitacion"></prompt>
</field>
</form>
```

Tabla 4 - Ejemplo <record>.

2.2.1.5. <SUBDIALOG>

Esta etiqueta se usa para invocar a un diálogo que previamente ya se ha escrito en el mismo documento o que está en otro fichero distinto, cuando se llama a esta etiqueta se ejecutara un nuevo proceso que contiene la información del

diálogo. Alguno de sus atributos más importantes son:

- **Name:** es imprescindible, sirve para definir el nombre de la variable que guarda el resultado del diálogo raíz.
- **Namelist:** atributo obligatorio, mantiene una lista de variables de la etiqueta.
- **Src:** atributo obligatorio, indica la URI del subdialogo, puede ser dentro del documento o fuera del mismo.
- **Method:** atributo obligatorio, indica el método que se usa para la respuesta del subdialogo, puede tomar los valores `get` y `post`.

A continuación se puede ver más claro, un ejemplo sobre esta etiqueta.

```
<form id="sub">
<block>
<prompt>
      Se está transfiriendo al subdialogo para recolectar la
información que nos aporta.
</prompt>
</block>
<subdialogsrc="http://dialogoejemplo.com/ejemplo1.vxml"
name="Resultado Subdialogo">
<filled>
<prompt>
      Bienvenido al diálogo principal, el subdialogo ha calculado
edad, y el resultado es <valueexpr="Resultado Subdialogo.edad">
</prompt>
</filled>
</subdialog>
</block>
</form>
```

Tabla 5 - Ejemplo <subdialog> (Fichero de diálogo principal)

```
<form id="Subdialogo">
<field name="edad" type="digits">
<prompt>Por favor diga su edad.</prompt>
<filled>
      <return namelist="age"/>
</filled>
</field>
</form>
```

Tabla 6 - Ejemplo <subdialog> (Fichero con el Subdialogo invocado por el principal).

2.2.2. ETIQUETA DE CONTROL DE DIÁLOGOS

Existen 2 elementos básicos para el control del diálogo dentro de la etiqueta `form`: "`<block>`" y "`<initial>`", se puede ver estas dos etiquetas con más profundidad.

2.2.2.1. <BLOCK>

Se utiliza para construir un bloque de contenidos ejecutables que contienen el texto, éste será reproducido íntegramente al usuario por parte de la aplicación.

Dispone de atributos, el más usual es "`name`" que se utiliza para asignar un nombre de variable a ese bloque para que pueda ser utilizado en otros lugares del documento. También dispone del elemento "`expr`" el cual indica el valor por defecto que tendrá ese bloque.

En el siguiente ejemplo se puede ver que se crea un bloque el cual tiene una variable Equipo_Futbol y con un valor por defecto "Real Madrid", en la siguiente frase se utiliza la etiqueta `value` con un el valor referente a la variable "Equipo_Futbol".

El Resultado que se le ofrecería al usuario sería: "El mejor equipo de fútbol es el Real Madrid".

```
<?xmlversion="1.0"?>
<vxmlversion="2.1"
xmlns="http://www.w3.org/2001/vxml">
<form>
<block>
<varname="Equipo_Futbol" expr="'Real Madrid'"/>
          El     mejor     equipo     de     fútbol     es     el
<valueexpr="Equipo_Futbol" />
</block>
</form>
</vxml>
```

Tabla 7 - Ejemplo <block>.

2.2.2.2. <INITIAL>

Esta etiqueta se utiliza para los diálogos de iniciativa mixta (donde las expresiones del primer usuario determinan el flujo de la aplicación), permitiendo al usuario completar toda la información del formulario y según la información dar una respuesta distinta al usuario.

Algunos de sus atributos más importantes son:

- **Name:** es imprescindible, sirve para definir el nombre de la variable que guarda la información que contiene la etiqueta "`<initial>`".
- **Cond:** atributo opcional, se usa para saber si este elemento ha sido utilizado por el documento, tiene como valores true o false.

En el siguiente ejemplo se puede ver como se utilizan 2 tipos distintos de obtener información uno de la manera fácil con el uso de 2 "`<field>`" uno para la ciudad de salida y otro para la ciudad de destino y la otra maneraes el uso de la etiqueta "`<initial>`" la cual pide al usuario a la vez tanto la ciudad de origen como la ciudad de destino. Para la búsqueda de gasolineras en su trayecto.

```
<form id="ObtenerInformacion">
      <grammar  src="http://gramaticaejemplo.com/gramatica1.vxml"
type="application/grammar+xml"/>
      <block>
            Bienvenido al Localizador de Gasolineras PETROLSEARCH
      </block>
      <initialname="Inicial">
            <prompt>Por favor indique la ciudad de salida y de
llegada de
su viaje para poder buscar todas las gasolineras que habrá en
su trayecto.</promt>
            <nomatchcount='1'>
Por ejemplo "De Madrid a Valencia"    </nomatch>
      </initial>

      <field name="Ciudad_Salida">
<grammar src="gramatica2.vxml" type="application/grammar+xml"/>
<prompt>Por favor indique la ciudad de salida.</promt>
      </field>

      <field name="Ciudad_Llegada">
<grammar src="gramatica3.vxml" type="application/grammar+xml"/>
<prompt>Por favor indique la ciudad de llegada.</promt>
      </field>
</form>
```

Tabla 8 - Ejemplo <initial>.

2.2.3. PASOS PARA LA CONSTRUCCIÓN DE DIÁLOGOS Y SUBDIALOGOS

A continuación se definen unos pasos básicos para realizar la construcción de manera correcta de una aplicación VXML, con unos flujos de dialogo sencillos.

2.2.3.1. PRIMER PASO: ESCRIBIR CORRECTAMENTE EL DOCUMENTO VOICEXML

En el desarrollo de una aplicación con VoiceXML muchas veces se puede encontrar que elementos y construcciones de código similares se repiten constantemente. Existen etiquetas muy potentes en este lenguaje para poder remediar repetir el mismo código durante "x" veces.

Ejemplo:

```
<?xmlversion="1.0" encoding="UTF-8"?>
<vxmlversion = "2.1">
<metaname="Autor:                    Alvaro                    Casado"
content="100077040@alumnos.uc3m.es"/>
<form id="Formulario1">
<field name="Campo1">
<prompt>
      ¿Cual es el coche que más le gusta?
      Para Todoterreno, pulse o diga one
      Para Deportivo, pulse o diga two
      Pata Monovolumen, pulse o diga three
</prompt>
<grammar type="text/gsl"> [dtmf-1 dtmf-2 dtmf-3] </grammar>
<filled>
        Usted ha elegido <valueexpr="Campo1"/>
</filled>
</field>
</form>
</vxml>
```

Tabla 9 - Ejemplo de formulario con un solo campo.

Este ejemplo es muy sencillo ya que sería un cuestionario con sólo una pregunta con tres posibles respuestas.

2.2.3.2. SEGUNDO PASO: APLICAR REFERENCIAS ENTRE SUBDIALOGOS

Como se puede apreciar el ejemplo anterior es extremadamente sencillo, por ello en el caso de este proyecto se realizará un formulario complejo con varios campos y también con campos de diferentes tipos.

En este ejemplo se puede ver como se van encadenando diferentes subdialogos, para realizar un formulario con varias preguntas y varias respuestas.

```
<form id="Formulario1">
<subdialogname="Subdialogo1" src="Subdialogo.xml">
<paramname="Pregunta"
expr="¿Cual Piensa que es la mejor marca de ordenadores?
Si piensa que es Acer, pulse o diga one
          Si piensa que es MAC, pulse o diga two
          Si piensa que es Toshiba, pulse o diga three"/>
<filled>
<gotonextitem="Subdialogo2"/>
</filled>
</subdialog>

<subdialogname="Subdialogo2" src="Subdialogo.xml">
<paramname="Pregunta"
expr="¿Cual cree que es la mejor marca de Tarjetas Gráficas?
Si piensa que es ATI, pulse o diga one
          Si piensa que es NVIDIA, pulse o diga two
          Si piensa que es IBM, pulse o diga three"/>
<filled>
<gotonextitem="Subdialogo3"/>
</filled>
</subdialog>

<subdialogname="Subdialogo3" src="Subdialogo.xml">
<paramname="Pregunta"
expr="¿Cual cree que es la mejor procesador de Texto?
Si piensa que es Microsoft Word, pulse o diga one
     Si piensa que es Open Office, pulse o diga two
     Si piensa que es Kwrite (Linux), pulse o diga three"/>
<filled>
<prompt>
       Felicidades, ya ha terminado su test.
</prompt>
</filled>
</subdialog>
</form>
```

Tabla 10- Ejemplo de formulario con varios campos.

En este ejemplo también se puede apreciar como se ha cambiado la estructura de dialogo cambiando las etiquetas "`<field>`" y "`<grammar>`" por una estructura de "`<subdialog>`".

El único lugar que se puede utilizar la etiqueta "<subdialog>" es como hijo del "<form>", también se utiliza la etiqueta"<param>"que es donde se guardan las preguntas y las posibles respuestas.

2.2.3.3. TERCER PASO: CONSTRUCCIÓN DEL SUBDIALOGO

Si se analiza el cuestionario de ejemplo se puede ver, que se conserva el formato, es decir, existe una pregunta y tres posibles respuestas a esa pregunta.

Se puede apreciar que las preguntas que se tienen que realizar en el dialogo,se invocan mediante un "<prompt>", todas las respuestas obtenidas se almacenan en la variable Respuestas, que se llama en la etiqueta "<filled>".

```
<?xmlversion="1.0" encoding="UTF-8"?>
<vxmlversion = "2.1">
<meta name="Alvaro Casado" content="100077040@alumnos.uc3m.es"/>
<form id="Formulario_con_Subdialogos">
<field name="Respuestas">
<grammar type="text/gsl"> [dtmf-1 dtmf-2 dtmf-3]</grammar>
<prompt><value expr="Pregunta"/></prompt>
<filled>
<return namelist="Respuestas"/>
</filled>
</field>
</form>
</vxml>
```

Tabla 11 - Fichero de invocación a subdialogos.

2.2.3.4. CUARTO PASO: MEJORA DE LA APLICACIÓN

Para pulir un la aplicación, se puede ampliar algunas funcionalidades, como es el disponer de manejadores de errores, estos manejadores indican si las respuestas a las preguntas son las correctas, o si por ejemplo no se ha entendido lo que ha dicho el usuario.

Para ello se usan en este ejemplo las etiquetas "<if>" y "<elseif>" con las que se manipula el flujo del dialogo, a merced del desarrollador de la aplicación.

```
<?xmlversion="1.0" encoding="UTF-8"?>
<vxmlversion = "2.1">
<meta name="Alvaro Casado" content="100077040@alumnos.uc3m.es"/>
<form id="Formulario1">
<subdialogname="Subdialogo1" src="Subdialogo1.xml">
<paramname="Pregunta" expr="¿Cual Piensa que es la mejor marca de
PC's?
    Si piensa que es Acer, pulse o diga one
    Si piensa que es MAC, pulse o diga two
    Si piensa que es Toshiba, pulse o diga three"/>
<filled>
<ifcond="Subdialogo1.Respuestas=='1'"> Disculpe, la respuesta es
incorrecta. La respuesta correcta es Toshiba.
<elseifcond="Subdialogo1.Respuestas=='2'"/> Disculpe, la respuesta
es incorrecta. La respuesta correcta es Toshiba.
<elseifcond="Subdialogo1.Respuestas=='3'"/>    Su    respuesta    es
correcta.
<elseifcond="Subdialogo1.Respuestas=='Pasapalabra'"/>Pasapalabra
vale para todo, es una respuesta comodín.
<else/>
<prompt> No ha dicho ninguna opción correcta. </prompt>
</if>
<gotonextitem="Subdialogo2"/>
</filled>
</subdialog>

<subdialogname="Subdialogo2" src="Subdialogo2.xml">
<paramname="Pregunta" expr="¿Cual es la mejor marca de Tarjetas
Gráficas?
    Si piensa que es ATI, pulse o diga one
    Si piensa que es NVIDIA, pulse o diga two
    Si piensa que es IBM, pulse o diga three"/>
<filled>
<ifcond="Subdialogo2.Respuestas=='1'"> Disculpe, la respuesta es
incorrecta. La respuesta correcta es IBM.
<elseifcond="Subdialogo2.Respuestas=='2'"/> Disculpe, la respuesta
es incorrecta. La respuesta correcta es IBM.
<elseifcond="Subdialogo2.Respuestas=='3'"/>    Su    respuesta    es
correcta.
<elseifcond="Subdialogo2.Respuestas=='Pasapalabra'"/>Pasapalabra
vale para todo, es una respuesta comodín.
<else/>
<prompt> No ha dicho ninguna opción correcta. </prompt>
</if>
<gotonextitem="Subdialogo3"/>
</filled>
</subdialog>

<subdialogname="Subdialogo3" src="Subdialogo3.xml">
<paramname="Pregunta" expr="¿Cual cree que es la mejor procesador
de Texto?
    Si piensa que es Microsoft Word, pulse o diga one
    Si piensa que es Open Office, pulse o diga two
    Si piensa que es Kwrite (Linux), pulse o diga three"/>
<filled>
<ifcond="Subdialogo3.Respuestas=='1'"> Disculpe, la respuesta es
incorrecta. La respuesta correcta es MSWord.
<elseifcond="Subdialogo3.Respuestas=='2'"/> Disculpe, la respuesta
es incorrecta. La respuesta correcta es MSWord.
<elseifcond="Subdialogo3.Respuestas=='3'"/>    Su    respuesta    es
correcta.
<elseifcond="Subdialogo3.Respuestas=='Pasapalabra'"/>Pasapalabra
```

```
vale para todo, es una respuesta comodín.
<else/>
<prompt> No ha dicho ninguna opción correcta. </prompt>
</if>
<gotonextitem="Final"/>
</filled>
</subdialog>

<blockname="Final">
<prompt> Felicidades, ya ha terminado el cuestionario. </prompt>
</block>
</form>
</vxml>
```

Tabla 12 - Fichero con los subdialogos.

2.3. RECONOCIMIENTO DEL HABLA Y ESPECIFICACIÓN DE GRAMÁTICAS

SRGS - Speech Recognition Grammar Specification, acrónimo de Reconocimiento del Habla y Especificación de Gramáticas[8], este sublenguaje se encargade ofrecer al desarrollador varios la construcción de gramáticas, siguiendo los pasos necesarios para especificar palabras y posibles patrones de palabras.

```
<vxml:form id="TipodeComida">
  <vxml:block> Bienvenido al Restaurante Online </vxml:block>
  <vxml:field name="primerplato" xv:id="primerplato">
    <vxml: grammar type="application/x-jsgf">
      <![CDATA[
        grammar primeros;
        public <primeros> = ensalada | paella ;
      ]]>
    </vxml:grammar>
    <vxml:prompt> Elija el primer plato. </vxml:prompt>
  </vxml:field>
```

Ilustración 5 - Localización de la Gramática en un FORM.

2.3.1. INTRODUCCIÓN

Existen 2 formas de representar la sintaxis de una gramática, con una forma BNF aumentada y formato XML, cada una de ellas seguirá unas pautas para poder crearlas y desarrollarlas.

2.3.1.1. CONCEPTOS BÁSICOS

- **Dual-Tone Multi-Frequency (DTMF):** Se trata de un sistema de marcación por tonos, es decir cada número del teclado de un teléfono tiene asociado 1 tono con una frecuencia distinta, esto es para distinguir cada uno de los números.
- **Forma Normal de Backus Aumentada (ABNF):** Esta sintaxis consiste en texto plano (no XML) el formato es similar a la gramática BNF. En los siguientes apartados se pueden ver algunos ejemplos.

- **Formato XML:** Esta sintaxis utiliza elementos XML, lenguaje de etiquetas, para representar la construcción de la gramática.
- **User Agent:** es el procesador de gramática, que tendrá como labor reconocer los datos de entrada, intentar encajarlos en alguna de las gramáticas definidas previamente y producir un efecto o acción en el sistema acorde a la entrada recibida.
- **Reconocedor de Voz:** es un tipo especial de User Agent, que dispone de las siguientes características:

  Como entrada: dispone de una gramática para conocer cuales son las posibles palabras y patrones que recibe y también recibe palabras para reconocerlas dentro de nuestra gramática.

  Como salida: intenta comprobar las palabras obtenidas como entrada, a la gramática conocida, para posteriormente en caso favorable realizar un efecto sobre el sistema, y en caso negativo, que no se haya reconocido la palabra dentro de la gramática, realizar otro efecto distinto al anterior o no realizar ninguno.

2.3.1.2. ALCANCE Y USO

El principal uso y alcance de la gramática reconocedora de voz es la de permitir, que una aplicación de voz sea capaz de asociar lo que ha dicho el usuario (entrada), con lo que está permitido (reglas) en la gramática.

Algunas de las capacidades que tiene una gramática son las siguientes:

- **Adaptación de los datos del Hablante:** Algunos reconocedores de voz tienen la capacidad de ajustarse dinámicamente a la voz del usuario, por si éste usara el sistema en futuras ocasiones. Los datos del hablante u orador se pueden incluir listas de palabras de uso frecuente.

- **Configuracióndel reconocedor de voz:** El formato de la gramática no incorpora funciones para definir las características del reconocedor, tales como tiempos de espera, umbrales de reconocimiento ni tamaños de búsqueda.
- **Otras capacidades de procesamiento del habla:** La tecnología de procesamiento de voz existe para la identificación del idioma, la verificación correcta del habla y para el reconocimiento de los distintos hablantes que intervienen en una conversación.

2.3.1.3. CONVERSIONES DE GRAMÁTICA

Los formatos de gramáticas en XML y en ABNF pueden ser transformables, es decir sería posible convertir una gramática de forma ABNF en otra con forma XML, o inclusive, a la inversa, lo mismo ocurre con la semántica.Existen 2 equivalencias entre estos dos formatos:

Ambas gramáticas aceptan el mismo idioma como entrada y rechazan el mismo idioma como entrada.Ambas gramáticas analizan cualquier cadena de caracteres de manera idéntica.

Existen algunas limitaciones en la conversión de gramáticas de forma ABNF y forma XML:

Los espacios en blanco que no son parte de alguna estructura (p.e. etiquetas) se eliminan y no se tienen en cuenta para su conversión.

Algunas elementos de la forma XML no tienen equivalente en la forma ABNF como por ejemplo los Esquemas de XML, la estructura DTD, espacio de declaraciones y referencias (variables, referencias a otros XML...) y las instrucciones de procesamiento. Para un correcto proceso de conversión XML a ABNF, se debe tener escrita nuestra gramática XML la versión 1.0.

Comentarios propuestos por el constructor de la gramática pueden ser modificados.

2.3.1.4. INTERPRETACIÓN SEMÁNTICA

Un reconocedor de voz es capaz de hacer coincidir la entrada de audio del usuario comparándola con la gramática, para producir una transcripción de texto literal.

Por ejemplo, la frase: "Quiero unas gafas de cristales oscuros y montura de metal" se puede tener una estructura de datos XML del siguiente estilo*[Tabla 13]*:

```
<compra-gafas>
      <cristales>Oscuros</cristales>
<montura>Metal</montura>
</compra-gafas>
```

Tabla 13 - Ejemplo de Estructura Semántica en XML.

La construcción de etiquetas y el formato de la etiqueta proporcionan una semántica única para cada uno de los ejemplos que se aborden por un desarrollador.

La interpretación semántica que se lleva a cabo en el proceso de reconocimiento de voz, que se caracteriza por:

- **Contexto restringido:**La interpretación no resuelve las referencias anafóricas o que hacen referencia a algo ya mencionado.

  En el ejemplo *[Tabla13]* "Quiero unas gafas de cristales oscuros y montura de metal" le continuase otra frase "Éstas tienen que ser de color rojo", para se entiende que "Éstas" se refiere a "gafas", pero no para la interpretación semántica de VoiceXML, esto está fuera de su alcance.

- **Especificación de un dominio:** Una gramática de reconcomiendo de voz siempre está ligada y restringida a un dominio de entrada *(En el ejemplo [Tabla 13] es la "compra de un par de gafas")*, toda cualquier consulta fuera de ese dominio, está alejado del alcance de la interpretación semántica del sistema.
- **Específicas del idioma:** Cada idioma tiene unas estructuras lingüísticas únicas, y tendrán distintos resultados semánticos.

2.3.1.5. GRAMÁTICAS INTEGRADAS

El SRGS está diseñado para permitir que las gramáticas en forma XML y ABNF puedan ser incrustadas en otros documentos, como por ejemplo dentro de un documento VXML.

La inclusión de una gramática en forma XML, dentro de un documento XML puede lograrse con el XMLNS[4] (XML namespace) que permite definir vocabularios para diferentes instancias de XML (por ejemplo: usar el atributo nombre tanto para un cliente como para un vendedor), con un DTD[4] (Definición de tipo de documento) que es una descripción de estructura y función básica del formato de datos.

También cualquier gramática ABNF puede incluirse en cualquier documento XML, cambiando alguno de sus caracteres que no son compatibles.

2.3.2. DEFINICIÓN DE GRAMÁTICAS

Un buen proceso de reconocimiento de la voz, se basa por un lado en las gramáticas definidas en el problema y por otro lado en las palabras que pueda tener esa entrada de audio.

El reconocedor de voz analiza las palabras que le han sido introducidas y las tiene que comparar con las gramáticas que tiene definidas, para poder devolver una respuesta acorde con la entrada recibida.El formato de las gramáticas puede ser de 2 tipos como se ha visto anteriormente, en forma XML y en forma ABNF que son los más comunes. Ambas formas son equivalentes y pueden transformarse unas a partir de las otras, en algún caso perdiendo algo de semántica.

Mención aparte tendrá el idioma en el que se interactúa con el sistema, ya que las gramáticas se tendrán que basar en las reglas de cada idioma para que sean sintácticamente correctas.

2.3.2.1. TOKENS

Un "`token`" es la unidad básica de conocimiento de una gramática, son todo tipo de palabras o conjunto de ellas, que el desarrollador puede definir a su gusto y también es el elemento que permite el usuario pueda hablar e interaccionar con el sistema.

Algún ejemplo de token:

Madrid

"Proyecto Fin de Carrera"

2

Año 2009

Si se dispone de un espacio entre las palabras de un "`token`", el "`token`" debería ir entrecomillado como: "Proyecto Fin de Carrera". Existe una alternativa si no se quiere usar las comillas para definir nuestro "`token`",se pueden usar las siguientes etiquetas que definen de igual manera lo anterior:

```
<token>
      Proyecto Fin de Carrera
</token>
```

Tabla 14 - Definición de un token con más de una palabra.

Para dar una apropiada síntesis del habla, con propiedad y exactitud, se utiliza el atributo "`xml:lang`"dentro de la etiqueta "`token`", para definir el idioma, en el caso del español no es muy eficaz, pero por ejemplo en el inglés es bastante eficaz ya que existen distintas pronunciaciones de una misma palabra entre Inglés Americano e Inglés Británico. Un ejemplo podría ser:

```
<token xml:lang="en-US">
      Color
</token>


<token xml:lang="en-GB">
      Color
</token>
```

Tabla 15 - Uso del atributo xml:lang de la etiqueta token.

2.3.2.2. REFERENCIAS DE REGLAS

Las gramáticas se componen por un conjunto de reglas, que son las que definen el alcance de la gramática. En este apartado se tratan las posibles referencias que se pueden hacer a esas reglas, su descripción y su formato tanto en ABNF y en XML.

Las sentencias de la forma ABNF que se pueden apreciar en la siguiente tabla [Tabla 16] deben ir implementadas en el lugar correspondiente, para la forma XML, debe ir dentro de la etiquetas "`<grammar></grammar>`".

Tipo de referencia	Forma ABNF	Forma XML
Referencia explícita de una regla local.	`$rulename`	`<rulerefuri="#rulename"/>`
Referencia explícita de una regla nombrada anteriormente en una gramática identificada por una URI.	`$<grammarURI#rulename>`	`<rulerefuri="grammarURI#rulename"/>`
Referencia implícita de una regla raíz de una gramática identificada por una URI	`$<grammarURI>`	`<rulerefuri="grammarURI" />`
Referencia explícita de una regla nombrada anteriormente en una gramática identificada por una URI con un tipo de medio.	`$<grammarURI#rulename>~<media-type>`	`<rulerefuri="grammarURI#rulename" type="media-type"/>`

Referencia implícita de una regla raíz de una gramática identificada por una URI con un tipo de medio	`$<grammarURI>~<media-type>`	`<rulerefuri="grammarURI" type="media-type"/>`
Referencia especial de definición de reglas.	`$NULL` `$VOID` `$GARBAGE`	`<ruleref special="NULL"/>` `<ruleref special="VOID"/>` `<ruleref special="GARBAGE"/>`

Tabla 16 - Diferencias en las referencias de las reglas en ABNF y XML [8].

2.3.2.3. REFERENCIAS LOCALES

Para realizar referencias a reglas que estén definidas en la misma gramática, se debe utilizar una referencia a regla de nivel local.

Para la forma ABNF y forma XML existen sintaxis diferentes para representar una referencia de regla de nivel local.

- **Para la forma ABNF**, la una referencia a regla de nivel local o simple siempre está precedida por un carácter "`$`".

- **Para la forma XML,** se utiliza el elemento "`<ruleref>`" que es una etiqueta vacía, con un atributo "`uri`"que sirve para especificar la regla a la que se referencia, en este caso es local y se debe completar con el símbolo "`#`" seguido del identificador que indica a nivel local la regla a la que hace referencia.

ABNF Grammar	XML Grammar
`$ Teléfonos`	`<rulerefuri="#teléfonos"/>`
`$ Países`	`<rulerefuri="#países"/>`

Tabla 17 - Referencias Locales en Gramáticas ABNF y XML.

2.3.2.4. REFERENCIAS EXTERNAS

Se conoce como referencia externa, aquella referencia de una regla que está fuera de la gramática, ésta gramática debe ser identificada con una URI para poder obtener la regla.

En caso de que no se indicara el nombre de la regla a la que se hace referencia se procedería a utilizar los elementos de la raíz de la gramática.

A las referencias externas se les puede indicar el tipo de medio, en el que se encuentra cada gramática. Para indicarlo en la forma XML se utiliza el atributo`type` dentro de la etiqueta de "`<ruleref>`". Para indicarlo en la forma ABNF se utiliza las llaves "`<...>`" a continuación se verán unos ejemplos.

- **Parala forma ABNF,** la referencia de una regla a nivel externo siempre está precedida por un carácter "`$`", también debe de ir entre ángulos "`<...>`". Existen 3 tipos de referencias distintas:

ABNF Grammar

Para realizar referencias a reglas específicas de una gramática externa:

```
 $            <http://gramaticadeejemplo.com/ciudades.gram#spain>
;Contendrá las ciudades de España.
 $  <http://gramaticadeejemplo.com/paises.gram#asia>    ;Contendrá
los países de Asia
```

Para realizar referencias implícitas reglas raíz de una gramática externa:

```
 $ <../ciudades.gram>
```

Para realizar referencias asociadas con los distintos tipos de medios:

```
$      <http://gramaticadeejemplo.com/ciudades.gram#canada>      ~
<application/srgs>
$ <../ paises.gram> ~ <application/srgs>
```

Tabla 18 - Referencias Externas en Gramáticas ABNF.

- **Parala forma XML,** se utiliza el elemento "`<ruleref>`" que es una etiqueta

vacía, con un atributo "uri"que sirve para especificar la regla a la que se referencia, en este caso es una referencia externa y se debe completar con el símbolo "#" seguido de la URI entera y el identificador que indica a nivel local la regla específica a la que hace referencia. Existen 3 tipos de referencias distintas:

XML Grammar

Para realizar referencias a reglas específicas de una gramática externa:

```
<rulerefuri="http://gramaticadeejemplo.com/ciudades.
#canada"/>grxml
<rulerefuri="http://gramaticadeejemplo.com/paises.#asia"/>grxml
```

Para realizar referencias implícitas reglas raíz de una gramática externa:

```
<rulerefuri="../ciudades. grxml "/>
```

Para realizar referencias asociadas con los distintos tipos de medios:

```
<Rulerefuri="http://gramaticadeejemplo.com/ciudades.Grxml       #
Canadá"
type = "application / xml SRGS +" />
<rulerefuri="../ciudades.grxml "type="application/srgs+xml"/>
```

Tabla 19 - Referencias Externas en Gramáticas XML.

2.3.2.5. REGLAS ESPECIALES

Existen 3 tipos de reglas especiales para su utilización en las distintas gramáticas que se puedan definan.

Para el formato **XML** necesitan ir introducidas por el atributo"special" dentro de la etiqueta "<ruleref>".

Para la forma **ABNF** sólo se utiliza el símbolo del dólar "$" y a continuación se escribe el tipo de regla especial.

- **NULL:**Define una regla que automáticamente hace coincidir con la gramática, con lo que el usuario dice, también hace coincidir con la gramática aunque el usuario no haya dicho ni una sola palabra.

Forma ABNF: "`$NULL`"

Forma XML: "`<ruleref special="NULL"/>`"

- **VOID:** Define una regla que no puede ser dicha.

Forma ABNF: "`$VOID`"

Forma XML: "`<ruleref special="VOID"/>`"

- **GARBAGE:**Define una regla que hace coincidir todo lo que el usuario habla, hasta que la siguiente regla coincida, el siguiente token o hasta el final del discurso del hablante.

Forma ABNF: "`$GARBAGE`"

Forma XML: "`<ruleref special="GARBAGE"/>`"

2.3.3. ALTERNATIVAS

Una alternativa es un conjunto de opciones de las cuales el usuario puede escoger (definidas dentro de la gramática).

En la forma ABNF, las alternativas están separadas por barras verticales "`|`" y opcionalmente pueden estar contenidas en paréntesis:

En la forma XML, las alternativas están identificadas dentro de la etiqueta "`<one-of>`" contenidas en una etiqueta "`<ítem></ítem>`", opcionalmente dentro de la etiqueta "`<ítem>`"puede ir reflejado el ancho con el parámetro "`weight`", en

la siguiente tabla se puede ver un par de ejemplos.

ABNF Grammar	XML Grammar
Mostoles|Alcorcon|Leganes|Fuenlabrada|$Otras_ciudades_de_Madrid (uva|platano|piña|melocotón) /23/Ford|/34/Chovrolet|/4/Ferrari	<grammar> <one-of> <ítem>Mostoles<ítem> <ítem>Alcorcon<ítem> <ítem>Leganes<ítem> <ítem>Fuenlabrada<ítem> <rulerefuri='#Otras_ciudad_de_Madrid"/></ítem> </one-of> </grammar> <grammar> <one-of> <item weight="23">Ford<ítem> <item weight="34">>Chevrolet<ítem> <item weight="4">>Ferrari<item> </one-of> </grammar>

Tabla 20 - Ejemplo de Alternativas para Gramáticas ABNF y XML.

2.3.4. REPETICIONES

Para la confección de reglas es necesario el uso de operadores, es un avance ya que ofrecen gran abanico de posibilidades de repeticiones, tanto para la forma ABNF como para la forma XML son totalmente opcionales, por defecto se contabiliza como sólo 1 repetición del elemento.

En la siguiente tabla se pueden ver, los distintos tipos de repeticiones con los distintos valores que puede tomar:

Forma ABNF	Forma XML	Comportamiento
<n> <6>	repeat="n" repeat="6"	Esta expresión se repite exactamente "n" veces. "n" debe ser "0"o un entero positivo.

<m-n> <4-6>	repeat="m-n" repeat="4-6"	Esta expresión se repite entre "m" y "n" veces (ambos inclusive). "m" y "n" tiene que ser "o" o un entero positivo y "m" tiene que ser más pequeño o igual a "n".
<m-> <3->	repeat="m-" repeat="3-"	Esta expresión se repite "m" o más veces. "m" debe ser "0" o un entero positivo. Por ejemplo, "3-" indica que la expresión en la que está contenida puede repetirse tres, cuatro, cinco o más veces.
<0-1> [...]	repeat="0-1"	Esta expresión indica que es opcional.

Tabla 21 - Ejemplo Repeticiones para Gramáticas ABNF y XML[8].

A continuación se aprecian varios ejemplos para las distintas formas de representar gramáticas.

- **Para la forma ABNF:**

```
[very]
very<0-1>  // El token "very" es opcional
$digit<0-> // $digit puede repetirse cero, una o más veces
$digit<1-> // $digit puede una o más veces
$digit<4-6> // $digit puede repetirse cuatro, cinco o seis veces
$digit<10-> // $digit puede repetirse diez o más veces

// Ejemplo completo que abarca las siguientes expresiones
//   "pizza"
//   "big pizza with pepperoni"
//   "very big pizza with cheese and pepperoni"
[[very] big] pizza ([with | and] $topping) <0->
```

Tabla 22 - Ejemplo de Repetición en Gramática ABNF.

- **Para la forma XML:** Dentro de la etiqueta "`<item>`" existe un atributo para poder indicar las posibles repeticiones u ocurrencias que puede tener ese elemento, se llama "`repeat`", se verán en los siguientes ejemplos.

```
<item repeat="0-1">very</item> // El token "very" esopcional

<item repeat="0-"><rulerefuri="#digit"/></item>
// $digit puede repetirse cero, una o más veces

<item repeat="1-"><rulerefuri="#digit"/></item>
// $digit puede una o más veces

<item repeat="4-6"><rulerefuri="#digit"/></item>
// $digit puede repetirse cuatro, cinco o seis veces

<item repeat="10-"><rulerefuri="#digit"/></item>
// $digit puede repetirse diez o más veces

// Ejemplo completo que abarca las siguientes expresiones
//   "pizza"
//   "big pizza with pepperoni"
//   "very big pizza with cheese and pepperoni"

<item repeat="0-1">
<item repeat="0-1"> very </item>
big
</item>
pizza
<item repeat="0-">
<item repeat="0-1">
<one-of>
<item>with</item>
<item>and</item>
</one-of>
</item>
<rulerefuri="#topping"/>
</item>
```

Tabla 23 - Ejemplo de Repetición en Gramática XML[8].

2.3.5. PRECEDENCIA

Para la forma ABNF conviene definir un orden para realización de sus reglas, hay que cumplir una serie de normas para la correcta construcción de la regla.

Los paréntesis deben ser usados para explicitar la estructura de la regla.Uso de paréntesis "(" ")"para agrupar y corchetes "[" "]" para realizar una agrupación opcional de mayor nivel.

Los operadores unarios (+, -, *) y los adjuntos de las etiquetas tienen que mostrarse inmediatamente precediendo a la expansión de la regla.

Conjuntos de las alternativas de una regla, separadas por barras verticales "|".

2.3.6. DEFINICIÓN DE REGLAS

La definición de reglas tiene que comprobar el alcance y propósito de la misma, si es local a la gramática donde se define o intervienen otras gramáticas externas a la gramática en la que ha sido definida. El nombre de cada regla debe ser único para cada gramática, por lo tanto puede haber gramáticas diferentes, en el que coincidan algunas de las reglas.

2.3.6.1. DEFINICIÓN BÁSICA DE REGLAS

Una regla debe contener el nombre de su regla que es único, también hay que saber de qué tipo es la regla con su esquema XML. Podría ser de tipo ABNF también. Existe sensibilidad entre letras mayúsculas y minúsculas para la construcción de reglas.

- **Para laforma ABNF:** La definición de reglas consisten en una declaración del alcance opcional ("`public`", "`private`"), seguido del nombre de la regla, un signo de igual, y la declaración de la regla y el símbolo "`;`".
- **Para la forma XML:** La definición de una regla se representa como la etiqueta "`<rule>`" con el atributo "`id`", que indicará obligatoriamente un nombre único para la regla. A partir de ese momento se pueden utilizar las distintas etiquetas para formar reglas, como "`<one-of>`", "`<ítem>`"para moldear la regla a nuestro antojo.

ABNF Grammar	XML Grammar
`$NombreRegla = DefinicionRegla;` `public $ NombreRegla = DefinicionRegla;` `private $ NombreRegla = DefinicionRegla;` `Por ejemplo:` `$ciudad = Paris \| "New York" \| Bruselas;` `$comando = $accion $objeto;`	`<rule id="ciudad">` `<one-of>` `<item>Paris</item>` `<item>"New York"</item>` `<item>Bruselas</item>` `</one-of>` `</rule>` `<rule id="comando">` `<rulerefuri="#accion"/>` `<rulerefuri="#objeto"/>` `</rule>`

Tabla 24 - Ejemplo de Reglas Básicas de una Gramática.

2.3.7. DOCUMENTACIÓN DE LA GRAMÁTICA

Una gramática es un conjunto de reglas entrelazadas, pero también tiene asociada un conjunto de cabeceras y datos para poder comprender toda la semántica de la misma. Algunas de ellas son opcionales pero otras son obligatorias para definir cualquier gramática.

2.3.7.1. CABECERAS DE UNA GRAMÁTICA

Estas cabeceras facilitan información correspondiente a la semántica de la propia gramática, en la siguiente tabla se puede apreciar algunos de los atributos obligatorias y algunos de las opcionales más importantes que existen.

Tipo de Información	Requerida / Opcional	Forma ABNF	Forma XML
Versión de la Gramática	Requerida	ABNF self-identifyingheader	version attribute on grammar element
XML Namespace	Requerida (Sólo XML)	No es aplicable	xmlns attribute on grammar element
Tipo de Documento	Recomendada (Sólo XML)	No es aplicable	XML DOCTYPE
Codificación de los caracteres.	Recomendada	ABNF self-identifyingheader	encoding attribute in XML declaration
Idioma	Requerida en Modo Voz. Ignorada en modo DTMF	language declaration	xml:lang attribute on grammar element
Modo	Opcional	mode declaration	mode attribute on grammar element
Regla Raíz	Opcional	root declaration	root attribute on grammar element

Metadatos	Opcional. Permitidos	Varios	`meta and http-equivdeclarations`	`meta element`
Etiquetas	Opcional. Permitidos	Varios	`tag declaration`	`tag element`

Tabla 25 - Cabeceras de una gramática[8].

2.3.7.2. IDIOMA

Es un aspecto relevante para las aplicaciones que usen la voz como método de comunicación con el sistema, en caso de ser un modo DTMF (marcación de dígitos del teléfono), no es aplicable este parámetro.

- **La forma ABNF:** El encabezado ABNF debe contener cero o una declaración de idioma. Consiste en la palabra reservada "`language`", espacio en blanco, un identificador de idioma, opcional espacio en blanco y un carácter de punto y coma final "`;`".

- **Para la Forma XML:** El identificado de idioma se indica con atributo "`xml:lang`" en la etiqueta "`<grammar>`".

ABNF Grammar	XML Grammar
`language en-US;` `languagefr;`	`<grammar` `xmlns="http://www.w3.org/2001/06/grammar"` `xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"` `xsi:schemaLocation="http://www.w3.org/2001/06/grammar` `http://www.w3.org/TR/speech-grammar/grammar.xsd"` **`xml:lang="en-US"`** `version="1.0">` `<grammar` `xmlns="http://www.w3.org/2001/06/grammar"` `xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"` `xsi:schemaLocation="http://www.w3.org/2001/06/grammar` `http://www.w3.org/TR/speech-grammar/grammar.xsd"` **`xml:lang="fr"`** `version="1.0">`

Tabla 26 - Ejemplo Selección Idioma en Gramáticas.

2.3.7.3. MODO DE LA GRAMÁTICA

Existen 2 modos de gramática, el modo voz, que es para una gramática que se va activando con la interacciona vocal de un usuario y el modo DTMF, que se realiza para que el usuario interaccione con la aplicación con el teclado de un teléfono.

- **Para la forma ABNF:** Para indicar el modo en el que está la gramática, únicamente se usa la palabra reservada "`mode`", un espacio en blanco y a continuación una de las dos palabras reservadas que corresponden con el tipo de modos que existen: "`voice`" y "`dtmf`" y terminando con un símbolo de punto y coma "`;`". Si no se declara ninguna, por defecto se tomará el modo de "`voice`".

- **La forma XML:** La declaración del modo va impuesta en un atributo de la etiqueta `<grammar>` al comienzo de la misma, la palabra reservada para ese atributo es "`mode`" y existen 2 posibles valores para ese atributo "`voice`" o "`dtmf`", en el caso de que se omita este atributo se tomará por defecto el valor "`voice`" para la gramática.

ABNF Grammar	XML Grammar
`Modevoice;` `Modedtmf;`	`<grammar mode="voice"` `version="1.0"` `xml:lang="en-US"` `xmlns="http://www.w3.org/2001/06/grammar"` `xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"` `xsi:schemaLocation="http://www.w3.org/2001/06/grammar` `http://www.w3.org/TR/speech-grammar/grammar.xsd">` `<grammar mode="dtmf"` `version="1.0"` `xmlns="http://www.w3.org/2001/06/grammar"` `xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"` `xsi:schemaLocation="http://www.w3.org/2001/06/grammar` `http://www.w3.org/TR/speech-grammar/grammar.xsd">`

Tabla 27 - Ejemplo Selección Modo en Gramáticas.

2.3.7.4. DEFINICIÓN DE LA REGLA RAÍZ

Opcionalmente dentro de la gramática se puede definir una regla raíz, que es a la que se acudirá por defecto en el caso de que haya una llamada a una regla y no se especifique a cual de ellas.

- **Para la forma ABNF:** Es opcional y se puede implementar mediante la palabra reservada "`root`", un espacio en blanco, el símbolo del dólar "`$`", el nombre de la regla que se quiere que sea raíz y el símbolo de punto y coma "`;`".

- **Para la forma XML:** Dentro de la etiqueta "`<grammar>`"se puede declarar la regla raíz perteneciente a esa gramática, es opcional, en el caso que se use se utilizará el atributo "`root`" seguido de un símbolo de igualdad "`=`" y el nombre de la regla entrecomillado "`nombreregla`".

ABNF Grammar	XML Grammar
`root $rulename;`	`<grammarroot="rulename" ...>`

Tabla 28 - Ejemplo Selección Regla Raíz en Gramáticas.

2.4. SÍNTESIS DEL HABLA

SSML - Speech Synthesis Markup Languaje, acrónimo de Lenguaje para la Síntesis del Habla, este lenguaje está basado en el Lenguaje Java del Habla (JSML). Este lenguaje es la vía para los desarrolladores para controlar todos los elementos de la síntesis del habla, como el tono de voz, el ritmo, la pronunciación y volumen.

Para el desarrollo de este proyecto esta tecnología no es imprescindible, pero si sirve para dar un enfoque global de la potencia de la tecnología VoiceXML.

2.4.1. INTRODUCCIÓN

El uso de SSML tiene como finalidad mejorar la calidad del hablasintetizándolo. Existen diferentes elementos que tienen impacto, en las distintas etapas del proceso de síntesis.

2.4.1.1. ETAPAS DEL PROCESO DE SÍNTESIS DEL HABLA

Los siguientes seis etapas son las principales para el procesamiento del habla, estas etapas son realizadas por un procesador de síntesis del habla, que tendrá el objetivo de convertir el texto en voz.

La creación de un documento en formato SSML, sirve como entrada a un procesador de síntesis puede ser automática, llevada a cabo por un usuario humano o una combinación de ambas.

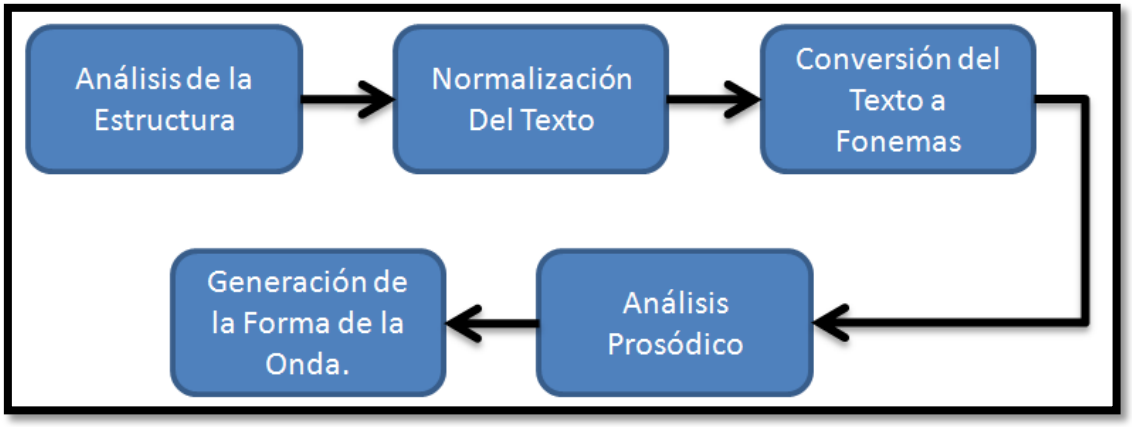

Ilustración 6 - Esquema de las etapas del Proceso de Síntesis del Habla [9].

2.4.1.1.1. ANÁLISIS GRAMATICAL DEL DOCUMENTO XML

El análisis se utiliza para extraer de la estructura del documento XML los posibles elementos como son: las etiquetas y atributos. Un buen análisis influye en cada uno de los pasos sucesivos.

2.4.1.1.2. ANÁLISIS DE LA ESTRUCTURA

La estructura de un documento, afecta a la forma en la queéste es leído. Existen patrones comunes asociadas con los párrafos y oraciones.

2.4.1.1.3. NORMALIZACIÓN DEL TEXTO

En todos los lenguajes existen construcciones especiales que necesitan una conversión especial de la forma escrita a la forma hablada (p.e 2/3 o 500 €).

Esta conversión es realizada automáticamente por el procesador de síntesis. La etiqueta "`<say-as>`" puede ser utilizada en el documento de entrada para indicar explícitamente la presencia de este tipo de construcciones.

De esta forma se produce la desambiguación de términos como 2/3 que puede tener múltiples significados como: (dos de tres, dos tercios, 2 de Marzo, etc.).

2.4.1.1.4. CONVERSIÓN DEL TEXTO A FONEMAS

Una vez que el procesador de síntesis ha determinado el conjunto de palabras que han de ser pronunciadas, se debe deducir la pronunciación de dichas palabras. La pronunciación de las palabras debe ser convenientemente descrita como secuencias de fonemas, que son unidades mínimas de sonido en un lenguaje, los fonemas sirven para distinguir una palabra de otra.

Existen lenguajes, que como el inglés, en el que existe ambigüedad en la conversión del texto en voz (p.e. "read" y "reed" se pronuncian igual).

2.4.1.1.5. ANÁLISIS PROSÓDICO

La prosodia se define como el conjunto de rasgos del habla que incluye el tono, el ritmo, las pausas, la velocidad y el énfasis. Conseguir una prosodia humana lo más cercana a la realidad es importante para lograr una voz natural y comprensible.

Para conseguir estos rasgos explícitamente SSML proporciona las etiquetas "`<emphasis>`", "`<break>`" y "`<prosody>`".

2.4.1.1.6. GENERACIÓN DE LA FORMA DE LA ONDA

Los fonemas y la información prosódica son utilizados por el procesador de síntesis en la producción de la formar de la onda correspondiente. SSML proporciona la etiqueta "`<voice>`" para solicitar una voz específica con unas determinadas cualidades (p.e. La voz de una mujer).

2.4.2. TERMINOLOGÍA

- **Síntesis del habla:** Es el proceso de generación automática de una salida de voz a partir de una entrada de datos que pueden incluir texto, texto marcado u objetos binarios.
- **Procesado de la síntesis:** Un sistema de paso de texto a voz, que acepta documentos SSML como entrada y los hace voz para su salida.
- **Sistema de Texto a Voz:** Proceso automático con salida en forma de voz, pero como entrada texto o texto marcado.

2.4.3. FORMATO DEL DOCUMENTO SSML

En este apartado se ve un pequeño resumen de los elementos básicos del formato de un documento de SSML.

2.4.3.1. CABECERAS DE UN DOCUMENTO SSML

Un documento SSML debe llevar una cabecera XML, la línea "`DOCTYPE`"en la que se indica el DTD debería ser similar a la siguiente:

```
<!DOCTYPE speak PUBLIC "-//W3C//DTD SYNTHESIS 1.0//EN"
  "http://www.w3.org/TR/speech-synthesis/synthesis.dtd">
```

Tabla 29 - Ejemplo DTD para SSML.

Después de la línea del "`DOCTYPE`", habrá que indicar la versión del documento XML, A continuación debe ir localizada la etiqueta "`<speak>`"en la cual se deben poner los atributos de versión, el espacio de nombres, la localización del esquema e inclusive añadir el idioma.

En la siguiente tabla se pueden ver 2 ejemplos de cabeceras de documentos SSML:

```
<?xml version="1.0"?>
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.w3.org/2001/10/synthesis
                    http://www.w3.org/TR/speech-
synthesis/synthesis.xsd"
xml:lang="en-US">




<?xml version="1.0"?>
<!DOCTYPE speak PUBLIC "-//W3C//DTD SYNTHESIS 1.0//EN"
                "http://www.w3.org/TR/speech-
synthesis/synthesis.dtd">
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis"
xml:lang="en-US">
```

Tabla 30 - Ejemplo cabeceras SSML.

2.4.3.2. ELEMENTOS Y ATRIBUTOS

A continuación se definen los elementos más comunes para el lenguaje SSML, se pueden ordenar en elementos de estructura, elementos de procesamiento de texto y elementos pronunciación.

2.4.3.2.1. <SPEAK> ELEMENTO RAÍZ

La etiqueta "`<speak>`"es el elemento raíz del documento, la especificación del lenguaje es obligatorio mediante el atributo "`xml:lang`", otro atributo obligatorio es la versión del documento que se denota por la palabra reservada "`version`", como etiqueta opcional se podrá especificar "`xml:base`", que indicará la URI base, del documento raíz.

```
<?xml version="1.0"?>
<speak version="1.0"
xmlns=http://www.w3.org/2001/10/synthesis
xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
     xsi:schemaLocation="http://www.w3.org/TR/speech-
synthesis/synthesis.xsd"
xml:lang="en-US">
... cuerpo de la aplicación ...
</speak>
```

Tabla 31 - Ejemplo cabecera <speak>.

El atributo "xml:lang", es el encargado de determinar el idioma de la entrada de texto y salida de voz.

```
<speak xml:lang="en-US">
      <paragraph> I don't speak Spanish. </paragraph>
<paragraph xml:lang="es"> Yo no hablo Español </paragraph>
</speak>
```

Tabla 32 - Ejemplo atributo xml:lang<speak>.

2.4.3.2.2. ESTRUCTURA DEL TEXTO: <P> Y <S>

La etiqueta "<p>"representa un párrafo y la etiqueta "<s>" representa una frase u oración, estas dos etiquetas tiene un atributo opcional que es el xml:lang, para indicar el idioma en el que deben ser expresados los párrafos o frases correspondientes.

```
<?xml version="1.0"?>
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.w3.org/2001/10/synthesis
 http://www.w3.org/TR/speech-synthesis/synthesis.xsd"
xml:lang="en-US">
<p>
<s> Esta es la primera sentencia del párrafo.</s>
<s> Aquí irá otra sentencia.</s>
</p>
</speak>
```

Tabla 33 - Ejemplo etiquetas <p> y <s>.

El uso de las etiquetas "<p>" y "<s>" dentro del elemento raíz "<speak>" es opcional, su uso ayudará al procesador de síntesis a comprender de mejor manera el texto escrito, y no lo tratará como texto plano.

2.4.3.2.3. <SAY-AS>

La etiqueta "<say-as>" permite al autor indicar información sobre el tipo de texto que hay dentro de la etiqueta.Esta información es usada para ayudar a aclarar la ambigüedad de pronunciación de algunas palabras de nuestro documento.

El atributo "type" de la etiqueta "<say-as>" es obligatorio de completar cada vez que se usa dicha etiqueta, este atributo indica el tipo de contenido que

tiene el texto que está dentro de la etiqueta. Algunos de los valores que puede tomar este atributo son: "spell-out"(contiene el texto pronunciado letra a letra – deletrear), "acronym" (lee la palabra como un acrónimo), "number", "date", "time, "duration", "currency", "telephone", "name", "net" (Identificador de Internet) y "address" (Indica la dirección postal).

```
<say-as type="spell-out">  USA </say-as>
<!--Para este ejemplo se utilizaría el deletreo, U. S. A.-->

La Jungla de Cristal <say-as type="number"> IV </say-as>
<!--Para este ejemplo el resultado sería "La Jungla de Cristal
cuatro"-->


Tengo en mi cuenta <say-as type="currency"> $250,30 </say-as>
<!--Para este ejemplo el resultado sería "Tengo en mi cuenta
doscientos cincuenta dólares con 30 centavos"-->
```

Tabla 34 - Ejemplo etiqueta <say as>.

2.4.3.2.4. <VOICE>

La etiqueta <voice>"tendrá como objetivo modificar la información del orador (sistema), que produce la salida por el altavoz. Algunos de sus atributos opcionales son:

- **Xml:lang:** es opcional e indica el idioma.
- **Gender:**genero de la voz del orador que dirá lo contenido en el texto entre las etiquetas "<voice>" algunos de los valores que puede tomar son: "male" (voz masculina), "female", (voz femenina) y "neutral".
- **Age:** Edad de la voz del orador, los valores que puede tomar son enteros positivos.
- **Category:**Categorías de edad predefinidas para la voz del orador, los valores que puede tomar son: "child"(niño), "teenager" (adolescente), "adult" (adulto) y "elderly" (persona mayor).
- **Variant:**Variantes predefinidas de las voces, toma valores enteros positivos.

En caso de no indicar alguno de los anteriores atributos, el sistema utilizará los valores por defecto.

```
<?xml version="1.0"?>
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.w3.org/2001/10/synthesis
http://www.w3.org/TR/speech-synthesis/synthesis.xsd"
xml:lang="en-US">

<voice gender="female">María vive en la Calle Madrid</voice>
<!—Ahora se utilizará otro tipo de voz femenina -->
<voice gender="female" variant="2">
    Pero, Antonio vive en la Calle Getafe.
</voice>
</speak>


<?xml version="1.0"?>
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.w3.org/2001/10/synthesis
http://www.w3.org/TR/speech-synthesis/synthesis.xsd"
xml:lang="en-US">


<voice gender="female"> Una voz de mujer </voice>
<voice age="6"> Una voz de niña aquí </voice>
<p xml:lang="ja"><!-- Una voz de niña japonesa --></p>
</speak>
```

Tabla 35 - Ejemplos etiqueta <voice>.

2.4.3.2.5. <EMPHASIS>

La etiqueta "`<emphasis>`"indica que el texto que está dentro de la citada etiqueta tiene que ser hablado con un cierto énfasis (prominencia o estrés), el atributo "`level`", será el que tenga el valor del énfasis.

Los valores que puede tomar el atributo "`level`" pueden ser: "`none`" (no le dará énfasis), "`x-weak`" (poco énfasis), "`weak`" (énfasis moderado), "`medium`" (énfasis por defecto), "`strong`" (énfasis fuerte), o "`x-strong`" (énfasis muy fuerte). Ejemplo:

```
<voice>
Hoy tengo un <emphasis> buen </emphasis> día.
Estoy <emphasis level="strong"> muy </emphasis> contento
</voice>
```

Tabla 36 - Ejemplo etiqueta <emphasis>.

Capítulo 3:

Análisis

CAPÍTULO 3: ANÁLISIS

En este capítulo se detalla en profundidad la fase de análisis, en la que se especificará detalladamente los objetivos y alcance de la aplicación, así como sus restricciones.

Se explica la fase de análisis con los Casos de Uso [12] de UML, y con una adaptación de la metodología sobre Requisitos Funcionales y No Funcionales [11].

3.1. CASOS DE USO

Para conocer mejor la información sobre los distintos tipos de casos de uso, se ha creado una tabla de valores con los siguientes atributos a rellenar:

- **Identificador:** clave que identifica el caso de uso de manera única.
- **Nombre:** nombre único del caso de uso.
- **Actores:** actores que participan en el caso de uso.
- **Descripción:** explicación detallada del caso de uso.
- **Pre-condiciones:** para utilizar este caso de uso se deben cumplir algunas acciones previamente.
- **Post-condiciones:** la salida que realiza el caso de uso.

3.1.1. DIAGRAMA DE CASOS DE USO

Para representar los Casos de Uso de este sistema, se ha decidido usar la representación de la que dispone el lenguaje UML[12]

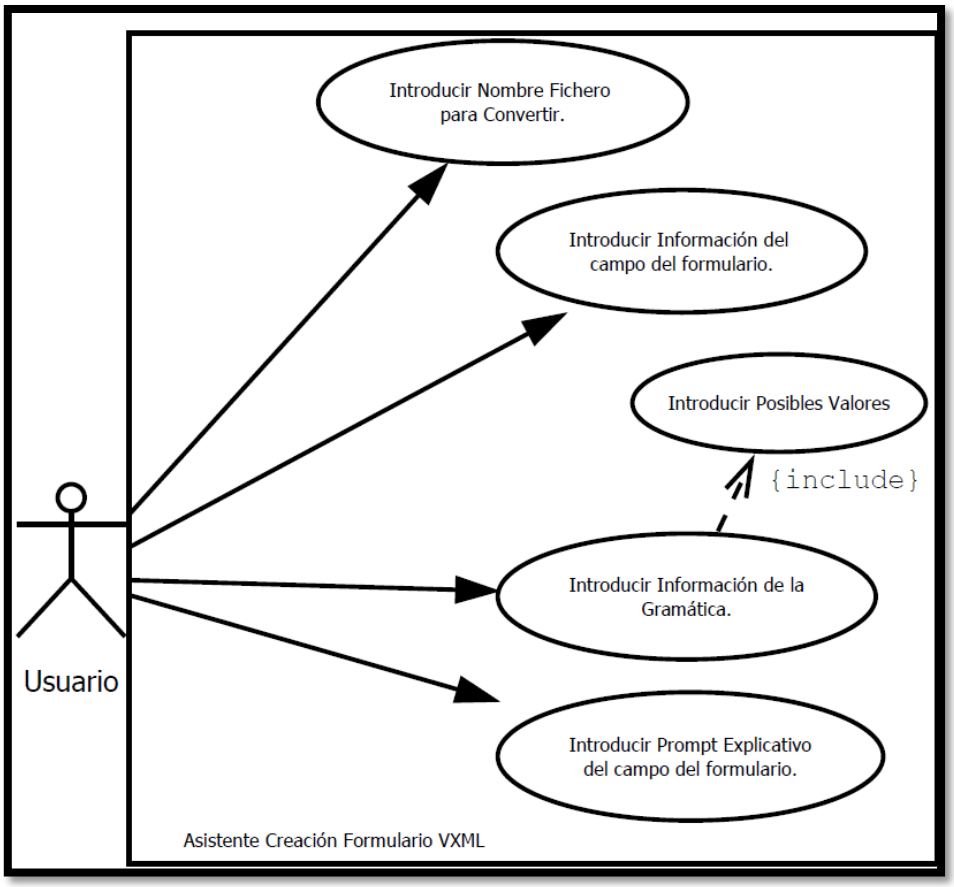

Ilustración 7 - Diagrama de Casos de Uso.

3.1.2. DESCRIPCIÓN DE CASOS DE USO

A continuación se definen los casos de uso que se han propuesto para este sistema.

ID:	CU-01
Nombre:	Introducir Nombre Formulario a Convertir
Actores:	Usuario
Descripción:	
El usuario introduce por teclado el nombre del formulario que quiere convertir en VoiceXML.	
Pre-condiciones:	El usuario tiene que haber iniciado la aplicación.
Post-condiciones	El sistema conocerá el nombre del formulario que se quiere convertir.
Escenario:	1. El Usuario inicia la aplicación. 2. El Sistema solicita al usuario que introduzca el nombre del fichero con la extensión. a. El usuario introduce bien el nombre del fichero. b. El usuario introduce mal el nombre del fichero. i. El sistema informa del error. ii. El sistema vuelve a solicitar al usuario (pto. 2.). 3. El sistema finaliza imprimiendo un mensaje por pantalla.

Tabla 37 - Caso de uso 1 - Introducir Nombre Formulario a Convertir.

ID:	CU-02
Nombre:	Introducir Información del Campo del Formulario.
Actores:	Usuario
Descripción:	
El usuario introduce mediante teclado la información del campo del formulario que se trate, cuando se le solicite la información al usuario, es porque el sistema no la ha podido recuperar del formulario de entrada.	
Pre-condiciones:	- El Usuario inicia la aplicación. - El Sistema solicita al usuario que introduzca el nombre del fichero con la extensión.
Post-condiciones	- Se conocerán todos los datos del campo del formulario.
Escenario:	1. El usuario introduce bien el nombre del fichero. 2. El Sistema lee e interpreta el formulario. a. Si a algún "`input`" le falta el atributo "`id`". i. El sistema informa de que falta ese "`id`" y se lo solicita al usuario. ii. El usuario introduce un "`id`" identificativo del campo. 3. El sistema finaliza imprimiendo un mensaje por pantalla.

Tabla 38 - Caso de Uso 2 - Introducir Información del Campo del Formulario.

ID:	CU-03
Nombre:	Introducir "`Prompt`" Explicativo para del campo del formulario.
Actores:	Usuario.
Descripción:	
El usuario puede introducir o no un "`prompt`" explicativo que se reproducirá antes del campo indicado.	
Pre-condiciones:	- El Usuario inicia la aplicación. - El Sistema solicita al usuario que introduzca el nombre del fichero con la extensión. - El usuario introduce bien el nombre del fichero. - El Sistema lee e interpreta el formulario.
Post-condiciones	- "`Prompt`" introducido para el campo indicado.
Escenario:	1. El sistema para cada input solicita si se quiere o no mostrar un "`prompt`". i. Si se solicita un "`prompt`", se escribe el mismo. ii. Si se indica que no se quiere input el programa continúa. 2. El sistema finaliza imprimiendo un mensaje por pantalla.

Tabla 39 - Caso de Uso 3 - Introducir "Prompt" Explicativo para del campo del formulario.

ID:	CU-04
Nombre:	Introducir Información de la Gramática.
Actores:	Usuario.
Descripción:	
El usuario introduce información sobre la gramática que se asociará a un campo determinado del formulario.	
Pre-condiciones:	- El Usuario inicia la aplicación. - El Sistema solicita al usuario que introduzca el nombre del fichero con la extensión. - El usuario introduce bien el nombre del fichero. - El Sistema lee e interpreta el formulario.
Post-condiciones	Información sobre la gramática introducida.
Escenario:	1. Si a alguna gramática le falta el atributo "`id`". i. El sistema informa de que falta ese "`id`" y se lo solicita al usuario. ii. El usuario introduce un "`id`" identificativo de la gramática. 2. El sistema finaliza imprimiendo un mensaje por pantalla.

Tabla 40 - Caso de Uso 4 - Introducir Información de la Gramática.

ID:	CU-05
Nombre:	Introducir Posibles Valores.
Actores:	Usuario.
Descripción:	
El usuario introduce los valores que puede reconocer la gramática.	
Pre-condiciones:	- El Usuario inicia la aplicación. - El Sistema solicita al usuario que introduzca el nombre del fichero con la extensión. - El usuario introduce bien el nombre del fichero. - El Sistema lee e interpreta el formulario.
Post-condiciones	- Valores introducidos en la gramática.
Escenario:	1. Si el "`input`" que se lee es un "`select`". i. El sistema crea la gramática con los valores de los que dispone el "`select`". 2. Si el input que se lee es un input de tipo "`text`" o "`textarea`". i. El sistema solicita al usuario los posibles valores que se pueden introducir. ii. El usuario introduce los valores correspondientes. 3. El sistema crea la gramática con los valores asociados a ese input.

Tabla 41 - Caso de Uso 5 - Introducir Posibles Valores.

3.2. ESPECIFICACIÓN DE REQUISITOS

En este apartado se define el catálogo de requisitos funcionales y no funcionales del sistema que se tiene que desarrollar.

Para conocer mejor la información sobre los distintos tipos de requisitos, se ha creado una tabla de valores con los siguientes atributos a rellenar:

- **Identificador:** clave que identifica el requisito de manera única.
- **Nombre:** nombre único del requisito.
- **Descripción:** explicación detallada del requisito.
- **Verificabilidad:** indicar si es verificable en la sección de pruebas.
- **Prioridad:** necesidad de la implementación del requisito.
- **Impacto:** indicar si tiene un gran impacto en el sistema general.

3.2.1. REQUISITOS FUNCIONALES

A continuación se definen los requisitos funcionales que se han propuesto para este sistema.

ID:	RF-01
Nombre:	Elección Formulario.
Descripción:	
El sistema debe ser capaz de que, el usuario pueda introducir el nombredel formulario HTML que quiere convertir en VoiceXML.	
Verificable:	Si
Prioridad:	Alta
Impacto:	Si

Tabla 42 - Requisito Funcional: RF01 – Elección Formulario.

ID:	RF-02
Nombre:	Solicitud de "Prompt".
Descripción:	
El sistema después de analizar el formulario de entrada, preguntará al usuario, si quiere poner "`prompts`" explicativos para cada uno de los inputs.	
Verificable:	Si
Prioridad:	Alta
Impacto:	Si

Tabla 43 - Requisito Funcional: RF02 – Solicitud de Prompt.

ID:	RF-03
Nombre:	Solicitud Valores Inputs.
Descripción:	
Después del análisis del formulario de entrada, el sistema preguntará al usuario, sobre los valores que pueden tomar los inputs (sólo para los input de tipo "`text`" o "`textarea`").	
Verificable:	Si
Prioridad:	Alta
Impacto:	Si

Tabla 44 - Requisito Funcional: RF03 – Solicitud Valores Inputs.

ID:	RF-04
Nombre:	Información del input.
Descripción:	
La aplicación debe ser capaz de acceder a la información, ("`tipo`", "`name`" e "`id`")de cada "`input`" del formulario de entrada, para poder utilizar esa información en el formulario VoiceXML y evitar preguntársela al usuario.	
Verificable:	Si
Prioridad:	Alta
Impacto:	Si

Tabla 45 - Requisito Funcional: RF04 – Información del input.

ID:	RF-05
Nombre:	Valores del input del tipo “select”.
Descripción:	
El sistema deberá obtener toda información sobre los posibles valores que puede tomar el “`input`”del tipo “`select`”, con estos valores se construirá la gramática de palabras reconocedoras, el “`input`” del tipo “`select`” tiene que estar existente en el formulario de entrada.	
Verificable:	Si
Prioridad:	Medio-Alto
Impacto:	Si

Tabla 46 - Requisito Funcional: RF05 – Valores del input del tipo Select.

ID:	RF-06
Nombre:	Generación de Gramáticas.
Descripción:	
La aplicación después de analizar los inputs del formulario de entrada, deberá generar una gramática para cada tipo de “`input`” que se haya encontrado en el mismo. Estas gramáticas, deben contener los valores que se han obtenido del análisis de los “`inputs`” del formulario o bien de la respuesta a las preguntas realizadas al usuario.	
Verificable:	Si
Prioridad:	Alto
Impacto:	Si

Tabla 47 - Requisito Funcional: RF06 – Generación de Gramáticas.

ID:	RF-07
Nombre:	Construcción Fichero VoiceXML.
Descripción:	
La aplicación después de analizar los inputs del formulario de entrada, después de recibir información del usuario sobre los "`inputs`" y después de generar una gramática para cada tipo de "`input`". Deberá crear la estructura VoiceXML acorde a la información y datos de los que dispone, esta estructura tiene que se acorde con el estándar para poder conformar un fichero VoiceXML válido.	
Verificable:	Si
Prioridad:	Alto
Impacto:	Si

Tabla 48 - Requisito Funcional: RF07 – Construcción Fichero VXML.

3.2.2. REQUISITOS NO FUNCIONALES

A continuación se definen los requisitos no funcionales que se han propuesto para este sistema.

ID:	RNF-01
Nombre:	Formulario de entrada debe ser validado
Descripción:	
El formulario que se recibe como entrada en el sistema, para convertirlo a formato VoiceXML, deberá cumplir con las especificaciones del estándar W3C, esto se realizará mediante el validador (http://validator.w3.org/).	
Verificable:	Si
Prioridad:	Media.
Impacto:	Alto

Tabla 49 - Requisito No Funcional: RNF01 - Formulario de entrada debe ser validado.

ID:	RNF-02
Nombre:	Los “inputs” deben tener toda la información básica.
Descripción:	
Los inputs del formulario deben tener al menos los atributos obligatorios de cada tipo de “`input`”, para la correcta conversión del formulario a la tecnología VoiceXML.	
Verificable:	Si
Prioridad:	Media – Alta.
Impacto:	Alto

Tabla 50 - Requisito No Funcional: RNF02 - Los Inputs deben tener toda la información básica.

ID:	RNF-03
Nombre:	El Formulario HTML puede contener varios inputs y de distintos tipos.
Descripción:	
El formulario HTML puede tener uno o varios "`inputs`", algunos "`inputs`" no serán soportados por esta aplicación, es decir no podrán ser convertidos a la tecnología VoiceXML, estos son: - Input – tipo: "`image`". - Input – tipo: "`hidden`". - Input – tipo: "`password`". - Input – tipo: "`radio`". - Input – tipo: "`checkbox`". Los inputs que si son soportados por la aplicación son: - Input – tipo: "`text`". - Input – tipo: "`textarea`". - Input – tipo: "`select`". - Input – tipo: "`submit`". - Input – tipo: "`reset`".	
Verificable:	Si
Prioridad:	Medio.
Impacto:	Medio

Tabla 51 - Requisito No Funcional: RNF03 - El Formulario HTML puede contener varios inputs y de distintos tipos.

ID:	RNF-04
Nombre:	La información de entrada deberá ser en inglés.
Descripción:	
La información que se le ofrezca al sistema tanto en el Formulario HTML, como en las respuestas que de el usuario, deberán ser en introducidas en la aplicación en lengua inglesa. Esto es debido a que para el mejor funcionamiento de la aplicación se tratará tanto los input como los "`prompt`" en inglés, que es el lenguaje por defecto.	
Verificable:	Si
Prioridad:	Medio.
Impacto:	Medio

Tabla 52 - Requisito No Funcional: RNF04 - La información de entrada deberá ser en inglés.

ID:	RNF-05
Nombre:	Requisitos software para ejecutar el Formulario VoiceXML.
Descripción:	
Para la ejecución del Formulario VoiceXML transformado con esta herramienta, se deberá disponer de la versión 9.2 del navegador Opera, con las opciones habilitadas de Control de Navegación por Voz.	
Verificable:	Si
Prioridad:	Alto.
Impacto:	Alto.

Tabla 53 - Requisito No Funcional: RNF05 - Requisitos software para ejecutar el Formulario VoiceXML.

ID:	RNF-06
Nombre:	Requisitos hardware para ejecutar el Formulario VoiceXML.
Descripción:	
Para la ejecución del Formulario VoiceXML transformado con esta herramienta, se deberá disponer de una tarjeta de sonido básica instalada en el Ordenador donde se vayan a ejecutar las pruebas. También hará falta un micrófono para poder introducir la información.	
Verificable:	Si
Prioridad:	Alto.
Impacto:	Alto.

Tabla 54 - Requisito No Funcional: RNF06 - Requisitos hardware para ejecutar el Formulario VoiceXML.

ID:	RNF-07
Nombre:	Número limitado de opciones.
Descripción:	
La aplicación no puede procesar un número alto de opciones, ya que dispone de un buffer de 1024 caracteres para almacenar todas ellas, para cada uno de los inputs,	
Verificable:	Si
Prioridad:	Medium
Impacto:	Si

Tabla 55 - Requisito No Funcional: RNF07 – Tamaño Buffer de Opciones.

3.3. PLAN DE PRUEBAS.

A continuación se define una batería de pruebas que abarcan los requisitos, que previamente se han definido, estas pruebas se tienen que realizar y terminar con éxito para poder aprobar el correcto funcionamiento de este Proyecto Fin de Carrera.

Para conocer mejor la información sobre las distintas pruebas, se ha creado una tabla con los siguientes campos:

- **Identificador:** clave que identifica la prueba de manera única.
- **Nombre:** nombre único de la prueba.
- **Descripción:** explicación detallada de la prueba.

3.3.1. CATÁLOGO DE PRUEBAS

ID:	PR-01
Nombre:	Elección correcta de Formulario.
Descripción:	
Se inicia la aplicación, y se elige un formulario correcto que está en el directorio de la aplicación. En caso de que haya sido correcto, el programa continuará, pero en el caso contrario, el programa seguirá preguntando de nuevo y mostrando un mensaje de que el nombre introducido no es correcto.	
Requisitos que comprueba:	RF-01.

Tabla 56 - Prueba 1 - Elección correcta de Formulario.

ID:	PR-02
Nombre:	Elección de “prompt” Explicativo.
Descripción:	
Se inicia la aplicación, y se elige un formulario correcto. Cada vez que se analice un “`input`”, el sistema pregunta si el usuario quiere poner un “`prompt`”, si es así tendrá que poner el texto que quiere como “`prompt`”. En caso contrario el programa continuará con otro “`input`”.	
Requisitos que comprueba:	RF-01 y RF-02.

Tabla 57 - Prueba 2 - Elección de Prompt Explicativo.

ID:	PR-03
Nombre:	Elección de Valores Posibles del“input” de tipo "text" y "textarea".
Descripción:	
Se inicia la aplicación, y se elige un formulario correcto. Cada vez que se analice un “`input`” del tipo "`text`" o "`textarea`", el sistema pregunta al usuario por los valores posibles que tiene que tener ese campo. Cuando son introducidos con el formato adecuado, es decir, cada palabra o “`token`” tiene que estar separado por una barra horizontal “`|`”,continúa el programa.	
Requisitos que comprueba:	RF-01, RF-02 y RF-03.

Tabla 58 - Prueba 3 - Elección de Valores Posibles de Input "text" y "textarea".

ID:	PR-04
Nombre:	Elección de Valores Posibles de Input "select".
Descripción:	
Se inicia la aplicación, y se elige un formulario correcto. Cada vez que se analice un "`input`" del tipo "`select`", el sistema automáticamente obtendrá los valores de ese campo. Cuando son introducidos con el formato adecuado continúa el programa.	
Requisitos que comprueba:	RF-01, RF-02 y RF-05.

Tabla 59 - Prueba 4 - Elección de Valores Posibles de Input "select".

ID:	PR-05
Nombre:	Elección de la Información Obligatoria del Input.
Descripción:	
Se inicia la aplicación, y se elige un formulario correcto. Este formulario tiene que estar creado con la carencia de algún atributo ("`id`" o "`name`") de la etiqueta "`input`". Cada vez que se analice un "`input`" con esta carencia, el sistema pregunta al usuario por el valor de ese atributo. Cuando son introducidos por el usuario el programa continúa.	
Requisitos que comprueba:	RF-01, RF-02 y RF-04.

Tabla 60 - Prueba 5 - Elección de la Información Obligatoria del Input.

ID:	PR-06
Nombre:	Prueba de la generación de la Gramática.
Descripción:	
Se inicia la aplicación, y se elige un formulario correcto. Para cada uno de los "`input`" se crea una gramática asociada, la que contiene los valores posibles que puede tomar. Tras introducir los datos en la prueba PR-03 o PR-04, en el fichero de salida de la aplicación, se puede ver el resultado de esta prueba, es decir, se tiene que apreciar la estructura de la gramática dentro de cada uno de los campos del formulario.	
Requisitos que comprueba:	RF-01, RF-06 y RF-07.

Tabla 61 - Prueba 6 - Prueba de la generación de la Gramática.

ID:	PR-07
Nombre:	Comprobación de la construcción final del Fichero VoiceXML.
Descripción:	
Se inicia la aplicación, y se elige un formulario correcto. Se realiza la PR-02, PR-03 y PR-04 y finaliza la aplicación. Al final de la ejecución se abre el fichero generado para comprobar que está bien formado.	
Requisitos que comprueba:	RF-01 y RF-07.

Tabla 62 - Prueba 7 - Comprobación de la construcción final del Fichero VoiceXML.

3.3.2. MATRIZ DE PRUEBAS-REQUISITOS

En la siguiente matriz, se puede ver de manera resumida, la correspondencia que hay entre las pruebas y los requisitos que se validan con esas pruebas.

		Pruebas						
		PR-01	PR-02	PR-03	PR-04	PR-05	PR-06	PR-07
Requisitos	RF-01	**X**	**X**	**X**	**X**	**X**	**X**	**X**
	RF-02		**X**	**X**	**X**	**X**		
	RF-03			**X**				
	RF-04					**X**		
	RF-05				**X**			
	RF-06						**X**	
	RF-07						**X**	**X**

Tabla 63 - Matriz de pruebas-requisitos

Capítulo 4:

Diseño del Sistema

CAPITULO 4: DISEÑO DEL SISTEMA.

En el siguiente capítulose abordan los fundamentos básicos, para realizar una satisfactoria fase de diseño a partir de los requisitos que se tomaron en la fase de análisis.

Para esta aplicación se ha decidido diseñar una arquitectura de cliente, es decir, que solo se interviene desde el lado del cliente, se debe aclarar que no se realiza ningún acceso a ningún servidor, tampoco de conexión de red o bases de datos para recuperar información.

El patrón utilizado durante el desarrollo de esta aplicación ha sido, la descomposición en pequeños módulos (clases java).

Para ilustrar algunos conceptos de este diseño, se utilizandistintos diagramas de UML[12].

4.1. MODELO CONCEPTUAL

En este apartado se encarga de mostrar, el modelo conceptual de la aplicación, las divisiones de los distintos módulos que se han decidido diseñar, para que la implementación sea más sencilla. Para ello se ha usado el Diagrama de Clases propuesto por UML, del cual se mostrará el diagrama y un breve comentario sobre cada una de las clases y sus relaciones con otras clases.*[Ilustración 8]*

4.1.1. DIAGRAMA DE CLASES

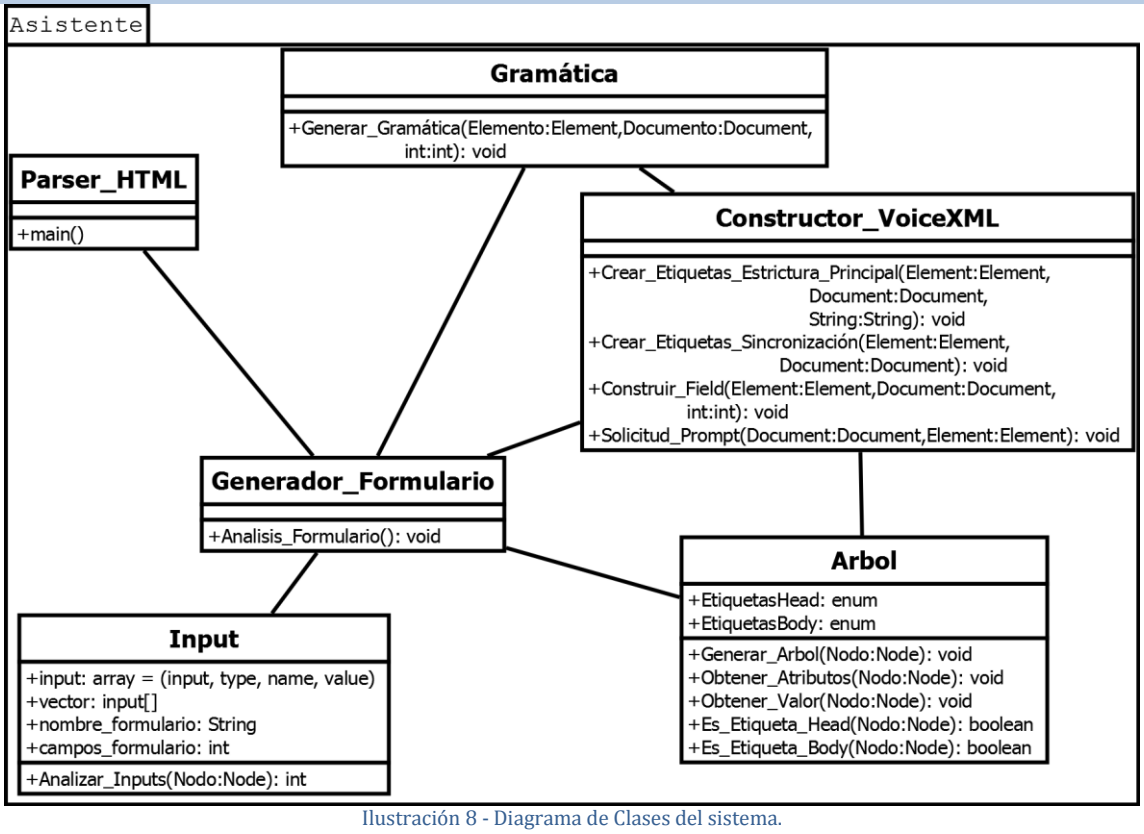

Ilustración 8 - Diagrama de Clases del sistema.

4.1.2. DESCRIPCIÓN DE LAS CLASES

En este apartado se realizará una descripción de cada una de las clases, en las que se ha decidido dividir el sistema. Todas las clases están dentro de un paquete llamado Asistente. Las clases son:

- **PARSER_HTML:** Será la clase generadora, que dispondrá del programa principal que accionará toda la aplicación.

  El procedimiento principal "`main`" llamará a un método llamado "`Análisis_Formulario()`", que está dispuesto en la clase Generador Formulario.

- **GENERADOR_FORMULARIO:** Dispone del método "`Análisis_Formulario()`", que realizará la creación de la estructura del formulario, éste que entra como "`input`" al sistema, esta labor se facilitará con la librería "`javax.xml.parsers.DocumentBuilder;`", que se usa para extraer toda la información y estructura del fichero que se introduce al sistema.

  La librería "`javax.xml.transform.Transformer;`", se utiliza una vez ya finalizados los cambios que se le han hecho a la estructura, en este caso incluir las etiquetas de VoiceXML en el formulario HTML, posteriormente se debe transformar la estructura creada que está en memoria, en un fichero VoiceXML válido.

  Esta misma clase invocará a los métodos "`Generar_Arbol`" dentro de un proceso recursivo, que es capaz de analizar en orden todas las etiquetas de un formulario HTML *[Ilustración 9]*, con este análisis previo, se podrá crear una estructura de árbol.

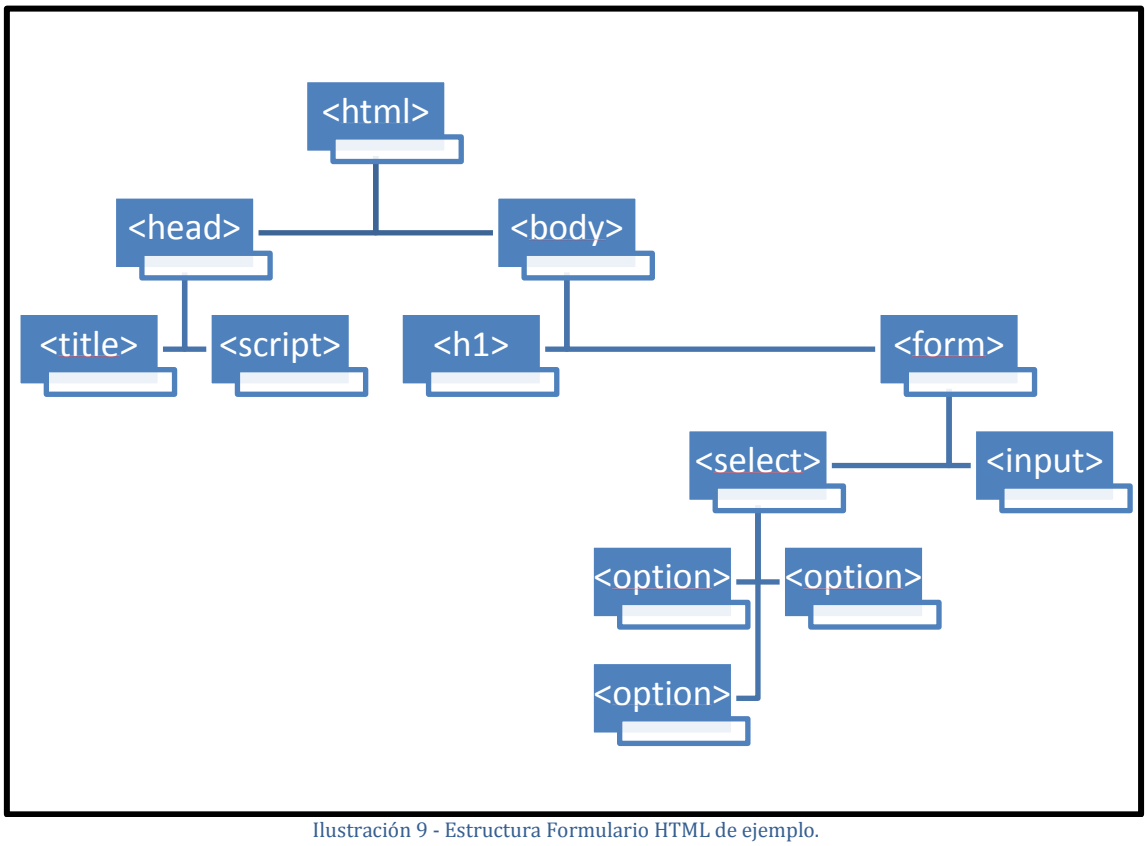

Ilustración 9 - Estructura Formulario HTML de ejemplo.

Este procedimiento también llamará a dos métodos de la clase Constructor_VoiceXML, estos son "`Crear_Etiquetas_Sincronizacion`", que se encargará de crear las etiquetas necesarias para la sincronización del formulario HTML con el formulario VOICEXML, es decir, que se puedan actualizar los datos del formulario HTML según lo que se le ha podido hablar a la aplicación.

El otro de los métodos llamado es "`Construir_Field`", que por cada uno de los campos que se hayan encontrado en el análisis del formulario, les deberá de construir la estructura VoiceXML correspondiente.

- **INPUT:**Dispone de un único método que es "`Analizar_Inputs`", este método será el encargado de recoger la información de todos y cada uno de los inputs del formulario HTML, es decir la información del input que es ("`select`" o "`input`") y del tipo de input ("`text`" o "`textarea`"). Esa información a posteriori se utilizará para la creación de la estructura.

- **ARBOL:**Dispone de 2 atributos propios que son "`EtiquetasHead`" y "`EtiquetasBody`", del tipo enumerado que contendrán todas las etiquetas posibles de las que se puede disponer debajo de las etiquetas "`head`" y del "`body`" respectivamente.

  También dispone de cinco métodos que ayudan al análisis y generación del árbol con la estructura del Formulario de entrada:

  El método "`Obtener_Atributos`", obtiene todos los atributos de los que dispone la etiqueta que se esté analizando.

  El método "`Obtener_Valor`", obtiene el valor que contiene la etiqueta, no se refiere al valor de ninguno de los atributos, si no al propio de la etiqueta, es decir, a lo encerrado entre el comienzo de la etiqueta y el final de la misma.

  Los métodos "`Es_Etiqueta_Head`" y "`Es_Etiqueta_Body`" son del tipo booleano, por lo que se utilizarán en distintas comparaciones que se

realicen en algunos métodos. Tendrán la finalidad de comparar si el valor del nodo que se le introduce como entrada está dentro del enumerado definido, tanto para las etiquetas "`head`" como para las etiquetas "`body`".

El método mas relevante es "`Generar_Arbol`", que utilizará todos los métodos mencionados anteriormente en esta clase, este método tendrá la finalidad de analizar el Formulario de entrada y poder representarlo para su comprobación.

- **GRAMATICA:**Esta clase dispone de un único método llamado "`Generar_Gramática`", tendrá como cometido el de generar la gramática correspondiente, al input que corresponda.

  Podrán distinguirse dos tipos de inputs, los inputs "`select`" o los inputs "`text` o `textarea`". Para los inputs "`select`" se ha decidido diseñar una gramática del tipo XML y para el tipo "`text` o `textarea`" se ha decidido diseñar una gramática ABNF.

  Ambos tipos de gramáticas estará embebidos en el código del fichero de salida.

- **CONSTRUCTOR_VOICEXML:**Esta clase dispone de 4 métodos, con éstos se realizarán todos los pasos necesarios para, crear la estructura VoiceXML e incorporarla al formulario de entrada. Todo ello servirá para que el Formulario pueda ser controlado por voz.

  Uno de los métodos es "`Crear_Etiquetas_Sincronizacion`", que sirve para construir las etiquetas que sincronizarán los elementos del formulario VoiceXML con los del formulario HTML que el usuario eligió, es decir, estas etiquetas sirven para que los valores dichos por el usuario en una ejecución serán interpretados por VoiceXML y por el formulario HTML.

El método "`Crear_Etiquetas_Estructura_Principal`", consiste en la creación estándar de la que dispone un formulario VoiceXML, se crean la etiqueta "`vxml:form`" con sus atributos específicos marcados por el estándar de la W3C.

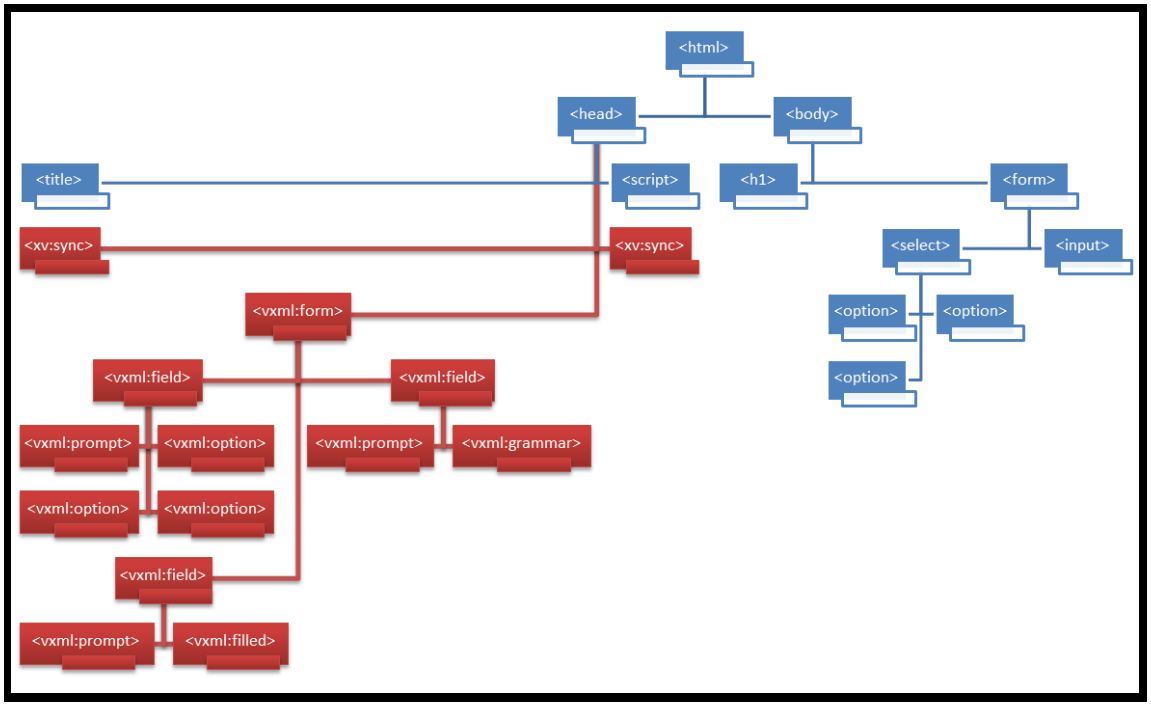

Ilustración 10 - Estructura Formulario HTML con VoiceXML de ejemplo.

El siguiente procedimiento es "`Construir_Field`" que realizará todo lo necesario para construcción de cada una de las etiquetas "`field`", para cada uno de los campos de los que dispone el formulario.

Dentro de este procedimiento será cuando se llamará al método "`Generar_Gramática`" dentro de la clase Gramática, que será el encargado de crear la estructura de la gramática y situarla en la estructura VoiceXML que se está creando.

A posteriori se llamará al procedimiento "`Solicitud_Prompt`", que es el cuarto método de esta clase, básicamente realiza una solicitud al usuario de que si quiere poner un "`prompt`" o mensaje explicativo, indicando la información que debe poner en cada campo.

4.2. FLUJO DE LA APLICACIÓN

A continuación se muestra un diagrama de secuencia sobre una ejecución de ejemplo del asistente.

4.2.1. DIAGRAMA DE ACTIVIDAD

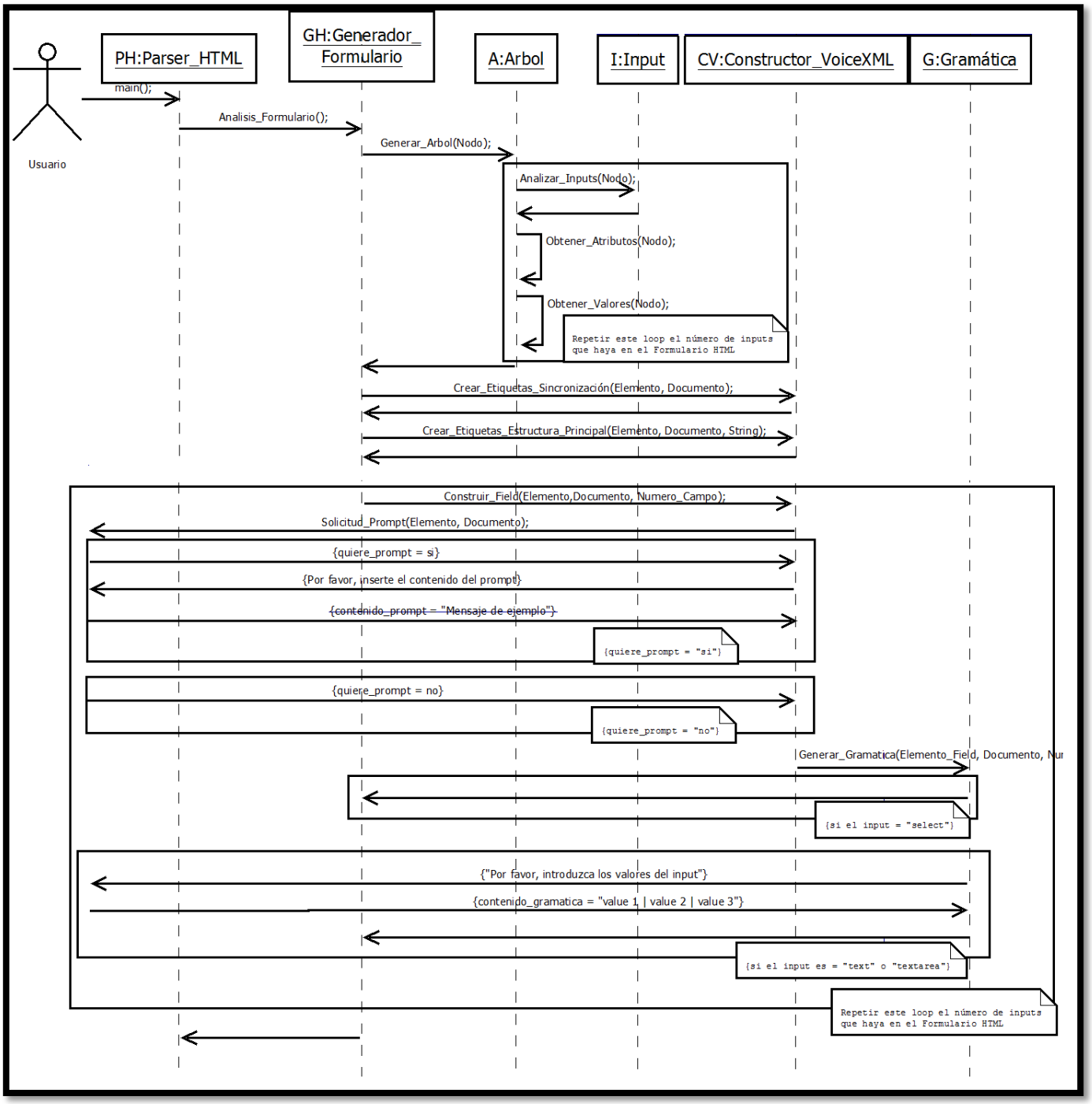

Ilustración 11 - Diagrama de Secuencia del Sistema.

4.2.2. DESCRIPCIÓN DEL DIAGRAMA DE ACTIVIDAD

El diagrama de la *Ilustración 11*, no muestra un escenario específico, se deja abierto al número de inputs que pueda tener el Formulario HTML de entrada.

En el citado diagrama se puede ver la interacción que tiene el usuario con la aplicación, y la comunicación que hay entre los distintos objetos.

El usuario iniciará el programa, que comunicará con la clase "`Parser_HTML`", que ejecutará el método llamado "`Generar_Arbol(Nodo)`", éste a su vez llamará a una pequeña estructura dentro de un bucle, que tendrá que repetirse tantas veces como inputs haya. Esta estructura está compuesta por los siguientes procedimientos "`Analizar_Inputs(Nodo)`", "`Obtener_Atributos(Nodo)`" y "`Obtener_Valores(Nodo)`".

Cuando ya se haya realizado las repeticiones del bucle correspondientes, el flujo llegará a la clase "`Generador_Formulario`", que invocará a los métodos "`Crear_Etiquetas_Sincronización(Elemento,Documento)`" y posteriormente a "`Crear_Etiquetas_Estructura_Principal(Elemento, Documento, String)`".

Tras estos dos procedimientos, la siguiente estructura se repetirá el mismo número de veces como "`inputs`" haya en el formulario, se refiere a la construcción de cada uno de los campos con sus correspondientes gramáticas.

Se accionará el método "`Construir_Field(Elemento,Documento,Número_Campo);`" que deberá preguntar al usuario si desea introducir un "`prompt`" o mensaje explicativo para ese "input", el usuario indicará una respuesta, en caso de ser positiva el sistema le

indicará al usuario que introduzca el "`prompt`" correspondiente, una vez introducido la aplicación continua. En caso de que la respuesta sea negativa, la ejecución continuará.

A continuación llega el momento de instanciar a la clase gramática con el método "`Generar_Gramatica(Elemento, Documento, numero_campo);`" que tendrá una nueva bifurcación, esta bifurcación la resolverá el sistema comparando el tipo de "`input`" que tiene que tratar, si es "`select`" tomará los valores que tenía del mismo; si es "`text`" o "`texarea`" tomará los valores que introduzca por teclado el usuario.

Al término de estos pasos, el flujo del sistema llega a la clase "`Constructor_Formulario`" y de ahí finalizará el programa.

Capítulo 5:

Implementación y Pruebas

CAPÍTULO 5: IMPLEMENTACIÓN Y PRUEBAS

En este capítulo se define la descripción sobre los métodos y procedimientos más relevantes de este sistema.

Este capítulo también tiene reservado un apartado para el chequeo de las pruebas que se han definido en el plan de pruebas.

5.1. MÉTODOS

A continuación se podrá ver una breve descripción de los métodos implementados más relevantes para el funcionamiento del sistema.

5.1.1. GENERAR ESTRUCTURA DEL FORMULARIO

El primer método que se usa en la aplicación será "`Análisis_Formulario()`", que corresponde a la clase Generador Formulario, consiste en el procedimiento principal de la aplicación, se encarga de la creación de la estructura del Formulario HTML [21], se ayuda de la librería "`javax.xml.parsers.DocumentBuilder;`".

Al final tras los ajustes que se han realizado en el formulario para que pueda utilizarse con VoiceXML se usará la siguiente librería "`javax.xml.transform.Transformer;`", servirá para transformar la estructura, en un fichero que posteriormente pueda ser utilizado y reproducido por VoiceXML.

Lo primero de todo será recorrer el árbol (estructura del documento) en forma recursiva, se ha decidido de esta manera para que no hubiera ningún problema a la hora de interpretarlo por el navegador. El algoritmo seleccionado ha sido el de amplitud, en la siguiente imagen *[Ilustración 11]*, se puede ver un árbol

de ejemplo con el orden en el que sus nodos han sido expandidos.

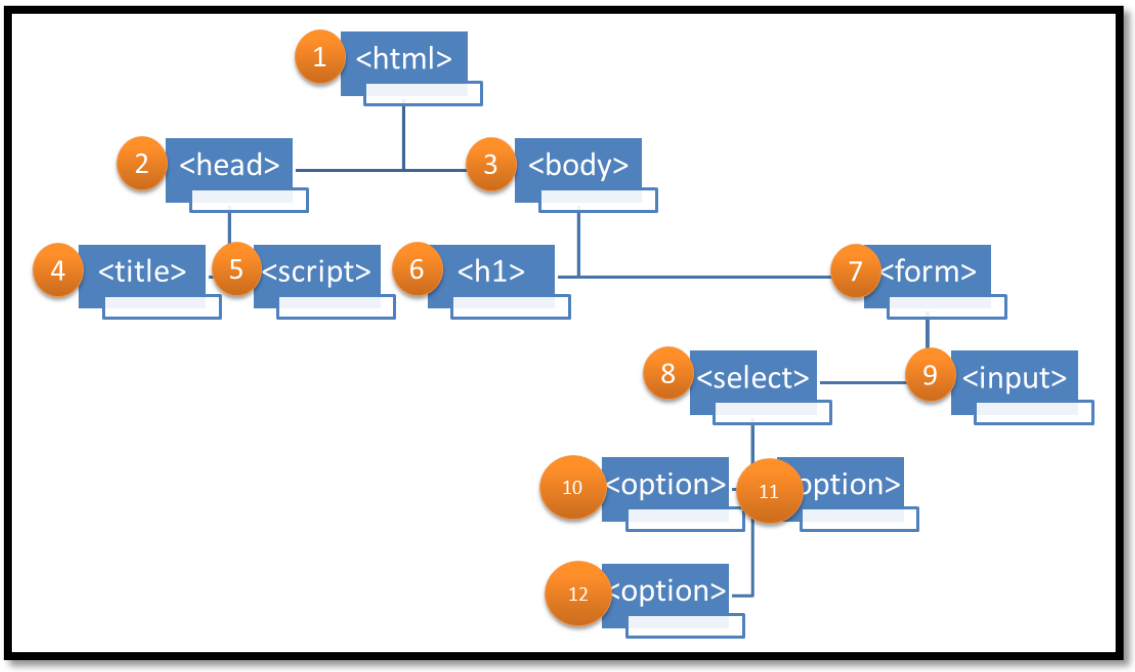

Ilustración 12 - Ejemplo Algoritmo de Amplitud.

En el momento que se analizan estos elementos, hay que recopilar algunos datos sobre el mismo que ayudarán a facilitar la labor del asistente, teniendo que pedirle los menos datos posibles al usuario.

5.1.2. OBTENER DATOS DEL FORMULARIO Y DEL USUARIO.

La estructura de datos creada para recopilar toda la información, que se ofrecen desde los inputs, es la siguiente:

```
static class input {
      String input="";
      String type="";
      String name="";
      String value="";
};

static input [] vector= new input [numero_campos_formulario];
```

Tabla 64 - Estructura de Datos para almacenamiento de información de los inputs.

Como se puede ver es una clase que contiene 4 campos, todos ellos de tipo "`String`", el primero de ellos es "`input`" que reflejará la característica de input del que se trata, si es "`select`" o "`input`". El siguiente campo es "`type`" que marcará el tipo de input del que se trata, para los "`select`" no tomará ningún valor pero para

los input puede tomar los siguientes valores "`text`", "`textarea`", "`submit`" y "`reset`".

El campo "`name`" indicará el nombre identificativo elegido para este "`input`", ese nombre tiene que ser único ya que cuando el usuario quiera introducir un "`prompt`" o mensaje declarativo, o los valores posibles que puede tomar ese campo, el usuario debe conocer de que "`input`" se trata.

Éste valor también es importante ya que, se utilizará para la creación de las etiquetas de sincronización entre el Formulario HTML y el Formulario VoiceXML que se crea.

Por último, la cadena de caracteres llamada "`value`", guardará todos los posibles valores que pueden tomar los "`inputs`", para los "`select`" ya vendrán plasmados en el código, pero para los "`input`" o tipo "`text`" o "`textarea`", el usuario los tendrá que definir introduciéndolos con el teclado.

El formato con el que el usuario tendrá que introducir estos valores para el caso en concreto será "`value1 | value2 | value 3`", los valores irán separados por barras delimitadoras "`|`".

Para guardar la información de los "`input`" y de los "`select`", en la clase Input, se declara el método "`Analizar_Inputs (Node Nodo)`" se utilizan estos procedimientos que a continuación se detallan.

```
if (Nodo.getNodeName().toString() == "select"){
  campos_formulario = campos_formulario + 1;
  vector[campos_formulario] = new input();
  vector[campos_formulario].input=Nodo.getNodeName().toString();
  vector[campos_formulario].name=Nodo.getAttributes().getNamedIte
  m("name").getTextContent().toString();
  NodeList NodoOpciones = Nodo.getChildNodes();
  int a = NodoOpciones.getLength();

  for (int h=0; h<a; h++){
if (NodoOpciones.item(h).getNodeName().toString()=="#text"){}
else{
      vector[campos_formulario].value=vector[campos_formulario].
      value+NodoOpciones.item(h).getAttributes().getNamedItem("v
      alue").getTextContent().toString()+ "·";
}
}
}
```

Tabla 65 - Método de capturar la información del input "select".

```
if ((Nodo.getNodeName().toString() == "input")){
  String
  contenido_etiqueta_tipo=Nodo.getAttributes().getNamedItem("type
  ").getTextContent().toString();
  if     ((contenido_etiqueta_tipo.contentEquals("text"))     ||
  (contenido_etiqueta_tipo.contentEquals("textarea"))){
      campos_formulario = campos_formulario + 1;
      vector[campos_formulario] = new input();
      vector[campos_formulario].input=Nodo.getNodeName().toStri
      ng();
      vector[campos_formulario].type=Nodo.getAttributes().getNa
      medItem("type").getTextContent().toString();;
      vector[campos_formulario].name=Nodo.getAttributes().getNa
      medItem("id").getTextContent().toString()
  }
```

Tabla 66 - Método de capturar la información del input "text" y "textarea".

Estos dos métodos anteriores tienen una estructura similar, primero de todo se comprueba que el nombre del nodo de la estructura es "`select`" o "`input`", a continuación en el "`input`" se comprueba que sea del tipo "`text`" o "`textarea`".

De manera paralela se aumenta la variable que se ha definido como contador de campos del formulario, que es un valor muy relevante, sobre todo para guardar en memoria las posiciones necesarias del vector del tipo input.

A continuación se extrae la información del nombre del Nodo de la estructura, que es el que lo contiene, se guarda en la estructura de datos creada para ello.

Para los "`select`", se realizará el traspaso de los valores de las opciones, se guardarán en la variable "`value`", al ser varios los valores posibles, se concatenan con el siguiente símbolo de separación entre cada uno de ellos "`·`".

5.1.3. CONSTRUCCIÓN DEL FICHERO VOICEXML

En esta se detallan los métodos más relevantes para la construcción del formulario VoiceXML.

5.1.3.1. ESTRUCTURA PRINCIPAL

Una vez recogidos estos datos, se decide ir desarrollando la estructura del VoiceXML, que dependerá del nodo Head del documento HTML. Para ello se desarrollan dos procedimientos "`staticElement Crear_Etiquetas_Estructura_Principal (Element Elemento, Document Documento, String nombre_del_formulario)`" y "`staticvoid Crear_Etiquetas_Sincronizacion (Element Elemento, Document Documento)`", que a continuación se detallan.

```
staticElement    Crear_Etiquetas_Estructura_Principal    (Element
Elemento, Document Documento, String nombre_del_formulario) {
Element Elemento_Form = Documento.createElement("vxml:form");
  Elemento_Form.setAttribute("xmlns",
"http://www.w3.org/2001/vxml");
Elemento_Form.setAttribute("id", nombre_del_formulario);
Elemento.appendChild(Elemento_Form);

Element Elemento_Block = Documento.createElement("vxml:block");
Elemento_Block.setTextContent("  WELCOME  TO  VOICEXML  SYSTEM.
PLEASE, FOLLOW THE INSTRUCTIONS TO COMPLETE THE FORM. ");
Elemento_Form.appendChild(Elemento_Block);
return Elemento_Form;
}

staticvoid  Crear_Etiquetas_Sincronizacion  (Element  Elemento,
Document Documento){
for (int i=1; i <= Inputs.campos_formulario; i++){
     Element Elemento_sinc=Documento.createElement("xv:sync");
     Elemento_sinc.setAttribute("xv:input",Inputs.vector[i].name.t
     oString());
     Elemento_sinc.setAttribute("xv:field","#"+Inputs.vector[i].na
     me.toString());
     Elemento.appendChild(Elemento_sinc);
}
}
```

Tabla 67 - Procedimientos Creación estructura VoiceXML (I).

El primer procedimiento será el encargado crear el elemento "`form`" con los atributos que marca el estándar, también se creará un elemento "`block`" para reproducir un mensaje nada más comenzar a ejecutar el formulario, será meramente informativo "`WELCOME TO VOICEXML SYSTEM. PLEASE, FOLLOW THE INSTRUCTIONS TO COMPLETE THE FORM.`". A continuación este elemento "`block`" se tiene que juntar con la estructura "`form`" ya que será un hijo más.

El segundo procedimiento constará en la creación de las etiquetas de sincronización, se encargará de construir las etiquetas necesarias para la sincronización de los elemento del formulario HTML con los elementos del formulario VoiceXML es decir, al introducir por voz el valor de uno de los campos del formulario se pueda actualizar en el formulario HTML. Se realizarán tantas etiquetas como campos existan en el formulario.

5.1.3.2. CAMPOS DEL FORMULARIO

El siguiente paso será la construcción de la estructura correspondiente a cada uno de los "input" que tenga el formulario, de eso se encargará el procedimiento Construir Field, que se puede ver a continuación.

```
static void Construir_Field (Element Elemento, Document
Documento, int Numero_Campo){
  Element Elemento_Field = Documento.createElement("vxml:field");
  System.out.println ("--> Las siguientes preguntas son acerca
  del campo nº: " + Numero_Campo + " del tipo: "+
  Inputs.vector[Numero_Campo].type.toString()+ " llamado: " +
  Inputs.vector[Numero_Campo].name.toString());

  Elemento.appendChild(Elemento_Field);
  Elemento_Field.setAttribute("name",Inputs.vector[Numero_Campo].
  name.toString());

  Elemento_Field.setAttribute("xv:id",Inputs.vector[Numero_Campo]
  .name.toString());

  Solicitud_Prompt(Elemento_Field, Documento);
  Gramatica.Generar_Gramatica(Elemento_Field, Documento,
  Numero_Campo);
}
```

Tabla 68 - Procedimientos Creación estructura VoiceXML (II).

En este método lo primero de todo será informar al usuario de cuales de los campos se está tratando, con la información del tipo y su nombre, esta información también tendrá que ser incluida en los atributos de la etiqueta "vxml:field". A continuación se le solicitará al usuario si desea un "prompt" o mensaje declarativo.

Para ello esta el método "Solicitud_Prompt(Elemento_Field, Documento)" que solicitará al usuario si desea introducir un "prompt" o mensaje explicativo de cada campo del formulario, estos mensajes pueden ser muy útiles para informar al usuario de la aplicación. Aprovechando que la respuesta es positiva a la creación de un prompt, se creará la estructura VoiceXML que le corresponde para incluirla dentro del campo correspondiente.

```
static void Solicitud_Prompt(Element Elemento, Document
Documento){
try{
    System.out.println("¿Quiere indicar algún mensaje explicativo
    sobre este campo? (sí o no)");
    String quiere_prompt = Entrada.readLine();
    if (quiere_prompt.equalsIgnoreCase("si")){
        System.out.println("Indique a continuación lo que quiere
        poner en el prompt");
        String valor_prompt = Entrada.readLine();
        Element                Elemento_Prompt                =
        Documento.createElement("vxml:prompt");
        Elemento.appendChild(Elemento_Prompt);
        Elemento_Prompt.setTextContent(valor_prompt);
    }
}
// Manejador de Excepciones
    catch (Exception e) {
    e.printStackTrace();
}
}
```

Tabla 69 - Procedimientos Creación estructura VoiceXML (III).

5.1.3.3. GRAMÁTICA

Por último se realizará la creación de la Gramática reconocedora de palabras, es decir todos los posibles valores que pueden introducirse en el formulario. Para ello se usa el método "`Generar_Gramatica(Elemento_Field, Documento, Numero_Campo)`".

Como se explicó en la fase de diseño para los "`select`" se decidió implementar gramáticas XML, y para los "`inputs`" se decidió hacer gramáticas ABNF, por ello al principio del método habrá una comparación, para saber si es de un tipo o de otro.

Para los "`select`" se tiene que extraer de la variable "`vector[x].value`" todos los valores, se debe recordar que estaban concatenados con un símbolo de separación, este símbolo es "`·`".

A continuación se crean etiquetas "<vxml:option>" que será el formato que tienen las opciones disponibles para cada campo, en las gramáticas XML. A estas etiquetas hay que introducirles el atributo "value" en el cual irá el valor final de la opción.

```
if ((Inputs.vector[i].input.toString().contentEquals("select")){
String Opciones = Inputs.vector[i].value.toString();
int Numero_Opciones = 0;
  /*Función para contar los campos que tiene el select
(miraremos las marcas que tiene el String)*/
for(int x=0;x<Opciones.length();x++) {
if ((Opciones.charAt(x)=='·')){
Numero_Opciones++;
     }
   }
for (int c=1; c<=Numero_Opciones; cont++){
Element                    Elemento_Option                    =
Documento.createElement("vxml:option");
String[] Array_Opciones = Opciones.split("·");
Elemento_Option.setAttribute("value",Array_Opciones[c-1]);
     Elemento_Option.setTextContent(Array_Opciones[cont-1]);
Elemento.appendChild(Elemento_Option);
}
}
```

Tabla 70 - Procedimientos Creación estructura VoiceXML (IV).

Para los "input" no se disponen de valores, para introducir en la gramática, por ello se deberán solicitar al usuario, que los deberá introducir por teclado. Con el siguiente formato "value1 | value2 | value 3", separados siempre por una barra horizontal "|".

A continuación se crean la estructura de la gramática con el elemento "<vxml:grammar>", siempre conservará este formato:

```
<vxml:grammar type="application/x-jsgf">
<![CDATA[grammarNOMBRE_CAMPO;
Public<NOMBRE_CAMPO>=VALUE1|VALUE2|VALUE3;]]>
</vxml:grammar>
```

Tabla 71 - Formato Gramática ABNF.

Variarán tanto en nombre del campo, como los valores que puede tomar, que serán los introducidos por el usuario.

El método completo es el siguiente:

```
if  (Inputs.vector[i].input.toString().contentEquals("input")  &&
(Inputs.vector[i].type.toString().contentEquals("text")        ||
Inputs.vector[i].type.toString().contentEquals("textarea"))){
try{
Element Grammar = Documento.createElement("vxml:grammar");
Grammar.setAttribute("type","application/x-jsgf");
String Comienzo_Grammar = "<"+"![CDATA[";
StringCabecera_Grammar=("grammar"                              +
Inputs.vector[i].name.toString() + ";");
String Final_Grammar = "]]"+">";

System.out.println  ("Indique  las  posibles  valores  que  puede
tomar este input, separelos con este caracter '|' ");
String Opciones_Grammar = Entrada.readLine();
String       Cuerpo_Grammar       =       ("public"+"<"        +
Inputs.vector[i].name.toString() + ">"+" =" + Opciones_Grammar +
";");
String Completa_Grammar = (Comienzo_Grammar + Cabecera_Grammar +
Cuerpo_Grammar + Final_Grammar);
System.out.println (Completa_Grammar);
Grammar.setTextContent(Completa_Grammar);
Elemento.appendChild(Grammar);
}
```

Tabla 72 - Procedimientos Creación estructura VoiceXML (V).

5.2. PRUEBAS

Para conocer mejor la información sobre las pruebas, se ha creado una tabla de valores con los siguientes atributos a rellenar:

- **Identificador:** clave que identifica la prueba de manera única.
- **Descripción:** explicación detallada la prueba.
- **Resultado:** resultado de la ejecución de la prueba.
- **Comentarios:** comentarios relevantes tras la ejecución de la prueba.

5.2.1. EJECUCIÓN DE LAS PRUEBAS

A continuación se reflejan los resultados a las ejecuciones de las siete pruebas definidas anteriormente.

ID:	PR-01
Descripción:	
Se inicia la aplicación, y se elige un formulario correcto que está en el directorio de la aplicación. En caso de que haya sido correcto, el programa continuará, pero en el caso contrario, el programa seguirá preguntando de nuevo y mostrando un mensaje de que el nombre introducido no es correcto.	
Resultado:	✓
Comentarios a la Prueba:	

Tabla 73 - Resultado de la prueba PR-01

ID:	PR-02
Descripción:	
Se inicia la aplicación, y se elige un formulario correcto. Cada vez que se analice un "`input`", el sistema pregunta si el usuario quiere poner un "`prompt`", si es así tendrá que poner el texto que quiere como "`prompt`". En caso contrario el programa continuará con otro "`input`".	
Resultado:	✓
Comentarios a la Prueba:	

Tabla 74 - Resultado de la prueba PR-02

ID:	PR-03
Descripción:	
Se inicia la aplicación, y se elige un formulario correcto. Cada vez que se analice un input del tipo "`text`" o "`textarea`", el sistema pregunta al usuario por los valores posibles que tiene que tener ese campo. Cuando son introducidos con el formato adecuado, es decir, cada palabra o "`token`" tiene que estar separado por una barra horizontal "`|`", continúa el programa.	
Resultado:	✓
Comentarios a la Prueba:	

Tabla 75 - Resultado de la prueba PR-03

ID:	PR-04
Descripción:	
Se inicia la aplicación, y se elige un formulario correcto. Cada vez que se analice un "`input`" del tipo "`select`", el sistema automáticamente obtendrá los valores de ese campo. Cuando son introducidos con el formato adecuado continúa el programa.	
Resultado:	✓
Comentarios a la Prueba:	

Tabla 76 - Resultado de la prueba PR-04

ID:	PR-05
Descripción:	
Se inicia la aplicación, y se elige un formulario correcto. Este formulario tiene que estar creado con la carencia de algún atributo ("`id`" o "`name`") de la etiqueta input. Cada vez que se analice un input con esta carencia, el sistema pregunta al usuario por el valor de ese atributo. Cuando son introducidos por el usuario el programa continúa.	
Resultado:	✓
Comentarios a la Prueba:	

Tabla 77 - - Resultado de la prueba PR-05

ID:	PR-06
Descripción:	
Se inicia la aplicación, y se elige un formulario correcto. Para cada uno de los "`input`" se crea una gramática asociada, la que contiene los valores posibles que puede tomar. Tras introducir los datos en la prueba PR-03 o PR-04, en el fichero de salida de la aplicación, se puede ver el resultado de esta prueba, es decir, se tiene que apreciar la estructura de la gramática dentro de cada uno de los campos del formulario.	
Resultado:	✓
Comentarios a la Prueba:	

Tabla 78 - Resultado de la prueba PR-06

ID:	PR-07
Descripción:	
Se inicia la aplicación, y se elige un formulario correcto. Se realiza la PR-02, PR-03 y PR-04 y finaliza la aplicación. Al final de la ejecución se abre el fichero generado para comprobar que está bien formado.	
Resultado:	✓
Comentarios a la Prueba:	

Tabla 79 - Resultado de la prueba PR-07

Capítulo 6:

Gestión del Proyecto.

CAPÍTULO6: GESTIÓN DEL PROYECTO

En este apartado se ilustra la planificación que se ha realizado al principio del proyecto y también los resultados reales obtenidos.

6.1. PLANIFICACIÓN DEL PROYECTO

Las tareas que se han propuesto para este proyecto son las siguientes:

Hito de Inicio del proyecto.

- **Fase de Análisis:**
  - **Estudio Tecnología VoiceXML:** estudio general de la tecnología VoiceXML y de alguno de sus sublenguajes más importantes.
    - **Estudio VoiceXML:** estudio del estándar de la entidad w3c sobre la tecnología VoiceXML.
    - **Estudio SRGS:** investigación sobre el sublenguaje de las gramáticas.
    - **Estudio SSML:** formarse sobre la síntesis del lenguaje en este sublenguaje.
  - **Análisis de Requisitos:** recopilación del catálogo de requisitos que tiene que cumplir la aplicación que posteriormente se desarrollará.
  - **Diseño Diagramas de Casos de Uso:** Representación en este diagrama UML de los casos de usos relevantes en el sistema.
  - **Estudio Java - Lenguaje de Programación:**aprendizaje del lenguaje de programación Java, así como distintas librerías de este lenguaje que facilitan la implementación de la aplicación.
  - **Reunión Seguimiento Fase Análisis:** Reunión para verificar los puntos más destacados de la fase de análisis, con el tutor del Proyecto Fin de Carrera.

Hito Fin Fase de Análisis

- **Fase de Diseño:**
    - **Diseño Diagrama de Clases:** Representación de la estructura modular del programa, para la posterior fácil implementación.
    - **Diseño de Diagramas de Actividad:** Representación de diagramas UML sobre el funcionamiento del sistema.
    - **Reunión Seguimiento Fase Diseño:** Reunión para verificar los puntos más destacados del diseño del sistema, con el tutor del Proyecto Fin de Carrera.

Hito Fin Fase de Diseño

- **Fase de Implementación:**
    - **Método Recorrer árbol:** Procedimiento que permite recorrer el documento HTML, y construir una estructura de árbol con esas etiquetas, para poder analizarlo.
    - **Método Obtener información usuario:** Función que obtiene información relevante para la estructura VXML, ya sea de lo introducido por teclado por el usuario o por algunos atributos característicos del documento HTML.
    - **Método Construir Estructura VXML:** Función, que a partir de la información recopilada en la fase anterior, podrá construir la estructura específica de VoiceXML.
    - **Reunión Seguimiento Fase Implementación:**Reunión para verificar los puntos más relevantes de la fase de Implementación, así como realizar algunas ejecuciones, todo ello con el tutor del Proyecto Fin de Carrera.

Hito Fin Fase de Implementación.

- **Fase de Pruebas:**
    - **Plan de pruebas:** Batería de pruebas que realiza el programador, testeando los puntos más críticos de la aplicación.
    - **Reunión Seguimiento Fase Pruebas:** Reunión con el tutor del proyecto fin de carrera, para poner analizar los resultados de las pruebas realizadas por el programador y por los usuarios.

Hito Fin Fase de Pruebas

- **Documentación:**
    - **Confección Memoria:** Tarea que se realiza al final de cada día de trabajo, en el que se recogen los aspectos mas importantes que se han ido realizando durante esa jornada.

Hito Fin Fase de Documentación

- **Presentación del Proyecto:**Preparación de la Presentación del Proyecto Fin de Carrera ante el Tribunal de profesores, esta etapa incluye la preparación de diapositivas y el estudio minucioso de toda la documentación obrante en la Memoria del Proyecto.

Hito Fin del Proyecto.

6.1.1. PLANIFICACIÓN INICIAL ESTIMADA

En la planificación estimada de la duración del proyecto se ha conseguido un total de ***167 días***. La fecha de comienzo fue el ***lunes 12 de diciembre de 2011.***

Realizando cinco jornadas laborales de 4 horas cada una de ellas, de 16:00 a 20.00 horas, a lo largo de la semana, se ha empleado un esfuerzo continúo del 90% en las tareas correspondiente al ciclo de vida (análisis, diseño, implementación y pruebas), y un 10% de esfuerzo en cada jornada, para la fase de documentación, en la que se recogían todos los avances del proyecto.

La fecha estimada de finalización del proyecto ha sido la del ***martes 31 de julio de 2012.***

En la siguiente figura *[Ilustración 13]* se puede ver el diagrama de Gantt de la planificación del proyecto.

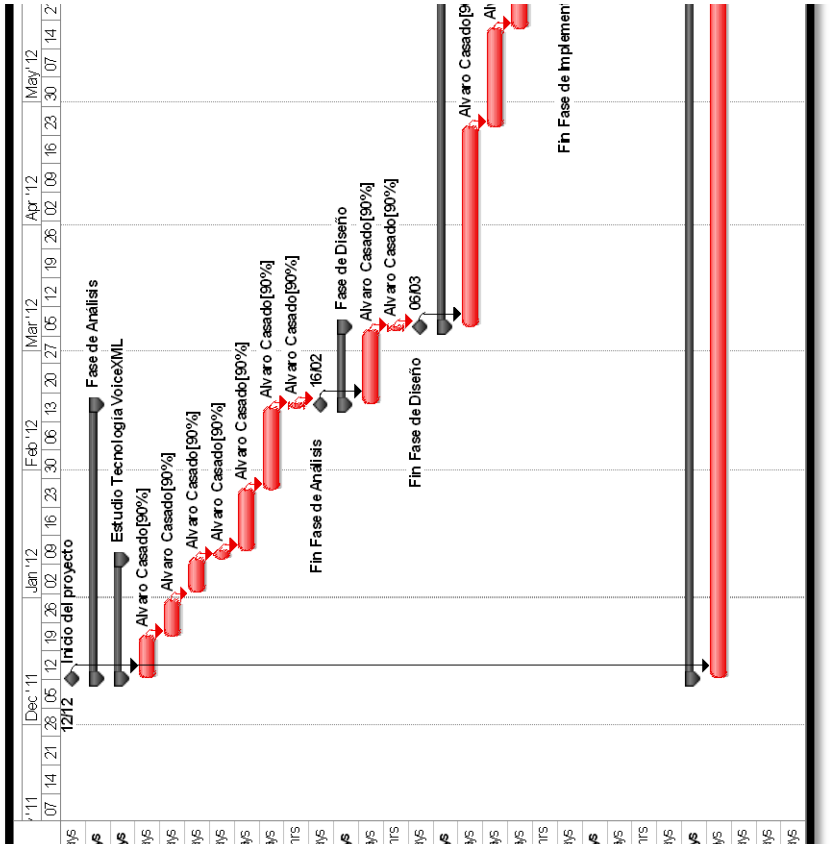

Ilustración 13-Microsoft Project: Diagrama de Gantt Planificación Proyecto.

6.1.2. PLANIFICACIÓN REAL

Tras la realización del proyecto, y llevar un seguimiento exhaustivo del mismo, la duración del proyecto se ha alargado hasta los ***215 días***, con una desviación de ***48 días***, al final del proyecto.

La fecha real para el final del proyecto, es decir, para la presentación del mismo ha sido la del ***05 de Octubre de 2012***.

En la siguiente figura *[Ilustración 14]* se puede ver el diagrama de Gantt de la planificación del proyecto.

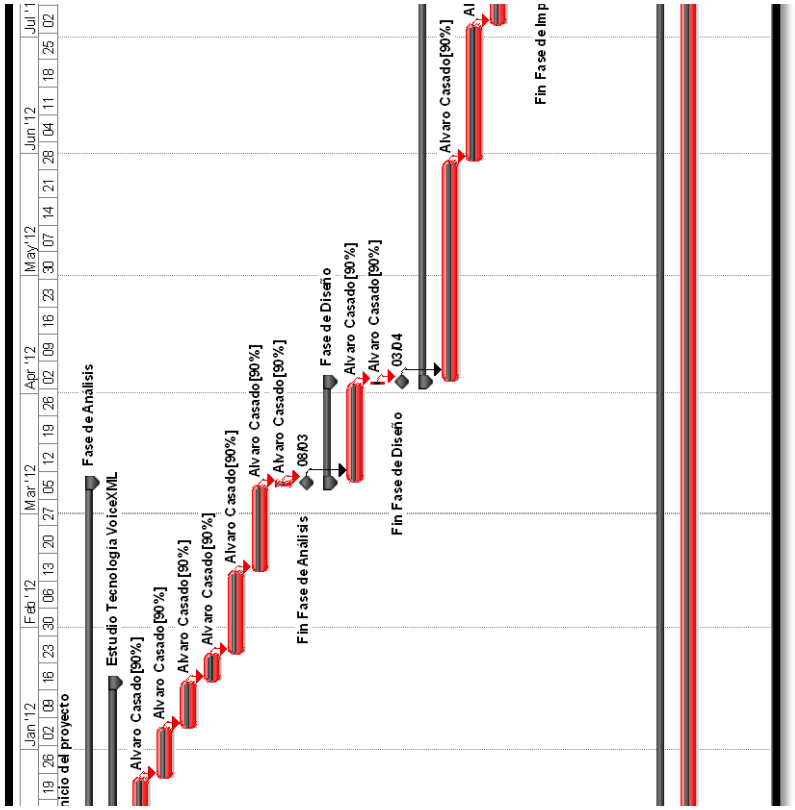

Ilustración 14 - Microsoft Project: Diagrama de Gantt Real del Proyecto.

6.1.3. ANÁLISIS DE PLANIFICACIÓN DEL PROYECTO

El principal motivo de esta desviación, es que se tuvo que compaginar el desarrollo del proyecto fin de carrera con el Curso de Adaptación al Grado de Ingeniería en Informática, que no se había planificado anteriormente, también no se habían contemplado periodo vacacional de verano, que sumó otras dos semanas de retraso al proyecto.

Por estos motivos la desviación del proyecto es de 48 días, que impidieron que se pudiera presentar el proyecto fin de carrera a finales de Julio, y se tuviera que hacer a principio de Octubre.

En esta imagen capturada el programa Microsoft Project, se puede ver algunos datos relevantes del proyecto, las líneas de "*baseline*", son las correspondientes a la planificación inicial, la línea de "*actual*" es la correspondiente a la planificación real.

Nota:El coste indicado en la imagen, únicamente corresponde con el coste del personal humano de este proyecto.

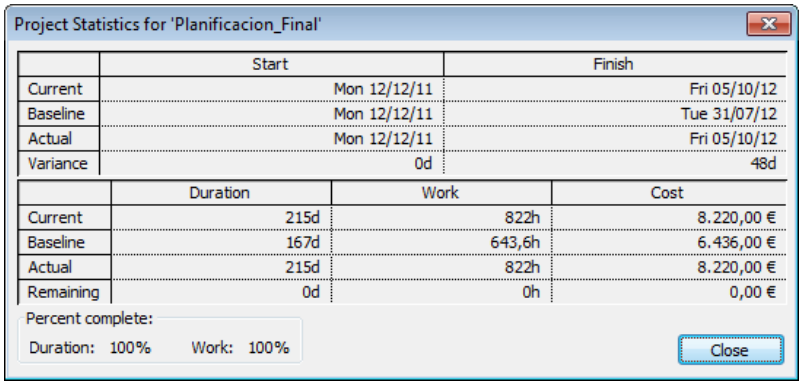

Ilustración 15 - Microsoft Project: Estadísticas de la Planificación del Proyecto.

En el siguiente diagrama*[Ilustración 16]*se puede apreciar el retraso que ha sufrido cada una de las tareas, en azul se puede ver la planificación final y en gris punteado el "*baseline"* que se realizó con la planificación inicial.

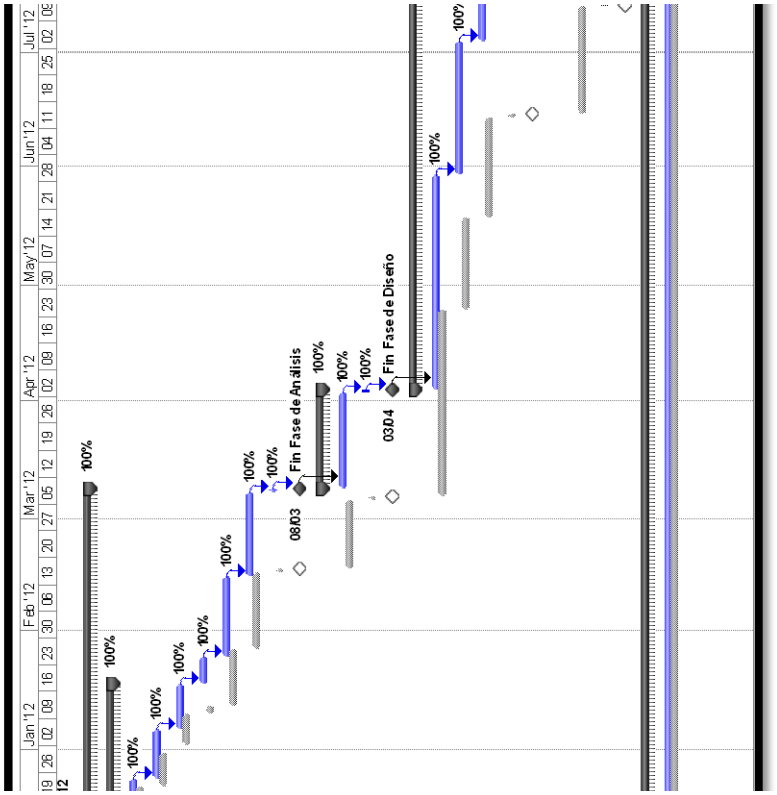

Ilustración 16 - Microsoft Project: Diagrama de Gantt con Seguimiento del Proyecto.

6.2. MEDIOS TÉCNICOS EMPLEADOS

En este apartado se ofrecen los distintos medios y herramientas, tanto hardware como software que se han utilizado durante todo el ciclo de vida de este Proyecto Fin Carrera.

Algunas de estas herramientas han sido imprescindibles para la realización de este proyecto, siempre complementadas por otras muy útiles.

6.2.1. MEDIOS HARDWARE

A continuación se puede apreciaruna tabla con los distintos dispositivos hardware, que se han necesitado para realizar el proyecto durante todo el ciclo de vida del mismo.

TIPO	NOMBRE	DISTRIBUIDOR

Ordenador.	Ordenador DELL con Windows 7	DELL
Impresora Multifunción.	Impresora HP Office jet J4585 All - in - one	Hewlett-Packard

Tabla 80 - Medios Hardware.

6.2.2. MEDIOS SOFTWARE

A continuación se puede visualizar una tabla con los distintos dispositivos software, que se han necesitado para realizar el Proyecto Fin de Carrera.

TIPO	NOMBRE	DISTRIBUIDOR
Sistema Operativo.	Windows 7, 64x – Enterprise Edition.	Microsoft.
Entorno de Programación.	Eclipse JDE	Oracle JAVA.
Navegador.	Opera Web Browser versión 9.2	Opera Software.
Navegador.	Internet Explorer 8.0.	Microsoft.
Editor de Código Fuente.	Notepad ++	Notepad Team (GNU).
Procesador de Texto.	Office - Word 2010.	Microsoft.
Tabla de cálculo.	Office - Excel 2010.	Microsoft.
Planificador Temporal y de Coste.	Office – Project 2010.	Microsoft.
Diagramas.	Office – Visio 2010.	Microsoft.
Diagramas UML.	Día Portable v.0,97	Día (GNU)
Presentación Diapositivas.	Office – Power Point 2010.	Microsoft.
Control de Versiones y Almacenamiento Web.	Dropbox	Dropbox.

Tabla 81 - Medios Software.

- **Navegador Ópera**, [14]en su versión 9.2, es uno de los navegadores web más relevantes dentro del mundo de la red, esta versión que se ha utilizado en el proyecto, es la más actual que dispone del "plugin", fácilmente descargable, para la interactividad con VoiceXML.

Ilustración 17 - Opera Software**[15]**.

- **Eclipse JDK,** [16]en su versión 4.2, es probablemente el software más utilizado para el desarrollo de aplicaciones para ordenadores y también para dispositivos móviles o "tablets".

Ilustración 18 - Eclipse JDK**[17].**

- Para manipular los ficheros del código fuente, se ha utilizado el software **Notepad++,** que dispone de una interfaz bastante amigable para tratar con el código.

- Paquete de herramientas de **Microsoft Office**:
  - *Word 2010:* procesador de textos para la redacción de la documentación.
  - *Excel 2010:* hoja de cálculo para la confección de tablas y del presupuesto.
  - *Project 2010*: software para la gestión y administración de proyectos. También dispone de algunos diagramas de planificación

(p.e. Gantt).

- *Visio 2010*: software encargado de representar diagramas UML del proyecto.
- *Power Point 2010*: software encargado de realizar presentaciones con diapositivas.

6.2.3. LENGUAJES DE PROGRAMACIÓN

Para la creación de la aplicación se ha utilizado el lenguaje de programación Java[16], que gracias al gran número de librerías y código de libre acceso, facilita la implementación en gran medida.

Uno de los grandes factores para el uso y elección de JAVA, fue la gran comunidad de desarrolladores que dispone, lo cual es un punto positivo para poder obtener ayuda, ya sea por otros usuarios de JAVA o a través de documentación oficial.

6.2.4. MANUALES Y ESTÁNDARES

Los estándares de la tecnología VXML están definidos por la "World Wide Web Consortium" (W3C) [2], estos estándares son en los que se ha basado todo el conocimiento para estudiar el lenguaje de VoiceXML[18], [19].

Los ejemplos prácticos que circulan por la red no son muy cuantiosos, existen algunos del ámbito académico y algunos pero más escasos de ámbito laboral[20]. Pero suficientes como para enseñar el funcionamiento básico de esta nueva tecnología.

6.3. GESTIÓN ECONÓMICA

Para realizar la gestión económica del proyecto, se ha seguido la plantilla de guía que ofrece la universidad para realizar los cálculos económicos.

Dentro de la gestión económica del proyecto se distinguen 3 tipos de gastos:

- **Recursos Humanos:** Personas físicas que participan en la realización de este proyecto fin de carrera.
- **Equipos:** Recursos materiales utilizados para la confección del proyecto.
- **Otros Gastos:**Listado de otros gastos asociados al proyecto.

Para obtener el coste total del proyecto, se determinará el sumatorio de todos los tipos de gastos.

6.3.1. COSTES ESTIMADOS

En este apartado se representa el presupuesto inicial del proyecto.

Los costes de los ***recursos humanos***durante todo el proyecto han sido los siguientes:

Puesto.	Coste/hora.	Total horas	Coste Total.
Álvaro Casado *(Programador Junior)*	10,00€ a la hora.	643,6 h	6436,00 €

Tabla 82 - Planificación de coste de recursos humanos.

Los costes de los ***equipos*** utilizados durante todo el proyecto han sido los siguientes:

Descripción	Coste (Euro)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable
Ordenador DELL con Windows 7	646,00 €	100	8	24	215,33 €
Paquete Microsoft Office Business 2010	340,00 €	100	8	24	113,33 €
Impresora HP Office jet J4585 All - in – one	100,00 €	100	8	24	33,33 €
				Total	362,00 €

Tabla 83 - Planificación de coste de los equipos utilizados en el proyecto.

Otros gastos del proyecto han sido estos:

Descripción	Empresa	Costes imputable
Material de Oficina		150,00 €
Consumibles (Tinta, Pen drive…)		60,00 €
	Total	210,00 €

Tabla 84 - Planificación de coste de otros gastos del proyecto.

A través del programa de planificación de proyectos Microsoft Project, se han obtenido los siguientes datos del proyecto, el coste solo indica los recursos humanos desplegados en el proyecto.

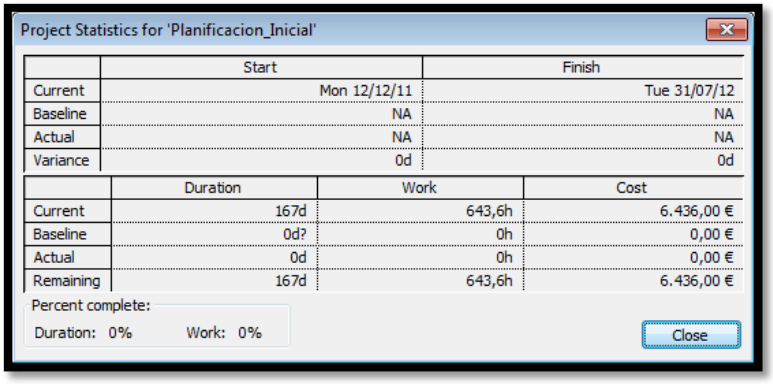

Ilustración 19 - Microsoft Project: Estadísticas de la Planificación del Proyecto.

El ***resumen del presupuesto*** es el siguiente:

Presupuesto Costes Totales	
Personal	6436,00 €
Amortización	362,00 €
Subcontratación de tareas	- €
Costes de funcionamiento	210,00 €
Costes Indirectos	1.401,60 €
Total	***8.409,60 €***

Tabla 85 - Planificación del Coste Total del proyecto.

6.3.2. COSTES REALES

En este apartado se reflejan los costes finales del proyecto.

Los costes de los ***recursos humanos*** durante todo el proyecto han sido los siguientes:

Puesto.	Coste/hora.	Total horas	Coste Total.
Álvaro Casado *(Programador Junior)*	10,00 € a la hora.	822 h	8220,00 €

Tabla 86 - Coste Final de recursos humanos.

Los costes de los ***equipos*** utilizados durante todo el proyecto han sido los siguientes:

Descripción	Coste (Euro)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable
Ordenador DELL con Windows 7	646,00 €	100	10	24	269,17 €
Paquete Microsoft Office Business 2010	340,00 €	100	10	24	141,67 €
Impresora HP Office jet J4585 All - in - one	100,00 €	100	10	24	41,67 €
				Total	**452,50 €**

Tabla 87 - Coste Final de los equipos utilizados en el proyecto.

Otros gastos del proyecto han sido estos:

Descripción	Empresa	Costes imputable
Material de Oficina		160,00 €
Consumibles (Tinta, Pen drive…)		60,00 €
	Total	220,00 €

Tabla 88 - Coste Final de otros gastos del proyecto.

En la siguiente captura de pantalla *[Ilustración 20]*se puede ver las estadísticas finales del proyecto que ofrece la herramienta Microsoft Project, el coste solo indica los recursos humanos desplegados en el proyecto.

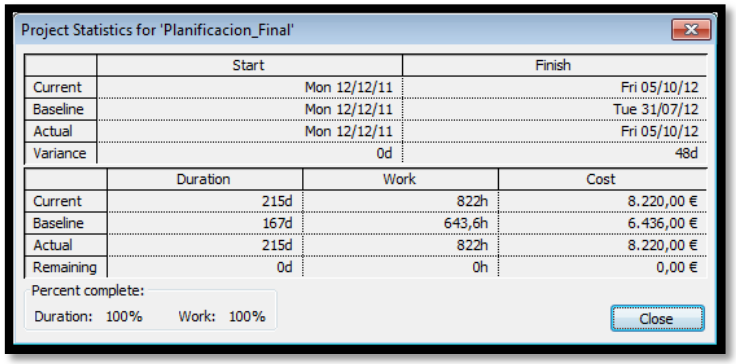

Ilustración 20 - Microsoft Project: Estadísticas Final del Proyecto.

El ***resumen de coste total*** es el siguiente:

Costes Totales	
Personal	8220,00€
Amortización	452,50 €
Subcontratación de tareas	- €
Costes de funcionamiento	220,00 €
Costes Indirectos	1.778,50 €
Total	***10.671,00 €***

Tabla 89 - Coste Total Final del proyecto.

6.3.3. ANÁLISIS DE COSTES DEL PROYECTO

Según se puede apreciar en los costes presupuestados el total daba como resultado***8.409,60 €*** mientras que en el final el valor del coste del proyecto era de ***10.671,00 €.***

Eso supone que la desviación del presupuesto, en estos 48 días extra que ha sido de (10.671,00 € - 8.409,60 €) =***2.261,40 €.***

La desviación económica del proyecto en datos porcentualesha sido de un ***26,89 %*** más, del presupuesto inicial, por los motivos mencionados en el apartado de Planificación Temporal.

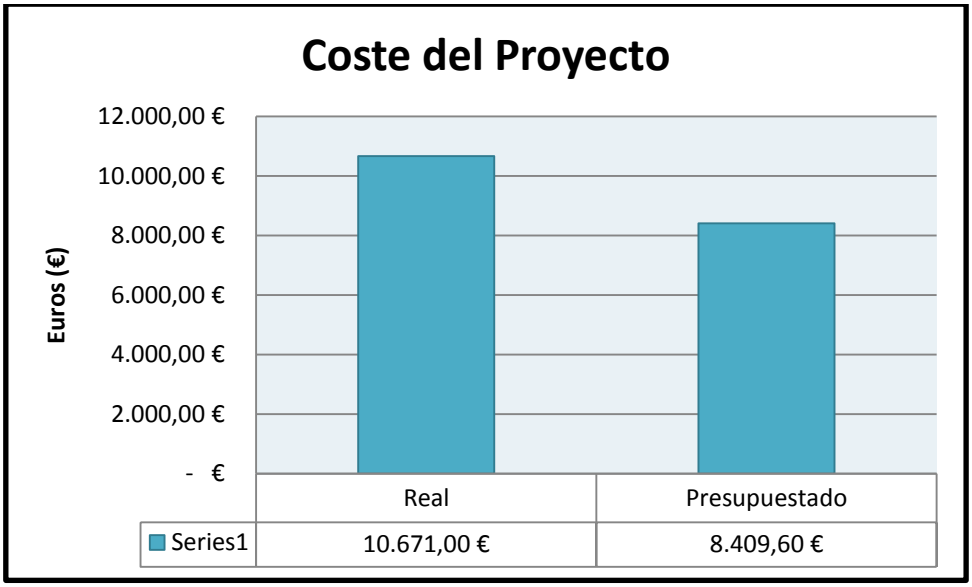

Ilustración 21 - Gráfica comparativa entre el coste del proyecto.

Capítulo 7:

Conclusiones y Líneas Futuras.

CAPÍTULO 7: CONCLUSIONES Y LÍNEAS FUTURAS

En este último capítulo de la memoria, se detallan las conclusiones extraídas durante el desarrollo de todas las fases de este proyecto fin de carrera.

También, en este capítulo se destina un pequeño apartado, a la aportación de algunas líneas o trabajos futuras sobre este proyecto, éstas tendrán relación con esta tecnología.

Estos trabajos añadirán nuevas funcionalidades a la aplicación, también podrán dar un nuevo enfoque al mismo.

7.1. CONCLUSIONES

Este proyecto ha sido muy interesante y productivo para la inmersión en el mundo de una tecnología desconocida, como era VoiceXML, esta tecnología puede llegar a ser muy útil para todo tipo de usuarios.

Esta herramienta puede proporcionar a personas con discapacidad visual, una manera de poder interactuar con una máquina, el alcance final de esta aplicación será poder rellenar y enviar un formulario sencillo con la voz.

Durante la fase de diseño e implementación se ha puesto un especial interés en que los mensajes que se le ofrecen al usuario por pantalla dentro del asistente, fueran fácilmente entendibles, y que no requirieran mucho aprendizaje más que el funcionamiento de un formulario web.

Con el estudio realizado de la tecnología y la puesta en práctica, mediante el desarrollo de esta aplicación, se puede afirmar que el uso de esta tecnología es manejable y no es complicado la realización de pequeñas aplicaciones para cualquier ámbito de negocio o entorno (educativo, social, deportivo…). La complejidad del mismo cambiaría si el sistema que se quiere desarrollar en un sistema amplio y complejo.

Ha sido un acierto usar el lenguaje de programación Java, ya que dispone de una gran documentación, programas de ejemplo y también existe gran cantidad de comunidades de desarrolladores en los que se ofrecen soluciones a problemas comunes, estos problemas pueden ser comunes en la fase de implementación.

El uso de librerías específicas para el reconocimiento y creación de distintas estructuras XML, ha sido de gran ayuda.

En el presente este tipo de aplicaciones y pequeños sistemas se está comenzando a distribuir, no solo para ordenadores, sino también para “smartphones” y “tablets”. Estas aplicaciones son cada día máscomunes y sobre todo mas utilizadas para la accesibilidad de todo tipo de personas a estas tecnologías, sin distinción de edad ni de discapacidad.

7.2. LÍNEAS FUTURAS

Para esta aplicación se podría pensar en unas cuantas mejoras, que propiciarían una ampliación de su funcionalidad.

- **Análisis completo de "inputs":** a partir de la aplicación realizada se podrían buscar soluciones a los otros "`inputs`" que no se han tratado en este proyecto, sería el caso de los tipo "`image`", "`checkbox`", "`radio`", "`hidden`" y "`password`". Para ellos habría que buscar una manera de tratarlos obteniendo información de los mismos y analizando sus estructuras y su conversión a VoiceXML.

- **Validarse en un sistema:** consistiría en poder hacer "`logan`" con usuario y su clave en un sistema, sin hacer falta teclearlo ni pulsar al teclado. Tiene que estar bien pensado ya que el "`logan`" se podría hacer a páginas con información sensible y la seguridad debería estar presente. Esta posible mejora va unida a la anterior mejora del "`input`" del tipo "`password`".

- **Implementación con otros lenguajes:** esto es para la creación de interfaces visuales que puedan agradar aún mas al usuario y lo principal, hacer lo mas sencillo posible su uso.

- **Generar GramáticasRetroalimentadas:** que la gramática aprenda y pueda ampliarse dinámicamente puede ser una solución a algunos problemas, de esta aplicación. Esto sería bueno para que la aplicación no fuera estática y cobrara dinamismo.

- **Selección de idioma:** podría buscarse una solución para que la aplicación no solo entendiera la lengua inglesa,es decir construir las gramáticas y el formulario VoiceXML detallando que el idioma no es inglés. De este posible trabajo futuro se obviarían idiomas con caracteres complicados de interpretar para el navegador.

- **Integración en web:** consiste en crear "plugin" que pudiera estar disponible en la web del formulario HTML, este "plugin"se iniciaría haciendo clic en un botón ("Haz clic aquí para convertir a VoiceXML"), justo a continuación se iniciaría el asistente, que convertiría el formulario, donde se emplaza el botón.

- **Integración en navegador:** el similar al anterior, pero esta vez sería concretar un "plugin" dentro del navegador Opera, éste ejecutaría el asistente con el formulario que tiene en la pestaña activa del navegador.

- **Integración en "smartphones" y "tablets":** como se menciona en las conclusiones, el uso de estos dispositivos hoy en día está al orden del día y cada vez están mas usados, un posible trabajo futuro sería el análisis de integración de esta tecnología en estos dispositivos, en los que se puede navegar por la red.

- **Inclusión de elementos SSML** Como se indica en el Estado del Arte, en esta aplicación no se usa este sublenguaje de VoiceXML, pero es muy potente para realizar mejoras en el habla, para la aplicación.

• GLOSARIO DE TÉRMINOS

Formulario HTML: Formulario que se utiliza como entrada al sistema, tendrá las características habituales de un formulario web.

Formulario VoiceXML: Formulario que se genera tras el paso por el sistema que se ha implementado, básicamente es la estructura inicial del Formulario HTML, pero con la estructura VoiceXML para el manejo del formulario con la voz.

Gramática Reconocedora de Palabras: Conjunto de palabras que son reconocidas por la tecnología VoiceXML para introducir como dato en un campo de un formulario, cada campo del formulario deberá tener una gramática. Hay de dos tipos ABNF y XML.

Sistema de diálogo: Softwareque tiene como objetivo interactuar conlos usuarios oralmente o de forma multimodal para ofrecer algunos servicios.

Comunicación Multimodal: Es la comunicación en la que intervienen distintos modos humanos con una máquina, como por ejemplo la voz. El objetivo principal de la comunicación multimodal es hacer más sencilla y natural la interacción entre hombre y máquina.

Interacción: Acción que se ejerce recíprocamente entre dos o más objetos, en este caso entre un humano y una máquina.

Accesibilidad: Capacidad de acceso a un entorno Web y a sus contenidos, sin distinción de agentes humanos o agentes humanos que disponen de una discapacidad (física o intelectual).

• LISTADO DE ACRÓNIMOS

VOICEXML: Voice eXtensible Markup Languaje.

W3C: World Wide Web Consortium.

HTML: HyperText Markup Language.

XHTML: Extensible HyperText Markup Language.

SRGS: Speech Recognition Grammar Specification.

SSML: Speech Synthesis Markup Language.

DTD: Document Type Definition.

DOM: Document Object Model.

XML: Extensible Markup Language.

DTMF: Dual-tone multi-frequency signaling.

ABNF: Augmented Backus–Naur Form.

URI: Uniform Resource Identifier.

• REFERENCIASY BIBLIOGRAFÍA

A continuación se listan las referencias a las que se hace durante todo el documento de la memoria del proyecto, todas ellas han servido como bibliografía para el mismo.

[1]Título: MANAGING THE DEVELOPMENT OF LARGE SOFTWARE SYSTEMS. Autor: Dr. Winston W. Royce

http://www.cs.umd.edu/class/spring2003/cmsc838p/Process/waterfall.pdf Último Acceso (Agosto 2012).

[2] http://www.w3c.es/Último Acceso: Agosto 2012.

[3] http://lema.rae.es/drae/?val=diálogo Último Acceso: Junio 2012.

[4] http://www.w3schools.com/tags/att_html_xmlns.asp Último Acceso: Agosto 2012.

[5] http://www.w3schools.com/dtd/default.asp Último Acceso: Agosto 2012.

[8] http://www.w3.org/TR/speech-grammar/Último Acceso: Agosto 2012.

[9]http://www.w3.org/TR/speech-synthesis/ Último Acceso: Agosto 2012.

[10] http://www.w3.org/TR/ccxml/ Último Acceso: Junio 2012.

[11] http://www.infor.uva.es/~mlaguna/is1/apuntes/2-requisitos.pdfÚltimo Acceso: Junio 2012.

[12] http://es.wikipedia.org/wiki/Lenguaje_Unificado_de_Modelado. Último Acceso: Agosto 2012.

[13] http://www.contabilidad.com.py/interna.php?id=73 Último Acceso: Agosto 2012.

[14] http://dev.opera.com/articles/view/getting-to-know-voice/ Último Acceso:

Junio 2012.

[15] http://www.cmswire.com/cms/web-cms/browser-wars-is-opera-96-a-hit-or-a-miss-003280.phpÚltimo Acceso: Julio 2012

[16] http://www.eclipse.org/downloads/moreinfo/java.php Último Acceso: Julio 2012.

[17] http://admmdn.blogspot.com/2010/07/empezando-con-java.html Último Acceso: Julio 2012

[18]http://www.w3.org/TR/voicexml20/Último Acceso: Julio 2012.

[19]Sharma,Cheetan;Kunins, Jeff: "VoiceXML: Strategic and Techniques for Effective Voice Applications Development with VoiceXML 2.0", (Ed.): 'Willey Computer Publishing'.

[20] http://www.larson-tech.com/james/?page_id=316 Último Acceso: Julio 2012.

[21] Goodman, Danny: "Dynamic HTML – The Definitive Reference", (Ed.): "O'REILLY" 1ª Edición Julio 1998.