



*Test your English! Desarrollo de un sistema multimodal mediante XHTML+Voice para un aprendizaje autónomo y personalizado de idiomas*



**Autor:** Ismael Baena Eguía

**Tutor:** David Griol Barres

Universidad Carlos III de Madrid

Ingeniería Técnica en Informática de  
Gestión.



**Título:** Test your English! Desarrollo de un sistema multimodal mediante XHTML+Voice para un aprendizaje autónomo y personalizado de idiomas.

**Autor:** Ismael Baena Eguía.

**Director:** David Griol Barres.

#### EL TRIBUNAL

Presidente: \_\_\_\_\_

Vocal: \_\_\_\_\_

Secretario: \_\_\_\_\_

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día \_\_ de \_\_\_\_\_ de 20\_\_ en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO

PRESIDENTE



# Agradecimientos

---

Me gustaría expresar mi agradecimiento a David Griol, tutor de este proyecto, por brindarme las ideas y herramientas necesarias y conseguir siempre, con motivadoras palabras de ánimo, que todo haya sido un poco más fácil.

Agradecer también la ayuda, y sobre todo paciencia, a mis padres y hermanos por ser el empujón necesario cuando más cuesta arrancar y a mis compañeros de carrera por los consejos, ideas y apoyo mostrados en todo momento.



# Resumen

---

El presente Proyecto Final de Carrera tiene como principal objetivo la implementación de una aplicación web cuyo cometido es la creación, gestión, visualización y resolución de ejercicios prácticos del idioma inglés con el fin de poder evaluar los conocimientos que tenemos y mejorar el aprendizaje del idioma.

La integración de un sistema de diálogo, desarrollado mediante el lenguaje XHTML+Voice, permite que la navegación por la aplicación y la resolución de ejercicios pueda realizarse indistintamente mediante escucha y voz o de forma convencional, mediante lectura e introducción de texto escrito. La posibilidad de ejercitar el habla en el proceso de práctica y aprendizaje de un idioma supone un valor añadido frente a los tradicionales métodos de ejercicios únicamente escritos.

El usuario puede realizar ejercicios típicos de gramática, mejorar su vocabulario o entrenar el oído realizando ejercicios de escucha. Para cada una de estas modalidades puede seleccionar la temática y ajustar el nivel deseado de acuerdo a sus conocimientos. Los resultados obtenidos en las pruebas son considerados por la aplicación a la hora de recomendar o lanzar nuevos ejercicios.

La aplicación no solo está orientada a usuario final, sino que incluye un módulo que completa las tareas típicas de administración con el fin de que los contenidos sean editados o ampliados de manera sencilla y rápida cuando así se requiera.

El funcionamiento del sistema requiere a su vez de la interacción con otras tecnologías actuales que aseguran su dinamismo y aumentan su eficiencia y flexibilidad. Por este motivo, la implementación ha supuesto el aprendizaje y utilización de servidores web, bases de datos y diferentes lenguajes de desarrollo adicionales (PHP y MySQL entre otros).

El ingrediente esencial que aumenta la exclusividad del producto es la integración del sistema de diálogo cuyo estudio y análisis sirve de introducción a este campo todavía en plena explotación y mejora, pero cuyas posibilidades lo convierten en un producto de actualidad y a tener muy en cuenta en el futuro.





# Abstract

---

The main objective of this bachelor's project is the implementation of a web application whose purpose is the creation, management, visualization and resolution of practical exercises in English in order to evaluate the knowledge that we have and improve the learning of this language.

The integration of a dialogue system, developed by means of the XHTML+Voice language, allows the navigation through the application and resolution of exercises to be done either by listening and speaking, or conventionally by reading and writing text inputs. The introduction of speech interaction in the process of learning a language provides an added value over traditional methods that only include written exercises.

The user can perform typical grammar exercises, improve his vocabulary or train the ear by doing listening exercises. For each of these modalities can select the topic and adjust the desired level according to his knowledge. The results obtained in the tests are taken into account to recommend and select new exercises.

The application is not only end-user oriented, but also it includes a module to complete typical administration tasks in order to make content edited or expanded easily and quickly when required.

The operation of the system requires interaction with other prevailing technologies that ensure its dynamism and increase efficiency and flexibility. For this reason, its implementation has involved the use of web servers, databases and different additional development languages (PHP and MySQL among others).

The essential feature that increases the exclusivity of this product is the integration of the dialogue system, whose study and analysis has served as an introduction to this field that is in a full operation and improvement in this moment and whose prospects make it a prevalent product to be considered in the future.



# Índice General

---

<b>1. Introducción y objetivos</b>	
1.1. Introducción .....	19
1.2. Objetivos .....	21
1.3. Fases de Desarrollo .....	23
1.4. Medios empleados .....	27
1.5. Estructura de la Memoria .....	28
<b>2. Estado del Arte</b>	
2.1. Introducción .....	31
2.2. Características de los Sistemas de Diálogo respecto al reconocimiento automático del habla .....	34
2.3. Evolución e Historia de los Sistemas de Diálogo .....	35
2.4. Introducción a VoiceXML .....	41
2.5. Introducción a XHTML+Voice .....	45
<b>3. Descripción General del Sistema Desarrollado</b>	
3.1. Introducción al Sistema .....	51
3.2. Tecnologías .....	53
3.2.1. Intérprete XHTML+Voice (Opera Voice) .....	53
3.2.2. MySQL .....	55
3.2.3. PhpMyAdmin .....	56
3.2.4. MySQL Administrator .....	58
3.2.5. Servidor HTTP (Apache) .....	59
3.3. Implementación de las Operaciones Generales .....	60
3.3.1. Desarrollo Dinámico .....	60
3.3.2. Estructura de las páginas PHP .....	61
3.3.3. Estilos .....	71
3.3.4. Bases de datos .....	73
<b>4. Descripción detallada de los Módulos del Sistema</b>	
4.1. Página principal o HOME .....	81
4.1.1. Funcionalidad .....	81
4.1.2. Arquitectura .....	82
4.2. Módulos de registro e inicio de sesión .....	83
4.2.1. Funcionalidad .....	83
4.2.2. Arquitectura y funcionamiento .....	85

4.3. Módulo de realización de ejercicios .....	89
4.3.1. Funcionalidad .....	89
4.3.2. Arquitectura, funcionamiento e interacción con bases de datos..	90
4.3.3. Esquema de Creación-Navegación de páginas e interacción con bases de datos .....	99
4.4. Módulo de Administración de Contenidos .....	102
4.4.1. Funcionalidad .....	102
4.4.2. Arquitectura, funcionamiento e interacción con bases de datos..	103
4.4.3. Esquema de Creación-Navegación de páginas e interacción con bases de datos .....	109
4.5. Módulo de puntuación y resultados (My Results) .....	112
4.5.1. Funcionalidad .....	112
4.5.2. Arquitectura, funcionamiento e interacción con bases de datos..	112
4.5.3. Esquema de Creación-Navegación de páginas e interacción con bases de datos .....	120

## Conclusiones y líneas de trabajo futuro

- Conclusiones .....	123
- Líneas de trabajo futuro .....	126

## Presupuesto

- Duración de Fases y Tareas .....	129
- Coste de los Recursos .....	129
- Coste Total .....	131

## Glosario

- Glosario .....	133
------------------	-----

## Anexo: Opción "Choose for Me"

1. Funcionalidad .....	137
2. Criterios y Arquitectura .....	138
3. Diagramas y escenarios .....	144

## Bibliografía

-	Capítulo 1	.....	147
-	Capítulo 2	.....	147
-	Capítulo 3	.....	150
-	Capítulo 4	.....	151



# Índice de Figuras

---

Figura 1.1: Diagrama de Gantt de planificación temporal del proyecto .....	26
Figura 2.1: Arquitectura modular de los Sistemas de Diálogo .....	32
Figura 2.2: Comparativa del Sistema de etiquetas HTML/XML .....	42
Figura 2.3: Arquitectura Modular definida en VoiceXML.....	44
Figura 2.4: Modelo 1 de uso de formularios VoiceXML .....	48
Figura 2.5: Modelo 2 de uso de formularios VoiceXML .....	48
Figura 2.6: Ejemplo de escenario típico de interacción multimodal .....	49
Figura 3.1: Esquema Cliente-Servidor de la aplicación desarrollada .....	53
Figura 3.2: Resumen de comandos de voz disponibles en Opera Voice .....	54
Figura 3.3: Opciones de Configuración de Opera Voice .....	55
Figura 3.4: Comparativa de uso de diferentes SGBD Actuales .....	56
Figura 3.5: Pantalla inicial y Vista de tablas de phpMyAdmin .....	58
Figura 3.6: Menú Inicial de Opciones de MySQL Administrator .....	59
Figura 3.7: Esquema de creación de páginas X+V dinámicas .....	61
Figura 3.8: Declaración y apertura de fichero XHTML resultante.....	61
Figura 3.9: Escritura en fichero XHTML resultante .....	61
Figura 3.10: Almacenamiento de variables pasadas a través de formulario .....	62
Figura 3.11: Ejemplo de uso de variables de Sesión .....	63
Figura 3.12: Eliminación de variables de sesión .....	63
Figura 3.13: Apertura y conexión a la Base de Datos .....	63
Figura 3.14: Ejemplo de consulta a la Base de Datos .....	64
Figura 3.15: Declaración y apertura del fichero de gramática externa .....	64
Figura 3.16: Escritura de la gramática en fichero externo .....	64
Figura 3.17: Declaración de cabecera XHTML .....	65
Figura 3.18: Declaración de la sección HEAD .....	65
Figura 3.19: Ejemplo de la sección de título .....	66
Figura 3.20: Ejemplo de mensaje de voz inicial .....	66
Figura 3.21: Declaración de campo de entrada de voz o "field" .....	66
Figura 3.22: Referencia a gramática en fichero externo .....	67
Figura 3.23: Ejemplo de declaración de gramática interna .....	67
Figura 3.24: Declaración de etiqueta "no match" .....	67
Figura 3.25: Sincronización de entradas de voz y entradas escritas .....	68
Figura 3.26: Ejemplo de declaración de formulario XHTML .....	69
Figura 3.27: Ejemplo de diferentes métodos de entrada de datos .....	69
Figura 3.28: Declaración de botón "submit" para el paso de variables .....	70
Figura 3.29: Carga de la página resultante mediante comando "header".....	70

Figura 3.30: Declaración del fichero de estilos en la cabecera de la sección HEAD....	71
Figura 3.31: Ejemplos de declaración de estilos en fichero externo .....	72
Figura 3.32: Ejemplo de llamadas a diferentes estilos .....	73
Figura 3.33: Vista de la tabla " <i>grammar_ex</i> " .....	74
Figura 3.34: Vista de la tabla " <i>vocabulary_ex</i> " .....	74
Figura 3.35: Vista de la tabla " <i>listening_texts</i> " .....	75
Figura 3.36: Vista de la tabla " <i>listening_ex</i> " .....	75
Figura 3.37: Vista de la tabla " <i>users</i> " .....	76
Figura 3.38: Vista de la tabla " <i>answers</i> " .....	77
Figura 3.39: Vista de la tabla " <i>results</i> " .....	78
Figura 3.40: Relación tablas " <i>users-answers</i> " .....	79
Figura 3.41: Relación tablas " <i>users-results</i> " .....	79
Figura 3.42: Opciones de programación de backups en MySQL Administrator .....	80
Figura 4.1: Página principal o HOME .....	81
Figura 4.2: Modo usuario activo con opción de cierre de sesión .....	82
Figura 4.3: Procedimiento de verificación de usuario activo .....	82
Figura 4.4: Pantalla de aviso de inicio de sesión requerido .....	83
Figura 4.5: Formulario de registro de usuario .....	84
Figura 4.6: Formulario de login de usuario .....	84
Figura 4.7: Validación del nick registrado por el usuario .....	85
Figura 4.8: Validación de la clave registrada por el usuario .....	86
Figura 4.9: Validación de login y password en proceso de inicio de sesión .....	86
Figura 4.10: Caso inicio de sesión satisfactorio .....	87
Figura 4.11: Ejemplo de errores en formulario de registro .....	88
Figura 4.12: Inserción de datos de registro en tabla " <i>users</i> " .....	88
Figura 4.13: Caso registro satisfactorio .....	89
Figura 4.14: Consultas de selección de categorías y niveles existentes .....	90
Figura 4.15: Declaración de gramática interna dinámica .....	91
Figura 4.16: Declaración de formulario de selección de categoría .....	92
Figura 4.17: Vista de formulario de selección de tema y nivel .....	92
Figura 4.18: Consulta de selección de ejercicios de tema y nivel introducido .....	93
Figura 4.19: Vista de página de resolución de ejercicios de gramática .....	94
Figura 4.20: Vista de página de resolución de ejercicios de vocabulario .....	95
Figura 4.21: Sección de lectura del fichero listening .....	95
Figura 4.22: Página de escucha y resolución del listening .....	96
Figura 4.23: Inserción de respuestas del usuario en tabla " <i>answers</i> " .....	97
Figura 4.24: Análisis de respuestas para escritura de página de corrección .....	98
Figura 4.25: Vista de inserción de respuesta en tabla " <i>answers</i> " .....	99
Figura 4.26: Esquema de creación-navegación de páginas .....	100
Figura 4.27: Esquema de interacción con bases de datos .....	101
Figura 4.28: Consulta de extracción de diferentes categorías existentes.....	103
Figura 4.29: Vista de selección de tema para administración de contenidos .....	103



Figura 4.30: Acceso a página de edición de ejercicio .....	104
Figura 4.31: Acceso a página de borrado de ejercicio .....	105
Figura 4.32: Vista de página principal de Administración de Contenidos.....	105
Figura 4.33: Formulario de creación de ejercicio en nuevo tema .....	106
Figura 4.34: Inserción de datos en tabla de ejercicios .....	107
Figura 4.35: Ejemplo de errores en formulario de creación de ejercicio .....	107
Figura 4.36: Consulta de actualización de datos de ejercicio .....	108
Figura 4.37: Consulta de borrado de ejercicio .....	108
Figura 4.38: Transición de páginas en proceso de edición de ejercicio .....	109
Figura 4.39: Esquema de creación-navegación de páginas .....	110
Figura 4.40: Esquema de interacción con bases de datos .....	111
Figura 4.41: Consulta y conteo de ejercicios existentes por categoría y nivel .....	113
Figura 4.42: Cálculo del total de ejercicios existentes .....	113
Figura 4.43: Consulta y conteo de ejercicios realizados por categoría y nivel .....	114
Figura 4.44: Cálculo del número total de ejercicios realizados de gramática .....	114
Figura 4.45: Cálculo del número total de ejercicios realizados .....	115
Figura 4.46: Consulta ejercicios correctos por familia y nivel .....	115
Figura 4.47: Cálculo del número total de ejercicios correctos .....	115
Figura 4.48: Vista de la primera sección de página de resultados .....	116
Figura 4.49: Vista primera inserción en tabla de resultados .....	117
Figura 4.50: Vista segunda inserción en tabla de resultados .....	117
Figura 4.51: Vista de porcentaje realizado y puntuación .....	118
Figura 4.52: Vista de gráficas comparativas de ejercicios correctos .....	118
Figura 4.53: Consulta para obtención de ejercicios realizados seleccionados .....	119
Figura 4.54: Vista de ejercicios seleccionados realizados .....	120
Figura 4.55: Esquema de creación-navegación de páginas .....	120
Figura 4.56: Esquema de interacción con bases de datos .....	121
Figura Anexo.1: Detalle de selección de la opción " <i>choose for me</i> " .....	137
Figura Anexo.2: Consulta de ejercicios realizados para cierta familia y nivel .....	138
Figura Anexo.3: Consulta ejercicios existentes del nivel del usuario .....	139
Figura Anexo.4: Acción llevada a cabo en función de ejercicios pendientes .....	140
Figura Anexo.5: Consulta de ejercicios incorrectos de cierta familia y nivel .....	140
Figura Anexo.6: Análisis de casos en función de los ejercicios incorrectos .....	141
Figura Anexo.7: Consulta para obtener los textos de Listening ya realizados por un usuario .....	143
Figura Anexo.8: Consulta para obtener todos los textos de cierto nivel .....	143
Figura Anexo.9: Diagrama de flujo " <i>choose_for_me</i> " gramática y vocabulario .....	144
Figura Anexo.10: Diagrama de flujo " <i>choose_for_me</i> " caso listening .....	145
Figura Anexo.11: Vista del mensaje resultante (Caso 1) .....	145
Figura Anexo.12: Vista del mensaje resultante (Caso 2) .....	146
Figura Anexo.13: Vista del mensaje resultante (Caso 3) .....	146
Figura Anexo.14: Vista del mensaje resultante (Caso 4) .....	146



# Capítulo 1: Introducción y objetivos

---

## 1.1 Introducción

Una de las formas de comunicación más eficientes y naturales que poseemos es el habla. Para la plena integración del mundo tecnológico en la sociedad actual supone un objetivo primordial el dotar a una maquina de la capacidad de producir y comprender mensajes orales. La emulación artificial del diálogo utilizando una maquina como interlocutor supone un reto de indudable interés científico, social y económico [1] [2].

Esencialmente los sistemas de diálogo actuales se utilizan para obtener y gestionar información de diversos tipos. Por ejemplo horarios y precios de transportes, información meteorológica o ciudadana sobre horarios de servicios, sobre tramites en la administración o para llevar a cabo transacciones (comercio electrónico, venta de entradas y billetes, banca electrónica, etc.). Otra aplicación sumamente relevante en la actualidad es la traducción automática del habla, de modo que combinando un sistema de diálogo con las tecnologías propias de la traducción automática se pueda llegar a conseguir que dos personas mantengan una conversación expresándose cada uno en su lengua [3].

Un sistema de diálogo ideal reconocería el habla espontanea, comprendería enunciados sin restricciones de contenido, proporcionaría respuestas con sentido, gramaticalmente bien formadas y pragmáticamente adecuadas, respondería con voz completamente natural y seria multimodal. Sin embargo los sistemas de diálogo que podemos utilizar hoy en día están sujetos a las limitaciones del reconocimiento del habla. En este sentido, no se dispone aún de reconocedores que puedan procesar con un cien por cien de fiabilidad la variación propia del habla completamente espontanea, la multiplicidad de realizaciones fonéticas de locutores de diversa procedencia geográfica y social así como las interferencias producidas por el entorno cuando el sistema tiene que utilizarse, por ejemplo, dentro de un vehículo en marcha o hablando por un teléfono móvil en plena calle [4].

Las interfaces basadas exclusivamente en el habla presentan diversos inconvenientes que pueden degradar considerablemente la interacción usuario-sistema. Uno de estos inconvenientes es el comentado anteriormente relacionado con el reconocimiento, pues incluso en dominios muy restringidos y utilizando vocabularios muy limitados, las tasas de acierto nunca son del 100%. Otro problema, observado especialmente en sistemas de información que devuelven muchos datos, está relacionado con la forma en que estos sistemas proporcionan la información a los usuarios. Se ha comprobado que frecuentemente tienen problemas para entender y anotar en papel toda la información generada [4].

Con el fin de intentar resolver estos problemas, en los últimos años ha crecido notablemente el interés por la utilización conjunta del habla y otras modalidades de comunicación en los sistemas de diálogo, dando lugar al desarrollo de los denominados **sistemas de diálogo multimodales**.

El objetivo fundamental de un sistema de diálogo multimodal (*multimodal dialogue system*) [5] es superar las limitaciones de la interacción basada exclusivamente en el habla. En una interacción multimodal el usuario no está restringido a utilizar el habla como único canal de comunicación, sino que puede utilizar varios dispositivos de entrada, como por ejemplo un teclado, un ratón, un micrófono, una cámara, una pantalla sensible al tacto, una PDA, etc. Asimismo, el sistema multimodal puede utilizar diversos canales de salida para proporcionar información al usuario como por ejemplo, voz, texto, gráficos o imágenes, con objeto de estimular varios sentidos del usuario de forma simultánea y así facilitar la recepción de la Información. Algunos sistemas de diálogo multimodal permiten incluso que los usuarios puedan elegir entre las diversas modalidades de entrada para llevar a cabo la interacción, permitiendo así una cierta adaptación a las condiciones ambientales de luz, ruido, etc. Esta ventaja permite además que personas con determinadas discapacidades (p. e. personas invidentes) puedan usar estos sistemas mediante alguna de las modalidades de interacción disponibles [3].

## 1.2 Objetivos

A partir de la descripción de sistema de diálogo multimodal concretada en el punto anterior, y con la idea y enfoque buscado que se explicó en el resumen inicial, ya puede entenderse que el objetivo formal del proyecto consiste en el desarrollo e implementación de un sistema de diálogo multimodal, con el que el usuario pueda interactuar de forma oral y/o escrita con la aplicación conllevando dicho diálogo un valor añadido en el objetivo que se busca, la práctica y evaluación de conocimientos en el ejercicio de aprendizaje del idioma inglés.

Las funciones de la aplicación desarrollada pueden quedar detalladas en función de los siguientes módulos:

- **Módulo o función de práctica:** Enfocado al usuario final. Será donde acceda con el fin de realizar los 3 tipos de ejercicios existentes:
  - *Ejercicios de gramática:* Consistentes en el clásico ejercicio de completar frases con el correspondiente verbo, adjetivo, adverbio, artículo o cualquier otro elemento gramatical perteneciente al tema que inicialmente haya sido seleccionado, junto con el nivel de dificultad, por el usuario.
  - *Ejercicios de vocabulario:* Tras escoger un determinado tema y nivel, el usuario debe dictar o escribir la palabras que definen un conjunto de imágenes que irán apareciendo de manera sucesiva relacionadas con el tema seleccionado.
  - *Ejercicios de escucha:* El sistema de voz leerá un texto seleccionado por el usuario. A continuación le serán formuladas una serie de preguntas tipo test relacionadas con lo que ha escuchado cuyas respuestas permitirán al usuario y al sistema analizar su capacidad de comprensión del texto.
- **Módulo de evaluación de resultados:** El usuario podrá acceder a esta funcionalidad con el fin de repasar las respuestas dadas a cualquiera de los ejercicios realizados junto con la corrección correspondiente en caso de haber fallado. Al hacer uso del módulo el sistema realiza además una evaluación de resultados ofreciendo

porcentajes, puntuaciones y comparativas al usuario con el fin de guiarle en el proceso de aprendizaje.

- **Módulo de gestión de contenidos:** Esta enfocado a la administración de los contenidos prácticos de la aplicación y por tanto queda reservado a usuarios restringidos (profesorado, administradores de la aplicación, etc.). Concretamente las funciones a realizar son la creación de nuevos ejercicios y la edición o borrado de ejercicios existentes.

El objetivo principal que se plantea a la hora de abordar un proyecto basado en un sistema de diálogo multimodal es la posibilidad de estudiar, aprender y desarrollar una tecnología de plena actualidad y cuyo potencial y posibilidades de aplicación y servicio se encuentran todavía en fase de explotación e investigación.

Pese a que prácticamente cualquier servicio o aplicación actual es factible programarla con el objetivo de ser manejada mediante un diálogo cliente-equipo, no siempre esto presentaría utilidad o ventajas reales. Lo que se ha buscado con esta aplicación en concreto es que además dicha interacción mediante la voz si suponga ventajas o valor añadido frente a los métodos tradicionales empleados con el mismo fin. No cabe duda de que tener la posibilidad de expresarse de manera oral en un idioma cuando se está practicando para su correcto aprendizaje supone una ventaja muy interesante y provechosa.

Para poder llevar a cabo la implementación de la aplicación y la resolución del proyecto en general, se definieron una serie de objetivos parciales que pueden quedar resumidos de esta manera:

- Llevar a cabo un estudio de las tecnologías VoiceXML y XHTML+Voice, referidas al desarrollo de interfaces orales unimodales y multimodales, con el fin de mostrar las posibilidades que ofrece, explicar su funcionamiento y poder decidir qué dirección y qué finalidad darle al proyecto.
- Desarrollar una aplicación que permita al usuario resolver una serie de ejercicios típicos orientados al aprendizaje del idioma Inglés, ofreciendo mediante el uso de la tecnología XHTML+Voice la posibilidad de que puedan ser resueltos mediante la voz, de forma escrita, o usando simultáneamente ambos métodos. Ofrecer al usuario además el resumen de todos sus ejercicios junto con puntuación, porcentajes, informes y evaluaciones que le permitan hacer un seguimiento de su evolución y le guíen en el proceso de estudio.

- Optimizar la gestión de contenidos y estructura del código mediante la implementación completa de la aplicación haciendo uso de páginas web dinámicas.
- Facilitar la edición y ampliación continua de ejercicios mediante el desarrollo de un módulo orientado a la administración sencilla de los contenidos prácticos de la aplicación.
- Dotar a la aplicación de un diseño e interfaz intuitivo y sencillo que permita que cualquier usuario, pese a no estar familiarizado con el uso y navegación a través de la web, pueda utilizar la aplicación sin problemas y de forma cómoda y productiva.

### 1.3 Fases de Desarrollo

El desarrollo del presente proyecto puede quedar dividido en las siguientes fases y subfases:

- **Fase1. Estudio de las herramientas:**
  - *Estudio* de los sistemas de diálogo, inicialmente como concepto general y a continuación profundizando en el manejo y funciones de los lenguajes de programación VoiceXML y XHTML + Voice, empleados para el desarrollo de la aplicación.
  - *Repaso* del lenguaje de programación XHTML y *estudio* del lenguaje PHP. Repaso de los conceptos aprendidos durante la carrera acerca del diseño y estructura de bases de datos y de las funciones de creación de tablas y consultas a través de MySQL.

Es importante destacar que, por supuesto, esta fase de estudio cubre la totalidad del proyecto. Durante las fases de planificación y desarrollo del sistema es necesario estar continuamente actualizando información, repasando conceptos y resolviendo dudas.

- **Fase2: Planificación y análisis de requisitos:**

Una vez que se conoce el funcionamiento y las posibilidades de las herramientas implicadas, es el momento de decidir qué es lo que se va a realizar con ellas, es decir, planificar la funcionalidad y uso de la aplicación desarrollada.

Con la idea anterior bien establecida es el momento de definir los requisitos concretos necesarios para el correcto desarrollo:

- Diseño de la estructura que presentará la aplicación, dividiéndola en módulos y submódulos funcionales y estableciendo el modo de navegación y la interfaz más adecuada.
- Estudio de la estructura y los requisitos a nivel de bases de datos. Para ello es necesario planificar que información vamos a necesitar y cuando y para que será solicitada. Conociendo estos requisitos ya se puede elaborar un modelo relacional de la Base de Datos.
- Estudio y planificación de los tipos de ejercicios y contenidos que presentara la aplicación y que más se ajusten a los objetivos y funciones de esta. Es necesario investigar de igual modo las fuentes de donde podrán ser extraídos dichos contenidos.

- **Fase3: Desarrollo:**

Implementación y desarrollo de todos los módulos y submódulos definidos anteriormente así como de la base de datos descrita. Una vez la estructura y programación de aplicación queda completada será necesario además:

- Introducir todos los contenidos. La inserción de los distintos ejercicios y usuarios puede suponer una tarea tediosa y muy repetitiva por lo que conviene definir algún procedimiento que pueda automatizar o hacer más sencilla, al menos en parte, esta labor. Para ello se ha hecho uso del módulo de administración implementado ya que facilita mucho las tareas con respecto a lo que habría sido la inserción y edición de contenidos mediante comandos de gestión de la base de datos. Esto además ha servido para realizar una completa y continua evaluación de dicho módulo.



- Dotar a la interfaz del aspecto y estructura que asegure que la navegación por los distintos módulos y contenidos sea la más adecuada, sencilla y cómoda.
- Estudio y realización del conjunto de pruebas unitarias que aseguren el correcto funcionamiento de todos los módulos de manera independiente y a continuación de pruebas de sistema completo que analicen la correcta interacción de todos los módulos anteriores. Puesto que se trata de un sistema multimodal, se deben establecer pruebas diferenciadas para el testeo del manejo mediante voz y de manera escrita.
- **Fase4: Documentación y presentación:**
  - Organizar y resumir toda la información recopilada en cada fase para estructurar y posteriormente redactar de manera clara y detallada la presente memoria explicativa del proyecto. Diseñar y recopilar imágenes y diagramas que faciliten la comprensión del texto. Una vez redactado revisar el documento completo llevando a cabo las correcciones pertinentes.
  - Estructurar y recopilar el contenido de la presentación lo que incluye la grabación de videos de ejemplo y muestra.

La Figura 1.1 incluye el desglose de las fases en las tareas más significativas representando la duración aproximada de tiempo en días que se han necesitado para completarlas. A partir de estos datos se genera un diagrama Gantt que muestra de manera gráfica los tiempos y el orden en que se han llevado a cabo. Es importante destacar la dificultad de establecer el número de horas de trabajo diario puesto que ha sido muy variable. El motivo es que prácticamente la totalidad de la realización del proyecto ha tenido que compaginarse con un empleo de horario dinámico y, principalmente durante la etapa de desarrollo, hubo algunos periodos de varios días en los que se le pudo dedicar menos tiempo o ninguno. Como consecuencia los plazos de finalización se fueron alargando quedando sobrepasado el tiempo inicial estimado para completarlo.



## 1.4 Medios Empleados

Los medios empleados para la realización del presente proyecto final de carrera quedan divididos de la siguiente manera:

- **Dispositivos hardware:**
  - Ordenador portátil.
  - Auriculares Estéreo.
  - Micrófono portátil externo.
  - Periféricos habituales (teclado, ratón, pantalla).
  - Disco duro externo para almacenaje y copias de seguridad.
  
- **Dispositivos software:**
  - Navegador Opera 10.10 (con complemento Opera Voice instalado y debidamente configurado).
  - Editor de programación HateML (Para el código XHTML, XHTML+Voice y PHP).
  - Editor de texto Notepad++.
  - Microsoft Office 2007 (Word y Powerpoint).
  - Foxit PDF reader.
  - MySQL 5.5.15 (junto con MySQL Administrator para realizar copias de seguridad).
  - Servidor Web Apache 2.2.11
  - PHP 5.2.9-2
  - phpMyAdmin 3.4.5 (Para la creación y gestión de la Base de Datos).
  - Rapid CSS (Editor de Hojas de Estilo).
  - PicPick (Capturador de pantalla y editor de imágenes).

La documentación empleada durante la realización del proyecto, tanto para el estudio de los sistemas de diálogo como de los lenguajes de programación aplicados, ha sido extraída en su mayoría de diversas fuentes de internet (tutoriales, manuales, foros). Toda la documentación y sus fuentes se encuentra detallada en el apartado "Bibliografía" de la presente memoria.

## 1.5 Estructura de la Memoria

La memoria está dividida en cinco capítulos. Se incluye después de estos el glosario de términos, un apartado de anexo y las fuentes y bibliografía de toda la documentación empleada. A continuación se describe brevemente el contenido de cada capítulo.

- **Capítulo 1: Introducción y Objetivos.**

Esta primera parte explica y resume el propósito y los objetivos globales de la totalidad del proyecto así como una pequeña introducción a conceptos referentes a los sistemas de diálogo. Incluye además las fases en las que queda dividido su desarrollo y la planificación de estas fases en el tiempo, los medios hardware y software que han sido empleados y la presente estructura de la memoria.

- **Capítulo 2: Estado del arte.**

Este capítulo describe qué se entiende por sistemas de diálogo, sus principales aplicaciones y evolución a lo largo de las últimas décadas. A continuación, el capítulo analiza las características concretas de las tecnologías empleadas VoiceXML y XHTML+Voice. Se detalla su sintaxis, funcionamiento y funcionalidad general de manera que pueda ser entendida la aplicación concreta de dicha tecnología en el sistema, que es analizada con más detalle en los dos capítulos que le suceden.

- **Capítulo 3: Descripción general de la aplicación desarrollada.**

Este capítulo proporciona una visión global de la aplicación desarrollada. Para ello se repasa la funcionalidad, los requisitos y los procedimientos de uso común en todos los módulos que la componen. Se explica la arquitectura general de las páginas creadas y se presenta el esquema de la base de datos empleada. De esta manera queda entendido el comportamiento, estructura e integración general del sistema con el fin de poder revisar más a fondo y por separado cada módulo en el siguiente capítulo.

- **Capítulo 4: Descripción detallada de los módulos del sistema.**

Ofrece una visión en profundidad de cada uno de los módulos que conforman la totalidad de la aplicación. Los módulos de realización de los diferentes tipos de ejercicios, el módulo de repaso y evaluación de los resultados así como la herramienta de edición y gestión de contenidos. Para cada uno de estos se realiza una descripción detallada de su estructura, funcionalidad y escenarios de uso.

- **Conclusiones y trabajo futuro.**

Resume las principales ideas y conclusiones a las que se ha llegado a partir de la implementación y documentación del proyecto completo, junto con un análisis de posibles ampliaciones y mejoras de la aplicación desarrollada. Se analizan las posibilidades que tienen las tecnologías empleadas en la actualidad y de cara al futuro.

**Anexo:** Incluye contenido e información adicional sobre algún concepto que ha sido referenciado en otro apartado del texto.

**Glosario:** Se definen los conceptos más importantes tratados a lo largo del proyecto que han sido mencionados en la presente documentación. El objetivo es ayudar al lector a comprender mejor el significado de éstos términos clave.

**Bibliografía:** Incluye las citas bibliográficas, fuentes de información y procedencia de toda la documentación cuya consulta ha sido necesaria para la realización de la aplicación y la presente memoria.



## Capítulo 2: Estado del Arte

---

Se amplía en este segundo capítulo el concepto de Sistema de Diálogo que ya se introdujo en el capítulo anterior. Serán explicadas las características generales en que se basan así como su evolución a lo largo de las últimas décadas. A continuación se analizarán las tecnologías VoiceXML y XHTML+Voice con el fin de que pueda entenderse de manera clara el Sistema de Diálogo desarrollado para la aplicación que se describe en el presente proyecto.

### 2.1 Introducción

Como ya quedó definido en el capítulo anterior, los sistemas de diálogo o sistemas conversacionales son una tecnología concebida para facilitar la interacción natural mediante el habla entre una persona y un ordenador. Constituyen una interfaz o intermediario entre un usuario y un sistema informático, que tiene la ventaja de no requerir del uso de periféricos (pantalla, ratón, teclado,...) y de recurrir, en cambio, al medio de comunicación propio de los seres humanos. Por ello, suelen emplearse para servicios que se ofrecen a través de teléfono, aunque también se usan en todos aquellos casos en los que la lengua oral permite llevar a cabo una determinada petición, tarea o función [4].

Se puede denominar entonces como sistema de diálogo a todos los procesos necesarios para la realización de un sistema automático capaz de emular a un ser humano en un diálogo con otra persona. Las fases de su funcionamiento y componentes podrían quedar englobados tal y como muestra la Figura 2.1.

En primer lugar la finalidad del reconocedor de habla es procesar la voz del usuario y transformarla en una secuencia de palabras reconocidas en forma de texto. Dicha secuencia constituye la entrada del módulo de análisis lingüístico, cuya finalidad es obtener la representación semántica (significado) de la frase reconocida.

La representación semántica obtenida constituye la entrada del módulo de gestión del diálogo, cuya finalidad consiste en determinar qué acción debe realizar el sistema en cada momento. Puede decirse que este es el módulo fundamental del sistema, pues trata de lograr que la interacción con el usuario sea lo más cómoda e "inteligente" posible. Para conseguir este objetivo, el módulo de gestión del diálogo suele realizar confirmaciones de los datos obtenidos del usuario, iniciar subdiálogos de corrección y generar expectativas respecto a las frases más probables del usuario en un momento dado.

Una vez el módulo de gestión del diálogo ha decidido la acción que debe realizar el sistema, el módulo de generación de respuestas construye la respuesta del sistema en formato de texto, la cual constituye la entrada del sintetizador de voz para generar la respuesta oral del sistema [3].

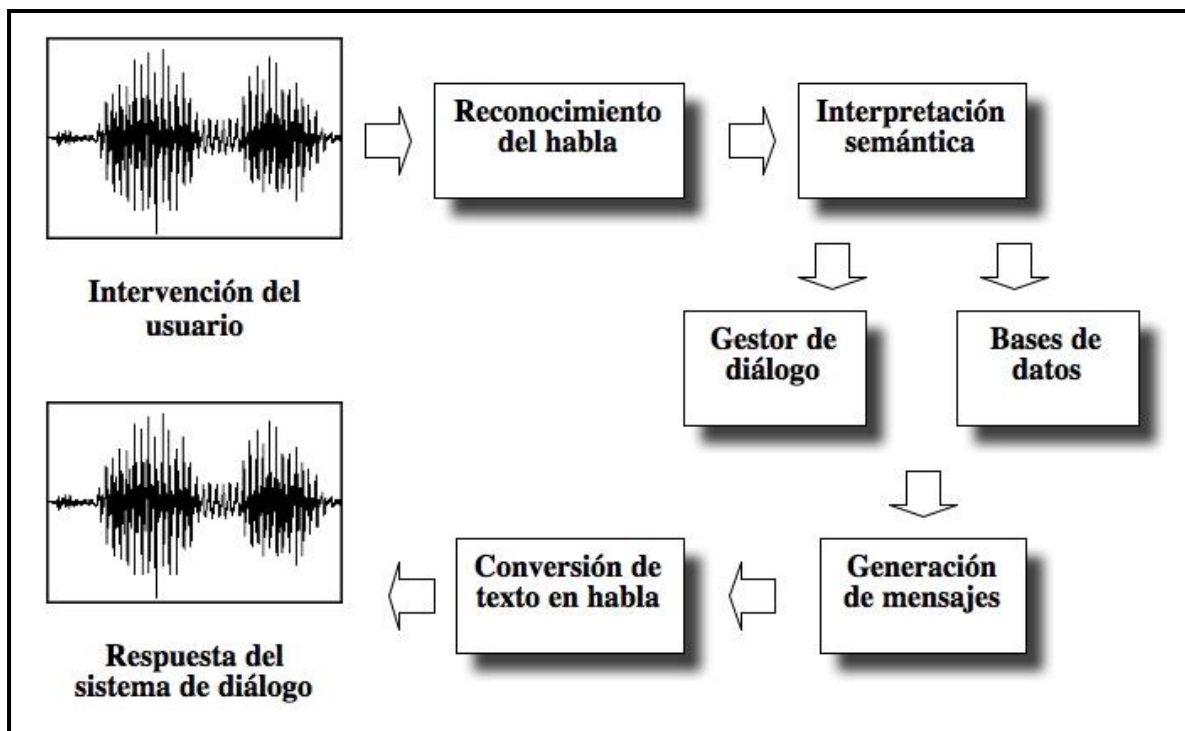


Figura 2.1: Arquitectura modular de los Sistemas de Diálogo.



Aunque en teoría cualquier tarea en la que se interactúe con un ordenador puede utilizar el reconocimiento de voz, actualmente las siguientes aplicaciones son las más comunes [8]:

- **Dictado automático:** El dictado automático fue, hacia mediados de la primera década del 2000, el uso más común de las tecnologías de reconocimiento de voz. En algunos casos, como en el dictado de recetas médicas y diagnósticos o el dictado de textos legales, se usan corpus especiales para incrementar la precisión del sistema.
- **Control por comandos:** Al sistema de reconocimiento de habla diseñado para dar órdenes a un computador se le denomina control por comandos. Estos sistemas reconocen un vocabulario reducido y de uso común, lo que incrementa su rendimiento. Más adelante se estudiará el funcionamiento del *plug-in* de voz del Navegador Opera que es un buen ejemplo de éste tipo de sistema.
- **Telefonía:** Con la llegada de los teléfonos inteligentes (*smartphones*), el número de aplicaciones desarrolladas que permiten utilizar la voz como fuente de entrada de datos se ha vuelto innumerable: traductores, aplicaciones basadas en GPS e incluso programas que permiten realizar prácticamente cualquier función del teléfono dictando la orden a través de la voz. En cuanto a la telefonía analógica convencional el reconocimiento de voz se usa fundamentalmente en centralitas que permiten a los usuarios ejecutar comandos mediante el habla, en lugar de pulsar tonos. También aceptan la entrada de números o palabras por voz con el fin de navegar por un menú de opciones.
- **Sistemas portátiles:** Los sistemas portátiles de pequeño tamaño, como los tablets-PC o los teléfonos móviles mencionados anteriormente, tienen unas restricciones muy concretas de tamaño y forma, así que el habla es una solución natural para introducir datos en estos dispositivos.
- **Sistemas diseñados para discapacitados:** Los sistemas de reconocimiento de voz pueden ser útiles para personas con discapacidades que les impidan teclear con fluidez, así como para personas con problemas visuales o auditivos, que pueden usarlos para obtener texto escrito a partir de habla lo que les permitiría, por ejemplo, que pudiesen recibir llamadas telefónicas.

La dificultad que plantea la implementación de un sistema de diálogo se encuentra principalmente en la necesidad de coordinar y analizar un conjunto de características intrínsecas al lenguaje oral y que provienen a su vez de diferentes campos de conocimiento. Propiedades acústicas, fonéticas, fonológicas, léxicas, sintácticas y semánticas que, además pueden encontrarse sometidas a ciertas ambigüedades, incertidumbres y errores inevitables que impiden en muchas ocasiones obtener una interpretación aceptable del mensaje acústico recibido.

## 2.2 Características de los Sistemas de Diálogo respecto al reconocimiento automático del habla

Los sistemas de diálogo pueden clasificarse atendiendo a diferentes criterios [6] [7], en cuanto a las características del reconocimiento del habla podemos distinguir entre:

- **Entrenabilidad:** Determina si el sistema necesita un entrenamiento previo antes de empezar a usarse.
- **Dependencia del hablante:** Indica si el sistema está especialmente preparado para la dicción y la forma de hablar de algún individuo general, de un equipo de personas en particular o de forma más general se adapta a todo usuario. Los sistemas dependientes del hablante son generalmente más fáciles de desarrollar, más baratos de comprar y más precisos que los sistemas independientes del hablante. Por contra no tan flexibles como estos últimos.
- **Continuidad:** Este parámetro determina la velocidad de encadenamiento entre las palabras que puede reconocer el programa. Se trata de un valor muy importante, porque aunque en situaciones ideales la pausa entre las palabras es muy amplia, en el lenguaje diario tendemos a encadenarlas. El lenguaje continuo es más difícil de tratar debido a la variedad de efectos. Primero, es difícil encontrar el comienzo y el final de las palabras. Otro problema es la coarticulación. La generación de cada fonema se ve afectada por la generación de los fonemas adyacentes, y de modo parecido, el comienzo y final de las palabras se ven afectados por las palabras que les preceden y suceden. El reconocimiento del lenguaje continuo también se ve condicionado por la frecuencia de habla, un discurso rápido suele ser más difícil de reconocer con éxito.

Un sistema de reconocimiento discreto funciona con palabras simples, necesitando de una pausa entre la dicción de cada palabra. Esta es la forma más sencilla de reconocimiento para llevar a cabo ya que los puntos de finalización son más fáciles de encontrar y la pronunciación de una palabra no afecta a las demás. De este modo y ya que las ocurrencias de las palabras son más consistentes, es más fácil reconocerlas.

- **Robustez:** determina si el sistema está diseñado para usarse con señales poco ruidosas o, por el contrario, puede funcionar aceptablemente en condiciones poco nítidas, ya sea ruido de fondo, ruido procedente del canal o la presencia de voces de otras personas.
- **Tamaño del diccionario:** Otro factor muy importante es el tamaño del vocabulario aceptado por nuestro sistema. A medida que este crece, aumenta obviamente la dificultad para el reconocimiento o surgen nuevos problemas, por ejemplo, aumenta la posibilidad de que se confunda una palabra con otra o el incremento del tiempo de localización de esa palabra.

Cabe mencionar en este punto el concepto de **multimodalidad** como otra característica con la que poder clasificar un sistema de diálogo. Como quedó explicado en la introducción del capítulo anterior, este concepto se refiere a la posibilidad de que el sistema no se base exclusivamente en el habla como método de manejo sino que permita la entrada y/o salida de datos o por vías alternativas a la voz.

## 2.3 Evolución e Historia de los Sistemas de Diálogo

La tecnología a lo largo de los años ha evolucionado de forma significativa en el objetivo de lograr que el esquema de comunicación de los seres humanos consiga una representación efectiva a través de la conjunción de una aplicación informática y su desarrollo en una interfaz sencilla de manejar, para que el usuario final pueda llevar a cabo las tareas cotidianas desde su casa, simplemente utilizando la voz como plataforma para la consecución de su objetivo: la compra de una entrada o billete de avión, la redacción de una carta, o bien, la consulta de los datos bancarios [10].

A continuación se recogen las fechas más significativas, así como las distintas investigaciones desarrolladas a lo largo de las últimas décadas en el campo de los sistemas de diálogo y de reconocimiento del habla.

**AT&T Bell Laboratories, 1910:** Construye la primera máquina capaz de reconocer voz (basada en plantillas) de los 10 dígitos del inglés. Requería un extenso reajuste a la voz de una persona, pero una vez logrado tenía un 99% de aciertos. Por lo tanto, surge la esperanza de que el reconocimiento de voz resulte simple y directo [10].

No resulta sencillo encontrar más referencias hasta la década de los 60, en la que la mayoría de los investigadores reconoce que era un proceso mucho más intrincado y sutil de lo que habían anticipado. Como consecuencia, comienzan a reducir los alcances y se enfocan a sistemas más específicos que presentan las siguientes características [9] [10]:

- *Dependientes del Locutor.*
- *Flujo discreto de habla* (con espacios bien delimitados, haciendo pausas entre palabras).
- *Vocabulario pequeño* (menor o igual a 50 palabras)

Estos sistemas empiezan a incorporar técnicas de normalización del tiempo con el fin de minimizar la diferencia en velocidad del habla. Por el momento no aspiran a conseguir una exactitud perfecta en el reconocimiento [10]. Cabe mencionar los siguientes proyectos referidos a esta década:

**ELIZA [14]:** Se trata de uno de los sistemas más conocidos que simula una conversación. Concretamente, se basaba en la simulación de un psicoterapeuta, usando el procesamiento y sustitución de palabras clave en frases modelo. Aunque las conversaciones podían parecer reales, en realidad las capacidades de ELIZA eran limitadas, y estaban basadas en el uso de palabras clave y respuestas asociadas.

**BASEBALL** [12]: Era un sistema de “pregunta-respuesta” capaz de contestar a cuestiones relativas a fechas, lugares, equipos y puntuaciones de partidos de beisbol. Para ilustrar la naturaleza ad hoc de parte del procesamiento del lenguaje, el significado de la palabra “quién” en el diccionario del sistema, era dado como “Equipo = ?”, un significado específico del dominio que no podría aplicarse a otros dominios.

**STUDENT** [13]: Fue un programa capaz de solucionar problemas de álgebra planteados con un lenguaje natural. El programa convertía el lenguaje natural en un conjunto de ecuaciones, desglosándolo en patrones simples y buscando palabras y frases que podían ser sustituidos por variables y expresiones aritméticas. Este sistema demostró algunas de las características más interesantes que vaticinaron desarrollos posteriores, como por ejemplo, que el sistema fuera capaz de tratar toda la conversación en conjunto, y no frases aisladas. Otra característica era que el sistema usaba un almacenamiento de conocimiento general, obteniendo más información del usuario.

Ambos sistemas se basaban generalmente en técnicas que podían ser aplicadas con éxito dentro de su dominio en cuestión, pero sin embargo no podían generalizarse a otros dominios. Conceptos como el procesamiento del lenguaje natural y la inteligencia artificial no serían aplicados hasta el transcurso de las dos décadas siguientes. En la década de los 70 las empresas IBM y CMV trabajan en el reconocimiento de voz continuo.

**Threshold Technology Inc , 1970-1980** [10]: Nace el VIP100, utilizaba un vocabulario pequeño, dependiente del locutor, y reconocía palabras discretas. Gana el U.S. National Award en 1972.

Nace el interés de ARPA, (Agencia de Proyectos de Investigación Avanzada), un organismo dependiente del Departamento de Defensa de los Estados Unidos de América. Se precipita la época de la inteligencia artificial. El proyecto financiado por esta institución buscó el reconocimiento del habla continua y de la ampliación del vocabulario [9] [10].

Este proyecto impulsa a los investigadores para que se centren en el entendimiento del habla. Los sistemas empiezan a incorporar los siguientes módulos:

- Análisis léxico (conocimiento léxico).
- Análisis sintáctico (estructura de palabras).
- Análisis semántico (significado, sentido o interpretación de signos lingüísticos).
- Análisis pragmático.

Asimismo, durante esta década comienza a implantarse el procesamiento de consultas de lenguaje natural, traduciéndolas a un lenguaje formal de consulta a base de datos. Este avance constituye uno de los más importantes éxitos del procesamiento del lenguaje natural. La década de los 70 deja los siguientes proyectos destacables:

**PARRY** [16]: Similarmente a ELIZA, simulaba a un paciente paranoico, y en ciertas pruebas los psiquiatras fueron incapaces de distinguir entre locuciones realizadas por PARRY y paranoicos reales. Disponía de un conjunto de más de 6000 patrones contra los que se contrastaba cualquier entrada. Mientras que ELIZA no tomaba la iniciativa en la conversación, sino que reflejaba lo que había dicho el usuario, PARRY era capaz de mantener una conversación ya que siempre tenía algo que decir.

**SHRDLU** [15]: Sistema desarrollado en el Instituto de Tecnología de Massachusetts (MIT), a principios de los años 70, fue un intento de modelar los procesos de entendimiento del lenguaje natural en un ordenador. El sistema operaba dentro de un dominio muy limitado, un pequeño espacio que contenía una caja, una mesa, bloques y pirámides que podían manipularse según las órdenes del usuario. Aparte, el sistema podía responder a preguntas, así como “aprender” a partir de las interacciones con el usuario.

El sistema tenía que procesar la entrada del usuario (compleja en algunos casos), no sólo lingüísticamente sino también en relación al mundo simulado en el que la posición de los objetos variaba continuamente como resultado de las órdenes del usuario. Los objetos podían referenciarse usando pronombres u otras expresiones, aunque a veces las referencias podían ser ambiguas; por ejemplo, “Coge la pirámide” o “Ponlo en la caja”, ya que puede haber varias pirámides y varias cajas. Para interactuar dentro del mundo de los bloques, SHRDLU usaba una combinación de análisis sintáctico (usando la gramática inglesa para asignar una estructura sintáctica a la entrada del usuario), semántico (para rechazar frases sin significado), y pragmático (para llevar la cuenta de los objetos del dominio).

En la **década de los 80** será **IBM** la empresa puntera que remonta el vuelo. Cabe recordar que inventa el ordenador personal en agosto de 1981. Los trabajos posteriores se basan en el desarrollo de modelos robustos, capaces de manejar locuciones continuas con amplio vocabulario (mayor de 1000 palabras), y superando dificultades como el habla a través de teléfono o conversaciones con ruido de fondo. Como resultado, la investigación reciente en la tecnología del diálogo hablado ha reunido conceptos del Procesamiento del Lenguaje Natural con los de la Inteligencia Artificial, que durante las décadas previas se habían desarrollado de forma independiente [10].

Durante estos años se volvió a la Lingüística como base teórica para el desarrollo de estos sistemas, además de retomarse modelos descartados previamente, como los modelos de estados finitos. Las teorías lingüísticas de esta época se orientan específicamente para su implementación informática, debido a su inspiración en lenguajes de programación.

Una de las principales tendencias durante la **década de los 90** estuvo relacionada con la definición de lenguajes estándar para el desarrollo de sistemas de diálogo orales. A finales de 1999, el W3C (World Wide Web Consortium, consorcio internacional dedicado a la creación de estándares y pautas para el desarrollo Web) presentó los primeros estudios de requisitos para navegadores web que supusieron la base de lenguajes de etiquetas, como VoiceXML, para el desarrollo de sistemas de diálogo oral. Cabe mencionar los siguientes proyectos destacables durante la década de los 90 y la primera década del 2000:

Un grupo de investigadores del **MIT** desarrolla **GALAXY** [11], una arquitectura de sistemas de diálogo adaptable a diferentes tareas. Su finalidad es el desarrollo de sistemas de diálogo, orientados a cumplir objetivos, con posibilidad de iniciativa mixta, y aplicados a tareas reales, de uso cotidiano. Ejemplos de su arquitectura de sistemas son **VOYAGER** [11], sistema de información para viajeros, **PEGASUS** [11], sistema de acceso al sistema de reservas on-line de SABRE, o **JUPITER** [11], sistema de información meteorológica.

Los investigadores de la **CMU** desarrollan el sistema **Carnegie Mellon Communicator** [11], que permite obtener información de itinerarios complejos que incluyen múltiples reservas en vuelos, hoteles y alquiler de coches. El sistema gestiona el diálogo usando planificaciones (*schema*) y agendas. Entre sus objetivos se encuentra también facilitar la

migración del sistema a otras tareas, separando en módulos agentes de dominio la gestión de la información más específica.

En el ámbito europeo, tuvo carácter pionero el proyecto **SUNDIAL (Speech Understanding and Dialogue)** [11], relativo a la consulta de horarios de trenes y aviones, y que fue desarrollado admitiendo cuatro idiomas: inglés y francés, para la información sobre vuelos, alemán e italiano, para la información sobre horarios de trenes.

Aunque es complicado realizar predicciones sobre la evolución de los sistemas de diálogo en los próximos años, se pueden establecer una serie de retos a cumplir por los mismos:

- Desarrollo de nuevas metodologías que permitan adaptar el funcionamiento de los sistemas de diálogo, teniendo en cuenta las necesidades y preferencias de cada usuario (**sistemas adaptativos**).
- La comunicación persona a persona es **multimodal**, combinando diferentes canales, por lo tanto, el desarrollo de sistemas multimodales y su evaluación en diferentes entornos como redes sociales, mundos virtuales, etc., se encuentra a la orden del día.
- Diseño de sistemas **multitarea**.
- Análogamente al reconocimiento del habla, será de gran utilidad el **reconocimiento de emociones**. Es fundamental el diseño de sistemas que tengan en cuenta el comportamiento emocional del usuario y que sean igualmente capaces de generar respuestas que sinteticen estas emociones.
- Abarcar con estos sistemas tecnologías de máxima actualidad y que representan un mercado puntero y en plena expansión como la **domótica** y la ayuda en carretera (**GPS**).



Algunas tecnologías, como la telefonía móvil inteligente, han puesto de manifiesto la versatilidad que posee la voz como herramienta de manejo de un sistema o aplicación. Cientos de aplicaciones actualmente en el mercado controladas por voz, y para todo tipo de funciones y cometidos, demuestran que el límite de éste complemento solo se encuentra en el límite de la imaginación.

## 2.4 Introducción a VoiceXML

Iniciamos esta sección introduciendo previamente el lenguaje XML, ya que el estándar **VoiceXML** fue diseñado basándose en éste. XML fue desarrollado por un grupo de trabajo bajo los auspicios del consorcio **World Wide Web (W3C)** a partir de 1996. Éste fue constituido en 1994 con el objetivo de desarrollar protocolos comunes para la evolución de Internet. Se trata de un consorcio de la industria internacional con sedes conjuntas en el Instituto Tecnológico de Massachussets, de Estados Unidos, el Instituto Nacional de Investigación en Informática y Automática europeo y la Keio University Shonan Fujisawa Campus de Japón. El W3C tiene como misión la publicación para uso público de protocolos o estándares globales de uso libre [18].

**XML** es un formato basado en texto, específicamente diseñado para almacenar y transmitir datos [19]. Un documento XML se compone de elementos XML, cada uno de los cuales consta de una etiqueta de inicio, de una etiqueta de fin y de los datos comprendidos entre ambas etiquetas. Al igual que los documentos HTML, un documento XML contiene texto anotado por etiquetas. Sin embargo, a diferencia de HTML, XML admite un conjunto ilimitado de etiquetas, no para indicar el aspecto que debe tener algo, sino lo que significa. Por ejemplo, un elemento XML puede estar etiquetado como precio, número de pedido o nombre. Podría decirse entonces que XML es un metalenguaje, dado que con él podemos definir nuestro propio lenguaje de presentación y, a diferencia del HTML, que se centra en la representación de la información, XML se centra en la información en sí misma. La particularidad más importante del XML es que no posee etiquetas prefijadas con anterioridad, ya que es el propio diseñador el que las crea a su antojo, dependiendo del contenido del documento [14].

Como ejemplo descriptivo, la Figura 2.2 muestra la información incluida en un código típico HTML y su versión equivalente en XML. Se puede apreciar que es más sencilla de entender la representación en XML [18] [19].

HTML	XML
<TABLE>	<LIBROS>
<TR>	<LIBRO>
<TD>Título</TD>	<TITULO>AutoSketch</TITULO>
<TD>Autor</TD>	<AUTOR>Ramón Montero</AUTOR>
<TD>Precio</TD>	<PRECIO>33</PRECIO>
</TR>	</LIBRO>
<TR>	<LIBRO>
<TD>AutoSketch</TD>	<TITULO>Windows 98</TITULO>
<TD>Ramón Montero</TD>	<AUTOR>Jaime Perez</AUTOR>
<TD>33</TD>	<PRECIO>3.250</PRECIO>
</TR>	</LIBRO>
<TR>	<LIBRO>
<TD>Windows 98</TD>	<TITULO>Web Graphics</TITULO>
<TD>Jaime Perez</TD>	<AUTOR>Ron Wodaski</AUTOR>
<TD>3.250</TD>	<PRECIO>8.975</PRECIO>
</TR>	</LIBRO>
<TR>	</LIBROS>
<TD>Web Graphics</TD>	
<TD>Ron Wodaski</TD>	
<TD>8.975</TD>	
</TR>	
</TABLE>	

Figura 2.2: Comparativa del Sistema de etiquetas HTML/XML

La investigación en sistemas de diálogo ha coincidido en el tiempo con el desarrollo de la red Internet y las aplicaciones Web y, en consecuencia, se ha planteado el acceso a los sistemas de diálogo mediante aplicaciones Web. El lenguaje estándar VoiceXML es el

resultado del trabajo conjunto de varias importantes compañías (AT&T, IBM, Lucent, Motorola, etc) [11].

Se trata de una especificación propuesta por la W3C en 1995 que tiene como objetivo crear archivos XML, llamados documentos, que puedan reproducir sonido digitalizado, sonido sintetizado usando la tecnología TTS, reconocer información ingresada por el usuario (tonos DTMF) y reconocer palabra y/o frases pronunciadas por una persona, todo esto usando un dispositivo telefónico (teléfono clásico, celular o cualquier otra variante). VoiceXML está basado completamente en XML, es decir necesita que el documento VoiceXML esté "bien formado" para que pueda ser reconocido como correcto [20].

El servidor de documentos (servidor web) procesa las peticiones enviadas por la aplicación cliente (intérprete de VoiceXML) a través del entorno del intérprete de VoiceXML, y proporciona como respuestas documentos VoiceXML que son procesados posteriormente por el intérprete de VoiceXML. El entorno del intérprete puede monitorizar las entradas del usuario en paralelo con el intérprete; por ejemplo, puede detectar una frase especial que inicie la ejecución de un agente de ayuda de alto nivel, o bien, una frase predeterminada para cambiar las preferencias del usuario. La plataforma de implementación contiene el hardware telefónico y otros recursos relacionados con el mismo, generando eventos en respuesta a las acciones del usuario (p. e. pulsaciones de botones o comandos orales). Además, es función de la plataforma ejecutar eventos del sistema [23]. La arquitectura puede ser comprendida más fácilmente a través del esquema y análisis de los componentes mostrados en la Figura 2.3 [20].

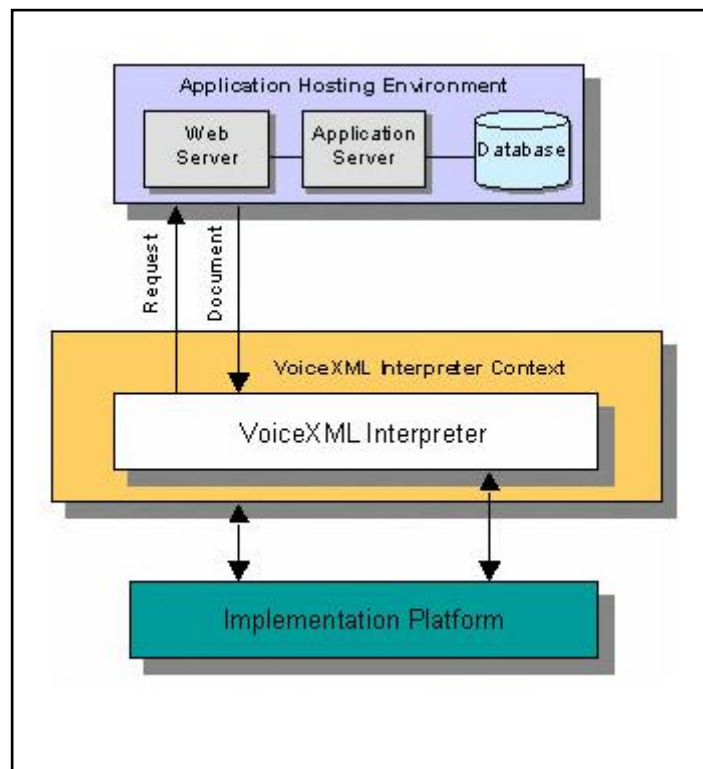


Figura 2.3: Arquitectura Modular definida en VoiceXML

- **Application Hosting Environment:** Llamado también "*Document Server*". Es un sistema que genera dinámicamente documentos VoiceXML. Básicamente está formado por 3 componentes:
  - *Web Server:* Servidor Web que recibe solicitudes HTTP y envía respuestas HTTP con un documento VoiceXML.
  - *Application Server:* Servidor de aplicaciones que mantiene una lógica de negocio que, sobre la base de los parámetros enviados por el Web Server, genera documentos VoiceXML.
  - *Database:* Base de Datos de la cual se obtiene información para generar los documentos VoiceXML.
- **VoiceXML Interpreter:** Aplicación que recibe un documento VoiceXML y lo interpreta, es decir, procesa las etiquetas que dicho documento contiene.
- **VoiceXML Interpreter Context:** Módulo del VoiceXML que monitoriza las posibles actividades que los usuarios realizan mientras se está interpretando un documento

VoiceXML, por ejemplo el usuario podría presionar desconectarse (colgar el teléfono), lo generaría que cancelación de la interpretación del documento.

- **Implementation Platform:** Este componente viene a ser el Browser en sí, pues cada empresa puede desarrollar su propio VoiceXML Browser el cual aparte de interpretar un documento VoiceXML puede implementar mecanismos de cache, procesamiento de llamadas telefónicas, etc.

## 2.5 Introducción a XHTML+Voice

**Extensible HyperText Markup Language (XHTML)** [24] es un lenguaje de etiquetas basado en XML para la creación de aplicaciones visuales en ordenadores o dispositivos portátiles. XHTML es la versión HTML 4.01 en XML, lo que quiere decir que mediante este estándar se pueden crear páginas que podrán ser leídas por todos los dispositivos compatibles con XML.

XHTML ha reemplazado a **HTML** como lenguaje estándar definido por el World Wide Web Consortium (W3C) para la implementación de aplicaciones web, de forma que éste estándar nos proporciona una sencilla adaptación al entorno multimodal y asegura que los usuarios con cualquier tipo de dispositivo puedan acceder correctamente a estas páginas [24].

Las diferencias a nivel sintáctico entre ambos lenguajes son sutiles pero muy importantes. XHTML trata de separar totalmente la forma del contenido, por ello se debe tener en cuenta que en XHTML se deben utilizar sólo las etiquetas y atributos que sirven para definir qué es cada elemento y no las que sirven para definir cómo se ha de ver cada uno de ellos. En XHTML se debe especificar obligatoriamente elementos como el juego de caracteres, el DTD que estamos utilizando con el *Doctype* o un nuevo elemento *Body* ligeramente más complejo.

- *Las etiquetas deben estar bien anidadas:* existen reglas estrictas en XHTML para la anidación de etiquetas y es obligatorio que se cierren antes las que se abrieron después. Es decir, no se puede alterar el orden de apertura y cierre.

- *Todas las etiquetas se cierran:* Todas las etiquetas deben tener su correspondiente etiqueta de cierre. Aquellas etiquetas como BR o IMG, que no tienen cierre en HTML, se deben cerrar con una barra al final de la apertura.
- *Todas las etiquetas y atributos van en minúsculas:* El lenguaje XML es sensible a mayúsculas y minúsculas, por ello para XML la etiqueta con las letras de su nombre o atributos en mayúsculas es distinta que las que van con ellas en minúsculas. Así pues, en XHTML se ha tomado la convención de escribirlo todo en minúsculas.
- *No se puede escribir contenido en el elemento BODY sin haberlo incluido en alguna etiqueta:* Para que el documento XHTML esté bien formado, no se puede meter un texto directamente en el cuerpo de la página sin haberlo metido en alguna etiqueta previamente (por defecto un "<div>").
- *Todos los valores de los atributos deben ir entrecomillados:* Es indiferente usar comillas dobles o simples, pero estamos obligados a usarlas en los valores que asignamos a cualquier atributo de las etiquetas.
- *Todos los atributos deben tener valor:* No se puede minimizar atributos, es decir, todos deben tener un valor asignado.
- *Los scripts y estilos deben colocarse en bloques CDATA:* Debido a las características de XML, caracteres como "<" o "&", pueden ser interpretados como parte del propio etiquetado del documento XHTML y para evitar que esto ocurra se encapsulan los bloques de script o estilos. De tal modo, cuando se abre un bloque de script o de estilos CSS, para evitar casos de incompatibilidad, debemos colocar su contenido dentro de un bloque CDATA.

Una vez introducido el concepto de VoiceXML, ya puede ser más fácilmente comprendida la funcionalidad y características de la herramienta de programación empleada en el desarrollo del presente proyecto: **XHTML+Voice**

**XHTML+Voice (X+V)** es un lenguaje de etiquetas basado en XML para el desarrollo de páginas web multimodales. X+V combina XHTML, para el desarrollo de los componentes visuales de la página, con VoiceXML como herramienta de interacción mediante voz y diálogo [24].

X+V usa un subconjunto de funciones básicas de VoiceXML para utilizarlas conjuntamente con XHTML creando así escenarios multimodales. Un ejemplo de escenario típico podría ser un formulario donde se nos solicita la entrada de datos en varios campos. Mediante la programación a través de X+V el formulario podría ser rellenado indistintamente mediante teclado (escrito) o mediante la voz. Las posibilidades de reconocimiento de VoiceXML van desde palabras concretas a frases. Puesto que la interfaz de la aplicación usará un motor gráfico y el reconocimiento y dictado de voz empleará un motor de diálogo, será necesario crear una sincronización de ambos contenidos, indicando a la herramienta qué campos de voz referencian a qué campos de texto.

Puesto que ciertas acciones de una página web van controladas a través de eventos, X+V incorpora el motor de eventos **Document Object Model (DOM)** usado en el estándar XML. Eventos típicos de HTML como puedan ser "*on mouse-over*" o "*input focus*" pueden ser usados con el fin de crear una correcta correlación entre los aspectos visuales y de voz de la aplicación.

Dado que todos los componentes de X+V son compatibles con el estándar XML, el desarrollo de los contenidos de voz puede ser escrito de dos maneras diferentes: en el mismo fichero que el contenido XHTML o en archivos separados. Separar el contenido de voz y el contenido visual (Figura 2.4) ofrece una mayor flexibilidad y ventajas en el desarrollo de las aplicaciones:

- Permite una mejor **organización y estructura** de los contenidos.
- Permite **reusar el mismo contenido** de voz en diferentes páginas XHTML.

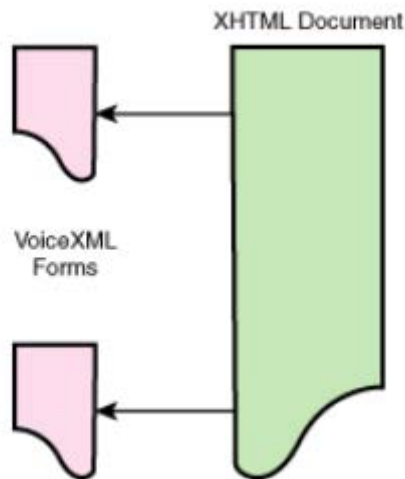


Figura 2.4: Modelo 1 de uso de formularios VoiceXML

Todavía podría crearse un nivel más de separación en el cual los formularios de voz quedarían escritos también de forma independiente respecto del documento VoiceXML. De esta manera podrían ser reusados por aplicaciones multimodales X+V o por aplicaciones de voz VoiceXML. La Figura 2.5 ayuda a entender el esquema descrito.

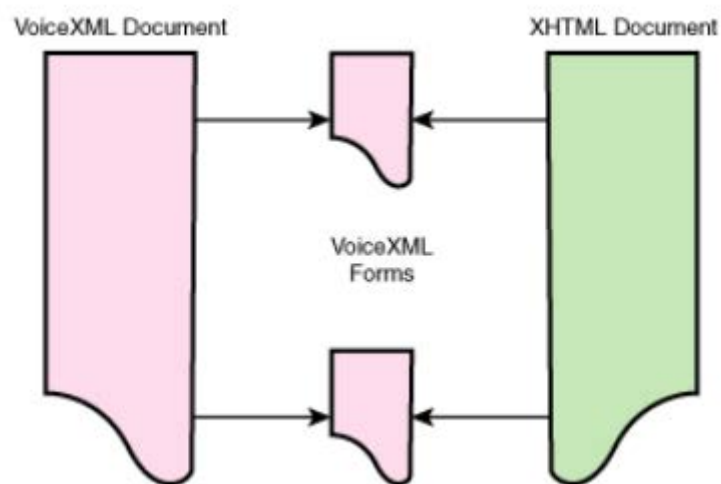


Figura 2.5: Modelo 2 de uso de formularios VoiceXML



Tal y como se ha explicado, X+V usa XHTML para la interacción visual, un subconjunto de funciones de VoiceXML para la interacción a través de voz y eventos XML para correlacionar ambos. La Figura 2.6 muestra un ejemplo típico de escenario multimodal en el que los tres tipos de contenido están presentes [24].

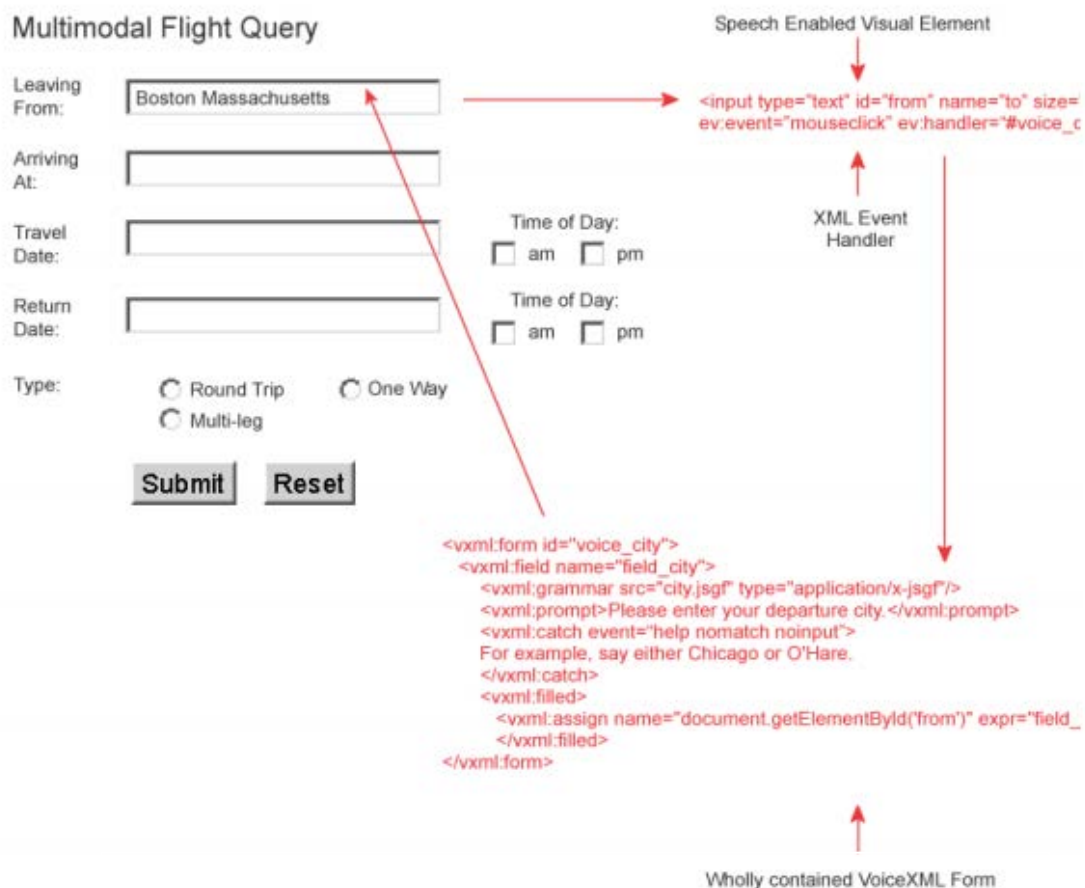


Figura 2.6: Ejemplo de escenario típico de interacción multimodal

En este escenario se solicita información al usuario de manera visual o mediante voz. El usuario responde a la primera cuestión "Enter the departure city" (introduzca la ciudad origen) con la entrada de voz "Boston, Massachusetts". El motor de diálogo reconoce la frase y devuelve una cadena de texto que es mostrada por pantalla. A continuación la aplicación mueve la propiedad *input focus* al siguiente campo que se requerirá en la siguiente interacción con el usuario.

La etiqueta XHTML para el campo "Departure City" es sencillamente una etiqueta *input* que genera un cuadro de texto. El código VoiceXML para ese mismo campo presenta una mayor complejidad. Posee los siguientes elementos:

- Un **mensaje de voz <prompt>** preguntado por la ciudad de origen.
- Una **gramática** compuesta de todos los aeropuertos.
- Una directiva que indica al motor de diálogo qué hacer con la **palabra resultante** de la interacción de voz.
- **Directivas** que indican qué hacer en caso de **ciertos eventos fijos** como son el caso de que el usuario diga "Help", que el motor de voz no haga coincidir la respuesta de voz con ninguna de las opciones posibles (*nomatch*), o que el usuario no diga nada (*noinput*).

Las **gramáticas** son la manera de que el desarrollador le indique al motor de reconocimiento de voz qué palabras o frases son las permitidas en cierto campo, para cierto formulario, para una página entera o para toda la aplicación. En el ejemplo, la gramática representa el conjunto de palabras que pueden ser reconocidas al contestar sobre el campo "field\_city" de ese formulario. Puede observarse en el código <vxml: grammar src="city.jsgf"....> que se referencia a una gramática que se encuentra en un fichero externo del tipo "jsgf". Como se ampliará en el siguiente capítulo mediante ejemplos, las gramáticas pueden ser:

- **Externas:** Permiten que una misma gramática pueda ser utilizada en diferentes secciones VoiceXML de distintas páginas, para distintos formularios de voz o para distintos campos. Basta con ser referenciada, como se ha visto en el ejemplo anterior.
- **Internas:** Se escriben en un determinado campo y formulario de la sección VoiceXML de una página X+V y solo sirven para indicar las entradas de voz que serán permitidas para ese campo concreto.

El uso de los comandos más comunes de la programación de voz de las páginas X+V será ampliado en el siguiente capítulo empleando para ello ejemplos obtenidos de secciones de código de nuestra aplicación. De igual manera se mostrarán ejemplos de gramáticas internas y externas con el fin de que los conceptos ahora introducidos puedan verse de manera más clara y completa.

## Capítulo 3: Descripción General del Sistema Desarrollado

---

Este capítulo detalla las características fundamentales y comunes al conjunto de módulos que conforman el sistema desarrollado, así como sus principales funcionalidades, arquitectura y diferentes anotaciones que ayuden a comprender de manera global la aplicación. De igual modo se analizan de manera detallada las diferentes tecnologías empleadas y su interacción con el sistema. Por último, se describen las operaciones generales junto con su codificación para que pueda ser comprendido su funcionamiento.

### 3.1 Introducción al Sistema

La aplicación desarrollada resume su funcionalidad y componentes en los siguientes módulos:

- **Página de inicio (Home page):** Se trata de una página inicial de interfaz muy sencillo e intuitivo que sirve como menú de acceso a las distintas funcionalidades y apartados de la aplicación (realización de ejercicios, consulta de resultados, registro, login y ayuda).
- **Registro y login:** Es necesario haber iniciado una sesión en la aplicación (y por tanto haber registrado nuestro usuario previamente) para hacer uso de los módulos de ejercicios y para consultar nuestros resultados. La página de registro es un sencillo formulario que recoge nuestros datos básicos y el login se realiza introduciendo nuestro usuario y contraseña.
- **Módulo de realización de ejercicios:** Tras la selección del tipo de ejercicio (gramática, vocabulario o escucha) se accede a una primera pantalla en la que se selecciona la categoría y nivel deseados de los ejercicios. Puede realizarse de manera manual o bien mediante el uso de la opción "*choose for me*", la cual, tras evaluar una serie de criterios que serán explicados de manera detallada más adelante, devolverá ejercicios al usuario de manera automática. La siguiente pantalla muestra los ejercicios seleccionados, y una vez realizados se pulsa la opción corregir la cual devolverá en una

nueva pantalla la corrección a nuestras respuestas y ofrecerá la posibilidad de volver a la página de inicio o home.

- **Consulta de resultados:** Este modulo devuelve información acerca del número de ejercicios hechos y correctos por familia y nivel añadiendo los porcentajes y la puntuación obtenida (del 1 al 100) en función de estos. Cada tipo de ejercicio es seleccionable y devolverá en una nueva pantalla el conjunto de ejercicios corregidos de esa familia y nivel con el fin de que el usuario pueda repasar sus aciertos y errores. Por último, la página muestra una serie de graficas que permiten comparar de manera visual la evolución de nuestra actividad en la aplicación.
- **Modulo de administración de ejercicios:** Permite crear ejercicios nuevos y editar o borrar ejercicios ya existentes. Tras seleccionar la familia de ejercicios, se accede a una pantalla en la cual podemos elegir la categoría en la que se incluirá el nuevo ejercicio o en la que ya existe el ejercicio a modificar o borrar. Aquí también será posible crear un ejercicio en una categoría nueva. Tras seleccionar la categoría se accede a una nueva pantalla que muestra los ejercicios ya existentes en esa categoría y la opción de editarlos o borrarlos. Por último, se muestra el formulario a rellenar si lo que se desea es crear un ejercicio nuevo.
- **Ayuda:** Se trata de una página que devuelve información con el fin del indicar al usuario como comenzar a utilizar la aplicación. Explica cómo realizar el inicio de sesión, así como los diferentes módulos y opciones de utilización posibles.

La estructura de la aplicación desarrollada sigue un esquema cliente servidor mostrado en la Figura 3.1. La aplicación web completa se encuentra contenida en un Servidor Web Apache. Las páginas PHP se encuentran en el servidor, donde son interpretadas, y generan las páginas XHTML+Voice resultantes que el cliente recibe. Asimismo el servidor web contiene el SGBD (MySQL) junto con las bases de datos creadas. También aloja todos los ficheros que utiliza la aplicación: imágenes, textos, ficheros de audio, gramáticas y estilos.

El cliente únicamente necesita un equipo con conexión a internet y. En caso de querer usar la voz para el manejo de la aplicación, deberá tener instalado el Navegador Opera junto con el plugin de voz debidamente configurado. Asimismo necesitará micrófono para dar las

instrucciones de voz y altavoces o auriculares para escuchar los diálogos generados con el sistema.

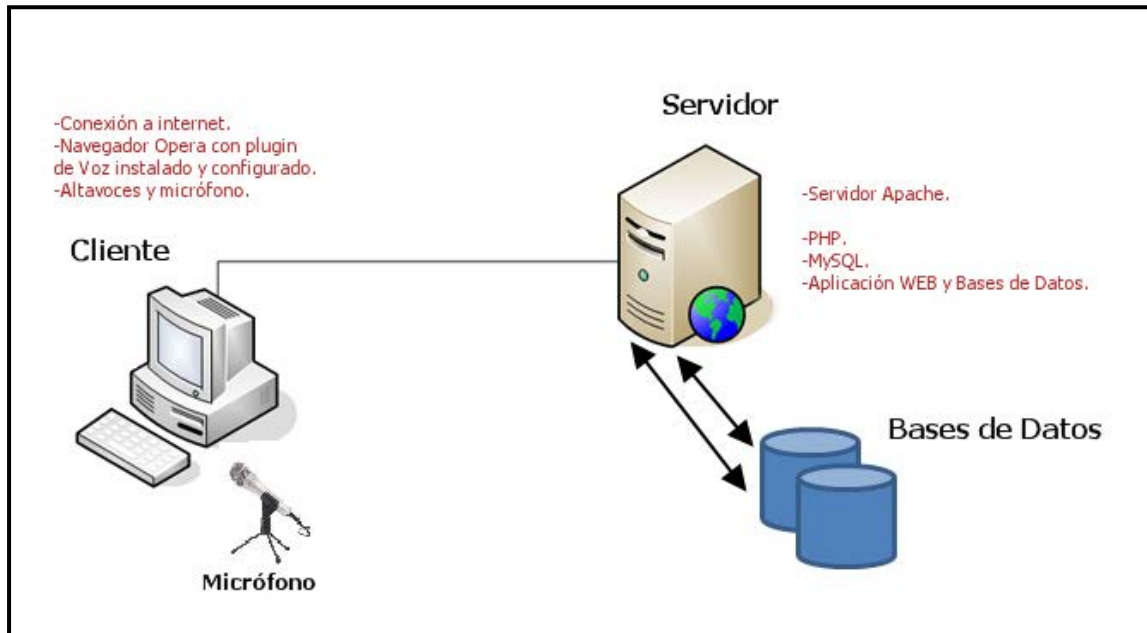


Figura 3.1: Esquema Cliente-Servidor de la aplicación desarrollada

### 3.2 Tecnologías

1. **Interprete XHTML+Voice (Opera Voice)** [26]: Opera Voice funciona como un *plug-in* o complemento de Opera cuyo objetivo principal es permitir al usuario el control del interfaz web "hablándole al navegador". Cualquier comando de uso frecuente puede ser invocado mediante la voz. La Figura 3.2 resume los más significativos.

#### Lectura

Comando	Acción de Opera
Speak	Lectura del texto seleccionado
Read	Selecciona el primero/siguiente bloque de texto en una página y lo lee
Page address	Lee la dirección (URL) de la página anterior
Link address	Lee la dirección (URL) del enlace que tiene el foco
Clipboard content	Lee el contenido del portapapeles

**General**

Comando	Acción de Opera
Voice commands	Lista de los comandos de Voz predeterminados (esta lista)
Voice help	Va a la página de ayuda de la Voz (este documento)
Help	Va a la página índice de los ficheros de ayuda de Opera
Back/Forward	Va a la página anterior o a la siguiente
Fast forward	Analiza la página y va a la que se supone que es la siguiente
Rewind	Vuelve al sitio anterior o a la primera página del sitio actual
Home	Va a la página principal predeterminada
Reload page	Recarga la página actual
Reload all	Recarga todas las páginas
Stop loading	Interrumpe la carga de la página actual
Log in	Usa la Varita (administrador de contraseñas) para entrar
Full screen	Entra/sale del modo de pantalla completa
Small screen	Emula un dispositivo de mano/vuelve al modo normal de pantalla
Next/previous page	Va a la página siguiente o a la anterior en el espacio de trabajo
Paste and go	Va directamente a la dirección Web (URL) del portapapeles
Duplicate page	Duplica la página activa
New page	Abre una página nueva y vacía
Close page/all	Cierra la página activa o todas las páginas abiertas
Close other	Cierra todas las páginas abiertas con la excepción de la activa
Reopen page	Reabre la última página cerrada
Quit application	Cierra Opera
Zoom in/out	Aumenta o disminuye la escala % en pasos de 10
Zoom normal	Vuelve la escala al 100%

**Figura 3.2:** Resumen de comandos de voz disponibles en Opera Voice

Pero la funcionalidad va mucho más allá. Opera Voice también puede leer el subconjunto de comandos VoiceXML que incluyen las páginas XHTML+Voice, lo que lo convierte en el intérprete de éste lenguaje multimodal.

Por el momento el complemento de voz está sólo disponible en inglés y para el sistema operativo Windows. Incluye diferentes voces (hombre-mujer) y la posibilidad de poder configurar diferentes parámetros como la tecla a usar antes de dictar un comando de voz y su uso, el tono, velocidad así como el volumen de la voz o el nivel de confianza (ver Figura 3.3).

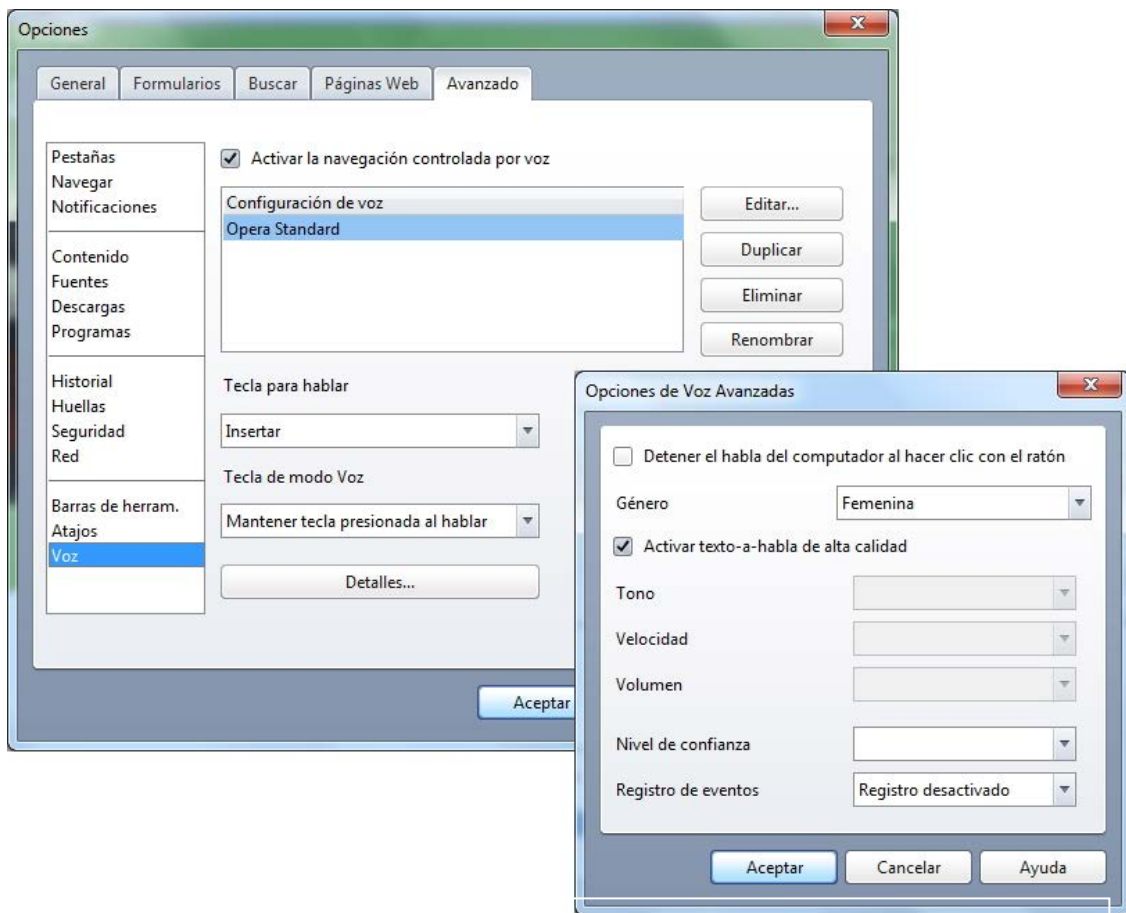


Figura 3.3: Opciones de Configuración de Opera Voice

2. **MySQL** [29]: Se trata de un sistema de gestión de bases de datos relacional, multihilo y multiusuario. MySQL AB, desde enero de 2008 una subsidiaria de Sun Microsystems y ésta a su vez de Oracle Corporation desde abril de 2009, desarrolla MySQL como software libre en un esquema de licenciamiento dual. Por un lado es Open Source lo que significa que la persona que quiera puede usarlo y modificarlo. Cualquiera puede descargar el software de MySQL de Internet y usarlo sin pagar por ello. Por otro lado, cualquiera que lo necesite puede estudiar el código fuente y cambiarlo de acuerdo a sus necesidades. MySQL usa la licencia GPL (Licencia Pública General GNU), para definir qué es lo que se puede y no se puede hacer con el software para diferentes situaciones. Sin embargo, si un usuario tiene problemas con este tipo de licencia o tiene la necesidad de incorporar código de MySQL en una aplicación comercial es posible comprar una versión de MySQL con una licencia comercial.

Existen varias interfaces de programación de aplicaciones que permiten, a aplicaciones escritas en diversos lenguajes de programación, acceder a las bases de datos MySQL, incluyendo C, C++, C#, Pascal, Delphi (via dbExpress), Java (con una implementación nativa del driver de Java), Lisp, Perl, PHP, Python, etc.; cada uno de estos utiliza un interfaz de programación de aplicaciones específica. También existe una interfaz ODBC, llamado MyODBC que permite a cualquier lenguaje de programación que soporte ODBC comunicarse con las bases de datos MySQL. También se puede acceder desde el sistema SAP, lenguaje ABAP. La Figura 3.4 muestra una comparativa de uso de los principales Sistemas Gestores de bases de datos situando a MySQL en una posición privilegiada [27].

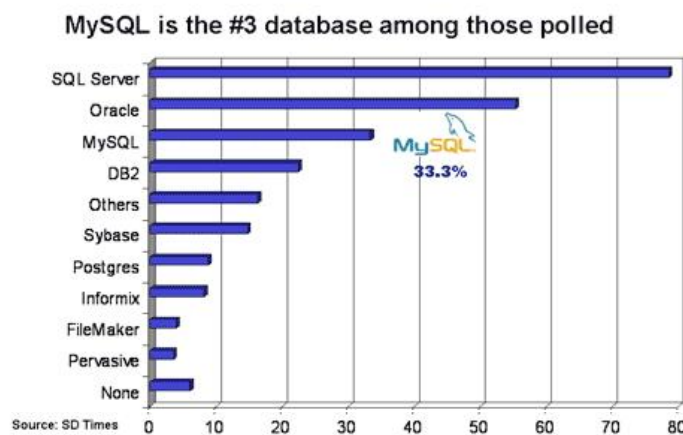


Figura 3.4: Comparativa de uso de diferentes Sistemas Gestores de bases de datos actuales

#### Administración de bases de datos. PhpMyAdmin y MySQL Administrator:

3. **phpMyAdmin** [31] es una herramienta escrita en PHP con la intención de manejar la administración de MySQL a través de páginas web. Se encuentra disponible bajo la licencia GPL, de libre distribución. Es una herramienta muy completa que permite acceder a todas las funciones típicas de la base de datos MySQL a través de una interfaz web muy intuitiva.

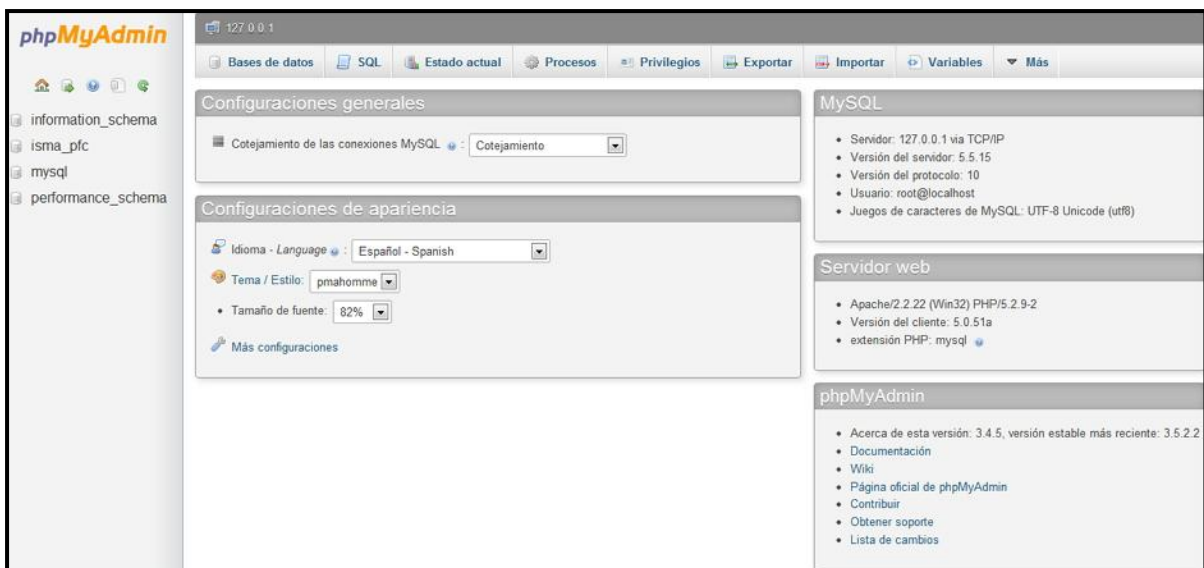
La aplicación en si no es más que un conjunto de archivos escritos en PHP que podemos copiar en un directorio de nuestro servidor web, de modo que, cuando accedemos a esos archivos, se nos muestra una interfaz donde podemos encontrar las bases de datos a las que tenemos acceso en nuestro servidor de bases de datos y todas sus tablas. La



herramienta nos permite crear tablas, insertar datos en las tablas existentes, navegar por los registros de las tablas, editarlos, borrarlos, borrar tablas incluso ejecutar sentencias SQL y hacer un backup de la base de datos entre otras muchas opciones [30].

Administrar las bases de datos mediante el uso de phpMyAdmin resulta muy práctico, rápido y visual teniendo en cuenta que, en caso de no tener esta herramienta, la solución consistiría en utilizar directamente lenguaje SQL (mucho más complejo), y, en caso de que la base de datos esté alojada remotamente en Internet, no podríamos hacerlo si no es con acceso por TELNET al servidor de la base de datos.

La Figura 3.5 muestra el interfaz de phpMyAdmin así como la ventana de opciones disponibles que tenemos para actuar sobre nuestra base de datos.



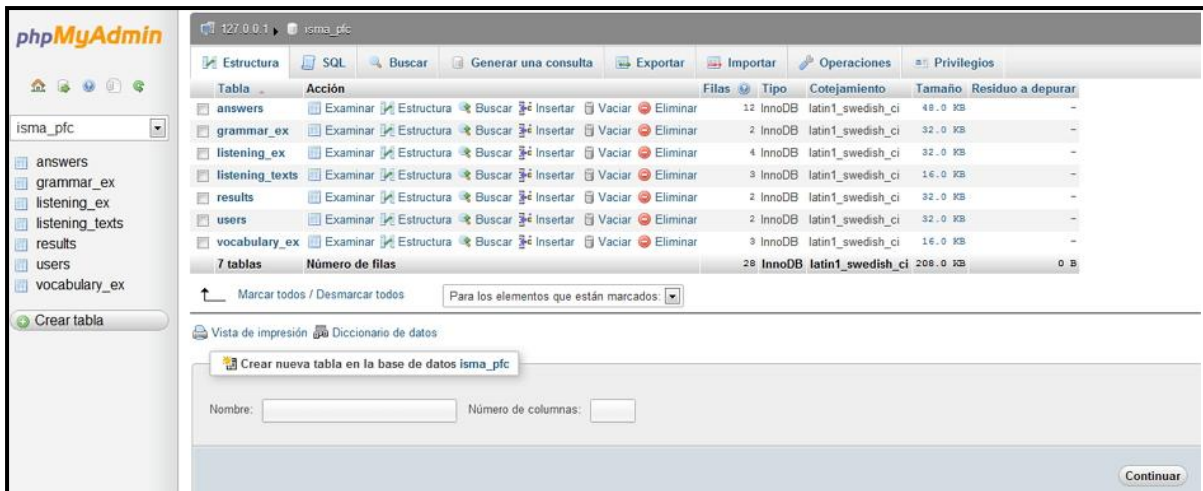


Figura 3.5: Pantalla inicial y Vista de tablas de phpMyAdmin

4. **MySQL Administrator** [32] es un software de administración de servidores de bases de datos de MySQL creado por MySQL AB. Se trata de un software multiplataforma, que por el momento se encuentra disponible para Linux y Microsoft Windows y que cuenta con un entorno gráfico de usuario muy intuitivo. A continuación se describen las tareas administrativas más comunes que permite llevar a cabo (Figura 3.6):

- Configuración de las opciones de inicio de los servidores.
- Inicio y detención de servidores.
- Monitorización de conexiones al servidor.
- Administración de usuarios.
- Monitorización del estado del servidor, incluyendo estadísticas de uso.
- Visualización de logs de eventos.
- Gestión de copias de seguridad y recuperaciones.
- Visualización de catálogos de datos.



Figura 3.6: Menú Inicial de Opciones de MySQL Administrator

5. **Servidor HTTP (Apache):** Un servidor web o servidor HTTP es un programa informático que procesa una aplicación del lado del servidor realizando conexiones bidireccionales y/o unidireccionales y síncronas o asíncronas con el cliente generando o cediendo una respuesta en cualquier lenguaje o Aplicación del lado del cliente.

El código recibido por el cliente suele ser compilado y ejecutado por un navegador web. Para la transmisión de estos datos suele utilizarse algún protocolo, generalmente el protocolo HTTP para estas comunicaciones, perteneciente a la capa de aplicación del modelo OSI [34].

El servidor HTTP Apache es un servidor web HTTP de distribución libre y de código abierto, siendo el más popular desde abril de 1996, con una penetración actual del 50% del total de servidores web del mundo (agosto de 2007). Compatible con plataformas Unix, Microsoft Windows, Macintosh y otras, implementa el protocolo HTTP/1.12 y la noción de sitio virtual [35].

La popularidad del sistema puede deberse, entre otras, a algunas de estas características [33]:

- Funciona en *multitud de Sistemas Operativos*, lo que lo hace prácticamente universal.
- Apache es una *tecnología gratuita de código fuente abierto*. El hecho de ser gratuita es importante pero no tanto como que se trate de código fuente abierto. Esto le añade transparencia al software de manera que si queremos ver que es lo que estamos instalando como servidor, lo podemos saber, sin ningún secreto o puerta trasera.
- Es un servidor *altamente configurable y de diseño modular*. Es muy sencillo ampliar sus capacidades. Actualmente existen muchos módulos adaptables para Apache, de instalación independiente y que ofrecen multitud de funcionalidades. Otro factor destacable es que cualquiera que posea una experiencia mediana en la programación mediante lenguaje C o Perl puede escribir un módulo para realizar una función determinada.
- Apache *trabaja ampliamente con Perl, PHP y otros lenguajes de script*. También con Java y páginas jsp, teniendo todo el soporte que se necesita para mantener páginas dinámicas.
- Permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. Es posible configurarlo para que ejecute un determinado script cuando ocurra un error concreto. Tiene una alta configurabilidad en la creación y gestión de logs a medida del administrador, de este modo puedes tener un mayor control sobre lo que sucede en tu servidor.

### 3.3 Implementación de las Operaciones Generales

1. **Desarrollo dinámico:** Como se ha mencionado en apartados anteriores, la aplicación ha sido desarrollada empleando PHP con la intención de crear páginas dinámicas. Esto quiere decir que las páginas XHTML+Voice estáticas que muestra la aplicación se han ido creando sólo cuando han sido solicitadas por el cliente a partir de la llamada a un fichero PHP, que analizará los datos y formularios enviados en la página anterior y generará la página siguiente en función de estos. Este esquema de funcionamiento puede entenderse mejor a partir del diagrama mostrado en la Figura 3.7 [36].

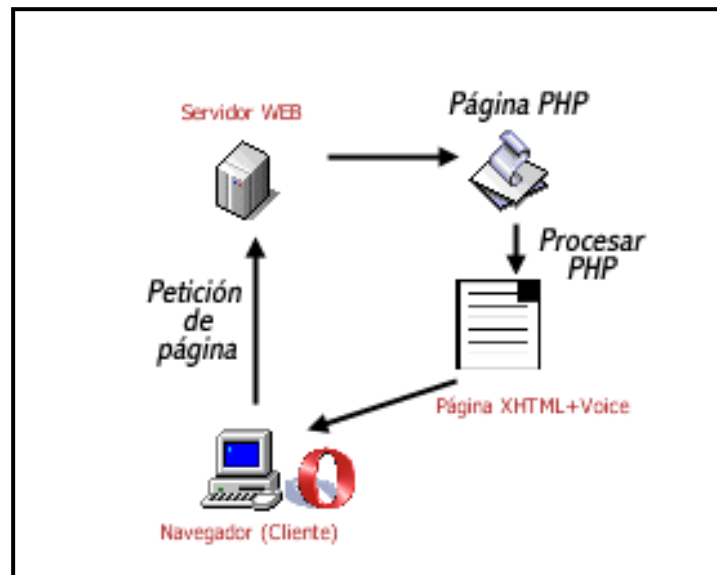


Figura 3.7: Esquema de creación de páginas X+V dinámicas

2. **Estructura de las páginas PHP:** Aunque cada página PHP realiza un tratamiento de la información diferente y escribe el contenido de manera particular, existe una estructura de operaciones general que se mantiene en todas ellas. Se resumen a continuación los puntos y el orden en que son codificadas:
  - Declaración y apertura del fichero de salida: Puesto que la página PHP tiene como finalidad la creación y escritura del fichero XHTML+Voice a ejecutar, en las primeras líneas se declara y abre dicho fichero (Figura 3.8).

```
$fichero_xhtml="grammar_exercise.xhtml";      -> Se le da nombre al fichero.  
$resultado_xhtml=fopen($fichero_xhtml,w);    -> Se abre para escritura.
```

Figura 3.8: Declaración y apertura de fichero XHTML resultante

A partir de este momento todas las operaciones de escritura a fichero destino presentarán el comando mostrado en la Figura 3.9.

```
fwrite($resultado_gramatica,'<texto o código a escribir en la página destino>');
```

Figura 3.9: Escritura en fichero XHTML resultante

Es decir, todo el contenido de salida será escrito en el fichero XHTML de salida guardado en la variable "\$fichero\_xhtml".

- Recogida de variables de formulario: Son las variables enviadas desde la página anterior a través de formulario. Las páginas XHTML creadas, presentan formularios cuyo contenido se envía a la presente página PHP con el fin de procesar la información y poder crear la página XHTML resultante tal y como se explicó en el esquema del principio del apartado. Para tal fin se emplea el comando \$\_GET (Figura 3.10).

```
$categoría=$_GET[categoría];  
$nivel=$_GET[nivel];
```

**Figura 3.10:** Almacenamiento de variables pasadas a través de formulario

En este ejemplo las variables "categoría" y "nivel" fueron enviadas a esta página a través del *submit* de un formulario de la página anterior. Con este comando las recogemos y las guardamos en nuevas variables ya que las necesitaremos para la creación de la nueva página.

- Recogida de variables de sesión: Además de las variables pasadas de una página a otra a través de formularios, en ocasiones se necesitan pasar otros datos, almacenados en variables y de muy diversa función, que no tienen por qué haber sido introducidos por el usuario y que nos sirven a nosotros para poder generar las páginas y el contenido dinámico. Lo más útil y efectivo es almacenar estos datos como variables de sesión. Podemos decir que es un método para hacer que variables estén disponibles en múltiples páginas sin tener que pasarlas de una a otra. Un ejemplo típico es, después de que el usuario haya iniciado sesión, guardar su identificador de usuario en una variable de sesión. A partir de entonces ya tendremos ese dato disponible en cualquier página sin tener que preguntarlo cada vez que lo necesitemos.

Para dotar a una página de la posibilidad de cargar variables de sesión ya hechas o de crear nuevas variables de este tipo insertaremos el siguiente código (Figura 3.11).

```
session_start();  
$_SESSION[idusuario] = $idusuario; //Para crear una variable de sesión.  
$idusuario = $_SESSION[idusuario]; //Para cargar una variable de sesión creada anteriormente.
```

Figura 3.11: Ejemplo de uso de variables de Sesión

Es sencillo comprobar que lo que se realiza al crear y cargar variables es el proceso inverso. En este ejemplo se ha almacenado previamente el identificador de usuario en la variable "\$idusuario" y a continuación se ha guardado en una variable de sesión llamada 'usuario'. Posteriormente, si en otra página necesitamos saber el identificador de usuario, simplemente guardaremos en una variable el valor de la variable de sesión que ya tenía almacenado el identificador buscado.

Las variables de sesión tienen un tiempo de existencia limitado tras el cual, si no se han solicitado más páginas de la aplicación, caducarán. En algunas ocasiones sin embargo es preferible que sean eliminadas por nosotros mismos, algo que hemos programado para que pase, por ejemplo, cuando el usuario hace el logout. Puesto que la información que se ha guardado en ellas es relativa a la sesión de un usuario, tiene sentido que se eliminen cuando el usuario termina su sesión para crear otras nuevas referidas al siguiente usuario. Para ello simplemente se utiliza el comando mostrado en la Figura 3.12 [37].

```
session_destroy();
```

Figura 3.12: Eliminación de variables de sesión

- Apertura de la Base de Datos: Puesto que las páginas PHP interactúan con la información almacenada en las bases de datos de la aplicación, será necesaria la apertura de dicha BBDD mediante los comandos mostrados en la Figura 3.13.

```
$ddb="isma_pfc"; -> Nombre de la Base de Datos.  
$conexion=mysql_connect("servidor","usuario","clave");  
mysql_select_db ($ddb, $conexion); -> Apertura de la Base de Datos.
```

Figura 3.13: Apertura y conexión a la Base de Datos

A partir de entonces al realizar las consultas deberá indicarse que se realiza sobre esta base de datos que hemos guardado en la variable "\$conexion" (Figura 3.14).

```
$sacar = "SELECT * FROM grammar_ex WHERE categoría = '$categoría'
AND nivel = '$nivel'";

$resultado = mysql_query($sacar,$conexion);
```

Figura 3.14: Ejemplo de consulta a la Base de Datos

La consulta almacenada en "\$sacar" sobre la base de datos almacenada en "\$conexion" guardará su resultado en la variable "\$resultado".

- Escritura en fichero de gramática: Este conjunto de operaciones sólo se lleva a cabo cuando se hace uso de una gramática externa. Puesto que la gramática externa se ubica en un fichero aparte, se declara inicialmente el fichero salida (una operación similar a la explicada en el punto 1 y mostrada en la Figura 3.15).

```
$fichero_gramatica="external_grammar.jsgf";->Nombre del fichero gramática.
$resultado_gramatica=fopen($fichero_gramatica,w);->Apertura del fichero para
escritura.
```

Figura 3.15: Declaración y apertura del fichero de gramática externa

Todo el contenido posterior de escritura de la gramática incluirá entonces el comando *fwrite* a archivo salida guardado en la variable "\$resultado\_gramatica" (Figura 3.16).

```
fwrite($resultado_gramatica,'#JSGFV1.oiso-8859-1;');
fwrite($resultado_gramatica,'grammar fichero_gramatica;');
fwrite($resultado_gramatica,'public <fichero_gramatica> =');
fwrite($resultado_gramatica,' ..... ')

.. <Escritura del contenido de la gramática>*
```

Figura 3.16: Escritura de la gramática en fichero externo



\* En el apartado de código de voz incluido en el siguiente punto será mostrado el código completo de una gramática ejemplo. La estructura y forma de la gramática en si es similar para gramáticas externas e internas.

Declaración de cabecera XHTML: Las páginas XHTML forzosamente deben abrirse indicando la normativa DTD. Si además van a hacer uso del sistema de diálogo se indican también las referencias XML en una cabecera fija como la mostrada en la Figura 3.17.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
xmlns:vxml="http://www.w3.org/2001/vxml"
xmlns:ev="http://www.w3.org/2001/xml-events"
xmlns:xv="http://www.voicexml.org/2002/xhtml+voice">
```

Figura 3.17: Declaración de cabecera XHTML

- Escritura de la cabecera (HEAD) en fichero de salida XHTML: La cabecera HEAD presenta inicialmente unas declaraciones fijas que definen básicamente el formato y el fichero de estilos que emplea la página (Figura 3.18).

```
fwrite($resultado_xhtml,'<head>');
fwrite($resultado_xhtml,'<meta http-equiv="Content-Type"
content="text/html; charset=utf-8" />');
fwrite($resultado_xhtml,'<meta name="language" content="es"/>');
fwrite($resultado_xhtml,'<linkrel="stylesheet" type="text/css" href="estilos
_contents.css" media="screen"/>');
```

Figura 3.18: Declaración de la sección HEAD

A continuación se escribe el elemento *"title"* que se mostrara como encabezado de la página o como título de la ventana (Figura 3.19).

```
fwrite($resultado_xhtml,'<title>');  
fwrite($resultado_xhtml,'GRAMMAR EXERCISE');  
fwrite($resultado_xhtml,'</title>');
```

Figura 3.19: Ejemplo de la sección de título

- Programación de la sección de voz: Tras el título del encabezado de la página, y todavía contenido dentro de la sección *<head>* es momento de incluir la programación del sistema de diálogo que presenta típicamente la siguiente estructura y contenido:
  - La **apertura del formulario, y un mensaje inicial** que será dictado por el navegador al cargar la página y que se programa mediante el comando *<vxml:block>*. Se usa en todas las páginas que emplean voz para darle instrucciones al usuario (Figura 3.20).

```
fwrite($resultado_xhtml,'<vxml:form id="getMessageInfo">');  
fwrite($resultado_xhtml,'<vxml:block>');  
fwrite($resultado_xhtml,'Fill the blanks using the verbs between brackets  
in present perfect');  
fwrite($resultado_xhtml,'</vxml:block>');
```

Figura 3.20: Ejemplo de mensaje de voz inicial

- **Declaración y contenido de los "fields":** Cada elemento de la página que espera para ser rellenado a través de la voz es un elemento *<vxml:field>*. Para declararlos a cada uno de ellos se le debe dar al menos un identificador (Figura 3.21).

```
fwrite($resultado_xhtml,'<vxml:field xv:id="identificador"  
name="nombre">');
```

Figura 3.21: Declaración de campo de entrada de voz o field

A continuación se indica la gramática que se va a utilizar para ese *"field"*, es decir, la gramática que contiene el conjunto de palabras permitidas como respuesta.

Como puede observarse, todas las gramáticas se crean de manera dinámica, utilizando en cada momento el conjunto de palabras necesarias tanto para gramáticas creadas en fichero externo como cuando se incluyen en la misma página resultante.

En caso de utilizar una gramática externa simplemente referenciaremos el fichero externo de gramática que fue creado anteriormente (Figura 3.22).

```
fwrite($resultado_xhtml,'<vxml:grammar src="external_grammar.jsgf">');
```

Figura 3.22: Referencia a gramática en fichero externo

En caso de tratarse de una gramática interna escribiremos el código dentro del *"field"*. Tanto en caso de ser externa como interna presenta la estructura mostrada en la Figura 3.23.

```
fwrite($resultado_xhtml,'#JSGF V1.0;');  
fwrite($resultado_xhtml,'grammar topics;');  
fwrite($resultado_xhtml,'public <topics> = <palabra 1> | <palabra 2> | <palabra n>');
```

Figura 3.23: Ejemplo de declaración de gramática interna

Como último elemento dentro del campo *"field"*, añadimos la etiqueta evento `<vxml:nomatch>`. Dentro de ella se incluyen las acciones que serán llevadas a cabo en caso de que la respuesta dictada por el usuario no coincida con ninguna de las palabras opción de la gramática (Figura 3.24).

```
fwrite($resultado_xhtml,'<vxml:nomatch>');  
fwrite($resultado_xhtml,'<vxml:prompt>Please repeat again, I can not  
understand your option.</vxml:prompt>');  
fwrite($resultado_xhtml,'</vxml:nomatch>');
```

Figura 3.24: Declaración de etiqueta "no match"

\* En el ejemplo, la acción a llevar a cabo es hacer que el navegador dicte (mediante el comando `<vxml:prompt>`) la frase indicada.

- **Fields completados:** Existen dos modalidades de acción a llevar a cabo al completar los campos o "fields". Con la opción `<filled mode="any">` indicamos acciones que se realizarán cada vez que se complete un "field" (y sólo uno). En el caso de nuestra aplicación la acción a llevar a cabo es comprobar si la palabra dictada por el usuario es "home" o "back", en cuyo caso se accederá a la página principal o a la anterior terminando la ejecución de la página actual.

Mediante la opción `<filled mode="all">` indicamos la acción a llevar a cabo una vez se ha completado de manera satisfactoria la introducción de voz para TODOS los campos "field". En el caso de nuestra aplicación la acción consiste en llevar a la siguiente página todas las respuestas con el fin de ser procesadas. Suele acompañarse también, mediante la etiqueta `<vxml:prompt>`, de un mensaje satisfactorio en el que se informa al usuario de que se tienen todas las respuestas.

- **Sincronización de valores dictados y valores escritos:** Puesto que los formularios pueden ser rellenados por voz, de manera escrita o incluso usando ambos métodos, es necesario indicar para cada "field" la variable de formulario de la parte escrita que le referencia. Esto se realiza con el comando `<xv:sync xv:input>` (Figura 3.25).

```
fwrite($resultado_xhtml,'<xv:sync xv:input="$cadena1" xv:field="#$cadena1"/>');
```

Figura 3.25: Sincronización de entradas de voz y entradas escritas

Lo que se está indicando en este fragmento de código es que si el usuario rellena por voz el *field* "\$cadena1" se le dará ese mismo valor a la variable "\$cadena1" del formulario escrito o viceversa. De esta manera sean completados por voz o texto las variables siempre tienen valor y coinciden. Para hacerlo más sencillo se ha utilizado el mismo nombre para los *fields* y para las variables del formulario pero no es obligatorio que esto sea así.

- Escritura del cuerpo (BODY) en fichero de salida XHTML: Todos los elementos visuales de la página irán contenidos en esta sección del código. Para la aplicación presenta típicamente los siguientes elementos:
  - **Título**: Empleando una tipografía amplia y clara que permite ver rápidamente en que sección nos encontramos.
  - **Instrucciones**: Presentan de manera escrita las acciones a llevar a cabo dependiendo de la página en la que nos encontremos. Vienen a dar un mensaje similar al dictado por la voz del navegador y que fueron explicados en la sección de voz.
  - **Formularios**: Todas las páginas que interactúan con el usuario solicitan datos de éste, ya sea por ejemplo para seleccionar y realizar los ejercicios o para logarse, registrarse o crear ejercicios. Para insertar los datos, y el posterior envío de información a la siguiente página, se emplean formularios GET. La declaración incluye el nombre del formulario, el método de envío de información empleado (GET) y la página PHP a la que deben ser enviados (Figura 3.26).

```
fwrite($resultado_xhtml,'<form name="send_answers" method="get"
action="grammar_answers.php">');
```

Figura 3.26: Ejemplo de declaración de formulario XHTML

Para la introducción de datos se emplean los típicos elementos *form* de HTML, seleccionando en cada caso el más adecuado, claro y cómodo (Figura 3.27).

Existing Topics:

present perfect-simple past ▾ → **Option**

Predicate:  → **Text Box**

Level:

Easy

Medium

Hard → **Radio Button**

Figura 3.27: Ejemplo de diferentes métodos de entrada de datos

El resto de variables que necesitan ser enviadas a la siguiente página y que no son introducidas por el usuario (puesto que son variables internas de control) y no directamente datos solicitados en el formulario, son guardadas como variables de sesión tal y como fue explicado al principio de este apartado.

El envío de los datos del formulario se realiza mediante un botón que realiza la operación *submit*. En caso de que el formulario haya sido rellenado a través de la voz no será necesario pulsar este botón. Automáticamente saltará a la siguiente página una vez hayan sido completados todos los campos requeridos (Figura 3.28).

```
fwrite($resultado_xhtml,'<input type="submit" value="Corregir"
class="button"/>');
```

Figura 3.28: Declaración de botón *submit* para el paso de variables

- **Header:** La función Header carga la página indicada una vez se ha terminado la ejecución del código de la página actual. Puesto que el cometido de la página PHP es escribir la página resultante XHTML, una vez escrita por completo se ejecuta el *header* para que sea cargada por el navegador (Figura 3.29).

```
header("location: http://localhost/grammar_exercise.xhtml");
(Llamada a la página que acaba de ser escrita por la página PHP).
```

Figura 3.29: Carga de la página resultante mediante comando header

3. **Estilos:** La maquetación de la aplicación ha sido programada en su totalidad empleando un único fichero externo de estilos "estilos\_contents.css" en donde se encuentran definidos todos los formatos empleados. Como ya se comentó en el apartado anterior la

declaración del fichero de estilos se realiza en la cabecera <HEAD> de cada una de las páginas (Figura 3.30).

```
fwrite($resultado_xhtml,'<head>');  
fwrite($resultado_xhtml,'<meta http-equiv="Content-Type"  
content="text/html; charset=utf-8" />');  
fwrite($resultado_xhtml,'<meta name="language" content="es"/>');  
fwrite($resultado_xhtml,'<link rel="stylesheet" type="text/css"  
href="estilos_contents.css" media="screen"/>');
```

Figura 3.30: Declaración del fichero de estilos en la cabecera de la sección HEAD

Para ver más claramente donde emplear cada estilo, y puesto que la página sigue un patrón claro de formatos y estilos, se emplean nombres personalizados para estos.

Se trata de una opción que da CSS para luego hacer más sencilla la llamada a cada estilo en la página. La Figura 3.31 muestra algunos formatos a modo de ejemplo para que pueda ser entendida la configuración y los parámetros empleados.

```
body (Formato que afecta a todo el contenido incluido en la sección <body>)  
{  
    margin:      0px;  
    padding:    0px;  
    background: #000000 url(images/imgo6.jpg);  
    font:       13px Arial, Helvetica, sans-serif;  
}  
  
body h2 (Formato que afecta a los títulos H2 incluidos en la sección <body>)  
{  
    font-family: Verdana, Arial, Sans-Serif;  
    font-size:   16pt;  
    font-weight: bold;  
    text-align:  center;  
    text-decoration: underline;  
}
```

```
#titulo (Formato que afecta a los elementos contenidos en el <DIV> titulo)
{
    padding:    opx opx opx 2opx;
    font-style: italic;
    font-weight: bold;
    font-size:  2opx;
    font-family: Brush Script MT;
    color:      white;
}

#description (Formato que afecta a los elementos contenidos en el <DIV>
description)
{
    padding:    opx opx opx 2opx;
    display:    block;
    font-family: Georgia, "Times New Roman", Times, serif;
    font-size:  18px;
    font-style: italic;
    font-weight: normal;
    text-decoration: none;
}

#instructions (Formato que afecta a los elementos contenidos en el <DIV>
instructions)
{
    padding:    opx opx opx 2opx;
    display:    block;
    font-family: Georgia, "Times New Roman", Times, serif;
    font-size:  2opx;
    font-style: italic;
    font-weight: normal;
    text-decoration: none;
    color: #000000;
}
```

Figura 3.31: Ejemplos de declaración de estilos en fichero externo



Es fácil reconocer las llamadas a estilos en el código de programación de las páginas. Como se ha visto en los ejemplos, pueden afectar a secciones enteras del código delimitadas por pestañas o por secciones más concretas delimitadas a mano mediante contenedores <div>. Lo más frecuente es que sean llamados empleando estos últimos. La Figura 3.32 muestra dos ejemplos del uso de dos contenedores definidos anteriormente y utilizados prácticamente en la totalidad de las páginas:

```
fwrite($resultado_xhtml,'<div id="titulo">'); //Llamada al formato o estilo "titulo".
    fwrite($resultado_xhtml,'<h1>');
    fwrite($resultado_xhtml,'GRAMMAR EXERCISE');
    fwrite($resultado_xhtml,'</h1>');
fwrite($resultado_xhtml,'</div>');

fwrite($resultado_xhtml,'<div id="instructions">'); //Llamada al formato o estilo
"instructions".
    fwrite($resultado_xhtml,'<p>');
    fwrite($resultado_xhtml,'Fill the blanks using the verbs between brackets
in past perfect.');
```

Figura 3.32: Ejemplo de llamadas a diferentes estilos

4. **Bases de Datos:** Como ya se ha descrito en el apartado anterior, ha sido empleado MySQL como SGBD para el conjunto de toda la aplicación. A través de phpMyAdmin se han llevado a cabo todas las tareas de creación de tablas, columnas, claves, propiedades de los campos, pruebas de consulta así como todas las tareas de edición. Para la conexión (inicio de instancia en MySQL) y copias de seguridad se ha empleado el software MySQL Administrator.

En este apartado se muestra la estructura, relaciones y propiedades generales de todas las tablas que componen la base de datos. Será en el próximo capítulo en el que se detallará para cada módulo en concreto el uso e interacción específica con tablas que realiza.

**Tablas:** Para el conjunto de la aplicación se han hecho uso de un total de siete tablas. A continuación se agrupan por familia y se comentan sus características principales:

- Tablas de ejercicios: Son las empleadas para guardar los contenidos de todos los ejercicios. Pertenecen a esta familia las siguientes tablas:
  - Tabla grammar\_ex: Contiene todos los datos referentes a los ejercicios de gramática. Presenta los campos mostrados en la Figura 3.33.

#	Columna	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra	Acción
1	ID_ex	int(11)			No	Ninguna	AUTO_INCREMENT	Cambiar Eliminar Más
2	categoria	varchar(35)	latin1_swedish_ci		Sí	NULL		Cambiar Eliminar Más
3	nivel	varchar(10)	latin1_swedish_ci		Sí	NULL		Cambiar Eliminar Más
4	texto1	varchar(100)	latin1_swedish_ci		Sí	NULL		Cambiar Eliminar Más
5	verbo	varchar(15)	latin1_swedish_ci		Sí	NULL		Cambiar Eliminar Más
6	texto2	varchar(100)	latin1_swedish_ci		Sí	NULL		Cambiar Eliminar Más
7	solucion	varchar(15)	latin1_swedish_ci		Sí	NULL		Cambiar Eliminar Más
8	opciones	varchar(200)	latin1_swedish_ci		No	Ninguna		Cambiar Eliminar Más

Figura 3.33: Vista de la tabla "grammar\_ex"

Queda identificada por una clave primaria (*ID\_ex*) numérica y autoincremental.

- Tabla vocabulary\_ex: Contiene todos los datos referentes a los ejercicios de vocabulario. Presenta los campos mostrados en la Figura 3.34.

#	Columna	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra	Acción
1	ID_im	int(11)			No	Ninguna	AUTO_INCREMENT	Cambiar Eliminar Más
2	nombre	varchar(40)	latin1_swedish_ci		No	Ninguna		Cambiar Eliminar Más
3	solucion	varchar(30)	latin1_swedish_ci		No	Ninguna		Cambiar Eliminar Más
4	nivel	varchar(15)	latin1_swedish_ci		No	Ninguna		Cambiar Eliminar Más
5	categoria	varchar(40)	latin1_swedish_ci		No	Ninguna		Cambiar Eliminar Más

Figura 3.34: Vista de la tabla "vocabulary\_ex"

Queda identificada por una clave primaria (*ID\_im*) numérica y autoincremental.

Con el fin de optimizar la estructura de los datos de los ejercicios de tipo *listening* se emplean dos tablas en lugar de una sola tabla como era en el caso de gramática y vocabulario. Puesto que cada texto de *listening* presenta varios ejercicios, se ha organizado la información en dos tablas relacionadas (Figuras 3.35 y 3.36).

- Tabla *listening\_texts*: Esta tabla contiene la información de los textos de listening. Presenta los campos mostrados en la Figura 3.35.

#	Columna	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra	Acción
1	<u>ID_texto</u>	int(5)			No	Ninguna	AUTO_INCREMENT	Cambiar Eliminar Más
2	categoria	varchar(30)	latin1_swedish_ci		No	Ninguna		Cambiar Eliminar Más
3	fichero	varchar(30)	latin1_swedish_ci		No	Ninguna		Cambiar Eliminar Más
4	nivel	varchar(15)	latin1_swedish_ci		No	Ninguna		Cambiar Eliminar Más

Figura 3.35: Vista de la tabla "*listening\_texts*"

Identifica a cada texto mediante la clave primaria (*ID\_texto*) de manera numérica y autoincremental.

1. Tabla *listening\_ex*: Esta tabla contiene la información de cada uno de los ejercicios de *listening* (preguntas), relacionándolo a su vez con el texto al que hace referencia. Presenta los campos mostrados en la Figura 3.36.

#	Columna	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra	Acción
1	<u>ID_texto</u>	int(5)			No	Ninguna		Cambiar Eliminar Más
2	<u>ID_ex</u>	int(5)			No	Ninguna	AUTO_INCREMENT	Cambiar Eliminar Más
3	texto	varchar(100)	latin1_swedish_ci		No	Ninguna		Cambiar Eliminar Más
4	solucion	varchar(20)	latin1_swedish_ci		No	Ninguna		Cambiar Eliminar Más
5	nivel	varchar(15)	latin1_swedish_ci		No	Ninguna		Cambiar Eliminar Más

Figura 3.36: Vista de la tabla "*listening\_ex*"

Cada ejercicio o pregunta se encuentra identificada mediante una clave primaria (*ID\_ex*) numérica y autoincremental.

- Tabla de usuarios (users): Guarda toda la información de los usuarios registrados compuesta por los campos mostrados en la Figura 3.37.



#	Columna	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra	Acción
<input type="checkbox"/>	1 <b>ID_us</b>	int(11)			No	Ninguna	AUTO_INCREMENT	Cambiar  Eliminar  Más
<input type="checkbox"/>	2 <b>nombre</b>	varchar(50)	latin1_swedish_ci		No	Ninguna		Cambiar  Eliminar  Más
<input type="checkbox"/>	3 <b>apellidos</b>	varchar(50)	latin1_swedish_ci		No	Ninguna		Cambiar  Eliminar  Más
<input type="checkbox"/>	4 <b>nick</b>	varchar(20)	latin1_swedish_ci		No	Ninguna		Cambiar  Eliminar  Más
<input type="checkbox"/>	5 <b>password</b>	varchar(40)	latin1_swedish_ci		No	Ninguna		Cambiar  Eliminar  Más
<input type="checkbox"/>	6 <b>fecha</b>	varchar(40)	latin1_swedish_ci		No	Ninguna		Cambiar  Eliminar  Más
<input type="checkbox"/>	7 <b>sexo</b>	varchar(10)	latin1_swedish_ci		No	Ninguna		Cambiar  Eliminar  Más
<input type="checkbox"/>	8 <b>nivel</b>	varchar(10)	latin1_swedish_ci		No	Ninguna		Cambiar  Eliminar  Más

Figura 3.37: Vista de la tabla "users"

Cada usuario se encuentra identificado por una clave primaria (*ID\_us*) numérica y autoincremental. Cuando el usuario se registra estima en el formulario de registro el nivel de inglés que posee. Esta información queda guardada en el campo "nivel" y será empleada con distintos fines que serán explicados en el siguiente capítulo.

- Tabla de respuestas (answers): Contiene información de las respuestas que ha realizado cada usuario en los ejercicios. Las consultas a esta tabla permiten luego crear las páginas de corrección de manera rápida y efectiva (Figura 3.38).



#	Columna	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra	Acción
<input type="checkbox"/>	1 <u>ID_us</u>	int(11)			No	Ninguna		Cambiar  Eliminar Más ▼
<input type="checkbox"/>	2 <u>ID_ex</u>	int(11)			No	Ninguna		Cambiar  Eliminar Más ▼
<input type="checkbox"/>	3 <u>introducido</u>	varchar(50)	latin1_swedish_ci		No	Ninguna		Cambiar  Eliminar Más ▼
<input type="checkbox"/>	4 <u>correcto</u>	tinyint(1)			No	Ninguna		Cambiar  Eliminar Más ▼
<input type="checkbox"/>	5 <u>familia</u>	varchar(20)	latin1_swedish_ci		No	Ninguna		Cambiar  Eliminar Más ▼

Figura 3.38: Vista de la tabla "answers"

Puesto que para cada familia existen diferentes ejercicios y un usuario realiza varios ejercicios de cada familia, la clave primaria queda formada por el identificador del usuario, el identificador del ejercicio y la familia. El campo booleano "correcto" guarda si ese ejercicio fue correctamente respondido o no.

- Tabla de resultados (results): Guarda información numérica acerca de los ejercicios realizados por un usuario y cuántos de ellos son correctos. Dicha información es empleada por el módulo "my\_results" para realizar los datos estadísticos, comparativos y de resultados de cada usuario. Guarda los campos de datos mostrados en la Figura 3.39.

#	Columna	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra	Acción
<input type="checkbox"/>	1 <u>num_consulta</u>	int(11)			No	Ninguna		Cambiar  Eliminar Más ▼
<input type="checkbox"/>	2 <u>ID_us</u>	int(11)			No	Ninguna		Cambiar  Eliminar Más ▼
<input type="checkbox"/>	3 <u>fecha</u>	datetime			No	Ninguna		Cambiar  Eliminar Más ▼
<input type="checkbox"/>	4 <u>gramm_easy</u>	int(11)			No	Ninguna		Cambiar  Eliminar Más ▼
<input type="checkbox"/>	5 <u>gramm_med</u>	int(11)			No	Ninguna		Cambiar  Eliminar Más ▼
<input type="checkbox"/>	6 <u>gramm_hard</u>	int(11)			No	Ninguna		Cambiar  Eliminar Más ▼
<input type="checkbox"/>	7 <u>voc_easy</u>	int(11)			No	Ninguna		Cambiar  Eliminar Más ▼
<input type="checkbox"/>	8 <u>voc_med</u>	int(11)			No	Ninguna		Cambiar  Eliminar Más ▼
<input type="checkbox"/>	9 <u>voc_hard</u>	int(11)			No	Ninguna		Cambiar  Eliminar Más ▼
<input type="checkbox"/>	10 <u>listen_easy</u>	int(11)			No	Ninguna		Cambiar  Eliminar Más ▼
<input type="checkbox"/>	11 <u>listen_med</u>	int(11)			No	Ninguna		Cambiar  Eliminar Más ▼
<input type="checkbox"/>	12 <u>listen_hard</u>	int(11)			No	Ninguna		Cambiar  Eliminar Más ▼
<input type="checkbox"/>	13 <u>done</u>	int(11)			No	Ninguna		Cambiar  Eliminar Más ▼
<input type="checkbox"/>	14 <u>score</u>	int(11)			No	Ninguna		Cambiar  Eliminar Más ▼

Figura 3.39: Vista de la tabla results

La tabla queda identificada por el identificador de usuario (*ID\_us*) y un identificador numérico (*num\_consulta*) cuya finalidad será explicada en el siguiente capítulo cuando se describa el módulo *my\_results*. Los campos "*done*" y "*score*" serán igualmente explicados en ese módulo.

**Relaciones e Integridad referencial:** La implementación del módulo de administración permite que las creaciones, ediciones y borrados de ejercicios se gestionen a través de la aplicación, permitiendo que estas operaciones se lleven a cabo a través de formularios, lo que supone que no es necesario que el cliente-administrador conozca y haga uso de comandos u operaciones de MySQL (dichas operaciones van por debajo y son transparentes al usuario). La programación llevada a cabo a través de dicho módulo de administración contempla y trata la integridad referencial de las bases de datos de ejercicios frente a borrados y modificaciones, sin embargo, puesto que la administración de usuarios no está contemplada en el módulo, ha sido necesario programar a nivel SQL las propiedades que aseguren la integridad de los datos que utilizan. Para lograr dicha integridad cabe contemplar las siguientes relaciones y propiedades:

- Relación Usuarios-respuestas (Figura 3.40): El identificador de la tabla respuestas "answers" (*ID\_us*) es clave ajena de la tabla de usuarios "users". Esta relación lleva asociada una restricción de borrado y modificación en cascada, lo que asegura que si se borra o modifica un usuario o sus datos se borran todas sus respuestas.



Figura 3.40: Relación tablas users-answers

- Relación usuarios-resultados (Figura 3.41): El identificador de la tabla de resultados "results" (*ID\_us*) es clave ajena de la tabla de usuarios "users". Al igual que en el caso anterior esta relación lleva asociada una restricción de borrado y modificación en cascada que asegura que si se borra un usuario se borran sus resultados.



Figura 3.41: Relación tablas users-results

**Copias de seguridad:** Pese a que la mayoría de operaciones a realizar sobre las bases de datos pueden ser llevadas a cabo a través de PHPMyAdmin, ciertas funciones como la realización de copias de seguridad o el inicio de instancias de servidor MySQL considero que son más sencillas y completas de realizar mediante el software MySQL Administrator (Figura 3.42). En concreto phpMyAdmin permite exportar a diversos formatos y con opciones configurables nuestra Base de Datos para realizar una copia que más tarde puede ser importada en caso de recuperación. MySQL Administrator permite además la programación automática de esta tarea, lo que presenta una gran ventaja teniendo en cuenta que si existen muchos movimientos de estructura y datos en las tablas, esta tarea debería llevarse a cabo con una frecuencia al menos diaria.

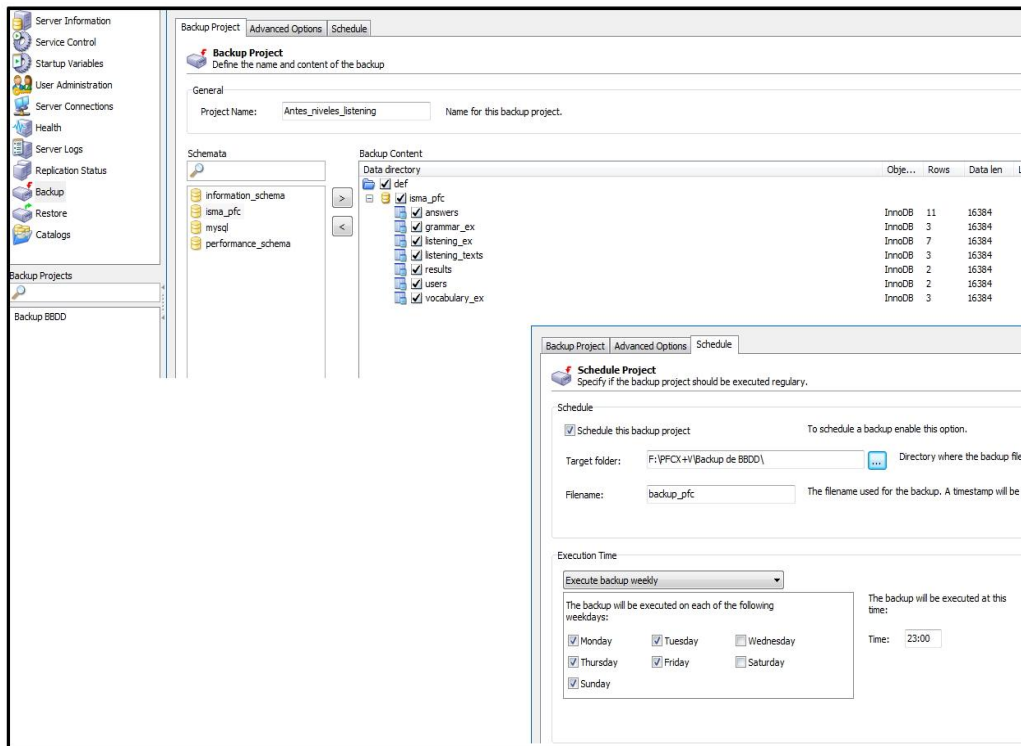


Figura 3.42: Opciones de programación de backups en MySQL Administrator



## Capítulo 4: Descripción detallada de los módulos del Sistema

El capítulo anterior reunía una serie de características de estructura y funcionalidad comunes a toda la aplicación, así como el conjunto de herramientas que ésta utiliza. El presente capítulo detalla cada módulo en concreto de forma que quede clara la funcionalidad, la arquitectura, el funcionamiento, la interacción con las bases de datos así como la presentación de diferentes escenarios de uso. Se describen los siguientes módulos:

### 1. Página principal o HOME

- 1.1. Funcionalidad: Es la página con la que se abre la aplicación y a través de la cual se accede a cualquiera de las herramientas y módulos de que ésta dispone. Se trata de una página dinámica y por tanto generada a partir de un fichero PHP. El motivo por el que es dinámica consiste en que su contenido cambia en función de si hay un usuario que ha iniciado sesión en la aplicación o no. Actúa como menú de las posibles opciones a llevar a cabo y no sólo se presenta inicialmente sino que es posible que la aplicación nos devuelva automáticamente a ésta página al llegar a la finalización del módulo escogido, con el objetivo de seleccionar una nueva operación a llevar a cabo (Figura 4.1).

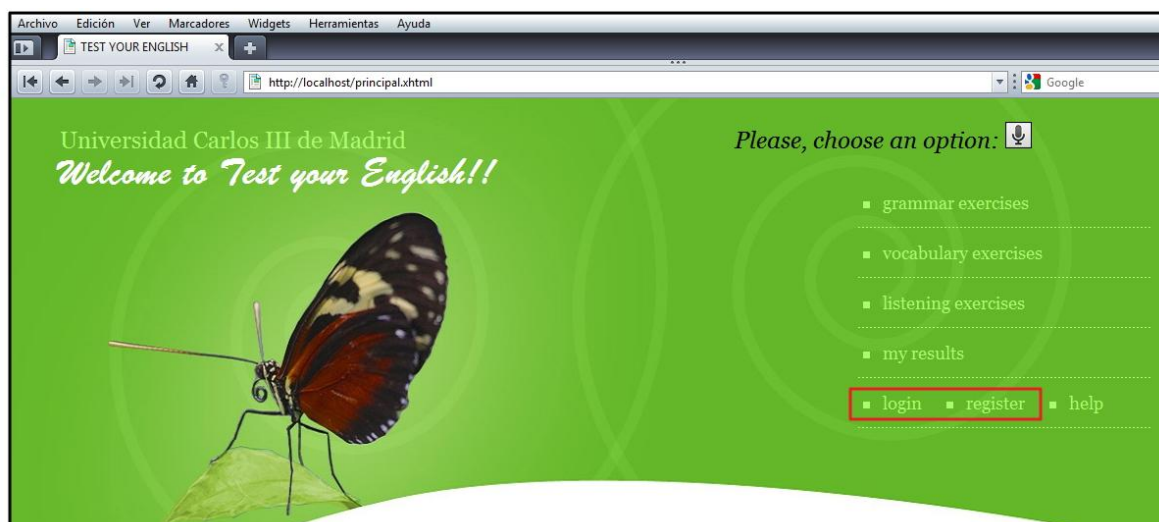


Figura 4.1: Página principal o HOME

La imagen muestra la apariencia y opciones de la página principal antes de realizar el inicio de sesión, modificándose entonces por el menú de opciones presentado en la Figura 4.2.

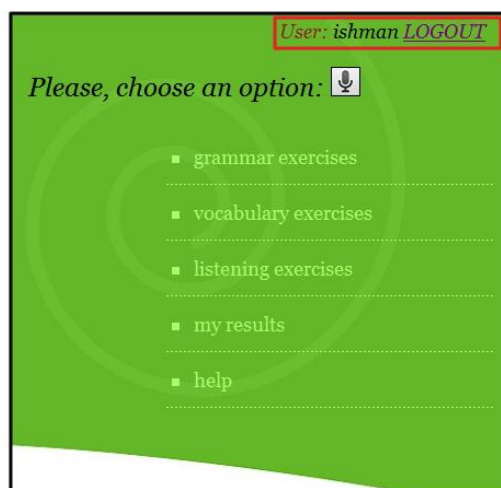


Figura 4.2: Modo usuario activo con opción de cierre de sesión

Desaparecen las opciones de iniciar sesión y registro y puede verse en la parte superior derecha de la pantalla el *nick* del usuario junto con la opción de realizar el *LOGOUT* para finalizar la sesión.

- 1.2. Arquitectura: Puesto que se trata de una página dinámica es generada por el fichero "**principal.php**" el cual se encarga de preguntar si la variable de sesión que almacena el usuario activo presenta contenido. En función de dicho análisis escribirá la página con el contenido presentado en la primera imagen o en la segunda en caso de que haya usuario con sesión iniciada. Dicho análisis se lleva a cabo mediante el código presentado en la Figura 4.3.

```
if (isset($_SESSION[idusuario])) //Existe un usuario activo. Se muestra el
nombre y la opción de cerrar sesión (logout).
{
    $idusuario = $_SESSION[idusuario];
    $nick = $_SESSION[nick];
    $logado=1;
}
else $logado=0; //Se muestra la opción login y registro.
```

Figura 4.3: Procedimiento de verificación de usuario activo

Dicho algoritmo de comprobación de *login* será utilizado en muchas otras páginas con diferentes fines, como será explicado de manera concreta en cada caso.

Puesto que la página únicamente actúa como menú principal de opciones, su programación visual se reduce a texto y enlaces de navegación a otras páginas. Puede ser controlada por voz y admite como entrada cualquiera de las opciones del menú. La opción *LOGOUT* realiza un borrado de todas las variables de sesión tal y como se explicó en el capítulo anterior, a través de comando "*session\_destroy()*".

## 2. Módulos de registro e inicio de sesión

2.1. Funcionalidad: Como ya se mencionó en capítulos anteriores, el proceso de registro e inicio de sesión de usuario es esencial y necesario para poder llevar a cabo la totalidad de funciones que presenta la aplicación. No es posible completar un histórico de resultados si la aplicación no sabe qué usuario la está utilizando, por lo que a la hora de realizar los ejercicios y consultar las respuestas o nuestras estadísticas el sistema nos exigirá antes que hayamos iniciado sesión (y por tanto estemos previamente registrados).

En caso de querer acceder a cualquiera de las categorías de ejercicios o al módulo de resultados sin haber iniciado previamente sesión, obtendremos como resultado la ventana mostrada en la Figura 4.4 que nos redirigirá a la página de inicio de sesión con el fin de que podamos completar el proceso.

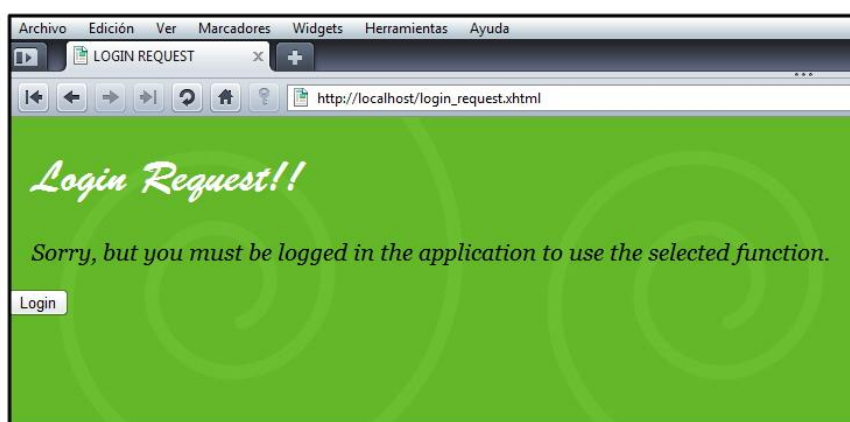
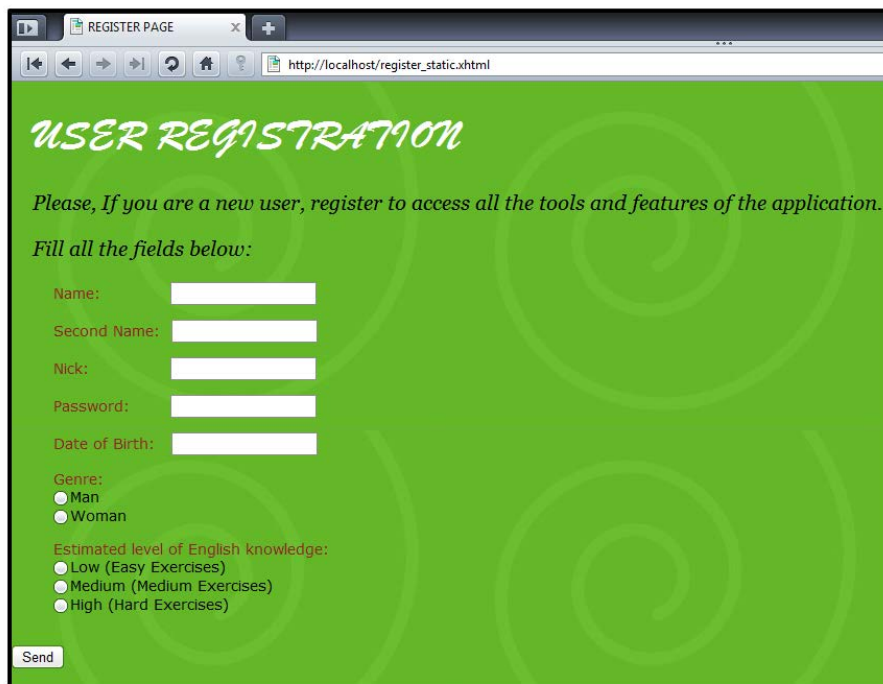


Figura 4.4: Pantalla de aviso de inicio de sesión requerido

Ambas operaciones se llevan a cabo a través de sencillos formularios que requieren la siguiente información:

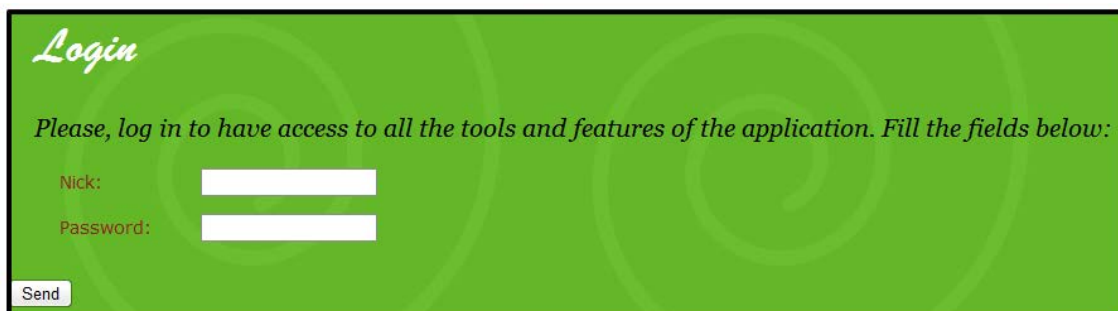
- *Formulario de Registro:* Con el fin de que la creación de usuario no resulte un proceso engorroso, se han minimizado los datos solicitados a los estrictamente necesarios para la aplicación (Figura 4.5).



The screenshot shows a web browser window with the title 'REGISTER PAGE' and the URL 'http://localhost/register\_static.xhtml'. The page content is on a green background with the heading 'USER REGISTRATION' in a stylized font. Below the heading is a message: 'Please, If you are a new user, register to access all the tools and features of the application.' followed by 'Fill all the fields below:'. The form fields are: 'Name:' with a text input, 'Second Name:' with a text input, 'Nick:' with a text input, 'Password:' with a text input, and 'Date of Birth:' with a text input. Below these are 'Genre:' with radio buttons for 'Man' and 'Woman', and 'Estimated level of English knowledge:' with radio buttons for 'Low (Easy Exercises)', 'Medium (Medium Exercises)', and 'High (Hard Exercises)'. A 'Send' button is located at the bottom left of the form area.

Figura 4.5: Formulario de registro de usuario

- *Formulario de inicio de sesión:* De todos los campos que componen la información de un usuario, el proceso de inicio de sesión sólo requiere del *nick* y el *password* para llevarse a cabo (Figura 4.6).



The screenshot shows a web browser window with the title 'REGISTER PAGE' and the URL 'http://localhost/register\_static.xhtml'. The page content is on a green background with the heading 'Login' in a stylized font. Below the heading is a message: 'Please, log in to have access to all the tools and features of the application. Fill the fields below:'. The form fields are: 'Nick:' with a text input and 'Password:' with a text input. A 'Send' button is located at the bottom left of the form area.

Figura 4.6: Formulario de login de usuario

2.2. Arquitectura y funcionamiento: Puesto que ambos formularios solicitan la información mínima exigible, es necesario que una vez realizado el *submit* unas páginas PHP intermedias (**login\_process.php** y **register\_process.php**) analicen si todos los campos han sido rellenados (el único campo opcional es el Apellido "*Second Name*" en el formulario de registro). Si no es así, se retornará al formulario informando de la ausencia de dicha información y solicitando que sea completada. En el caso de que todos los campos obligatorios hayan sido completados, estas páginas se encargan también de realizar una validación de los datos introducidos que realiza el siguiente análisis:

#### Registro:

- *Nick Único*: El campo nick rellenado por el usuario debe presentar un nombre que no haya sido usado con anterioridad por otro usuario del sistema. Aunque este campo no es el identificador (ya que la clave primaria queda formada por un valor numérico autoincremental) será controlado en la base de datos como campo único (Figura 4.7).

```
$resultado = mysql_query("SELECT * FROM users WHERE nick= '$nick';", $conexion);
$numero = mysql_num_rows($resultado); //Tuplas devueltas por la consulta.
if($numero != 0) //Ese nick ya existe luego no es válido.
{
    <Se crea texto informativo de error que será mostrado en la nueva carga
    del formulario con el fin de que sea corregido>
}
```

Figura 4.7: Validación del nick registrado por el usuario

- *Longitud de contraseña*: Por motivos de seguridad, la contraseña debe estar formada por al menos seis caracteres y no superar los doce caracteres alfanuméricos. El sistema no dejará usar una clave más corta ni que supere el límite de longitud (ver Figura 4.8).

```
$longitud=strlen($contrasena);  
if($longitud < 6)  
{  
    <Se crea texto informativo de error que será mostrado en la nueva carga  
    del formulario con el fin de que sea corregido>  
  
}if($longitud > 12){  
    <Se crea texto informativo de error que será mostrado en la nueva  
    carga del formulario con el fin de corregir el error>}  
}
```

Figura 4.8: Validación de la clave registrada por el usuario

- *Validación de fecha:* Los formatos admitidos serán (dd/mm/aaaa), (dd-mm-aaaa) o bien (dd.mm.aaaa). Cualquier otro formato no será aceptado por la aplicación.

#### Login:

- *Usuario existente y password correcta:* El sistema busca el nick introducido en la base de datos y comprueba que dicho nombre existe. A continuación verifica que el password introducido sea correcto (Figura 4.9).

```
$resultado = mysql_query("SELECT * FROM users WHERE nick= '$nick';", $conexion);  
$numero = mysql_num_rows($resultado); //Tuplas resultantes de la consulta  
$registro = mysql_fetch_array($resultado);  
if($numero == 0)//Ese nick no existe.  
{  
    <Se crea texto informativo de error que será mostrado en la nueva carga  
    del formulario con el fin de que sea corregido>  
}  
else  
{  
    if($contrasena!=$registro['password']) //El password introducido no es  
    correcto.  
    {  
        <Se crea texto informativo de error que será mostrado en la  
        nueva carga del formulario con el fin de que sea corregido>  
    }  
}  
}
```

Figura 4.9: Validación de login y password en proceso de inicio de sesión

Tanto en el proceso de inicio de sesión como en el de registro, en caso de que todas las condiciones anteriores hayan sido satisfechas los datos serán procesados. En ese caso se llamará al fichero "**operation\_done.php**", que se encargará de generar una página XHTML en la que se nos informará mediante un mensaje por pantalla y voz de que la operación concreta ha sido llevada a cabo con éxito (Figura 4.10).

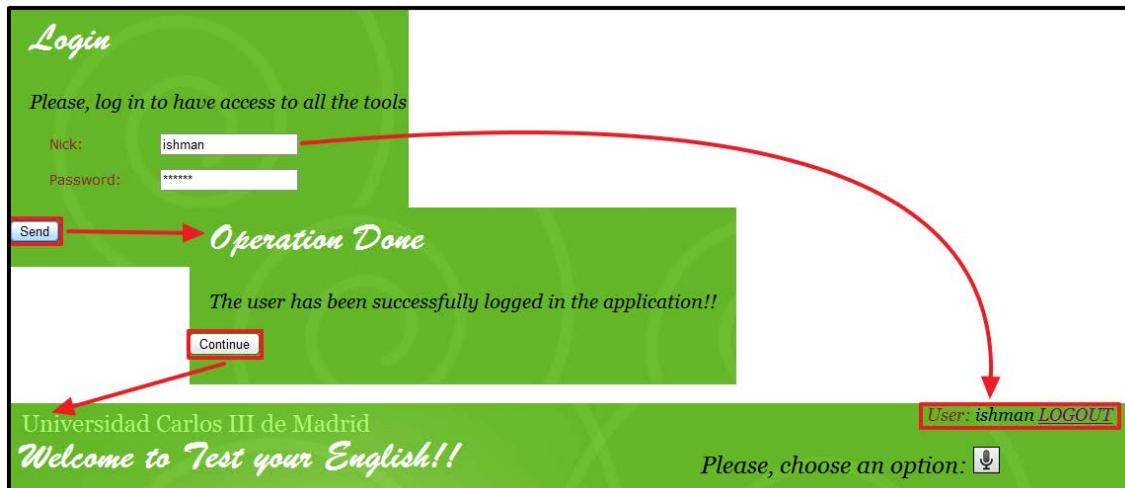


Figura 4.10: Caso inicio de sesión satisfactorio

En caso contrario se retornará al formulario indicando qué campos presentan errores y el problema a corregir (Figura 4.11).



Figura 4.11: Ejemplo de errores en formulario de registro

Este módulo emplea la tabla "users" para almacenar toda la información de usuario solicitada en el formulario de registro. Una vez que todos los campos han sido rellenados y sus valores validados se procede a realizar la inserción en la tabla (Figuras 4.12 y 4.13).

```
mysql_query("INSERT INTO users  
(nombre,apellidos,nick,password,fecha,sexo,nivel,logged) VALUES ('$nombre',  
 '$apellidos', '$nick', '$contrasena', '$fecha', '$sexo', '$nivel');", $conexion);
```

Figura 4.12: Inserción de datos de registro en tabla users



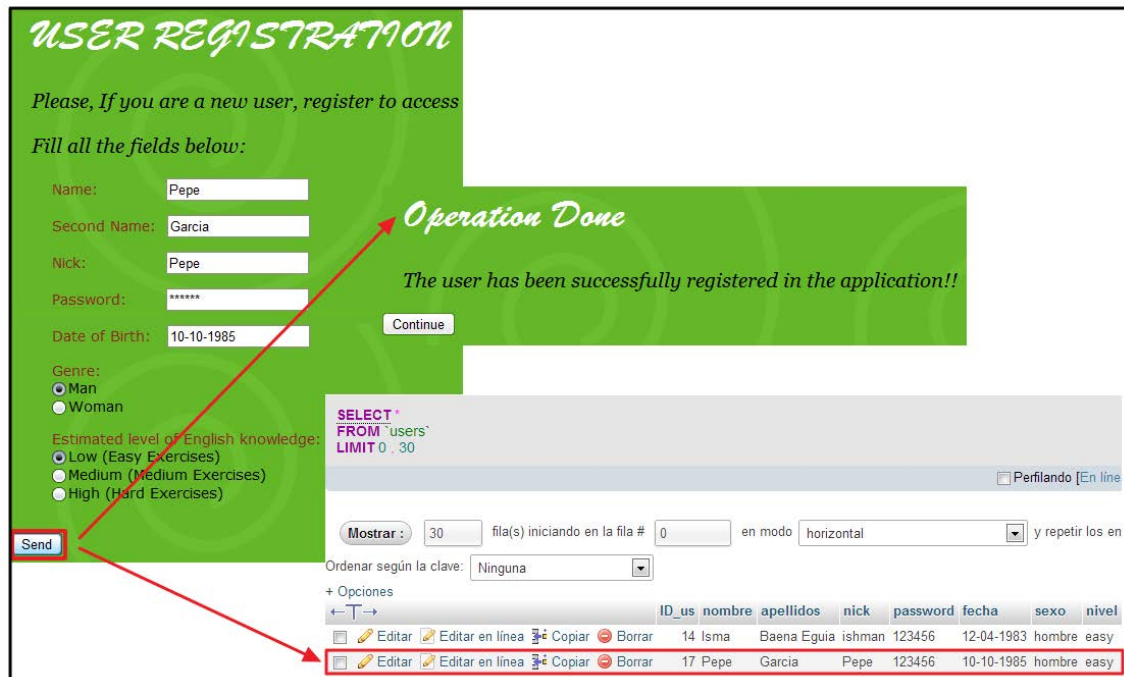


Figura 4.13: Caso registro satisfactorio

En lo que respecta al proceso de inicio de sesión la verificación de nick y contraseña se realiza, tal y como fue explicado en el apartado anterior, mediante una consulta a esta misma tabla "users".

### 3. Módulo de realización de ejercicios

3.1. Funcionalidad: Este módulo incluye tres apartados o pantallas para cada familia de ejercicios a seleccionar (gramática, vocabulario o *listening*):

- *Formulario de selección de tema y nivel*: Se trata de un formulario simple en el que se selecciona la categoría (tiempo verbal en los ejercicios de gramática, temática en los ejercicios de vocabulario y título del texto en los ejercicios de escucha) y el nivel de los ejercicios. Puede ser que para cierta categoría o tema no existan ejercicios del nivel seleccionado en cuyo caso la aplicación nos informaría solicitándonos introducir otra categoría o nivel. En el caso de los *listening* cada texto va asociado a un nivel concreto por lo que no se seleccionan las dos características sino simplemente un texto.

Además de poder escoger entre los diferentes temas y niveles, la aplicación está diseñada para llevar a cabo esta selección por nosotros utilizando la opción "*choose for me*". Los criterios tenidos en cuenta a la hora de automatizar esta selección pueden consultarse en el **Anexo "Choose for Me"**.

- *Formulario de realización de ejercicios seleccionados*: En el cual resolvemos mediante voz o texto los ejercicios propiamente dichos:
    - *Completar* la sentencia con la forma gramatical correspondiente a la familia seleccionada en el caso de ejercicios de gramática.
    - *Resolver* el sustantivo o verbo asociado a una imagen en el caso de vocabulario.
    - *Responder verdadero-falso* a diferentes cuestiones relacionadas con un texto que ha sido escuchado con anterioridad en el caso de *listening*.
  - *Página de resultados del ejercicio realizado*: En la que se nos muestra si hemos respondido de manera acertada o no y la respuesta correcta en caso de que hayamos fallado.
- 3.2. Arquitectura, funcionamiento e interacción con bases de datos: Como ya quedó explicado en el módulo de inicio de sesión, lo primero que se comprueba es que existe un usuario activo. En caso negativo se llama a la página "**login\_request.xhtml**" la cual nos informa de que debemos iniciar sesión y nos direcciona al formulario para que lo hagamos. En caso de ya haber iniciado sesión se procede con los siguientes pasos:
- *Formulario de selección de categoría/texto y nivel*: La página "**select\_contents.php**" es la encargada de consultar la tabla *grammar\_ex* o *vocabulary\_ex* de la base de datos, según el caso, para extraer todas las categorías y niveles existentes (Figura 4.14).

```
$sacarcats = "SELECT DISTINCT(categoria) FROM grammar_ex";  
$sacarniv = "SELECT DISTINCT(nivel) FROM grammar_ex";
```

**Figura 4.14:** Consultas de selección de categorías y niveles existentes

Una vez se tiene esta información guardada en una variable tipo *array* (*\$arraycat*), se utilizan los valores extraídos para crear la gramática interna correspondiente que permite programar la interacción mediante voz por la cual se podrá elegir cualquiera de las categoría o temas y los niveles (Figura 4.15).

```
fwrite($resultado_xhtml,'<vxml:field xv:id="categoria" name="categoria">');
fwrite($resultado_xhtml,'<vxml:prompt>Please select the
category.</vxml:prompt>');
fwrite($resultado_xhtml,'<vxml:grammar><![CDATA[#JSGF V1.0; 'grammar
categories;'];
fwrite($resultado_xhtml,'public <categories> = ');
$i=1;
while($i<=$num_categorias) //Se recorre el array de manera que se van
escribiendo los valores posibles.
{
    fwrite($resultado_xhtml,$arraycat[$i]);
    if($i<$numfilascats) fwrite($resultado_xhtml,'| ');
    $i++;
}
fwrite($resultado_xhtml,';]]>');
fwrite($resultado_xhtml,'</vxml:grammar>');
```

Figura 4.15: Declaración de gramática interna dinámica

Se utiliza un "field" de similar programación para seleccionar los niveles cuya gramática quedará formada por las palabras "easy", "medium" y "hard". De igual modo se utilizan los valores extraídos para crear el formulario de tipo select que será mostrado de manera visual en la página XHTML resultante (Figura 4.16).

```
fwrite($resultado_xhtml,'<div id="formulario">');
fwrite($resultado_xhtml,'<h5>1) Categoría</h5>'); //Titulo
fwrite($resultado_xhtml,'<select name="categoría" id="categoría">');
$i=1;
while ($i<=$num_categorias) //Tantas iteraciones como categorías o temas.
{
    fwrite($resultado_xhtml,'<option value =""');
    fwrite($resultado_xhtml,$arraycat[$i]);
    fwrite($resultado_xhtml,">');
    fwrite($resultado_xhtml,'</option>');
    $i++;
}
fwrite($resultado_xhtml,'</select>');
```

Figura 4.16: Declaración de formulario de selección de categoría

Se emplea un formulario de tipo "select" de similar programación para poder escoger el nivel deseado (Figura 4.17).



The screenshot shows a web interface with a green background. At the top, it says "Grammar Exercises" in a cursive font. Below that, there is a prompt: "Please, select topic and level for the exercises." followed by a microphone icon. The form is divided into two sections: "1) Topics" and "2) Level". Under "1) Topics", there is a dropdown menu currently showing "present perfect-simple past", with a list of options below it: "present perfect-simple past" and "present continuous". Under "2) Level", there is a dropdown menu currently showing "easy", with a list of options below it: "easy" and "hard". At the bottom of the form, there are two buttons: "Send" and "Choose for me!".

Figura 4.17: Vista de formulario de selección de tema y nivel

Para los ejercicios de escucha o *listening* se presenta un contenido de página ligeramente diferente, por lo que lo crea un fichero PHP distinto (**select\_listening.php**), pero con una estructura similar, que realiza las funciones explicadas anteriormente. Básicamente cambian las tablas consultadas para la extracción de categorías, con las que luego se realizará el "field" de selección por voz y un único formulario de tipo radio que permita seleccionar el texto que se desea.

- *Formularios de realización de ejercicios:*

Una vez seleccionada la categoría y el nivel del formulario anterior las páginas **grammar\_options.php**, **vocabulary\_options.php** y **listening\_options.php** son las encargadas de crear la gramática correspondiente y el formulario de resolución de ejercicios. Para ello en primer lugar se consulta a la tabla correspondiente por aquellos ejercicios que cumplan los requisitos de categoría y nivel seleccionados en el formulario anterior (Figura 4.18).

```
$sacar = "SELECT * FROM grammar_ex WHERE categoría = '$categoría' AND nivel = '$nivel';  
$resultado = mysql_query($sacar,$conexion);
```

**Figura 4.18:** Consulta de selección de ejercicios de tema y nivel introducido

Si el resultado devuelto por dicha consulta fuese de cero tuplas se nos redirigiría de nuevo al formulario anterior de selección de categoría y nivel informándonos de que actualmente no existen ejercicios que cumplan esas características.

En caso de que la consulta devuelva resultados se guardarán en un array los identificadores de ejercicio (*ID\_ex*) que cumplen las características. Conociendo el identificador ya tenemos acceso al resto de información de ese ejercicio. Cada uno de los ejercicios lleva asociado un conjunto de opciones posibles a responder las cuales se encuentran en el campo "opciones" de su tabla correspondiente. Obviamente sólo una de las opciones será la respuesta correcta sin embargo el sistema ha de reconocer de manera dictada cualquiera de ellas. La programación de voz consiste por tanto en crear tantos "*fields*" como ejercicios haya. Estos emplearán una gramática externa formada por el contenido del campo "opciones" de todos los ejercicios seleccionados.

Por último, se genera el formulario con los *textbox* que permiten al usuario rellenar la respuesta. En el caso de gramática, junto a la frase para completar se indica para cada ejercicio las posibles soluciones a escribir o dictar mediante el uso de voz (Figura 4.19).

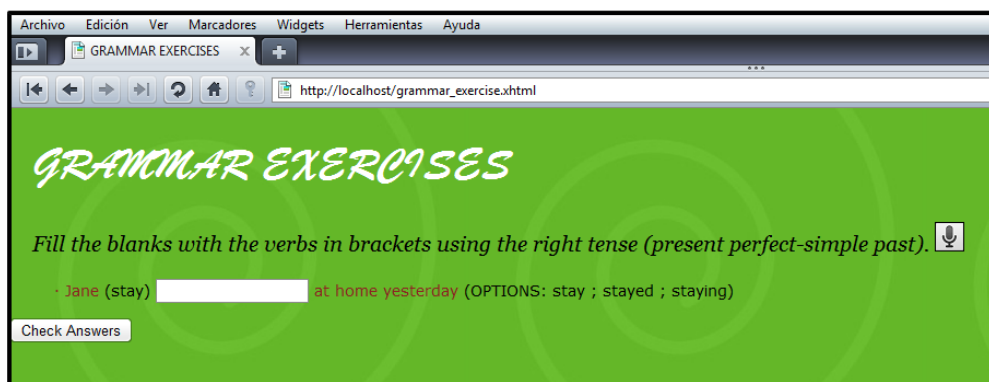


Figura 4.19: Vista de página de resolución de ejercicios de gramática

Las páginas *vocabulary\_options.php* y *listening\_options.php* realizan las mismas funciones y presentan una estructura similar. Lo que cambia es básicamente la manera de presentar de forma visual el formulario de resolución. En el caso de los ejercicios de vocabulario cada página muestra únicamente un ejercicio. Tras rellenar cada palabra realizaremos un *submit* de ella hasta que se llegue a la última palabra que cambiará la opción de pasar al siguiente por la opción "*Check Answers*" para pasar a la corrección (Figura 4.20).

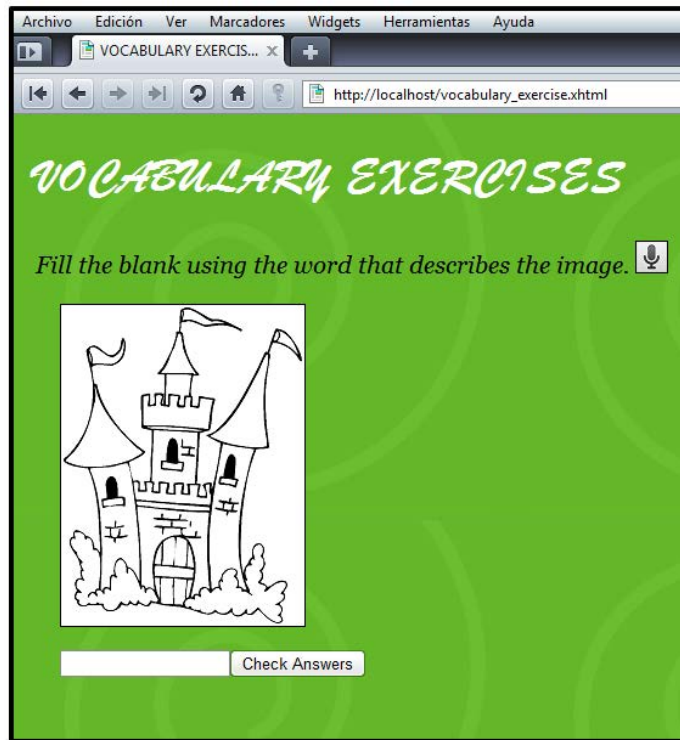


Figura 4.20: Vista de página de resolución de ejercicios de vocabulario

Los ejercicios de escucha son mostrados en dos pasos o páginas. La primera página nos advierte de que el inicio de la locución del texto empezará tan pronto pulsemos el botón "Start Listening" o digamos por voz la palabra "Start". Con esto se consigue que el usuario comience la escucha en el momento que desee y cuando esté preparado. Una vez pulsemos o digamos la palabra, se cambiará a una nueva página que comenzará el ejercicio nada mas cargar y que incluye además las preguntas de test a resolver. La tabla "listening\_text" presenta un campo "fichero" que contiene el nombre del fichero que contiene cada texto. Basta con volcar el contenido de ese fichero en una variable ("texto\_contenido") y a continuación realizar una lectura de ese contenido nada más cargar la parte de voz de la página (Figuras 4.21 y 4.22).

```
fwrite($resultado_xhtml,'<vxml:form id="getMessageInfo">');  
fwrite($resultado_xhtml,'<vxml:block>'); //Lee directamente el contenido.  
fwrite($resultado_xhtml,$texto_contenido);  
fwrite($resultado_xhtml,'</vxml:block>');
```

Figura 4.21: Sección de lectura del fichero *listening*



Figura 4.22: Página de escucha y resolución del *listening*

- *Página de resultados del ejercicio realizado.*

Una vez se ha terminado el ejercicio completo y se ha pulsado corregir, son las páginas `grammar_answers.php`, `vocabulary_answers.php` y `listening_answers.php` las encargadas de generar la página resultante en la que se mostrará si las respuestas introducidas son las correctas. En caso de no serlo nos indicará cuales son. Para poder proporcionar estos resultados, dichas páginas PHP llevan a cabo las siguientes funciones:

En primer lugar se extrae el identificador del usuario que ha realizado el ejercicio (el usuario con la sesión iniciada). El motivo es que el siguiente paso consiste en introducir en la tabla "answers" todas sus respuestas. Si el usuario ya había realizado con anterioridad alguno de los ejercicios se realiza un *UPDATE* de esa tupla actualizando la antigua respuesta con la nueva. Si es la primera vez que realiza el ejercicio lo que se lleva a cabo una nueva inserción en la tabla (Figura 4.23).



```
while($i<=$numfilas) //$numfilas contiene el número total de ejercicios
realizados.
$хid=$idarray[$i];      //$idarray es un array que contiene los identificadores de los
ejercicios.
$resultado = mysql_query("SELECT * FROM answers WHERE ID_us= '$idusuario'
AND ID_ex= '$хid' AND familia= 'grammar';", $conexion); //Se consulta si el
usuario ya había hecho ese ejercicio antes.

$numero = mysql_num_rows($resultado); //Tuplas resultantes de la consulta.
if($numero == 0)        //Caso de ser la primera vez que realiza ese ejercicio.
mysql_query("INSERT INTO answers (ID_us, ID_ex, introducido, correcto, familia)
VALUES ('$idusuario', '$хid', '$cadena[$i]', '0', 'grammar');", $conexion);
//INSERCION

else //Ya había realizado ese ejercicio con anterioridad.
mysql_query("UPDATE answers SET introducido = '$cadena[$i]' WHERE ID_ex=
'$хid' AND ID_us= '$idusuario' AND familia= 'grammar';", $conexion);
//ACTUALIZACION
```

Figura 4.23: Inserción de respuestas del usuario en tabla *answers*

Puesto que la página de resultados es una página de carácter informativo y no requiere de la interacción del usuario no es necesario programar gramáticas ni introducción de datos por voz ni texto. Por lo tanto el siguiente paso es directamente mostrar los resultados. A la vez que se escriben cada una de las correcciones, se lleva a cabo la comprobación de si se trata de la respuesta correcta o no. Para realizar esta comprobación es necesario comparar, para cada ejercicio, el valor del campo "introducido" de la tabla "answers" con el campo "solución" de la tabla "grammar\_ex" que contiene el valor correcto de ese ejercicio. Una vez llevada a cabo dicha comparación, de nuevo en la tabla "answers" se actualizará el campo "correcto" de cada tupla correspondiente al ejercicio, de manera booleana (1=correcto; 0=error) tal y como muestran las Figuras 4.24 y 4.25.

```

while($i<=$numfilas) //Se repite tantas veces como ejercicios haya.
{
    $id=$idarray[$i]; // $idarray es un array que contiene los
                    // identificadores de los ejercicios.
                    // Se localiza el ejercicio en la tabla grammar_ex:
    $resultado1 = mysql_query("SELECT * FROM
    grammar_ex WHERE ID_ex=
    '$id';", $conexion);
    // Se localiza el ejercicio en la tabla answers:
    $resultado2 = mysql_query("SELECT * FROM answers WHERE
    ID_ex= '$id' AND ID_us= '$idusuario' AND familia=
    'grammar';", $conexion);
    ..
    .. <Se escribe el ejercicio incluyendo nuestra respuesta>
    ..
    // Se compara el valor introducido con el valor solución de las tablas
    mencionadas antes:
    if($registro2['introducido'] == $registro1['solución']) // Caso de
    respuesta correcta
    {
        // Se dibuja un "check" y se actualiza la tupla de dicho
        ejercicio en la tabla "answers" con un 1=correcto.
        mysql_query("UPDATE answers SET correcto = '1' WHERE
        ID_ex= '$id' AND ID_us= '$idusuario' AND
        familia='grammar';", $conexion);
    }
    else // Caso de respuesta incorrecta.
    {
        // Se dibuja un aspa y se actualiza la tupla de dicho ejercicio
        en la tabla "answers" con un 0=error.
        mysql_query("UPDATE answers SET correcto = '0'
        WHERE ID_ex= '$id' AND ID_us= '$idusuario' AND
        familia= 'grammar';", $conexion); // Por último se
        muestra la respuesta correcta.
    }
}

```

Figura 4.24: Análisis de respuestas para escritura de página de corrección

**VOCABULARY RESULTS**

Check your answers:

Your Answer: **house** ✓

Your Answer: **tower** ✗  
The correct answer is: **castle**

isma_pfc		+ Opciones		ID_us	ID_ex	introducido	correcto	familia
answers	Editar	Editar en línea	Copiar	14	1	stayed		1 grammar
grammar_ex	Editar	Editar en línea	Copiar	14	1	house	1	vocabulary
listening_ex	Editar	Editar en línea	Copiar	14	2	true		1 listening
listening_texts	Editar	Editar en línea	Copiar	14	2	tower	0	vocabulary
results	Editar	Editar en línea	Copiar	14	7	wolf		1 vocabulary
users	Editar	Editar en línea	Copiar	14	9	playing		1 grammar
vocabulary_ex	Editar	Editar en línea	Copiar	14	10	will go		1 grammar
	Editar	Editar en línea	Copiar	14	23	false		0 listening
	Editar	Editar en línea	Copiar	14	37	true		1 listening
	Editar	Editar en línea	Copiar	14	38	false		1 listening
	Editar	Editar en línea	Copiar	14	39	true		1 listening

Figura 4.25: Vista de inserción de respuesta en tabla *answers*

Al igual que sucedía con el apartado anterior, las páginas *vocabulary\_answers.php* y *listening\_answers.php* llevan a cabo y presentan funciones y estructura similar. Lo que cambia es la manera de presentar de forma visual los resultados y corrección de respuestas. Una vez los resultados y correcciones han sido mostrados y la información guardada en las bases de datos se redirige al usuario al menú principal con el fin de que pueda realizar otras funciones o continuar resolviendo ejercicios.

### 3.3. Esquema de Creación /Navegación de páginas e interacción con base de datos

Las figuras 4.26 y 4.27 muestran dos esquemas cuya finalidad es facilitar la comprensión y servir de guía de la estructura y funcionamiento descrito anteriormente. El primer esquema presenta la secuencia de creación de páginas, a través de ficheros PHP, que se lleva a cabo al navegar por las distintas opciones que presenta el módulo de creación de ejercicios. El segundo permite ver qué páginas interactúan con qué tablas de la Base de Datos, en qué orden y que datos envían o extraen de ellas.

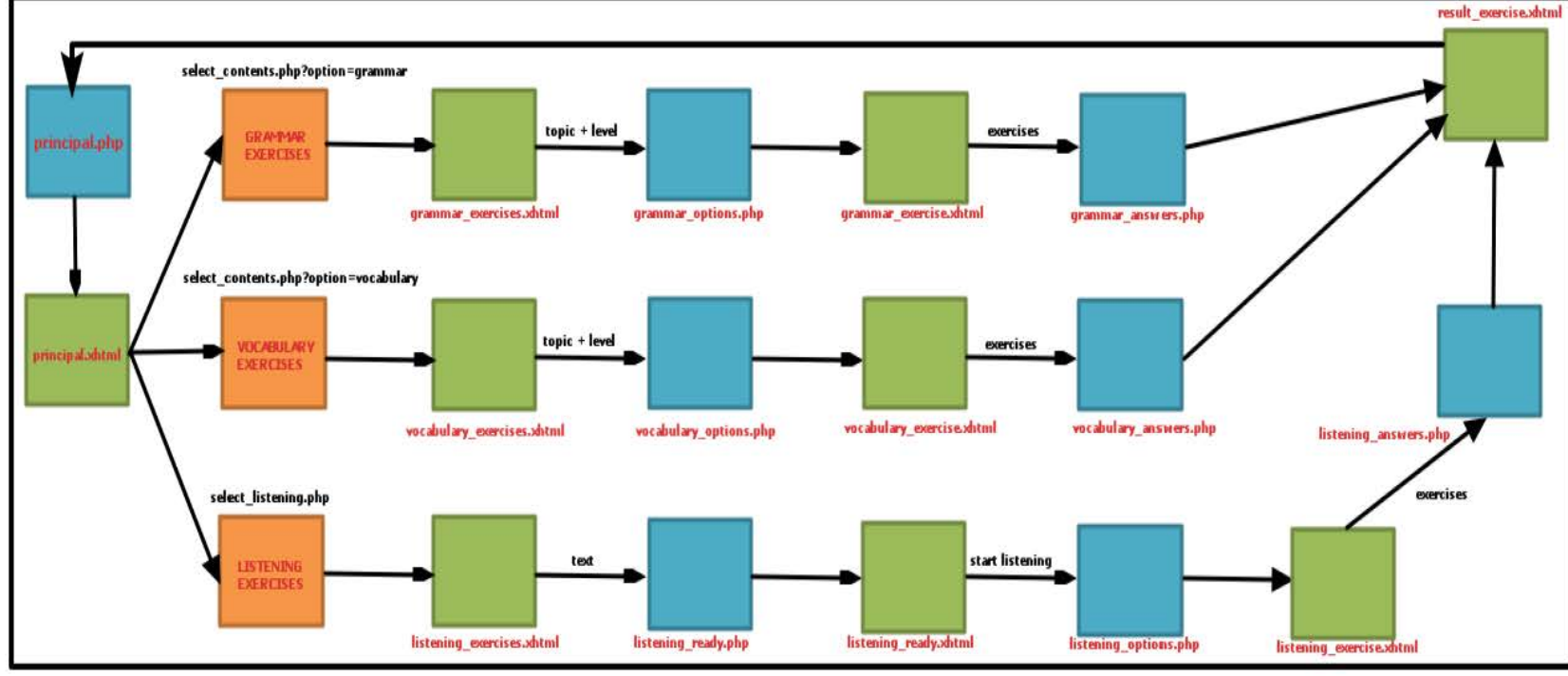


Figura 4.26: Esquema de creación-navegación de páginas

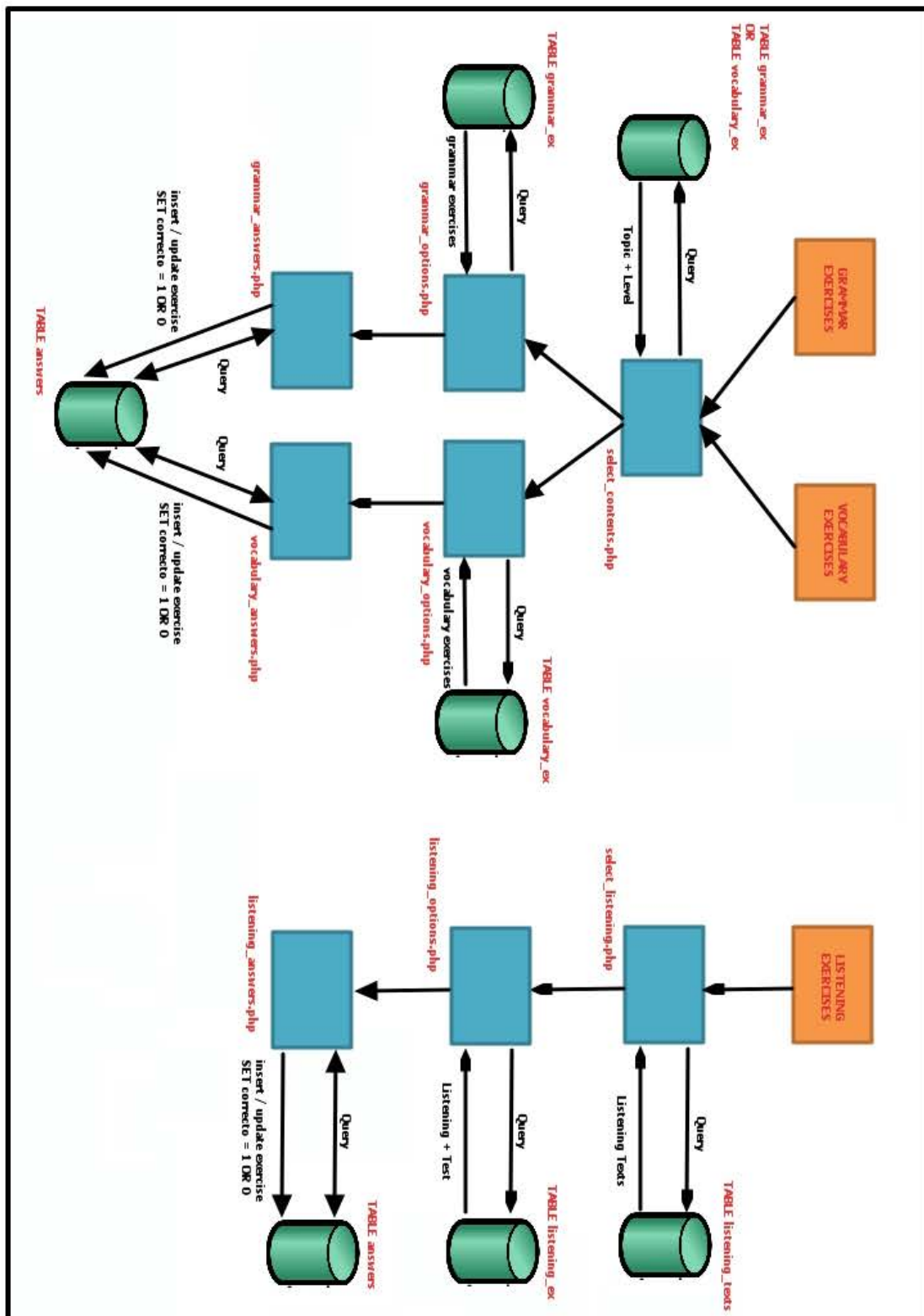


Figura 4.27: Esquema de interacción con bases de datos

#### 4. Módulo de Administración de Contenidos

4.1. Funcionalidad: La principal finalidad del módulo de administración de ejercicios es la de ofrecer al usuario responsable de añadir nuevo contenido (administradores, profesores o alumnos con dicho permiso) la posibilidad de crear nuevos ejercicios y/o editar y borrar ejercicios existentes sin tener que llevar a cabo estas tareas directamente sobre las bases de datos, lo que implicaría el obligado conocimiento del lenguaje y las herramientas. No sólo da la posibilidad de esta administración a cualquier usuario, incluso sin conocimientos informáticos previos, sino que además permite llevar a cabo estas tareas desde la misma aplicación web y mediante un interfaz de formularios que convierte el trabajo en una operación sencilla, clara y rápida. El módulo presenta el siguiente contenido:

- *Página inicial o menú*: En la que se selecciona la familia de ejercicios sobre la que queremos llevar a cabo las tareas de creación, edición o borrado.
- *Página de selección de la categoría del ejercicio*: Una vez seleccionada la familia se muestran todas las categorías existentes en ella con el fin de que seleccionemos la correspondiente al ejercicio que queremos crear, editar o borrar. También puede darse el caso de querer crear un ejercicio de una categoría nueva por lo que esta opción también será ofrecida.
- *Página de creación, edición o borrado de los ejercicios*: En las que se llevan a cabo las tareas propiamente dichas. Esta página queda dividida en dos partes. La primera parte muestra el conjunto de ejercicios ya existentes de la categoría seleccionada con las opciones de editar o borrar para cada ejercicio. La segunda parte muestra el formulario a rellenar en caso de que se quiera crear un ejercicio nuevo. Cabe tener en cuenta que obviamente la primera parte no se mostrará en el caso de que hayamos seleccionado "categoría nueva" en la página anterior puesto que al ser nueva todavía no existen ejercicios. La opción de editar un ejercicio existente muestra el formulario en una nueva página, arrastrando como contenido de los campos los actuales (esto es cómodo por si solo se quiere modificar un campo no tener que volver a rellenar toda la información).
- *Página de aviso de operación* llevada a cabo de manera satisfactoria.

#### 4.2. Arquitectura, funcionamiento e interacción con la base de datos:

- *Página inicial:* Ya que la página inicial o menú no requiere de contenido dinámico ni depende de variables traídas de páginas anteriores, está programada directamente como página XHTML (**create\_main.xhtml**). Su contenido es muy sencillo ya que consiste únicamente en enlaces a las tres posibles opciones a elegir.
- *Página de selección de la categoría del ejercicio:* Sea cual sea la opción elegida en la página anterior, será la página "**create\_options.php**" la encargada de generar la página de selección de categoría. Simplemente llevándonos hasta esta página una variable que indica la familia, ya se ejecutará el código correspondiente. Lo primero que hace la página es extraer las diferentes categorías de la familia concreta elegida en la página anterior, tal y como muestra la figura 4.28.

```
if($select==grammar) $sacarcat = "SELECT DISTINCT(categoria) FROM grammar_ex";
```

Figura 4.28: Consulta de extracción de diferentes categorías existentes

De manera similar se procede en caso de vocabulario y *listening*. A continuación simplemente queda crear el formulario de tipo *select* con las diferentes categorías existentes, anteriormente extraídas de la base de datos, con el fin de que el usuario pueda realizar la selección (Figura 4.29).

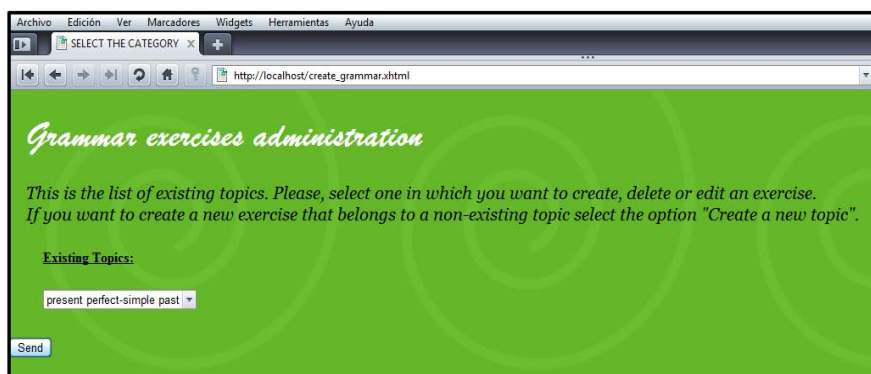


Figura 4.29: Vista de selección de tema para administración de contenidos

Como ya se mencionó en el apartado anterior, existe la posibilidad de que se requiera crear un ejercicio en una categoría nueva. En este caso se creará conjuntamente la categoría y el ejercicio. La opción "*create a new category*" está disponible al final del formulario de selección para llevar a cabo esta opción.

Puesto que para mostrar la lista de categorías se consulta directamente, como hemos visto, a la tabla de ejercicios correspondiente, no puede darse el caso de que se muestre una categoría para la que no existen ejercicios. Si son borrados todos los ejercicios de una categoría concreta ésta ya no será mostrada en el listado.

- *Página de creación, edición o borrado de los ejercicios:* Las páginas *create\_grammar\_form.php*, *create\_vocabulary\_form.php* , *create\_listening\_form.php* son las encargadas de mostrar los ejercicios existentes de la categoría seleccionada acompañados de las opciones de editar/borrar y a continuación de generar el formulario de creación de un ejercicio nuevo.

Existen dos posibles situaciones en función de las opciones seleccionadas en la página anterior y que influyen en las funciones generadas por estas páginas PHP. Son las siguientes:

- Ya existe la categoría y por tanto ya hay ejercicios asociados a ésta: Este es el caso más completo. Dada esta situación el PHP genera lo comentado en el apartado anterior, es decir, un listado de los ejercicios ya existentes acompañado de los botones de edición y borrado de ese ejercicio:

#### **Edición.**

```
fwrite($resultado_xhtml,'<form action="create_grammar_process.php">');  
fwrite($resultado_xhtml,'<input type="submit" value="Edit" class="button"/>');
```

**Figura 4.30:** Acceso a página de edición de ejercicio



### Borrado.

```
fwrite($resultado_xhtml,'<form action="delete_process.php">'); //PHP que se encargará del borrado del ejercicio.  
fwrite($resultado_xhtml,'<input type="submit" value="Delete" class="button"/>');
```

Figura 4.31: Acceso a página de borrado de ejercicio

En el caso del borrado será necesario llevar a la página "*delete\_process.php*" el identificador del ejercicio a borrar, la categoría del ejercicio y la familia a la que pertenece. Estos tres campos son los mínimos y necesarios para identificar a un ejercicio de manera inequívoca. A continuación se añade el formulario de creación de un nuevo ejercicio por si es ésta la operación que se desea llevar a cabo. La página completa generada presentará entonces la estructura mostrada en la Figura 4.32.

The screenshot shows a web browser window with the URL `http://localhost/create_grammar_form.shtml`. The page title is "Exercise Administration". Below the title, there is a paragraph: "This is the list of actual exercises that belong to 'present perfect-simple past' topic. You can fill the form below to create a new exercise." The main content area has a green background. It features a list of exercises, with one example: "Jane (stay) at home yesterday". Below this example are two buttons: "Delete" and "Edit". A red arrow points from the "Delete" button to the text "Ejercicios Existentes.". Below the list, there is a section for creating a new exercise, with the instruction: "Please, fill the form to create a new exercise following the scheme: [Subject]+[Verb Tense]+[Predicate]". This section contains several input fields: "Subjects:", "Verb tense:", "Predicate:", "Level:" (with radio buttons for "Easy", "Medium", and "Hard"), "Correct answer:", and "Possible Answers:". A red arrow points from the "Level:" section to the text "Formulario de creación de nuevo ejercicio.". At the bottom left, there is a "Send" button.

Figura 4.32: Vista de página principal de Administración de Contenidos

- Se ha seleccionado la opción de crear una categoría nueva: Obviamente en este caso no se muestran los ejercicios existentes ya que todavía no hay. Simplemente se escribe el formulario de creación de nuevo ejercicio de los apartados anteriores pero añadiéndole el campo de nombre de categoría que el usuario tendrá que rellenar (Figura 4.33).

Figura 4.33: Formulario de creación de ejercicio en nuevo tema

- *Realización de funciones:* Los ficheros PHP **create\_grammar\_process.php**, **create\_vocabulary\_process.php** y **create\_listening\_process.php** son los encargados de llevar a cabo la acción elegida en la página anterior (edición, borrado o creación). En función de la operación solicitada realizan diferentes funciones:
  - Creación de nuevo ejercicio: En primer lugar se controla que han sido rellenados todos los campos (condición indispensable). En caso afirmativo se procede a insertar en la base de datos y tabla correspondiente los valores del nuevo ejercicio (Figura 4.34).

```
INSERT INTO grammar_ex
(categoria,texto1,verbo,texto2,nivel,solucion,opciones) VALUES ('$categoria',
'$texto1', '$verbo', '$texto2', '$nivel', '$solución', '$opciones');
```

Figura 4.34: Inserción de datos en tabla de ejercicios

Si existen errores se lanza de nuevo en una nueva página el formulario advirtiéndole de estos. Una vez corregidos se vuelve a esta página PHP para comprobar de nuevo. Esta situación se dará en bucle hasta que todos los campos sean rellenados (Figura 4.35).



Figura 4.35: Ejemplo de errores en formulario de creación de ejercicio

- Edición de un ejercicio existente: Dado este caso se escribe un formulario similar al de creación de nuevo ejercicio pero con la particularidad de que los campos ya están rellenos con los valores actuales de ese ejercicio. Como se comentó en el apartado de funcionalidad esto permite que si se quiere cambiar sólo un dato no haya que verse obligado a rellenar de nuevo todos los campos. De igual manera que en el caso anterior una vez introducidos se valida que estén todos los campos completados. En caso afirmativo se procede a la actualización de los valores de ese ejercicio en la base de datos (Figura 4.36).

```
"UPDATE grammar_ex SET
texto1='$texto1', verbo='$verbo', texto2='$texto2', nivel='$nivel',
solución='$solución', opciones='$opciones' WHERE ID_ex='$ID_ex';"
```

Figura 4.36: Consulta de actualización de datos de ejercicio

Al igual que sucedía en el caso anterior, si existen errores volverá a lanzarse el formulario en una nueva página y será verificado de nuevo. Así hasta que todos los campos hayan sido completados.

- Borrado de un ejercicio: Esta acción la lleva a cabo un nuevo archivo "**delete\_process.php**" independientemente de la familia y categoría del ejercicio. El identificador del ejercicio, su categoría y la familia a la que pertenece son enviados hasta ésta página cuya función es básicamente lanzar la petición correspondiente de borrado (Figura 4.37).

```
if($type_ex=="grammar_ex") mysql_query ("DELETE FROM grammar_ex
WHERE ID_ex = '$ID_ex';", $conexion); //Se borra por el identificador de
ejercicio.
```

Figura 4.37: Consulta de borrado de ejercicio

Tras la finalización satisfactoria de cualquiera de estos tres procesos se realiza una llamada al fichero **operation\_done.php** que se encargará de generar una página XHTML en la que se nos informará mediante un mensaje por pantalla de que la operación concreta ha sido llevada a cabo con éxito. Este fichero es siempre el encargado de generar todos los mensajes que informan de operaciones terminadas satisfactoriamente y a continuación redirigirnos mediante un enlace al menú o la página que se considere más adecuada (Figura 4.38).

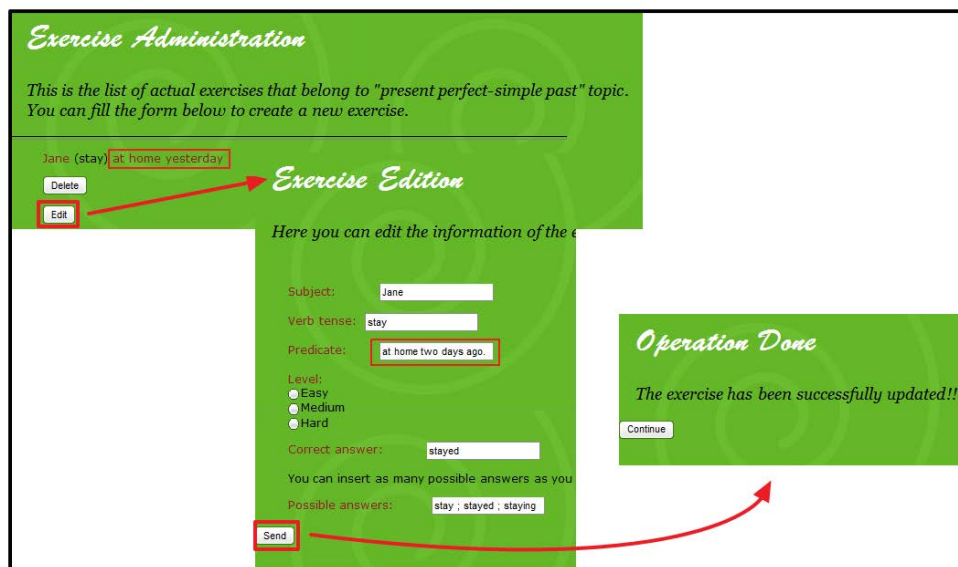


Figura 4.38: Transición de páginas en proceso de edición de ejercicio

#### 4.3. Esquema de Creación/Navegación de páginas e interacción con base de datos:

Al igual que en el apartado anterior, las Figuras 4.39 y 4.40 presentan dos esquemas cuya finalidad es facilitar la comprensión y servir de guía de la estructura y funcionamiento descrito anteriormente. El primer esquema presenta la secuencia de creación de páginas, a través de ficheros PHP, que se lleva a cabo al navegar por las distintas opciones que presenta el módulo de creación de ejercicios. El segundo permite ver qué páginas interactúan con qué tablas de la Base de Datos, en qué orden y que datos envían o extraen de ellas.

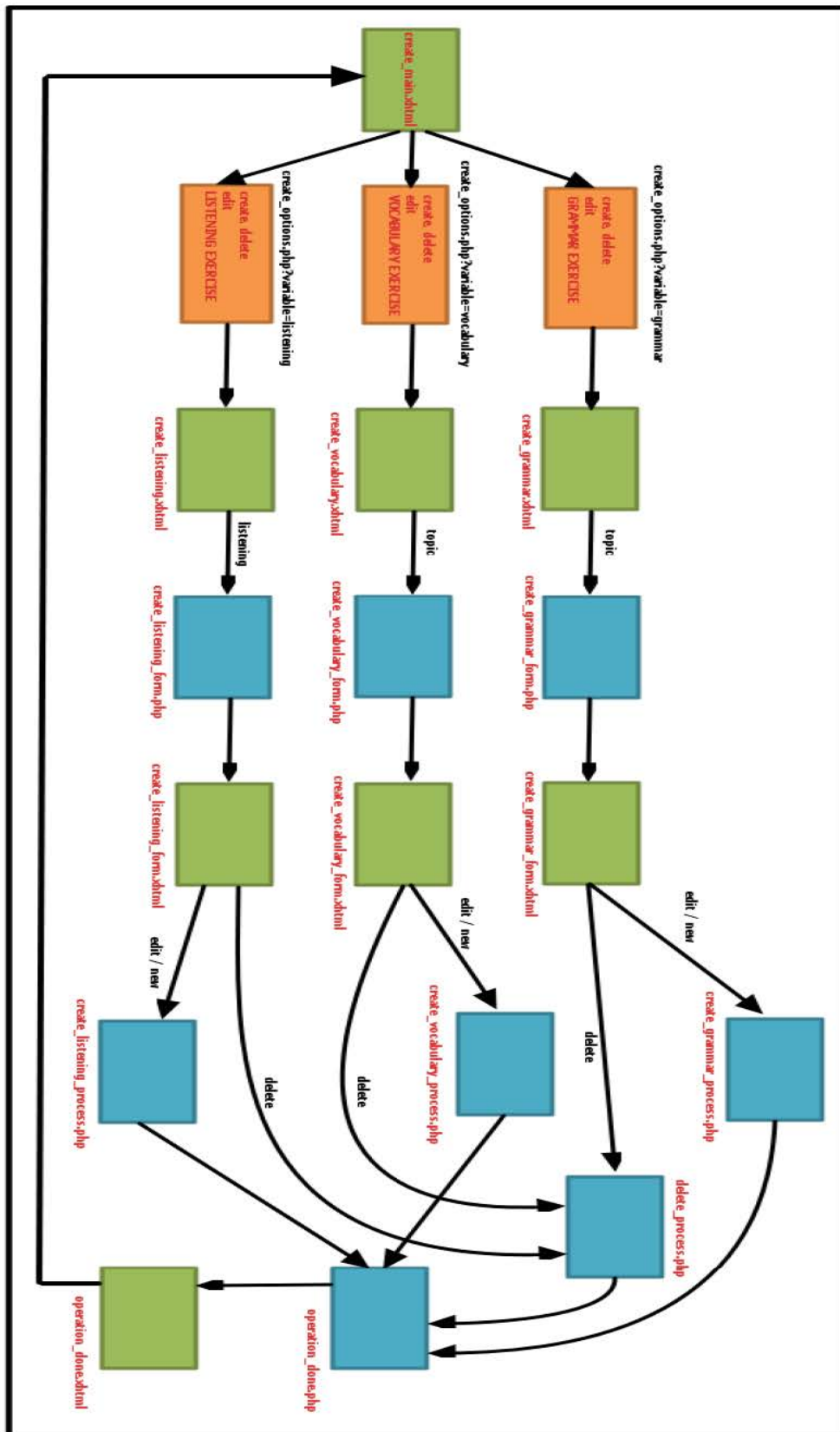


Figura 4.39: Esquema de creación-navegación de páginas

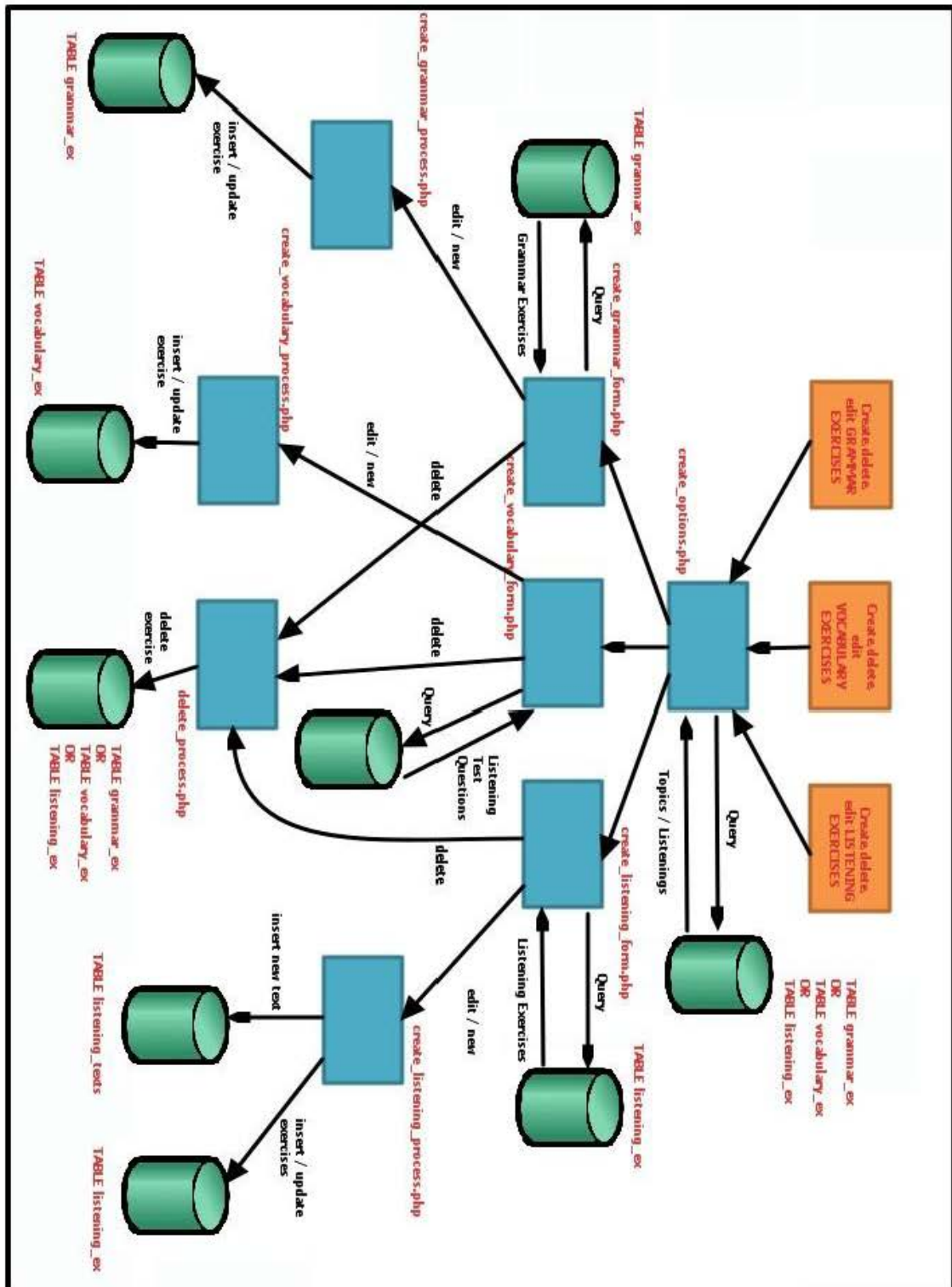


Figura 4.40: Esquema de interacción con bases de datos

## 5. Modulo de puntuación y resultados (My Results).

5.1. Funcionalidad: El modulo de resultados presenta dos objetivos diferenciados.

### **Mostrar resultados:**

- Mostrar el porcentaje de ejercicios ya realizados por familia y nivel.
- Mostrar el porcentaje de ejercicios realizados respecto al total.
- Mostrar el porcentaje de ejercicios correctos, respecto a los realizados, por familia y nivel.
- Establecer a partir de los ejercicios correctos una puntuación que sirva de referencia al usuario para controlar su avance y evolución.

### **Mostrar ejercicios realizados:**

El usuario puede seleccionar, de manera escrita o por voz, cualquier familia y nivel existentes y le serán mostrados todos los ejercicios con esas características que ha realizado durante todo su tiempo de uso de la aplicación. El objetivo es que pueda emplear esta funcionalidad para repasar todos sus errores y aciertos.

5.2. Arquitectura, funcionamiento e interacción con Base de Datos:

### *Mostrar resultados:*

Al igual que sucedía con el módulo de realización de ejercicios, para la consulta de puntuaciones y resultados necesitamos de igual forma haber iniciado sesión en el sistema.

La operación es la misma a la que ya se explicó en ese módulo: Verificación de usuario activo mediante la variable de sesión de usuario. En caso negativo la página **login\_request.xhtml** nos lanzará un mensaje informativo y nos llevará hasta la página de logado. En caso afirmativo se continuará con los siguientes pasos.

La página "**my\_results.php**" es la encargada de extraer todos los datos y realizar los cálculos para mostrar luego toda la información en la página XHTML correspondiente.



En primer lugar extrae el identificador del usuario activo. Una vez que ya conoce el usuario comienza a realizar consultas a la base de datos con el fin de realizar los cálculos necesarios. Lo primero que se necesita conocer es el *número total de ejercicios existentes por familia y nivel*. Esta operación se lleva a cabo mediante una consulta a la tabla correspondiente y un sencillo contador (Figura 4.41).

```
$sacar1 = "SELECT * FROM grammar_ex";
$resultado1 = mysql_query($sacar1,$conexion);
$tot_gramm = mysql_num_rows($resultado1); //El numero de filas que
devuelve esta consulta nos da el número total de ejercicios de gramática.

while ($registro1 = mysql_fetch_array($resultado1)) //Se repite tantas veces como
ejercicios de gramática haya.
{
    $nivel=$registro1['nivel']; //Mira el nivel de cada ejercicio y
suma 1 al contador correspondiente.
    switch ($nivel) {
        case "easy": $gramm_easy_tot++;
        case "medium": $gramm_med_tot++;
        case "hard": $gramm_hard_tot++;
    }
}
```

Figura 4.41: Consulta y conteo de ejercicios existentes por categoría y nivel

Se realiza exactamente el mismo procedimiento con los ejercicios de vocabulario y *listening*. Sumando el total de ejercicios de cada una de las familias guardamos el total de ejercicios existentes (Figura 4.42).

```
$tot_contents=$tot_gramm + $tot_voc + $tot_listen;
```

Figura 4.42: Cálculo del total de ejercicios existentes

A continuación se realiza un algoritmo similar al anterior para extraer el *número de ejercicios realizados* por familia y nivel. La diferencia es que el anterior extraía los existentes (y por tanto usaba la tabla de ejercicios "grammar\_ex"). Puesto que ahora

nos interesan los realizados, la tabla a la que se consulta es la de respuestas "answers" para extraer los ejercicios hechos, y una vez extraídos se consulta sobre su nivel en la tabla "grammar\_ex" con el fin de realizar la cuenta de manera similar a la explicada anteriormente (Figura 4.43).

```
$sacar1 = "SELECT * FROM answers WHERE ID_us = '$idusuario' AND familia = 'grammar'"; //Extraemos los ejercicios realizados.
$resultado1 = mysql_query($sacar1,$conexion);
    while ($registro1 = mysql_fetch_array($resultado1)) //Por cada ejercicio.
    {
        $ID_ex = $registro1['ID_ex']; //Obtenemos el identificador de cada ejercicio realizado por ese usuario.

//Se extrae el nivel del ejercicio y se añade al contador correspondiente.

        $sacar2 = "SELECT * FROM grammar_ex WHERE ID_ex = '$ID_ex'";
        $resultado2 = mysql_query($sacar2,$conexion);
        $registro2 = mysql_fetch_array($resultado2);
        $nivel = $registro2['nivel'];
        switch ($nivel)
        {
            case "easy": $gramm_easy_done++;
            case "medium": $gramm_med_done++;
            case "hard": $gramm_hard_done++;
        }
    }
}
```

Figura 4.43: Consulta y conteo de ejercicios realizados por categoría y nivel

Sumando el total de ejercicios realizados de cada una de cada uno de los niveles de gramática guardamos el total de ejercicios realizados de gramática (Figura 4.44).

```
$tot_gramm_done=$gramm_easy_done + $gramm_med_done +
$gramm_hard_done;
```

Figura 4.44: Cálculo del número total de ejercicios realizados de gramática

Se realiza exactamente la misma operación con los ejercicios de vocabulario y *listening*. Sumando los totales realizados de las tres familias de ejercicios obtenemos el total de ejercicios realizados (Figura 4.45).

```
$tot_done=$tot_gramm_done + $tot_voc_done + $tot_listen_done;
```

Figura 4.45: Cálculo del número total de ejercicios realizados

Por último, se obtienen los ejercicios correctos realizados por familia y nivel. El algoritmo es similar al anterior, lo único que cambia es que hay que añadir la condición de que sean correctos (Figura 4.46).

```
$sacar1 = "SELECT * FROM answers WHERE ID_us = '$idusuario' AND familia = 'grammar' AND correcto = '1'";
```

Figura 4.46: Consulta ejercicios correctos por familia y nivel

La cuenta se lleva a cabo de manera similar a la anteriormente descrita. Se emplea el mismo procedimiento para los ejercicios de vocabulario y *listening*. Se suma el total de ejercicios correctos de cada familia para obtener el total de ejercicios correctos (Figura 4.47).

```
$tot_ok=$tot_gramm_ok + $tot_voc_ok + $tot_listen_ok;
```

Figura 4.47: Cálculo del número total de ejercicios correctos

Ahora ya tenemos todos los valores necesarios. Se escriben en la página:

- Los ejercicios realizados por familia y nivel respecto del total existentes de esa familia y nivel.
- El valor del porcentaje realizado respecto del total. Se emplea la fórmula:

$(100 * \text{numero\_total\_ejercicios\_realizados}) / \text{numero\_total\_ejercicios\_existente}$

- Numero de ejercicios correctos por familia y nivel respecto al total de ejercicios realizados para esa familia y nivel.
- Puntuación total obtenida. Se emplea la formula:  

$$(100 * \text{total\_ejercicios\_correctos}) / \text{total\_ejercicios\_realizados}$$

Los resultados son mostrados en tres tablas diferentes separadas por familias con el fin de facilitar su lectura y comprensión tal y como puede observarse en la Figura 4.48.

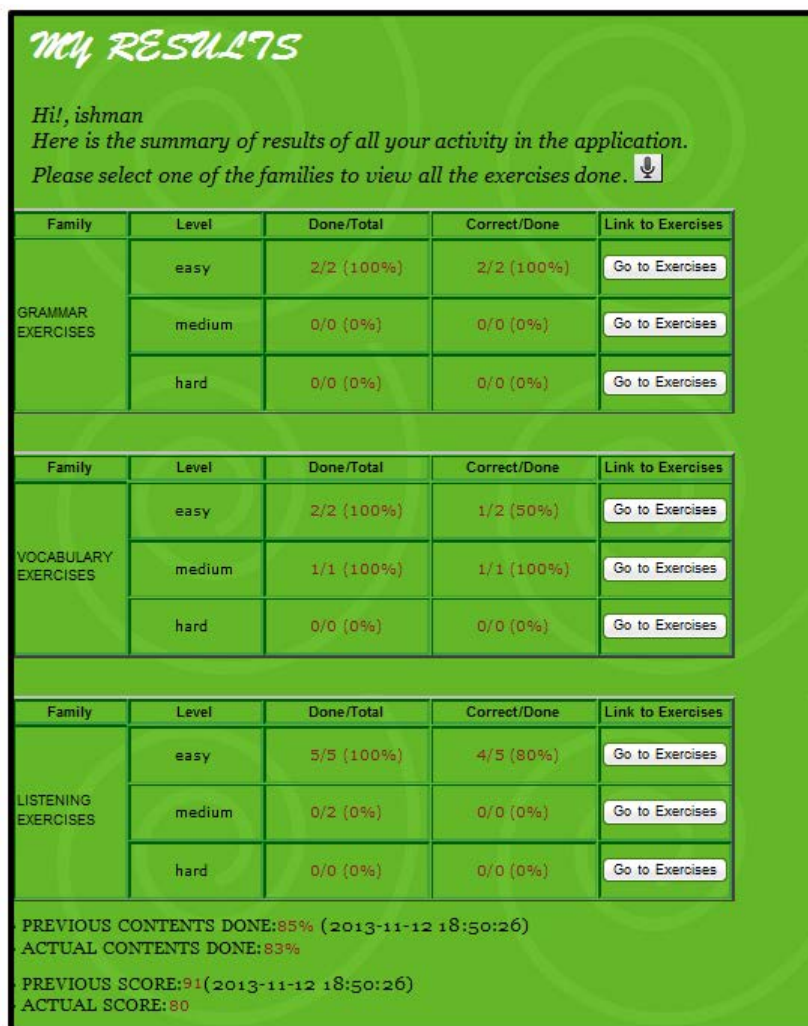


Figura 4.48: Vista de la primera sección de página de resultados

*Inserción de resultados en la base de datos:*

Una vez mostrados los resultados, se introducen estos valores en la tabla "results" de la base de datos. Para cada usuario existe un máximo de dos entradas en esta tabla. El almacenamiento funciona de la siguiente manera:

La primera vez que un usuario accede al módulo de resultados, sus datos se guardan en la tabla como "actuales" (Tupla 1, Figura 4.49).

num_consulta	ID_us	fecha	gramm_easy	gramm_med	gramm_hard	voc_easy	voc_med	voc_hard	listen_easy	listen_med	listen_hard	done	score
1	14	25-11-2012 17:30:47	2	0	0	1	1	0	4	0	0	83	80

**Figura 4.49:** Vista primera inserción en tabla de resultados

La segunda vez los datos anteriores pasan a ser "previos" (Tupla 2) y los nuevos a ser los "actuales" (Figura 4.50).

num_consulta	ID_us	fecha	gramm_easy	gramm_med	gramm_hard	voc_easy	voc_med	voc_hard	listen_easy	listen_med	listen_hard	done	score
1	14	25-11-2012 17:39:39	2	0	0	1	1	0	4	0	0	83	80
2	14	25-11-2012 17:30:47	2	0	0	1	1	0	4	0	0	83	80

**Figura 4.50:** Vista segunda inserción en tabla de resultados

Las siguientes veces los que figuran como "previos" desaparecerán, los que había como actuales pasarán a ser "previos" y los últimos calculados serán los "actuales" (almacenamiento en una pila de dos elementos).

Tal y como muestra la figura 4.51, todos los resultados se guardan junto con la fecha. El sentido de esta operación es que si el usuario ya ha hecho al menos dos consultas al módulo de resultados los cálculos de porcentaje realizado respecto del total existente y la puntuación (porcentaje del realizado correcto respecto del realizado) aparecerán dos veces, uno con los valores anteriores a la nueva consulta y

el otro con los valores referentes a la nueva. El objetivo es que el usuario pueda comparar el momento actual con la anterior vez que realizo la consulta de resultados. Obviamente si el usuario sólo ha realizado una consulta al modulo de resultados, estos valores solo aparecerán una vez refiriéndose al cálculo actual (Figura 4.51).

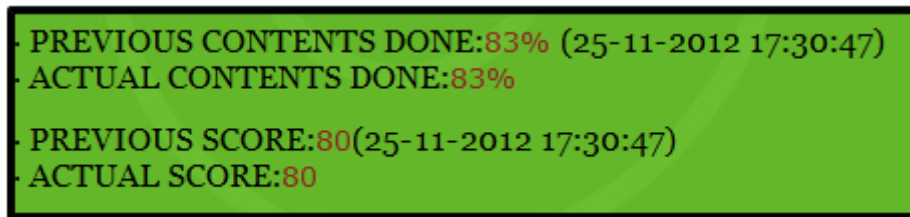


Figura 4.51: Vista de porcentaje realizado y puntuación

De igual manera si el usuario ha accedido a sus resultados al menos dos veces, utilizando los valores de los registros previo-actual comentados, la página creará mediante la librería JGraph tres gráficas dinámicas comparativas por nivel y familia diferenciando de manera gráfica los valores de la consulta anterior y los de la nueva consulta (Figura 4.53). Si es la primera vez que accede dichas gráficas no aparecerán ya que se realizan con carácter comparativo y no hay todavía nada que comparar (Figura 4.52).

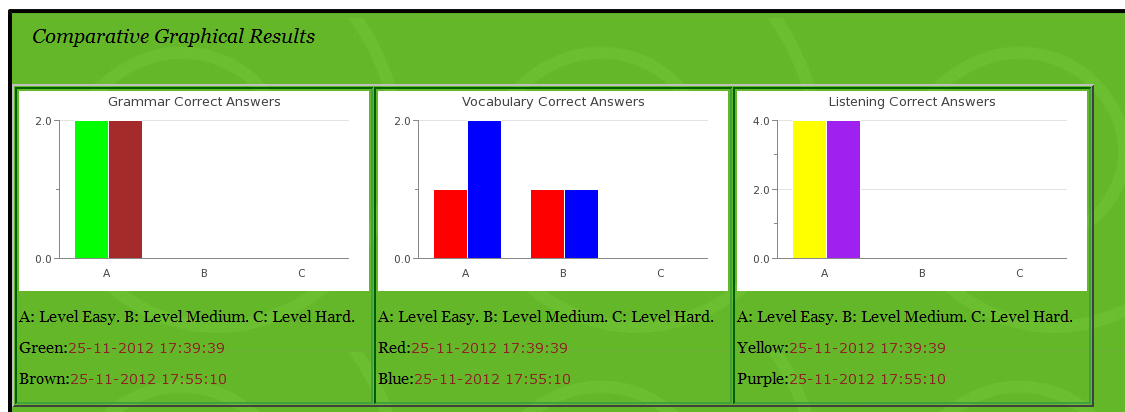


Figura 4.52: Vista de gráficas comparativas de ejercicios correctos

*Mostrar Contenido Realizado:*

En las tablas de resultados, para cada familia y nivel existe un botón que permite acceder al listado de ejercicios que el usuario a realizado con esas características. Al hacerlo se accede a una nueva página creada por el fichero "**results\_selected.php**". Este fichero PHP se encarga de mostrar los ejercicios de esa familia y nivel que ese usuario ha hecho a lo largo de todo el tiempo de uso de la aplicación. Si no ha realizado ejercicios con esas características la página informará de este hecho. Independientemente de la familia y nivel es el mismo PHP descrito el que actúa, por lo que controla su acción llevándose una variable que le indica la selección realizada. Para obtener este listado se realiza una consulta a la tabla correspondiente de la base de datos. La figura 4.53 muestra un ejemplo de consulta para ejercicios realizados de gramática nivel fácil.

```
switch($results)    //La variable results contiene la familia y el nivel
{
    case "easy grammar done":
        /*Se muestran primero los correctos.*/

        $sacar_correctos = "SELECT * FROM answers JOIN grammar_ex ON
answers.ID_ex=grammar_ex.ID_ex WHERE nivel='easy' AND correcto='1'
AND familia='grammar' AND ID_us='$idusuario";
        $resultado_correctos = mysql_query($sacar_correctos,$conexion);

        //A continuación los incorrectos, que presentan la misma consulta pero con
la condicion "AND correcto='0'".

        //De igual modo aquí se continuaría con el resto de los casos del "switch",
es decir todos los niveles para cada familia.
```

**Figura 4.53:** Consulta para obtención de ejercicios realizados seleccionados

Puesto que los campos necesarios para extraer estos ejercicios están repartidos en las tablas "answers" y "grammar\_ex", se realiza un *JOIN* de ambas tablas relacionándolas por su campo común *ID\_ex* (clave primaria), lo que nos permite poner como condición de búsqueda parámetros de cualquier campo de esas dos tablas.

Una vez ya tenemos todos los ejercicios que cumplen las condiciones seleccionadas solo queda proceder a su escritura en la página. Dependiendo de la familia los datos serán mostrados de una manera u otra y los parámetros de la consulta variarán pero la estructura y funciones de los ficheros PHP que generan la página resultante es similar en todos los casos (Figura 4.54).



Figura 4.54: Vista de ejercicios seleccionados realizados

### 5.3. Esquema de Creación/Navegación de páginas e interacción con base de datos:

Las Figuras 4.55 y 4.56 presentan dos esquemas cuya finalidad es facilitar la comprensión y servir de guía de la estructura y funcionamiento descrito anteriormente. El primer esquema presenta la secuencia de creación de páginas, a través de ficheros PHP, que se lleva a cabo al navegar por las distintas opciones que presenta el módulo de resultados. El segundo permite ver qué páginas interactúan con qué tablas de la base de datos, en qué orden y qué datos envían o extraen de ellas.

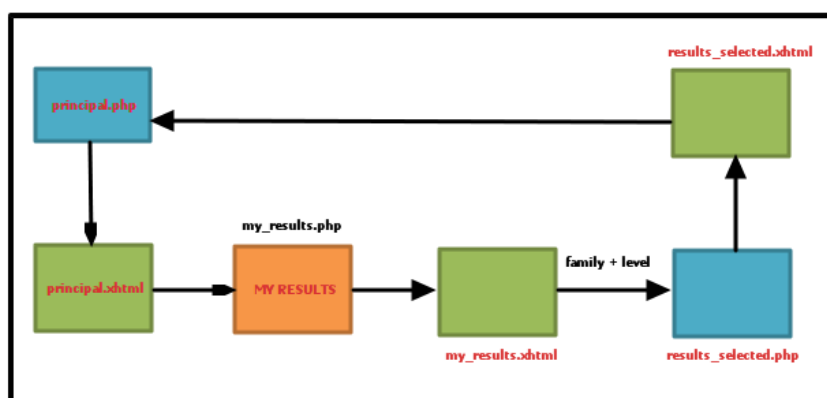


Figura 4.55: Esquema de creación-navegación de páginas



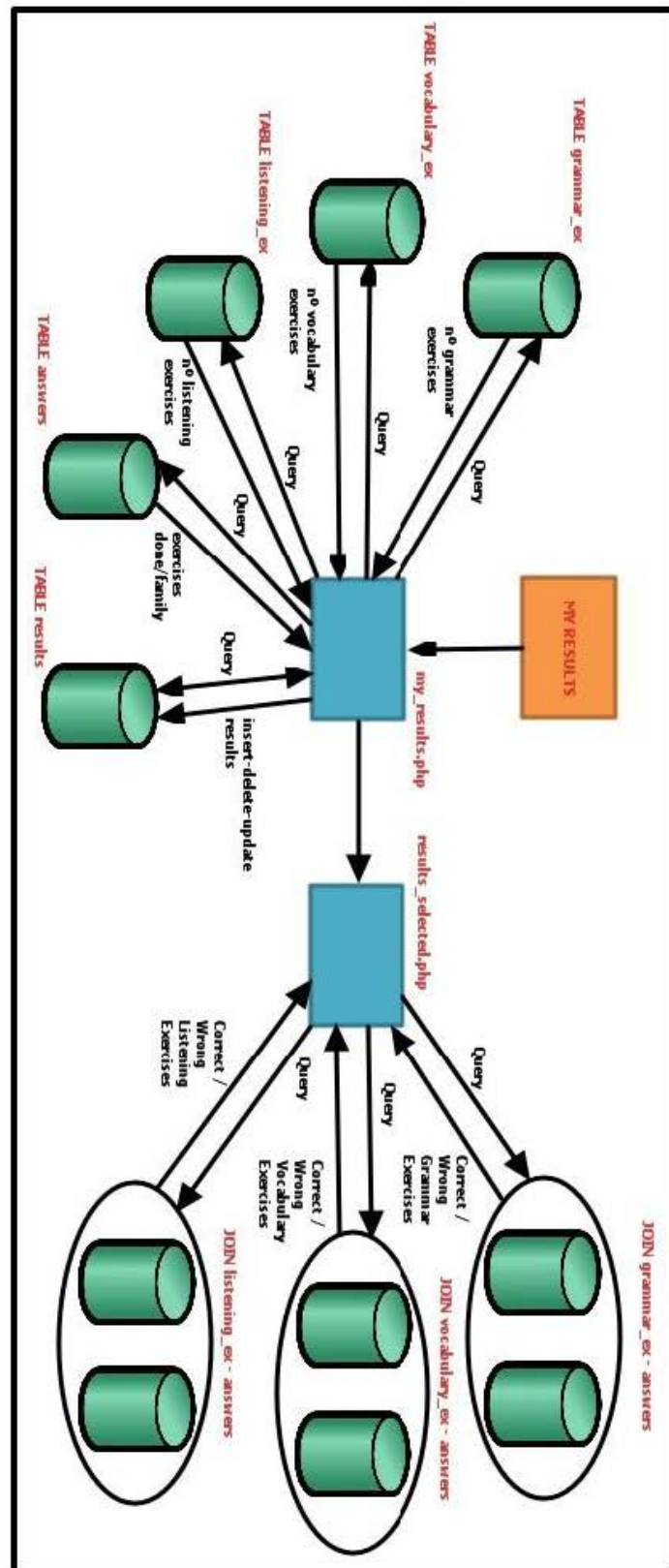


Figura 4.56: Esquema de interacción con bases de datos



## Capítulo 5: Conclusiones y líneas de trabajo futuro

---

### Conclusiones

No hay más que observar el mercado actual de las nuevas tecnologías para darse cuenta de que los sistemas de diálogo representan una herramienta de plena actualidad, con infinidad de aplicaciones y usos y que seguirá evolucionando en el futuro para continuar cumpliendo con la amplia demanda funcional que ya posee. Como ya se mencionó en el segundo capítulo de ésta memoria, si ya inicialmente cobró un auge importante en la telefonía fija con el uso de centralitas y asistentes de voz, los nuevos dispositivos móviles (en especial los *smartphones* o teléfonos inteligentes) han supuesto una explosión de nuevas ideas en lo que se refiere a la interacción mediante voz para el control y manejo de aplicaciones diseñadas para todo tipo de funciones. Se concluye de esta manera que los sistemas multimodales van un paso más allá en lo que a funcionalidad, creatividad puede dar de sí el uso de la voz como complemento a una aplicación de base visual.

A través de la aplicación desarrollada se pretende, también, de algún modo, demostrar que la voz no sólo representa un complemento que permite aumentar la comodidad de manejo sino que se busca específicamente el darle un sentido y funcionalidad en el que el uso de la voz se convierta en una característica que potencie su utilidad y beneficio y lo diferencie claramente de lo que supondría el mismo sistema basado exclusivamente en el manejo tradicional. Si bien considero que el objetivo de la aplicación cumple con esas expectativas, me parece necesario también hacer hincapié en una serie de aspectos a los que se he llegado a la conclusión durante el proceso, y que podrían considerarse en cierto modo características, que ha día de hoy, suponen las limitaciones más claras de estos sistemas. El primero de estos aspectos y fundamental se basa en el necesario empleo de gramáticas, es decir, de la necesidad de definir el conjunto de palabras que nuestra aplicación va a ser capaz de entender en cada momento. Sucede entonces que aquellas páginas o módulos que requieran de información totalmente variable, como por ejemplo el formulario a rellenar para registrar un usuario, no pueden ser llevadas a cabo mediante la voz, o no al menos de una manera efectiva o que suponga una ventaja respecto al sistema tradicional de entrada por

teclado. La única opción para la entrada de datos variables consiste en crear una gramática que asocie palabras a las letras del abecedario, de tal manera que se delecte lo que queremos escribir a través de ciertas palabras, proceso que puede resultar bastante tedioso en el caso de tener que escribir textos de muchos caracteres. Por ese motivo se ha optado por no implementar la funcionalidad de voz en aquellas páginas o contenidos que requieren de entradas totalmente variables. Esta limitación también supone que tenga que darse un conjunto de opciones respuesta en el caso de los ejercicios de gramática (si no el usuario podría estar diciendo términos y fallando infinitamente hasta que diese con la palabra concreta incluida en la gramática). Los ejercicios de vocabulario presentan otra solución al problema consistente en, como se ha visto, permitir al usuario decir una palabra determinada (en este caso "next") para pasar al siguiente ejercicio y no caer en la repetición ilimitada de intento y error. Estos dos métodos recogen las soluciones más efectivas que encuentro a la hora de acotar las palabras permitidas cuando el dato a entrar no es en principio acotado. La limitación genérica consiste entonces en la complejidad de dotar a una aplicación completa de un sistema multimodal, es decir, que pueda ser manejada íntegramente mediante la voz de una manera sencilla y cómoda ya que es muy probable que ciertos componentes de su contenido dependan, sin alternativa, de entradas de datos totalmente variables.

La segunda limitación tiene que ver con la capacidad de reconocimiento. Como se explicó en capítulos anteriores, hay muchos factores los que depende que el sistema "nos escuche" bien. El ruido ambiente, las limitaciones y configuración del dispositivo de entrada, el tono de la voz, etc. Estos factores podrían considerarse como externos al sistema, sin embargo el grado de exigencia de reconocimiento que requiere en muchos casos la entrada de voz, hace que incluso en condiciones óptimas de dichos factores el sistema sea susceptible de errores o equívocos. Un ejemplo muy claro en el ámbito de esta aplicación se da especialmente en el dictado de verbos para la resolución de ejercicios de gramática. Se tiene que tener en cuenta que en ciertas ocasiones la potencia de reconocimiento exigible es máxima al tener, como opciones de respuesta, palabras que presentan variaciones muy sutiles a la hora de ser pronunciadas. Por ejemplo, al realizar las pruebas se comprobó un alto índice de equívocos entre las respuestas opcionales como "clean" y "cleaned". Tanto mayor es la posibilidad de que tome la opción equivocada si se trata de palabras cortas, como habría mucha posibilidad de que pasase con verbos como "run" y "ran".

Como conclusión puede entonces resumirse que los sistemas de diálogo, y de manera especial los sistemas multimodales, pueden potenciar la funcionalidad, prestaciones y el manejo de una aplicación siempre y cuando se hayan estudiado muy bien características como las anteriormente descritas. Es decir, se debe estudiar si la aplicación exige mucho contenido de entrada variable y, si es el caso, si se puede llegar a algún método que lo acote sin afectar la efectividad y comodidad al introducirlo mediante voz. Si presenta mucho contenido de este tipo nos veríamos en la situación de tener que crear una aplicación mixta, no íntegramente multimodal, y el cambio continuo obligado del formato de entrada es algo que puede incomodar o desorientar al usuario. De igual modo habría que analizar el grado de complejidad de reconocimiento que exige el contenido de entrada de la aplicación para evitar un alto grado de equívocos o errores que casi con total seguridad terminarían siendo la causa de que el usuario buscase otra aplicación o desistiese en el uso de la voz para su manejo. Si este análisis no fuese llevado a cabo de antemano y de manera rigurosa las mismas características positivas a las que puede dotar a nuestra aplicación el uso de la voz, pueden convertirse en factores determinantes para que sea descartada por los usuarios.

En cuanto al uso del estándar VoiceXML para el desarrollo de sistemas de diálogo, me ha parecido una herramienta potente y que no presenta gran complejidad de uso por lo que se pueden conseguir resultados efectivos en pocas líneas y sin que requiera una preparación previa muy exhaustiva. Sin embargo he echado de menos algunas de sus funciones y comandos que no han sido adaptados al formato XHTML+Voice. El ejemplo más claro es la función SUBMIT que permite, de manera muy cómoda y sencilla, pasar los valores de los "fields" o entradas de voz a otras páginas o secciones. Al no estar implementada en XHTML+Voice se vuelve bastante compleja esta operación que sólo puede ser resuelta creando una expresión regular dinámica que incluye tantos elementos como variables queramos pasar. Esta ausencia de ciertas funciones delimita las posibilidades que ofrece el lenguaje o bien obliga a que para conseguir cierta funcionalidad adicional se requiera el manejo y conocimiento de otros lenguajes de script.

Considero que el objetivo principal del proyecto consistente en el aprendizaje de la tecnología XHTML+Voice y el concepto y posibilidades de los sistemas de diálogo ha quedado cumplido al igual que la ampliación de conocimiento y manejo del resto de tecnologías empleadas, como XHTML, MySQL y PHP, todas de plena actualidad y uso, y herramientas fundamentales en la creación de páginas y aplicaciones web.

## Líneas de trabajo futuro

Las líneas de trabajo futuro podrían quedar clasificadas como ampliaciones o mejoras:

- *Ampliaciones:*

Dada la naturaleza de la aplicación implementada es sencillo llegar a una primera ocurrencia como posibilidad de ampliación, consistente en el desarrollo de nuevos tipos de ejercicios, que complementen los ya incluidos, para cualquiera de las familias: gramática, vocabulario o *listening*. No se requiere de un proceso creativo avanzado para inventar nuevos juegos o actividades, ya que basta con observar cualquier libro de ejercicios de inglés para obtener suficientes ejemplos de ejercicios típicos que podrían ser llevados a cabo sin excesiva dificultad desde el punto de vista del diseño y programación. Si sería interesante siempre mantener la característica multimodal en todos ellos asegurando que pueda adaptarse la voz como método de manejo alternativo al ratón o teclado.

Podría ser interesante también incluir una pequeña guía de gramática con el fin de que la aplicación no solo sirviese de apoyo práctico sino que también fuese usada como herramienta de estudio y referencia teórica. De esa manera el usuario no se vería forzado siempre a tener que acudir a páginas externas en caso de dudas o necesidad de estudio y repaso de conceptos.

- *Mejoras:*

Puesto que los textos que dan instrucciones a las páginas han sido escritos en inglés, y la voz de salida de XHTML+V solo está implementada para ese mismo idioma, puede darse el caso de que un usuario con muy poco nivel de inglés se encuentre algo perdido al empezar a usar la aplicación. Al igual que sucede cuando alguien recibe clases, la intención de ésta situación es que ya desde el principio los usuarios vayan acostumbrándose a trabajar íntegramente en el idioma objetivo. Sin embargo una opción interesante sería ofrecer al usuario la posibilidad de elegir el idioma de las páginas por si encuentra dificultades para orientarse en el manejo e instrucciones de la aplicación.

La segunda mejora tiene que ver con los textos de salida emitidos por la voz de XHTML+V. Sucede que aunque la pronunciación de la voz es muy buena y fácilmente entendible, el sintetizador de voz no tiene en cuenta los signos de puntuación del texto. Eso hace que los textos largos, como los emitidos en los ejercicios de *listening*, sean leídos sin ninguna pausa ni expresión lo que dificulta en muchos casos el entendimiento. La mejora consistiría en pasar dichos textos a voz real grabada en archivos de sonido lo que además permitiría que pudiesen ser pausados, repetidos, o se pudiese modificar su volumen e incluso la velocidad de lectura. Obviamente la dificultad de esta mejora reside en el trabajo que conlleva pasar todos los textos a voz real en el caso de que sean muy numerosos, y además contar con la voz de una persona con la capacidad de pronunciarlos a la perfección.

Pese a que el módulo de Administración de Contenidos asegura que puedan llevarse a cabo todas las tareas de creación, edición y borrado de ejercicios sin tener que trabajar directamente sobre la base de datos, todavía sería posible añadirle algunas opciones más como la modificación de datos de los usuarios registrados o incluso el borrado de una cuenta en caso de que el usuario no quisiese usar más la aplicación.





# Presupuesto

Se presenta a continuación el presupuesto del proyecto completo. Para realizar el cálculo se deben tener en cuenta la duración de las fases y tareas, los costes de personal y el coste de material.

## 1. Duración fases y tareas

Como ya se comentó en la sección *Fases de Desarrollo* del Capítulo 1, por motivos laborales los tiempos que he necesitado para completar las diferentes fases del proyecto se han visto dependientes de la variabilidad de horas dedicadas por día y de ciertos periodos de tiempo en los que la dedicación tuvo que ser menor. Sin embargo en un entorno de trabajo real se parte del hecho de que existe un número de horas de dedicación prefijadas y constantes. Puesto que el presente presupuesto pretende dar un coste en un entorno como el comentado he realizado los siguientes promedios para adaptar los tiempos que yo he empleado a una situación profesional real.

El promedio de horas dedicadas al día por mí ha sido de **4 horas**. Puesto que no todos los días he podido dedicarle tiempo la cifra podría quedar ajustada a **3 horas** (lo que supondría uno de cada cuatro o un cuarto de los días totales sin dedicación). Las fases mantienen su duración pero teniendo en cuenta que nuestro programador trabaja 8 horas al día:

- Fase1: Formación y preparación. (60 días x 3horas/día) ≈ (**23 días** x 8horas/día)
- Fase2: Análisis y planificación. (27 días x 3horas/día) ≈ (**10 días** x 8horas/día)
- Fase3: Desarrollo. (350 días x 3horas/día) ≈ (**131 días** x 8horas/día)
- Fase4: Documentación. (122 días x 3horas/día) ≈ (**46 días** x 8horas/día)
- Fase5: Presentación. (15 días x 3horas/día) ≈ (**6 días** x 8horas/día)

## 2. Coste de los Recursos

- *Recursos Software:*

- Navegador Opera 10.10 (con complemento Opera Voice): **0 €**

- Editor de programación HateML (Para el código XHTML, XHTML+Voice y PHP): **0 €**
- Editor de texto Notepad++: **0 €**
- Microsoft Office 2007 (Word y Powerpoint): **200 €**
- Foxit PDF reader: **0 €**
- MySQL 5.5.15 (junto con MySQL Administrator para realizar copias de seguridad): **0 €**
- Servidor Web Apache 2.2.11: **0 €**
- PHP 5.2.9-2: **0 €**
- phpMyAdmin 3.4.5 (Para la creación y gestión de la Base de Datos): **0 €**

- *Recursos Hardware:*

- Ordenador portátil: **700 €**
- Auriculares Estéreo: **50 €**
- Micrófono portátil externo: **40 €**

- *Recursos Humanos:*

Para la realización del proyecto se requiere de un Ingeniero Sénior (**20 €/hora**) y un Jefe de Proyecto (**40 €/hora**). A continuación se desglosa la duración estimada y la participación de los miembros para cada fase.

- *Fase de Formación y Preparación*

- |  |              |
|--|--------------|
| - XHTML+Voice (Ingeniero, 6 días)                                      | <b>960 €</b> |
| - XHTML / PHP / CSS (Ingeniero, 0 días)*                               | <b>0 €</b>   |
| - Repaso teoría BBDD, MySQL, PhpMyAdmin (Ingeniero, 0 días) *          | <b>0 €</b>   |
| - Instalación y configuración de software necesario (Ingeniero, 1 día) | <b>160 €</b> |

- *Fase de Análisis y Planificación*

- |  |               |
|--|---------------|
| - Planificación y análisis de los requisitos de la aplicación (Jefe + Ing. 2 días) | <b>960 €</b>  |
| - Planificación de la Base de Datos necesaria (Jefe + Ing. 2 días)                 | <b>960 €</b>  |
| - Análisis de Diseño y Esquema de Navegación (Jefe + Ing. 5 días)                  | <b>2400 €</b> |

- *Fase de Desarrollo*

- |   |                |
|---|----------------|
| - Programación de Páginas (XHTML y PHP) (Ingeniero, 90 días)        | <b>14400 €</b> |
| - Programación de Sistema de Diálogo (XHTML+V) (Ingeniero, 23 días) | <b>3680 €</b>  |
| - Programación de estilos (maquetación) (Ingeniero, 4 días) **      | <b>640 €</b>   |
| - Pruebas Unitarias (Jefe + Ing. 2 días) **                         | <b>960 €</b>   |
| - Pruebas de Integración (Jefe + Ing. 2 días) **                    | <b>960 €</b>   |

- *Fase de Documentación*
- Redacción del contenido (Ingeniero, 18 días) \*\* **2880 €**
- Maquetación de la Memoria (Ingeniero, 17 días) **2720 €**
- *Diseño de la presentación* (Ingeniero, 5 días) **800 €**

(\*)Se parte del hecho de que la persona contratada ya cuenta en su expediente académico o profesional con la titulación que certifica su experiencia y conocimiento de los lenguajes indicados. No sería por tanto necesario un periodo de formación.

(\*\*)Puesto que se sigue un esquema de tiempos similar al llevado a cabo por mí, se ha de tener en cuenta que la redacción del contenido de la memoria se realiza en paralelo a la programación de estilos y a las pruebas unitarias y de integración. Dicho solapamiento se ha tenido en cuenta la hora de realizar la estimación tiempo-coste de estas tareas.

### 3. Coste total

CONCEPTO	IMPORTE
Recursos Software	<b>200 €</b>
Recursos Hardware	<b>790 €</b>
Recursos Humanos	<b>32480 €</b>
<b>TOTAL</b>	<b>33470 €</b>

*El presupuesto total de este proyecto asciende a la cantidad de Treinta y Tres Mil Cuatrocientos Setenta Euros.*

*Madrid a 15 de Diciembre de 2012*

*Fdo. Ismael Baena Eguía.*



## Glosario

<b>XHTML</b>	(eXtensible HyperText Markup Language). XHTML es básicamente HTML expresado como XML válido.
<b>PHP</b>	(Hypertext Pre-processor). Lenguaje de programación de uso general de script del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico.
<b>MySQL</b>	Sistema de gestión de bases de datos relacional, multihilo y multiusuario.
<b>PDA</b>	(Personal Digital Assistant). Ordenador de bolsillo, organizador personal o agenda electrónica de bolsillo.
<b>PDF</b>	(Portable Document Format). Formato de almacenamiento de documentos digitales independiente de plataformas de software o hardware.
<b>CSS</b>	(Cascading Style Sheets). Lenguaje usado para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML).
<b>VoiceXML</b>	(Voice Extensible Markup Language). Formato XML estándar del W3C para la especificación de diálogos interactivos entre un humano y una máquina.
<b>W3C</b>	(World Wide Web Consortium). Consorcio Internacional que trabaja para el desarrollo de estándares, protocolos y pautas web.

<b>MIT</b>	(Massachusetts Institute of Technology). Institución de educación superior privada situada en Cambridge, Massachusetts, Estados Unidos.
<b>CMU</b>	(Carnegie Mellon University). Ubicado en la ciudad de Pittsburgh (Pensilvania) y es uno de los más destacados centros de investigación superior de los Estados Unidos en el área de informática y robótica.
<b>GPS</b>	(Global Positioning System). Sistema global de navegación por satélite.
<b>XML</b>	(eXtensible Markup Language). Metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C).
<b>TTS</b>	(Text to speech). Sistemas que permiten la conversión de textos en voz sintética.
<b>DTMF</b>	(Dual-Tone Multi-Frequency). En telefonía, el sistema de marcación por tonos empleado por los teléfonos analógicos y fijos convencionales, también llamado sistema multifrecuencial.
<b>DTD</b>	(Document Type Definition). Es una descripción de estructura y sintaxis de un documento XML o SGML.
<b>Script</b>	Archivo de órdenes o archivo de procesamiento por lotes, es un programa usualmente de poca extensión, que por lo regular se almacena en un archivo de texto plano.
<b>CDATA</b>	Sección que permite introducir caracteres especiales sin que sean interpretados por el navegador como código ejecutable, sino como texto. Compatible con ciertas versiones de XHTML, HTML y XML.

<b>DOM</b>	(Document Object Model). Es esencialmente una interfaz de programación de aplicaciones (API) que proporciona un conjunto estándar de objetos para representar documentos HTML y XML, un modelo estándar sobre cómo pueden combinarse dichos objetos, y una interfaz estándar para acceder a ellos y manipularlos.
<b>SGDB</b>	(Sistemas de Gestión de Bases de Datos). Es un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan.
<b>GNU</b>	El proyecto GNU fue iniciado por Richard Stallman con el objetivo de crear un sistema operativo completamente libre: el sistema GNU.
<b>GPL</b>	(General Public License) de GNU, licencia orientada principalmente a proteger la libre distribución, modificación y uso del software.
<b>MyISAM</b>	Tecnología de almacenamiento de datos usada por defecto por el sistema administrador de bases de datos relacionales MySQL.
<b>ODBC</b>	(Open DataBase Connectivity). Es un estándar de acceso a las bases de datos cuyo objetivo es hacer posible el acceder a cualquier dato desde cualquier aplicación, sin importar qué sistema de gestión de bases de datos almacene los datos.
<b>SAP</b>	Conjunto de programas que permiten a las empresas ejecutar y optimizar distintos aspectos como los sistemas de ventas, finanzas, operaciones bancarias, compras, fabricación, inventarios y relaciones con los clientes.
<b>TELNET</b>	(TELEcommunication NETwork). Protocolo de red que permite conectarnos a otra máquina para manejarla remotamente.

<b>HTTP</b>	(Hypertext Transfer Protocol). Es el protocolo más común de intercambio de información en la world wide web, el método mediante el cual se transfieren las páginas web a un ordenador.
<b>OSI</b>	(Open System Interconnection). modelo de red descriptivo creado por la Organización Internacional para la Estandarización (ISO). Marco de referencia para la definición de arquitecturas de interconexión de sistemas de comunicaciones.
<b>SmartPhone</b>	Teléfono móvil construido sobre una plataforma informática móvil, con una mayor capacidad de computación y conectividad que un teléfono móvil convencional.

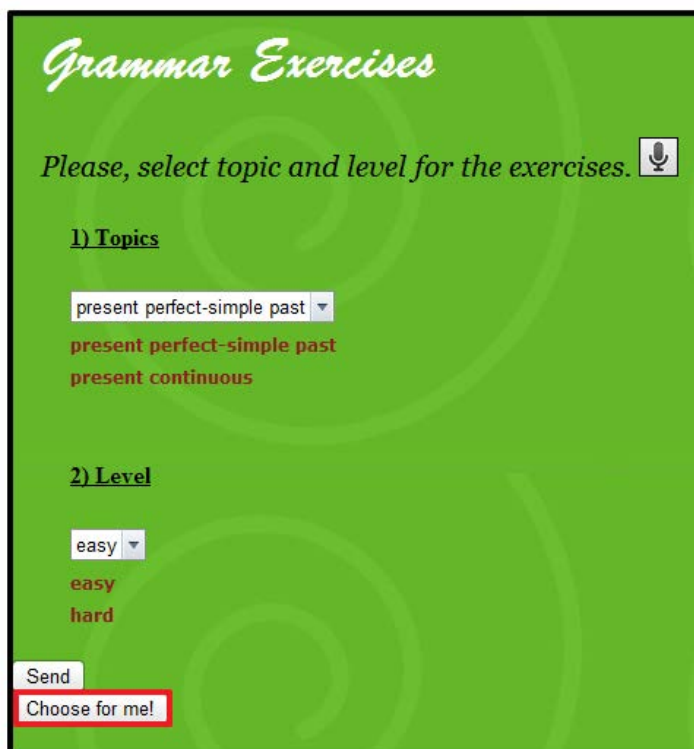


## Anexo


### Opción "Choose for me"

1. Funcionalidad: La opción "*choose for me*" es un complemento de uso opcional que se presenta en la página de selección de categoría y nivel, dentro del módulo de realización de ejercicios, para cualquiera de las familias (gramática, vocabulario y *listening*). La función de dicha opción es que el propio sistema decida automáticamente la categoría y nivel de los ejercicios estudiando una serie de criterios que se refieren al usuario concreto con sesión iniciada en la aplicación (Figura Anexo.1).

Los criterios de selección son similares para los ejercicios de gramática y vocabulario, sin embargo presentan ligeras variaciones en el caso de la selección de ejercicios de *listening*, por ese motivo han sido separados en el siguiente apartado donde se detallan.



*Grammar Exercises*

Please, select topic and level for the exercises. 

**1) Topics**

present perfect-simple past ▼

present perfect-simple past

present continuous

**2) Level**

easy ▼

easy

hard

Send

Choose for me!

Figura Anexo.1: Detalle de selección de la opción "*choose for me*"

1. Criterios y arquitectura

- *Opción choose for me ejercicios gramática y vocabulario:*

El archivo "**choose\_for\_me.php**" es el encargado de recibir la familia seleccionada (*grammar o vocabulary*) y realizar el estudio de criterios con el fin de seleccionar automáticamente los ejercicios que el usuario realizará. El primer paso, por tanto, y como sucedía en el resto de módulos, consiste en que el sistema determine la identidad del usuario. Como en módulos anteriores esta acción la realiza leyendo las variables de sesión que contienen esta información. Los pasos y criterios son los siguientes:

- Criterio 1: Se buscarán ejercicios que el usuario no haya hecho todavía y que pertenezcan al nivel que él mismo estimó e introdujo en el momento de hacer el registro de usuario. Para ello se consulta por los ejercicios que el usuario ha hecho ya de esa familia y nivel (Figura Anexo.2).

```
$sacar = "SELECT * FROM answers JOIN grammar_ex ON
        answers.ID_ex=grammar_ex.ID_ex WHERE nivel='$nivel' AND
        familia='grammar' AND ID_us='$idusuario";

$resultado = mysql_query($sacar,$conexion);
while ($registro = mysql_fetch_array($resultado))
{
    $done_array[$i] = $registro['ID_ex'];
    $i++;
}
```

Figura Anexo.2: Consulta de ejercicios realizados para cierta familia y nivel

\*Puesto que los criterios de consulta pertenecen a dos tablas diferentes es necesario realizar un *JOIN* que combine la información de ambas.

Tras la consulta se guardan en un vector (*\$done\_array*) los identificadores "*ID\_ex*" de los ejercicios que el usuario ya ha realizado.

A continuación, guardamos en otro vector (*\$contents\_array*) los ejercicios existentes (esta vez todos, no solo los ya realizados) del nivel correspondiente al usuario (Figura Anexo 3).

```
$sacar = "SELECT * FROM grammar_ex WHERE nivel = '$nivel'";

$resultado = mysql_query($sacar,$conexion);
while ($registro = mysql_fetch_array($resultado))
{
    $contents_array[$i] = $registro['ID_ex'];
    $i++;
}
```

**Figura Anexo 3:** Consulta ejercicios existentes del nivel del usuario

\*Si se diese el caso de que no existen ejercicios del nivel que introdujo el usuario como propio estimado, entonces se termina todo el proceso de estudio y se informa de este hecho en la página resultante.

A continuación una función compara los vectores que hemos generado anteriormente, es decir, compara el vector de ejercicios hechos con el de ejercicios existentes con el fin de guardar en otro vector los ejercicios "pendientes".

Una vez han quedado identificados los ejercicios pendientes se obtiene su número. Si es la mitad o más del número de ejercicios existentes entonces se envían esos ejercicios pendientes como resultante de la selección del *choose\_for\_me* (Figura Anexo.4).

```
if(($num_pendientes*2)>=$num_contents) /*Le quedan la mitad o más de
                                     ejercicios por hacer.*/
{
    /*En este caso se pasan los ID_ex de este array ($pendientes_array)
    a la siguiente página como ejercicios a realizar.
```

```
$caso=1; /*Se lleva una variable caso que guarda la justificación de  
dicha selección y que será mostrada en la siguiente página junto  
con los ejercicios seleccionados.  
  
enviar_array($pendientes_array,$caso,$familia,$idusuario,$nivel); //Se  
llama a la función encargada de pasar los ejercicios seleccionados a  
la siguiente página, junto con otros datos necesarios.  
  
}
```

**Figura Anexo 4:** Acción llevada a cabo en función de ejercicios pendientes

Si el número de ejercicios pendientes es menos de la mitad de los existentes, entonces se pasa el análisis al siguiente criterio.

- Criterio 2: Consiste en analizar si los ejercicios que el usuario ha realizado de su nivel presentan un número elevado de errores. Puesto que ya se tenía un vector con los ejercicios realizados por el usuario, tan sólo necesitamos ahora saber cuántos de estos son incorrectos (Figura Anexo.5). La consulta es similar a la anterior pero con una condición nueva ("correcto=0").

```
$sacar = "SELECT * FROM answers JOIN grammar_ex ON  
answers.ID_ex=grammar_ex.ID_ex WHERE nivel='$nivel' AND familia='grammar'  
AND ID_us='$idusuario' AND correcto='0'; //Hechos incorrectos de grammar de  
ese nivel.
```

**Figura Anexo.5:** Consulta de ejercicios incorrectos de cierta familia y nivel

Tras realizar la consulta se guardan los resultados en un (*\$num\_wrong*) y se cuentan. Si la mitad o más de ejercicios que ya ha hechos son incorrectos los seleccionados por el *choose for me* serán estos ejercicios incorrectos. La motivación es que el usuario repase ejercicios de su nivel ya que ha tenido un número elevado de errores. Si por el contrario, más de la mitad son correctos entonces se

establece que el nivel del usuario es el siguiente superior al que introdujo y se vuelve a realizar el análisis desde el principio pero teniendo en cuenta el nuevo nivel ( Figura Anexo.6).

```
if(($num_wrong*2)>=$num_done) /*La mitad o más de ejercicios que ha hecho
                               son incorrectos.*/
{
    $caso=2; /*La mitad o más de ejercicios que ha hecho son
            incorrectos.*/

    enviar_array($done_wrong_array,$caso,$familia,$idusuario,$nivel);
    //Se envía el array de ejercicios incorrectos a la siguiente página para que
    sean realizados.
}
else /*Mas de la mitad de ejercicios hechos están correctos.*/
{
    if($nivel=="hard") //Situación en que no se puede pasar de nivel porque
                       el usuario ya tenía el nivel establecido en "high".
    {
        $cambio_familia[1]="SI";
        $caso=3; /*Nivel high por lo que no se sube nivel y se le
                recomienda hacer ejercicios de otra familia.*/

        enviar_array($cambio_familia,$caso,$familia,$idusuario,$nivel);

    }
    else //La mitad o más de ejercicios correctos y nivel <> high.
    {

        $nivel=pasar_nivel(&$nivel); //Llamada a la función que sube el
                                     nivel del usuario.

        /*Volveríamos a hacer el estudio desde el principio pero con el
        nuevo nivel.*/
    }
}
```

Figura Anexo.6: Análisis de casos en función de los ejercicios incorrectos

Cabe comentar el caso observado en el código en el que el usuario ya ha hecho la mitad o más de ejercicios correctos del máximo nivel. En ese caso, el resultante de "choose for me" consiste en proponer al usuario que realice ejercicios de otra familia (se entiende que el usuario ha superado toda la dificultad de la aplicación para esa familia).

Cuando en los supuestos anteriores se dice que "se envían" los resultados del *choose\_for\_me* a la siguiente página, quiere decirse que el proceso de creación de la página resultante .xhtml se lleva a cabo en dos páginas o a través de dos ficheros PHP diferentes. El primero "*choose\_for\_me.php*" analiza y envía el resultado del análisis al siguiente PHP "*grammar\_options\_cfm.php*", que será el encargado de escribir la página resultante. Esto sucede de igual manera para los ejercicios de vocabulario y *listening*:

- ❖ Gramática: '*choose\_for\_me.php*' (Análisis y envío de resultados) -> '*grammar\_options\_cfm.php*' (escritura de la página a partir de los resultados recibidos) -> *grammar\_exercise.xhtml* (página resultante).
  - ❖ Vocabulario: '*choose\_for\_me.php*' (Análisis y envío de resultados) -> '*vocabulary\_options\_cfm.php*' (escritura de la página a partir de los resultados recibidos) -> *vocabulary\_exercise.xhtml* (página resultante).
  - ❖ Listening: '*choose\_for\_me\_listening.php*' (Análisis y envío de resultados) -> '*listening\_options\_cfm.php*' (escritura de la página a partir de los resultados recibidos) -> *listening\_exercise.xhtml* (página resultante).
- *Opción choose for me ejercicios de listening*

El criterio de selección de ejercicios y los algoritmos empleados para su análisis en la opción *choose for me* para el caso de ejercicios de *listening* es similar a la explicada anteriormente para ejercicios de gramática y vocabulario. El motivo de que presente cierta variación radica en el hecho de que mientras que en gramática y vocabulario los ejercicios son independientes y pueden ser mostrados así, en la categoría *listening* los ejercicios (preguntas de test) van asociadas a un único texto por lo que lo que el criterio de selección estudia los textos y no los ejercicios de manera individual. Los pasos y criterios (evitando repetir código y algoritmos similares a los anteriores) serían los siguientes:

- Criterio 1: Se guardan en un vector los textos que el usuario ya ha hecho (Figura Anexo.7).

```
$sacar = "SELECT DISTINCT ID_texto FROM answers JOIN listening_ex ON  
answers.ID_ex=listening_ex.ID_ex WHERE familia='listening' AND  
ID_us='$idusuario';
```

**Figura Anexo.7:** Consulta para obtener los textos de listening ya realizados por un usuario

\*Empleando un *JOIN* conseguimos esta información usando una única consulta.

Se guardan en otro vector los textos existentes de ese nivel (Figura Anexo.8).

```
$sacar = "SELECT * FROM listening_texts WHERE nivel = '$nivel';
```

**Figura Anexo.8:** Consulta para obtener todos los textos de cierto nivel

\*Al igual que antes si no hay ejercicios del nivel del usuario se termina el análisis del *choose for me* y se informa de este hecho en la página resultante.

Se comparan los vectores anteriores obteniendo con ello los textos que el usuario no ha hecho. Si le falta algún texto de su nivel se le lanza ese texto como resultante del *choose for me*.

Si tiene todos los textos de su nivel hechos se pasa al siguiente criterio.

- Criterio 2: Se analizan los fallos de cada texto que ha hecho. Si hay algún texto con la mitad o más de preguntas de test mal hechas será éste texto el que se obtenga como resultante.

Si todos los textos de su nivel tienen la mitad o más de preguntas de test bien hechas se establece que el nivel del usuario es el siguiente al actual y se vuelve a realizar el análisis desde el principio pero teniendo en cuenta el nuevo nivel.

\*Al igual que antes si tiene todos los textos de nivel "*hard*" hechos y todos con al menos la mitad de preguntas correctas se le sugerirá al usuario como resultante del *choose for me* que pruebe a realizar ejercicios de otra familia (entendiendo que ya ha superado en *listening* el nivel de la aplicación completa).

3. Diagramas y escenarios

Las Figuras Anexo.9 y Anexo.10 muestran diagramas de flujo para facilitar la comprensión del funcionamiento de la opción para los casos de ejercicios de gramática y vocabulario y por otro lado los ejercicios de *listening*.

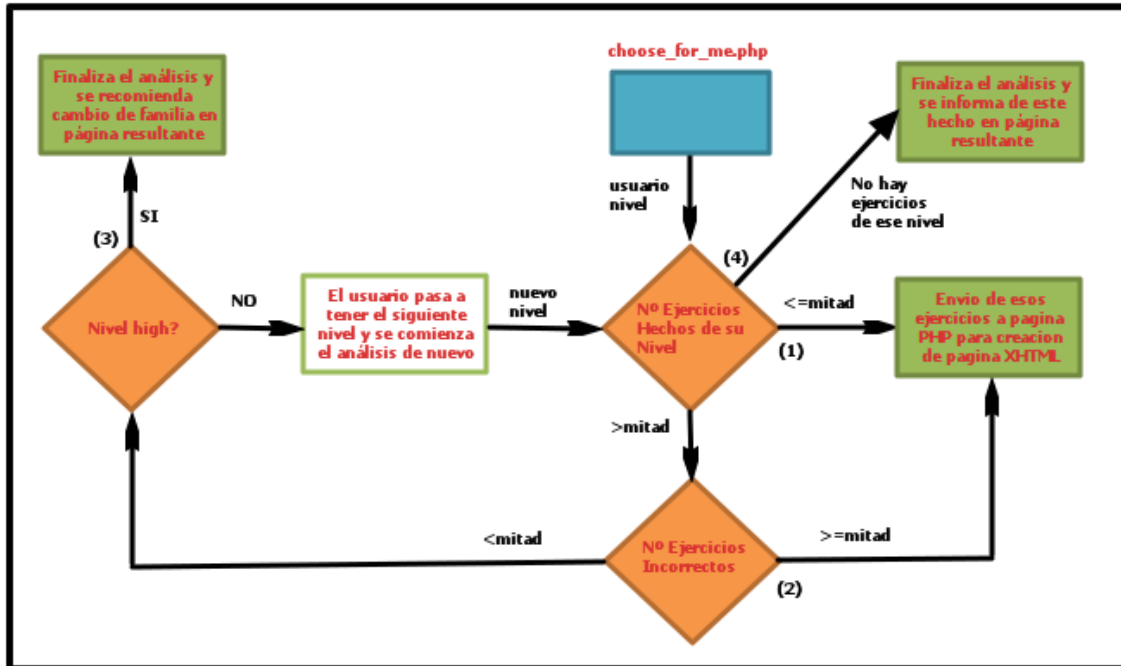


Figura Anexo.9: Diagrama de flujo "choose\_for\_me" gramática y vocabulario



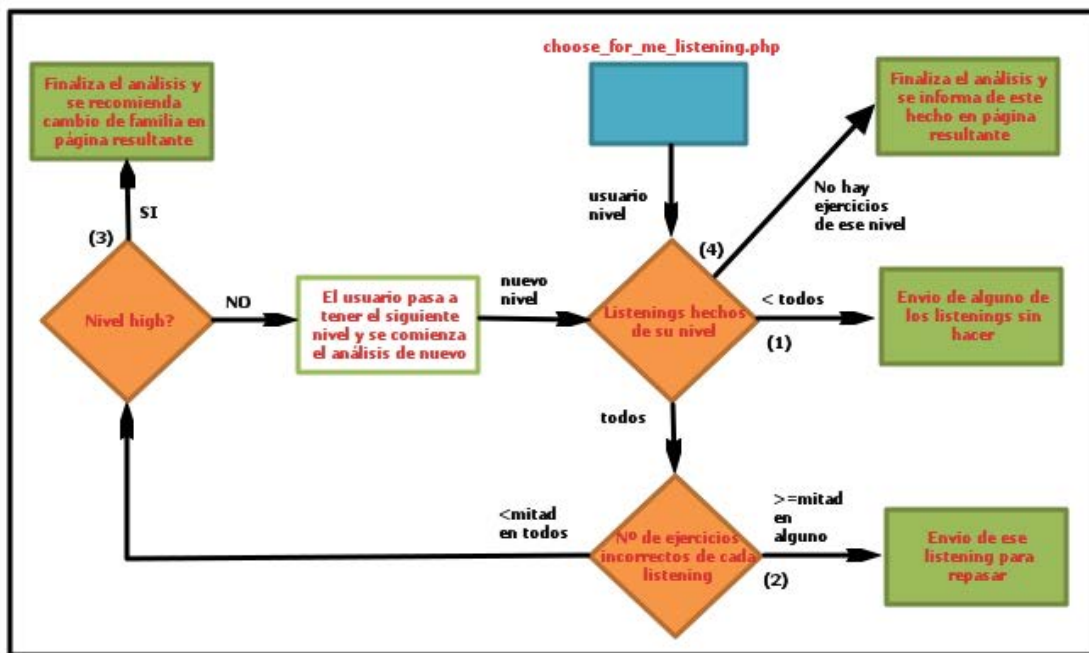


Figura Anexo.10: Diagrama de flujo "choose\_for\_me" caso listening

En ambos diagramas se muestran entre paréntesis los diferentes casos posibles. La página resultante XHTML muestra en cada caso una explicación de por qué el sistema ha escogido los ejercicios seleccionados. Las Figuras Anexo.11 a Anexo.14 muestran a continuación ejemplos de dichas explicaciones.

- Caso 1:

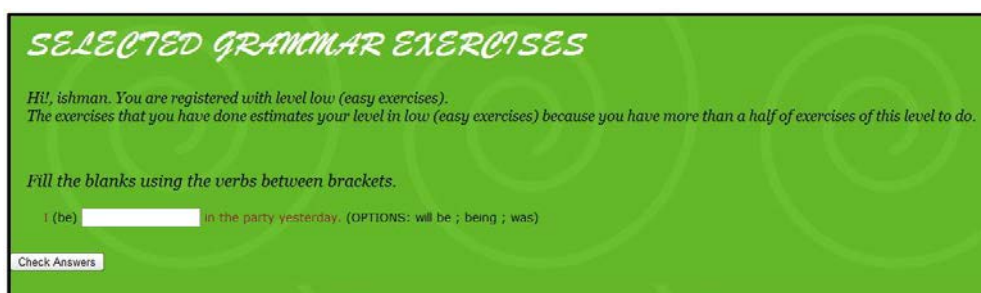


Figura Anexo.11: Vista del mensaje resultante (Caso 1)

- Caso 2:



Figura Anexo.12: Vista del mensaje resultante (Caso 2)

- Caso 3:

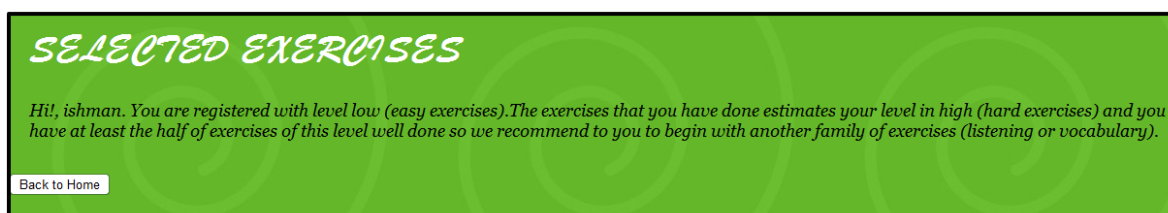


Figura Anexo.13: Vista del mensaje resultante (Caso 3)

- Caso 4:

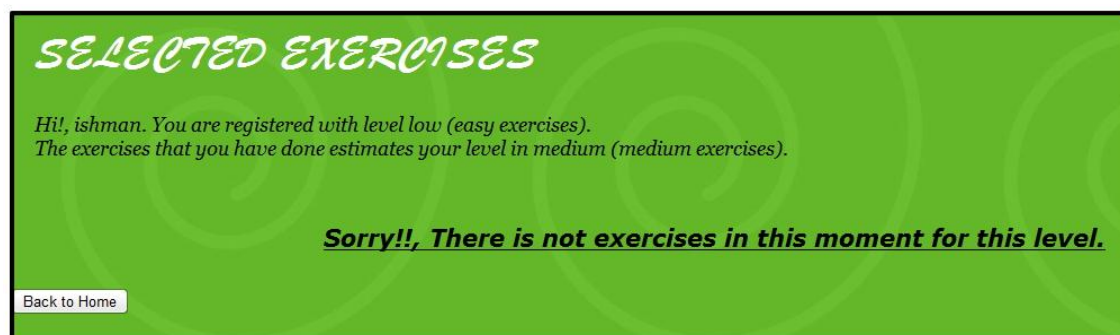


Figura Anexo.14: Vista del mensaje resultante (Caso 4)

\*Los textos mostrados son ejemplos extraídos de la opción "choose\_for\_me" para la familia de ejercicios de gramática. En el caso de vocabulario y *listening* las explicaciones son las mismas pero adaptadas a esos casos.

## Bibliografía

---

### ▪ Capítulo 1:

#### - *Introducción:*

[1] R. Pieraccini and L. Rabiner. *The Voice in the Machine: Building Computers that Understand Speech*. The MIT Press, 2012.

[2] R. López-Cózar and M. Araki. *Spoken, Multilingual and Multimodal Dialogue Systems*. John Wiley & Sons Publishers, 2005.

[3] R. López-Cózar, M. Gea, F. L.Gutiérrez. *Uso de Estrategias Adaptativas en Sistemas de Diálogo*. Ingeniería Informática Universidad de Granada, Dpto. Lenguajes y Sistemas Informáticos. Disponible en:  
<http://lsi.ugr.es/~fguti/taller/o6/cozar.pdf>  
[http://www.ugr.es/~rlopezc/sistemas\\_dialogo.htm](http://www.ugr.es/~rlopezc/sistemas_dialogo.htm)

[4] Antonio Bonafonte, Joaquim Llisterri, María J.Machuca. *Los sistemas de diálogo* Ed. Fundación Duques de Soria, Universitat Autònoma de Barcelona, 2006. (Sección Capítulo 1).Disponible en:  
[http://books.google.com/books?id=oYyBKB3JwoC&pg=PAg&hl=ca&source=gbv\\_toc\\_r&cad=4#v=onepage&q&f=false](http://books.google.com/books?id=oYyBKB3JwoC&pg=PAg&hl=ca&source=gbv_toc_r&cad=4#v=onepage&q&f=false)

[5] Wolfgang Wahlster. *Smartkom: Foundations of Multimodal Dialogue Systems (Cognitive Technologies)*. Springer-Verlag New York, Secaucus, NJ, USA 2006.

### ▪ Capítulo 2:

#### - *Características de los Sistemas de Diálogo:*

[6] Blog "Codificando Escarabajos". Entrada: Reconocimiento de Voz, Noviembre 2010. Disponible en:

<http://codificandoescarabajos.wordpress.com/2010/11/20/reconocimiento-de-voz/>

[7] Página Web: "Reconocimiento de voz Automático". Disponible en: <http://www.euskalnet.net/iosus/speech/recog.html>

[8] WIKIPEDIA. Concepto "Reconocimiento del Habla". Disponible en: [http://es.wikipedia.org/wiki/Reconocimiento\\_del\\_habla#Clasificaci.C3.B3n](http://es.wikipedia.org/wiki/Reconocimiento_del_habla#Clasificaci.C3.B3n)

- *Historia de los sistemas de Diálogo:*

[9] Angélica Ahuactzin Larios .*Diccionario español/inglés para el aprendizaje de vocabulario utilizando una interfaz de voz*. Universidad de las Américas Puebla, Escuela de Ingeniería Departamento de Ingeniería en Sistemas Computacionales (1999) (Sección Capitulo 1). Disponible en: [http://catarina.udlap.mx/u\\_dl\\_a/tales/documentos/lis/ahuactzin\\_l\\_a/capitulo\\_1.html#](http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/ahuactzin_l_a/capitulo_1.html#)

[10] Jorge Hierro Álvarez. *Informe técnico sobre los sistemas de reconocimiento de voz*. Departamento de Prensa Assit. Madrid, Junio de 2004. Disponible en: <http://jorgehierro.files.wordpress.com/2008/02/voice-reconigntionii.pdf>

[11] Francisco Torres Goterris. *Sistemas de diálogo basados en modelos estocásticos*. Tesis Doctoral. Departamento de Sistemas Informáticos y Computación Universidad Politécnica de Valencia, Valencia 2006. Disponible en: <http://www.dsic.upv.es/docs/bib-dig/tesis/etd-11092005-131117/tesisREVISADA6.pdf>

[12] B. Green, A. Wolf, C. Chomsky, K. Laughery. *BASEBALL: An Automatic Question* .Proc.of the Western Joint Computer Conference. 1963, páginas 219-224.

[13] D.G.Bobrow. *Natural Language Input for a Computer Problem-solving System*. M. Minsky B.(ed.) Semantic Information Processing. Cambridge, Mass: MIT Press.1968, páginas 146-226.

[14] Joseph Weizenbaum. *ELIZA, A Computer Program for the Study of Natural Language Communication between Man and Machine*. Communications of the ACM. 1965, volumen 9, páginas 36-45.

[15] T. Winograd. *Understanding Natural Language*. Academic Press Eds. 1972.

[16] Kenneth Mark Colby. *Artificial Paranoia: A Computer Simulation of Paranoid Processes*. Elsevier Science Inc. New York, NY. 1975.

- *Introducción a VoiceXML / XHTML + V:*

[17] Web MSDN en línea. ¿Por qué XML?. Disponible en:  
<http://www.microsoft.com/latam/msdn/articulos/2000/03/arto3/#top>

[18] Young M. *XML Step by Step*, Washington, Microsoft Press, página 12.

[19] Web monografias.com. "Manual XML". Disponible en:  
<http://www.monografias.com/trabajos7/xml/xml.shtml>

[20] Deusdit A. Correa Cornejo. *La Nueva Era de la Voz*. Technical Manager - VOXIVA, Br. en Ciencias de la Computación. Disponible en:  
[http://www.informatizate.net/articulos/la\\_nueva\\_era\\_de\\_la\\_voz\\_parte\\_02\\_12052004.html](http://www.informatizate.net/articulos/la_nueva_era_de_la_voz_parte_02_12052004.html)

[21] VoiceXML Tutorial. Disponible en:  
<http://www.voicexml.org/voicexml-tutorials/introduction>

[22] VoiceXML Forum. Disponible en:  
<http://www.voicexml.org/>

[23] R. López-Cózar, E. Sanchís, E. Segarra, J. M. Benedí. *Incorporación de VoiceXML en el Sistema de Diálogo DIHAN*. Dpto. Lenguajes y Sistemas Informáticos E.T.S. Ingeniería Informática Universidad de Granada. Dpto. Sistemas Informáticos y Computación Universidad Politécnica de Valencia. Disponible en:  
<http://www.aipo.es/articulos/3/36.pdf>

**[24]** XHTML+Voice Programmer's Guide. IBM Corporation. 2004. Disponible en:

[ftp://ftp.software.ibm.com/software/pervasive/info/multimodal/XHTML\\_voice\\_programmers\\_guide.pdf](ftp://ftp.software.ibm.com/software/pervasive/info/multimodal/XHTML_voice_programmers_guide.pdf)

**[25]** Voice Extensible Markup Language (VoiceXML) 3.0. Especificaciones del W3C. Disponible en:

<http://www.w3.org/TR/2010/WD-voicexml30-20100831/>

XHTML+Voice Profile 1.0. Especificaciones del W3C. Disponible en:

<http://www.w3.org/TR/xhtml+voice/>

▪ **Capítulo 3:**

- *Tecnologías:*

**[26]** Web: Opera Software Instrucciones Uso de Voz disponible en:

<http://help.opera.com/Windows/8.54/es-ES/voice.html>

**[27]** Web: Comparison of Oracle, MySQL and PostgreSQL DBMS. Disponible en:

<http://www-css.fnal.gov/dsg/external/freeware/mysql-vs-pgsql.html>

**[28]** Blog "Indira Informática". ¿Qué es MySQL?. Disponible en:

<http://indira-informatica.blogspot.com.es/2007/09/qu-es-mysql.html>

**[29]** WIKIPEDIA. Concepto "MySQL". Disponible en:

<http://es.wikipedia.org/wiki/MySQL>

**[30]** Web: desarrolloWeb.com. "Descripción y funcionalidades de phpMyAdmin". Disponible en:

<http://www.desarrolloweb.com/articulos/844.php>

**[31]** WIKIPEDIA. Concepto "phpMyAdmin". Disponible en:

<http://es.wikipedia.org/wiki/PhpMyAdmin>

**[32]** Web: desarrolloWeb.com."Descripción y funcionalidades de MySQL Administrator", disponible en:

<http://www.desarrolloweb.com/articulos/1798.php>

**[33]** Web: CiberAula Linux. "Una Introducción a Apache". Disponible en:

[http://linux.ciberaula.com/articulo/linux\\_apache\\_intro/](http://linux.ciberaula.com/articulo/linux_apache_intro/)

**[34]** WIKIPEDIA. Concepto "Servidor Web". Disponible en:

[http://es.wikipedia.org/wiki/Servidor\\_web](http://es.wikipedia.org/wiki/Servidor_web)

**[35]** WIKIPEDIA. Concepto "Servidor HTTP Apache". Disponible en:

[http://es.wikipedia.org/wiki/Servidor\\_HTTP\\_Apache](http://es.wikipedia.org/wiki/Servidor_HTTP_Apache)

- *Implementación de las Operaciones Generales:*

**[36]** Christian Pelissier Q. "Programación con PHP". Disponible en:

<http://profesores.elo.utfsm.cl/~agv/elo330/2so2/projects/pelissier/informe.html>

**[37]** Web: desarrolloWeb.com. "Sesiones en PHP". Disponible en:

<http://www.desarrolloweb.com/articulos/321.php>

▪ **Capítulo 4:**

**[38]** XHTML+Voice Profile 1.0. Especificaciones del W3C. Disponible en:

<http://www.w3.org/TR/xhtml+voice/>