UNIVERSIDAD
REY JUAN CARLOS

ESCUELA SUPERIOR DE INGENIERÍA INFORMÁTICA

INGENIERÍA INFORMÁTICA

Curso Académico 2009/2010

Proyecto de Fin de Carrera

Generación de esquemas de diálogos a partir de modelos conceptuales

Autor: Carlos Caballero

Tutores: Diana Pérez

Índice

Índice de figuras

Índice de tablas

Resumen

La principal motivación de este proyecto reside en la necesidad de explorar formas avanzadas de proporcionar un repaso de los contenidos vistos en clase a los estudiantes. En particular, el objetivo es poder proporcionar la posibilidad de interactuar en lenguaje natural con los estudiantes para realizar un repaso de los conceptos vistos en clase, usando esquemas de diálogo generados a partir de plantillas XML.

Para ello, en este proyecto se ha implementado un sistema interactivo de repaso en lenguaje natural via web llamado JARO. La metodología utilizada para el desarrollo de esta herramienta ha sido el ciclo de vida en cascada con iteración. El lenguaje de programación escogido es PHP en un servidor Apache con MySQL y XML como lenguaje de intercambio para otras plataformas.

Para poder repasar con JARO, en primer lugar, el profesor debe completar una plantilla en XML que contiene la información conceptual del tema. JARO implementa un algoritmo capaz de generar a partir de la plantilla XML el diálogo de turnos pregunta-respuesta con el estudiante. Además, también guarda la información del XML en la base de datos para poder ir actualizando la información de cada estudiante según interacciona con el sistema.

Otro de los objetivos de este trabajo es desarrollar JARO para que pueda ser utilizado tanto por usuarios con y sin conocimientos técnicos. Por lo tanto, se ha puesto especial interés en diseñar una interfaz “amigable” y en permitir varios modos de introducir la información en el sistema para dar las máximas facilidades a los estudiantes que repasen con el sistema.

También se ha trabajado en la idea de que el sistema tenga la mayor aceptación posible y pueda ser distribuido mediante herramientas de software libre e integrado con otros sistemas e-learning.

Las pruebas realizadas confirman que JARO cumple los requisitos especificados, y en una experiencia piloto con 6 usuarios (algunos sin experiencia informática) se ha podido comprobar cómo repasar con el sistema les resulta sencillo y útil.

1 Introducción

El objetivo principal de este proyecto es realizar un sistema interactivo de repaso mediante preguntas generadas a partir de un esquema conceptual en XML. Los archivos XML seguirán el formato de la herramienta e-learning que se utilice para enseñar, como por ejemplo podría ser Willow [1]. Para ello, se ha diseñado e implementado un algoritmo para generar esquemas de diálogo a partir de ficheros XML.

Para este propósito se ha desarrollado e implementado un algoritmo para generar de manera automática un esquema de diálogo en XML en función del modelo conceptual seleccionado. Un modelo conceptual se puede definir como una red de conceptos relacionados entre sí por su contenido semántico.

El primer paso del algoritmo consiste en analizar los ficheros XML, y guardar su contenido en una base de datos con información del estudiante. El siguiente paso es interactuar con el estudiante, actualizando los valores de la base de datos y generando nuevas preguntas en función de los resultados obtenidos previamente.

La hipótesis sobre la que se trabaja en el proyecto es que esta estructura de modelo conceptual se pueda usar como base para guiar la conversación con un usuario en la aplicación de e-learning. Se ha implementado una aplicación, llamada JARO, en la que con el uso de XML como esquema para el soporte de la estructura conceptual, consigue guiar la conversación-aprendizaje entre el alumno-maquina.

El contexto de este proyecto está en el auge que los sistemas de e-learning en los últimos años, ya que estos están siendo utilizados cada vez en más campos de la sociedad, tanto en el ámbito privado (formación a personal en empresas) o público (interacción usuario-máquina en temas de la administración). Además, hay que tener en cuenta que los sistemas e-learning van dirigidos también a personas sin una formación informático técnica, población que dispone únicamente de conocimientos básicos de uso de ordenadores.

Por lo tanto los sistemas e-learning deben ser usables y facilitar el acceso a estas herramientas por parte de población que no tenga una formación técnica en la materia [2]. Su manejo e interacción debe ser intuitivo y agradable. En caso contrario, si son herramientas demasiadas complejas de manejar o en las que el alumno no se sienta cómodo, pueden originar abandonos en el programa de aprendizaje [3].

En particular en este trabajo, nos centramos en el campo de la Interacción en Lenguaje Natural, según el cual el usuario interacciona con el ordenador en lenguaje natural [4]. Se pretende así que los sistemas e-learning sean más usables por la parte de la población que no tiene una formación técnica, y que se encuentren en un entorno más natural (similar a la interacción humano-humano) a la hora de interactuar con el ordenador.

JARO se asemeja a una herramienta de chat, en la que alumno, mediante lenguaje natural, decide qué acciones realizar. La aplicación ha sido probada con 6 usuarios de diferentes perfiles, para comprobar si es usable incluso para personas sin conocimientos técnicos. Los resultados de la experiencia realizada han sido positivos.

Un ejemplo de JARO sería el siguiente:

1. El diseñador del curso crea el archivo XML según la guía proporcionada con la información del curso y el esquema del modelo conceptual inicializado del alumno.
2. Se valida el archivo XML con la DTD correspondiente.
3. Se aplica el algoritmo para generar el esquema de diálogo a partir del XML validado.
4. La aplicación JARO usa el esquema conceptual generado para formular las preguntas al alumno.
5. El alumno interactúa con la herramienta JARO para ir modificando el valor de confianza de los conceptos aprendidos o repasados.

2 Objetivos

2.1 Descripción del problema

La necesidad de este proyecto radica en proporcionar mecanismos de repaso interactivo que apoyen al estudiante. Para ello, este proyecto tiene como objetivos la generación de esquemas de diálogo a partir de modelos conceptuales. Un modelo conceptual es un diagrama que ilustra una serie de relaciones entre ciertos factores que se cree impactan o conducen a una condición de interés en un cierto dominio.

Estos diálogos consisten en turnos de pregunta-respuesta que se implementan en el sistema web llamada JARO.

Las preguntas que se le formulan al alumno, no son realizadas al azar, sino que siguen estos criterios:

- Para poder repasar un concepto, éste previamente debe haber sido aprendido.
- Se repasan los conceptos que peor conoce el alumno.
- Si una respuesta es correcta aumenta el conocimiento del alumno y no se repite.
- Un concepto aprendido pasa en los siguientes turnos a ser repasado.

También se pretende que los alumnos no necesiten tener conocimientos técnicos para poder usar la herramienta. En la descripción del problema es importante tener en cuenta el factor de la usabilidad, y no excluir a las personas sin conocimientos técnicos.

Para ello, se ha propuesto como medio de comunicación principal para la interacción estudiante-máquina el lenguaje natural, a través del uso de patrones del lenguaje y autoevaluaciones por parte del alumno [5]. También se ha permitido el uso de menús para aquellos usuarios que prefieran no hacer uso de la Interacción en Lenguaje Natural, habilitando de esta forma varios modos de interacción.

2.2 Requisitos

Los requisitos para la elaboración del proyecto han sido los siguientes:

1) Facilidad de uso: La aplicación se debe poder usar tanto por usuarios con y sin conocimientos técnicos.

2) Interacción en lenguaje natural: Uso de patrones de lenguaje para la formulación y corrección de preguntas. Con este punto se pretende que las preguntas que se formulen sean lo más parecidas a una interacción en lenguaje natural.

3) Intercambio de información en XML: Validación de un fichero XML de entrada mediante la estructura correspondiente, validaremos la estructura del fichero contra la DTD, para que siempre tenga la misma estructura y validación de un fichero XML de salida mediante la DTD correspondiente, en este caso realizaremos el mismo procedimiento que en el caso anterior.

4) Permanencia y actualización de los datos: Almacenamiento de los datos de entrada en el sistema gestor de bases de datos, para luego poder trabajar con ellos y realizar posteriormente modificaciones sobre los mismos.

5) Interacción entre distintas plataformas e-learning: Posibilidad de aplicación del algoritmo a otros sistemas e-learning, ya que mediante la descarga de los ficheros XML que se generan es fácilmente portable.

6) Accesible vía web: Permitir que los usuarios usen la herramienta desde cualquier ordenador conectado a Internet y repasar con ella todo el tiempo que desee.

2.3 Estudio de alternativas

En este apartado se proporciona el estudio de las distintas alternativas posibles y, el porqué de las tecnologías usadas.

Se especificarán las tecnologías aceptadas para el proyecto y la elección de la más adecuada para el mismo. Se estudiarán en primer lugar los distintos servidores web, para luego pasar a las bases de datos, las distintas alternativas para el intercambio estructurado de la información, los lenguajes de programación, y finalmente las aplicaciones unificadas.

2.3.1 Servidores web

A continuación se revisan algunos posibles servidores:

2.3.1.1 Servidor ISS

Este servidor, es de distribución no libre por parte de Microsoft, por lo que para su uso, se requiere de licencias software. Permite la publicación de páginas web y ftp, tanto en local como en red pero para su uso se necesita que corra bajo la plataforma Windows Server, con licencia implícita [6].

Las ventajas de elegir este servidor son**:**

- Si se tiene una licencia para un sistema operativo Microsoft, (Windows Server) este software ya viene incluido.
- Facilidad de manejo.
- Aconsejable para usuarios con pocos conocimientos que busquen un servidor web, fácil y usable bajo Windows.

Los inconvenientes de elegir este servidor son:

- Solo se puede usar bajo licencia.
- No es compatible con otros sistemas que no sean Microsoft.
- No permite una administración para temas complejos.

2.3.1.2 Servidor GlassFISH

Este servidor web, desarrollado por Sun Microsystems permite servir páginas web, ftp y tecnología Java. Además, es de libre distribución bajo una licencia GPL.

La tecnología de este servidor deriva principalmente de tecnología usado por Sun y Apache Tomcat.

Las ventajas de elegir este servidor son:

- Multiplataforma, por lo que se puede utilizar bajo distintos sistemas operativos.
- No requiere de licencia propietaria, por lo que no es necesario adquirir una licencia para su uso.

Los inconvenientes de elegir este servidor son:

- Menor cantidad de documentación y ejemplos en la red.
- Proyecto reciente con una menor consolidación.

2.3.1.3 Servidor Web Apache

Este servidor es de código abierto y está respaldado por una gran comunidad de desarrolladores que resuelven los posibles fallos y vulnerabilidades de este software. Su creación es anterior a los dos anteriormente citados, y la experiencia en problemas y vulnerabilidades en la mayoría de los casos está documentada en la web.

Las ventajas de elegir este servidor son:

- Gran uso en la red, un porcentaje alrededor del 60% de las páginas servidas en la red, son servidas por Apache.
- Software Libre y código abierto. Multiplataforma.
- Configuración fácil y sencilla a través de fichero de configuración

No se han encontrado inconvenientes de elegir este servidor para este proyecto.

En la Tabla 1 se puede observar la comparativa entre las alternativas de servidores Web que se han analizado previamente.

	ISS	*GlassFish*	*Apache*
Licencia Libre	No	Si	Si
Multiplataforma	No	Si	Si
Información On-Line	Poco	Media	Alta
Complejidad Administración	Ninguna	Alta	Media

Tabla 1. Comparativa de servidores Web

En función de esta comparativa, se escoge el servidor Apache, puesto que es de licencia libre, multiplataforma y está más probado al llevar más tiempo en uso. Además, se dispone de más documentación técnica de este servidor.

2.3.2 Bases de datos

En este apartado, se revisan varios gestores de bases de datos relacionales.

2.3.2.1 MySQL

Es un sistema de gestión relacional de bases de datos, mutihilo y multiusuario, que funciona bajo licencia GNU GPL [7].

Además, es un sistema multiplataforma, y se caracteriza, por tener una potencial relativamente alto. Al tener libre licencia, esto le coloca en una posición privilegiada para ser usado en aplicaciones de pequeño y mediano tamaño.

Las ventajas de este gestor son:

- Software de libre licenciamiento, por lo que para la realización de nuestro proyecto no necesitamos la adquisición de una licencia.
- Manejable y administrable, en aplicaciones pequeñas y medianas.

Los inconvenientes de elegir este gestor son:

- Futuro incierto tras la adquisición de MySQL por parte de Oracle.

2.3.2.2 ORACLE

Este sistema de gestión relacional de bases de datos está orientado a grandes aplicaciones y empresas. El licenciamiento de este producto es bajo pago. Es el principal proveedor y distribuidor de aplicaciones de gestión de base de datos [8].

Las ventajas de elegir este gestor son:

- Gran escalabilidad.
- Soporte para cualquier dificultad o problema en el mantenimiento o creación de la base de datos.
- Estabilidad de la plataforma.
- Soporte multiplataforma.

Los inconvenientes de elegir este gestor son:

- La integración en un futuro con otras aplicaciones de e-learning, será de mayor dificultad, debido al problema de tener una licencia propietaria.
- Alta complejidad de administración.

2.3.2.3 Microsoft Access

Es un software propietario de Microsoft que permite el tratamiento de datos. Está enfocado a un uso ofimático o de pequeñas aplicaciones [9].

Las ventajas de elegir este gestor son:

- Es de muy fácil manejo.
- Interfaz visual muy clara.

Los inconvenientes de elegir este gestor son:

- Licenciamiento de pago.
- Posibilidades muy limitadas.
- Obligación de que este software corra bajo plataforma tipo Windows.
- Pocas posibilidades de integración con distintas aplicaciones.

En la Tabla 2 se muestra la comparativa de los gestores analizados.

	MySQL	*Oracle*	*Microsoft Access*
Licencia libre	Si	No	No
Soporte On-Line	Si	Si	No
Escalabilidad en medianos proyectos	Si	Si	No
Curva de Aprendizaje	Alta	Media	Baja
Aprovechamiento de Recursos	Alta	Baja	Alta
Multiplataforma	Si	Si	No

Tabla 2 Comparativa Bases de datos

Según la comparativa, se elige el gestor de base de datos relacional MySQL como más apropiado para nuestras necesidades.

2.3.3 Intercambio estructurado de información

Para este propósito utilizaremos XML, aunque en un principio XML fue ideado como un metalenguaje, se han observado su efectividad a la hora de utilizarlo como protocolo de intercambio de datos, no obstante también hay que mencionar que este lenguaje, no fue creado para su uso en Internet, sino que observando las posibilidades que ofrece, se ha decidido usar casi como estándar en intercambio de datos [10].

Otras posibilidades de lenguajes para intercambio estructurado de información son XML, RDF, OIL y DAML. Finalmente, se ha optado por XML porque para la realización de este proyecto se prefiere la versatilidad de poder crear cualquier conjunto de etiquetas frente a la complejidad de aprender otros lenguajes que puedan ser más adecuados para el manejo de ontologías y gestión de relaciones conceptuales más complejas.

Mediante XML, exportaremos la información de nuestra aplicación, para posteriormente y mediante un tratamiento en HTML, generar una página Web visible para el usuario.

2.3.4 Lenguajes de Programación web

En este apartado, se describirán varios lenguajes de programación web para comparar sus características y en base a esto decidir cual puede ser el más adecuado para este proyecto.

2.3.4.1 PHP

Es un lenguaje de programación interpretado, que se usa fundamentalmente para la realización de scripts, incrustados dentro de código HTML, con lo que se consigue el resultado de páginas Web dinámicas. La ejecución de PHP, se realiza como norma dentro de un servidor web (ej. Apache).

Las ventajas de elegir este lenguaje son:

- Es ampliamente utilizado en la programación web.
- Posee una licencia GPL que permite su uso sin un pago previo.
- Curva de aprendizaje corta.
- Interacción con bases de datos de cualquier tipo.
- Portabilidad a distintas plataformas.

Los inconvenientes de elegir este lenguaje web son:

- El modelo de objetos no es demasiado claro.
- Sin actualizaciones automáticas.

2.3.4.2 Java Server Pages (JSP)

Java Server Pages es una tecnología propietaria de Java, que al igual que PHP y ASP permite la generación de páginas Web con contenido dinámico. Otro de los usos más importantes que tiene JSP, es la construcción de servlets. Pero más allá de estas aplicaciones JSP, es un lenguaje de propósito general [11].

Las ventajas de elegir este lenguaje web:

- Fácil interacción con distintos servidores.
- Se pueden crear aplicaciones vistosas, más que con PHP o ASP.
- Portabilidad a distintas plataformas.

Los inconvenientes de elegir este lenguaje web:

- Lentitud debido al uso de Java.

2.3.4.3 Active Server Pages (ASP)

Es la alternativa de Microsoft a PHP. Viene como un complemento a ISS y se encarga de generar páginas web de manera dinámica. Es un lenguaje pensado a propósito para la web.

Las ventajas de elegir este lenguaje son:

- Se puede obtener como ampliación de ISS.
- Facilidad de trabajo con ActiveX.
- Actualizaciones automáticas de software.

Los inconvenientes de elegir este lenguaje son:

- Licencia propietaria de Microsoft.
- Solo corre en sistemas Windows.

La Tabla 3 recoge la comparativa entre los lenguajes de programación web revisados.

	PHP	*JSP*	*ASP*
Licencia libre	Si	Si	No
Curva de aprendizaje	Alta	Media	Media
Velocidad de ejecución	Rápida	Baja	Media
Integración con demás soluciones	Muy buena	Buena	Mala
Propósito del lenguaje	Específico Web	General	Especifico Web

Tabla 3 Comparativa lenguajes de programación

Se decide escoger PHP, debido a la buena integración que muestra con las demás tecnologías escogidas, además de su buena respuesta de ejecución y su licencia libre.

2.3.5 Aplicaciones unificadas

Con estas aplicaciones unificadas nos vamos a referir a aquellas que unen, en un solo instalador, una base de datos, un lenguaje de programación script (o mas de uno) y un servidor Web .

2.3.5.1 WAMP

Es un acrónimo utilizado para definir una infraestructura, para el servicio de páginas Web. En este caso, sus siglas significan Windows, Apache, MySQL y PHP.

Las ventajas de elegir esta aplicación unificada son:

- Fácil instalación tipo y de fácil manejo
- Durante la instalación no se requieren demasiados conocimientos de lo que se va a instalar.
- Ahorro de tiempo, frente a la instalación de herramienta por herramienta por herramienta.

Los inconvenientes de elegir esta aplicación unificada son:

- Es necesario un sistema Windows, que requiere de licencia para la instalación del mismo WAMP.

2.3.5.2 XAMPP

Este software a diferencia de WAMPP es multiplataforma, y como diferencia incluye PERL en lugar de PHP.

Las ventajas de elegir esta aplicación unificada son:

- Fácil instalación.
- Ahorro de tiempo.
- Incluye también Perl.

Los inconvenientes de elegir esta aplicación unificada son**:**

- No se utlizará PERL por lo que no se requiere su instalación.

2.3.5.3 LAMP

Contiene las mismas herramientas que WAMP pero se ejecuta en Linux en lugar de en Windows.

Las ventajas de elegir esta aplicación unificada son:

- Fácil instalación tipo Windows, de fácil manejo.
- Durante la instalación no se requieren demasiados conocimientos de lo que se va a instalar.
- Ahorro de tiempo de instalación.
- Licenciamiento libre a la hora de tener la plataforma de instalación.

Los inconvenientes de elegir esta aplicación unificada son:

- Requiere conocimientos de administración de Unix.
- Por defecto tiene configuraciones de poca utilidad para este proyecto, como puede ser una política excesiva de seguridad en el servidor web

La Tabla 4 recoge la comparativa de las aplicaciones unificadas.

	WAMP	*XAMPP*	*LAMP*
Requiere algún software propietario	Si	Si	No
Servidor Web	Si	Si	Si
Lenguaje Script	Si	Si	Si
Base de datos	Si	Si	Si
Fácil instalación y documentación abundante	Medio	Medio	Alto

Tabla 4 Comparativa aplicaciones unificadas

En función de las decisiones tomadas, la aplicación unificada más adecuada a este proyecto es LAMP.

2.3.6 Toma de decisiones

En función del análisis de las alternativas realizadas anteriormente se escogen:

- Se utilizará la aplicación unificada LAMP con modificaciones que realizaremos bajo los distintos tipos de software que engloba, para adecuarlos lo más posible a nuestras necesidades.

- La transmisión de datos será bajo un formato XML, que será validado contra un XML esquema y una DTD.

En general, siempre se dará prioridad a tecnologías libres. Para que nuestra aplicación, en un futuro sea lo más portable e integrable en distintas plataformas de e-learning. Así, el sistema operativo que se escoge también será un sistema operativo de software libre. En particular, Debian Lenny con gran disposición de paquetes, una abundante documentación en línea, estable y administrable. Características que le hacen adecuado para poder instalar LAMP y trabajar en el desarrollo y servicio de la aplicación informática JARO.

2.4 Metodología empleada

El objetivo principal de esta sección es analizar la metodología utilizada para el desarrollo del sistema web (JARO).

Para ello, se utilizarán varios diagramas en UML. UML es un lenguaje de modelado que proporciona un vocabulario y unas reglas para la representación conceptual y física de un sistema, este modelado nos va a permitir tener varias vistas (modelos) que nos ayudarán a comprender dicho sistema. En definitiva, UML cubre la especificación de todas las decisiones de análisis, diseño e implementación que deben realizarse al desarrollar y desplegar el sistema software que nos ocupa.

La metodología que se emplea en este trabajo sigue el ciclo de vida software en cascada con iteración.

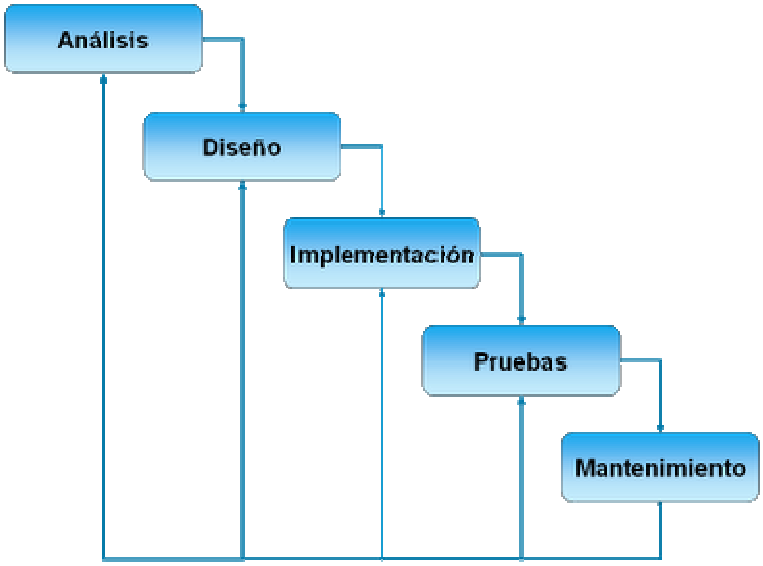


Figura 1 Metodología en cascada

2.4.1 Captura de requisitos y obtención de casos de uso

A continuación, se analiza el sistema utilizando los casos de uso que es la forma más simple para modelar los requisitos del sistema desde la perspectiva del usuario.

2.4.1.1. Requisitos funcionales

En este apartado se presentan los requisitos funcionales que deberán ser satisfechos por el sistema. Todas estas características son esenciales, es decir, no sería aceptable si el sistema desarrollado no satisface alguna de ellas.

Características generales:

- REQ01general: El alumno tendrá la opción de aprender conceptos.
- REQ02general: El alumno tendrá la opción de repasar conceptos.
- REQ03general: Las preguntas se generarán a partir del esquema conceptual previamente generado por el algoritmo.

- REQ04general: Las respuestas del alumno serán evaluadas, y en función de la evaluación, el valor de confianza del alumno para un determinado concepto se actualizará en la base de datos.
- Req05general: El alumno podrá elegir salir del módulo en cualquier momento guardándose los progresos del mismo de manera automática.
- REQ06general: Las peticiones del alumno podrán ser mediante interacción con la herramienta JARO en lenguaje natural o a través de menús.

2.4.1.2. Requisitos no funcionales

Los requisitos que se especifican a continuación están relacionados con la facilidad de uso, la fiabilidad, el rendimiento y el soporte, los cuales describen atributos del sistema o atributos del ambiente del sistema.

- REQ01NF: Nuestro sistema tendrá un interfaz amigable e intuitiva para el usuario.
- REQ02NF: El sistema debe responder en tiempo real a las peticiones del usuario. Esto se debe a que, aunque la velocidad no es un factor primordial para este proyecto, si se debe tener en cuenta que si la respuesta del sistema es muy lenta puede provocar el abandono por parte de algunos estudiantes.
- REQ03NF: El producto debe poderse utilizar desde cualquier ordenador conectado a Internet.

2.4.1.3. Identificación de actores

Se define como actor a cualquier agente externo que interactúe con la aplicación. Un actor será, por tanto, una entidad externa al sistema que inicia una interacción con el mismo. En el caso del módulo de estructuración se identifica a un único actor.

- Alumno: Persona que va a interactuar con la aplicación a partir de un modelo conceptual, realizando distintas opciones como aprender o repasar.

A continuación, se proporcionan los casos de uso que describirán una serie de secuencias de acciones que ejecuta el sistema produciendo un resultado de interés para un actor.

2.4.1.4.1. Estrada a la plataforma

- Caso de uso “Entrada Plataforma”

  Actores: Alumno.
  Descripción: Se permite el acceso al usuario.

ACCIÓN DEL ACTOR	RESPUESTA DEL SISTEMA
1. Este Caso de uso comienza cuando el alumno introduce en el navegador la URL de la aplicación.	2. El sistema muestra un saludo de bienvenida con el nombre del alumno
	3. El sistema muestra las opciones disponibles para el alumno

Tabla 5 Caso de uso “Entrada a la plataforma”

Caminos alternativos:

Evento 3: El usuario cierra el navegador.

2.4.1.4.2. Acciones de la plataforma

- Caso de uso “Aprender a través de lenguaje natural”

  Actores: Alumno.
  Descripción: El alumno introduce en el campo una frase que contenga la palabra aprender

ACCIÓN DEL ACTOR	RESPUESTA DEL SISTEMA
1. Este Caso de uso comienza cuando el alumno previamente ha entrado en la	2. El sistema comprueba que este usuario tiene conceptos que aprender

aplicación y decide escribir en el campo habilitado la palabra aprender	
	3. El sistema muestra el concepto con su definición y relaciones
4 El alumno asimila los conocimientos	

Tabla 6 Caso de uso “Aprender a través de lenguaje natural”

Caminos alternativos:

Evento 3: El usuario no tiene conceptos que aprender por lo que se muestra un mensaje diciendo que el alumno no tiene mas conceptos que aprender

- Caso de uso “Aprender a través de hipervínculo”

Actores: Alumno.

Descripción: El alumno introduce pincha sobre el hipervínculo aprender

ACCIÓN DEL ACTOR	RESPUESTA DEL SISTEMA
1. Este caso de uso comienza cuando el alumno selecciona el hipervínculo aprender	2. El sistema comprueba que este usuario tiene conceptos que aprender
	3. El sistema muestra el concepto con su definición y relaciones
4 El alumno asimila los conocimientos	

Tabla 7 Caso de uso “Aprender a través de hipervínculo”

Caminos alternativos:

Evento 3: El usuario no tiene conceptos que aprender por lo que se muestra un mensaje diciendo que el alumno no tiene más conceptos que aprender.

- Caso de uso “Repasar a través de lenguaje natural”

  Actores: Alumno.

  Descripción: El alumno introduce en el campo una frase que contenga la palabra aprender

ACCIÓN DEL ACTOR	RESPUESTA DEL SISTEMA
1. Este caso de uso comienza cuando el alumno previamente ha entrado en la aplicación y decide escribir en el campo habilitado la palabra aprender	2. El sistema comprueba que este usuario tiene conceptos que repasar
	3. El sistema muestra una pregunta de repaso
4 El alumno responde la pregunta	El sistema evalúa la certeza de la respuesta
	El sistema muestra la evolución del alumno

Tabla 8 Caso de uso “Repasar a través de lenguaje natural”

Caminos alternativos:

Evento 3: El usuario no tiene conceptos que repasar por lo que se muestra un mensaje diciendo que el alumno no tiene más conceptos que repasar.

- Caso de uso “Repasar a través de hipervínculo”

  Actores: Alumno.

ACCIÓN DEL ACTOR	RESPUESTA DEL SISTEMA
1. Este caso de uso comienza cuando el alumno previamente ha entrado en la aplicación y decide pinchar sobre el hipervínculo repasar	2. El sistema comprueba que este usuario tiene conceptos que repasar

	3. El sistema muestra una pregunta de repaso
4 El alumno responde la pregunta	El sistema evalúa la certeza de la respuesta
	El sistema muestra la evolución del alumno

Tabla 9 Caso de uso “Repasar a través de hipervínculo”

Caminos alternativos:

Evento 3: El usuario no tiene conceptos que repasar por lo que se muestra un mensaje diciendo que el alumno no tiene más conceptos que repasar.

2.4.2 Diagrama de Casos de uso

2.4.2.1 Diagrama de Casos de uso “Aprender”

En este diagrama se muestran las diferentes interacciones que puede tener el usuario de la plataforma JARO con ella:

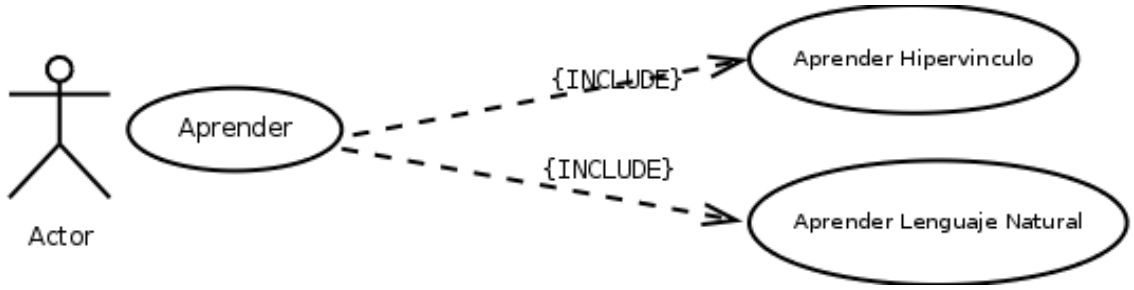


Figura 2 diagrama caso de uso aprender

2.4.2.2 Diagrama de casos de Uso Repasar

En este diagrama se muestran, los distintos tipos de repaso que puede realizar el usuario de la aplicación:

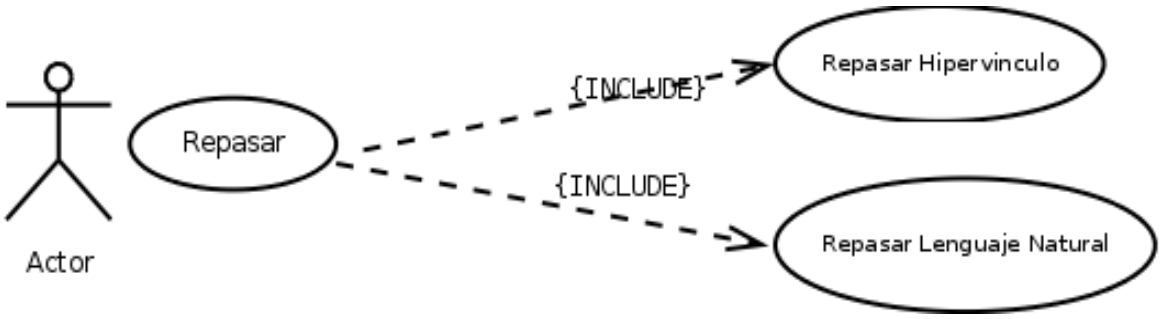


Figura 3 diagrama caso de uso repasar

2.4.2.3 Diagrama de casos de uso general de la plataforma

En este diagrama se muestran todas las opciones que puede tener un usuario dentro de la plataforma:

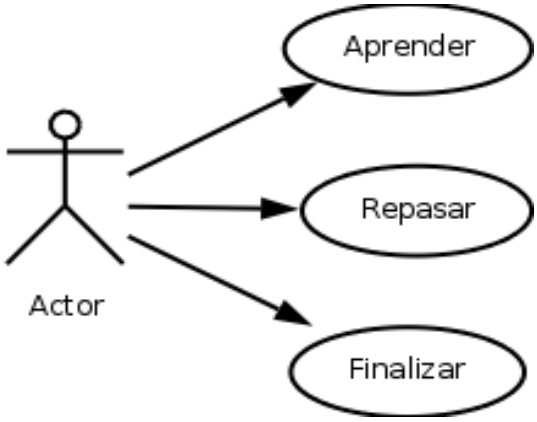


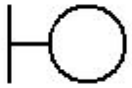
Figura 4 diagrama caso de uso de la plataforma general

2.4.3 Análisis

2.4.3.1. Descripción de las clases de análisis

- Clase límite o interfaz: modela la interacción entre el sistema y los actores. Representa la

interfaz del sistema con poco detalle, sólo a nivel conceptual. Describe la información presentada al actor y las peticiones que hace el actor al sistema.



Interfaz

- Clase control: representa la coordinación entre objetos, encapsulan el flujo de control de un determinado caso de uso. Ni interaccionan con el usuario ni se ocupan de almacenar información, sólo se encarga de la lógica del sistema.

Gestor Aplicación

- Clase Entidad: se utiliza para mantener la información permanente relacionada con un caso de uso, muestran una estructura de datos lógica y contribuyen a entender que información manipular.

Base de datos

En los siguientes diagramas se mostrarán las relaciones que existen en las clases definidas anteriormente.

2.4.3.2 Diagrama de interacción para el caso de uso Aprender lenguaje natural

En este diagrama se muestra como es la interacción usuario-maquina para una petición de aprendizaje mediante lenguaje natural.

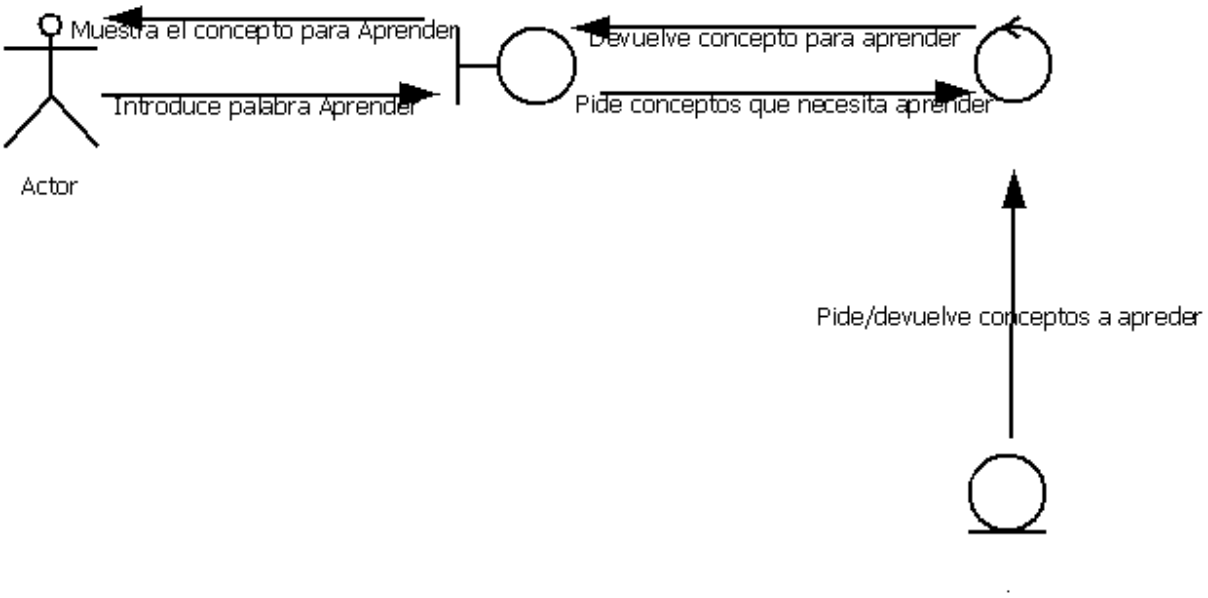


Figura 5 Diagrama interacción casos de uso Aprender lenguaje natural

2.4.3.3 Diagrama interacción casos de uso Aprender hipervínculo

En este diagrama se muestra como es la interacción usuario-maquina para una petición de aprendizaje mediante un hipervínculo:

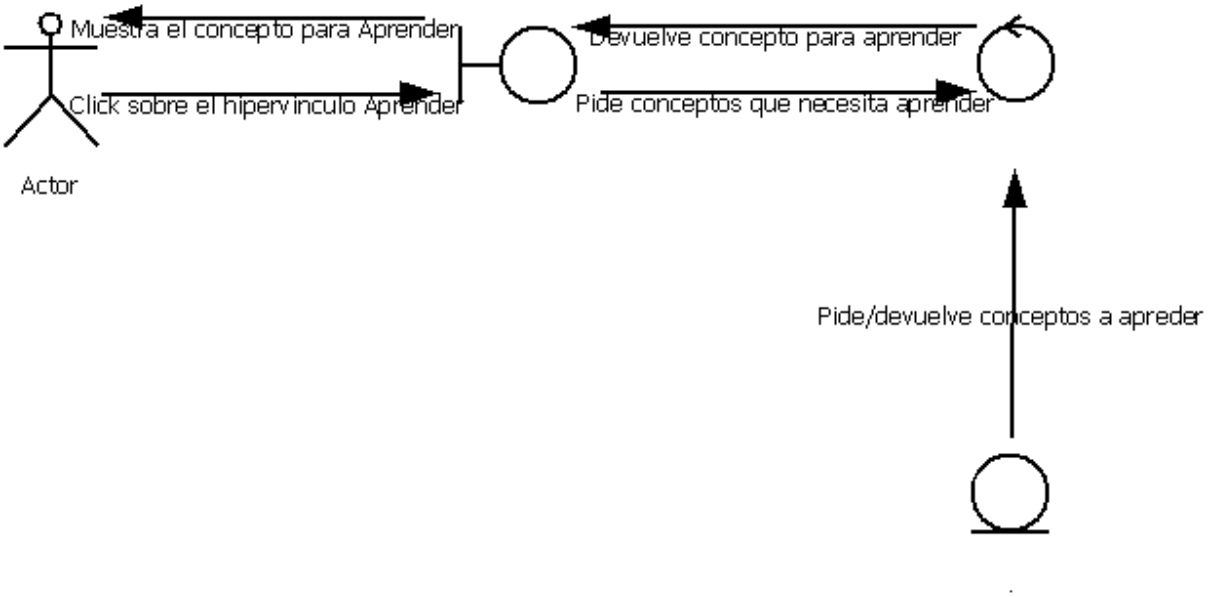


Figura 6 Diagrama interacción casos de uso Aprender hipervínculo

2.4.3.3 Diagrama interacción casos de uso repasar lenguaje natural

En este diagrama se muestra como es la interacción usuario-maquina para una petición de repasar aprendizaje mediante lenguaje natural

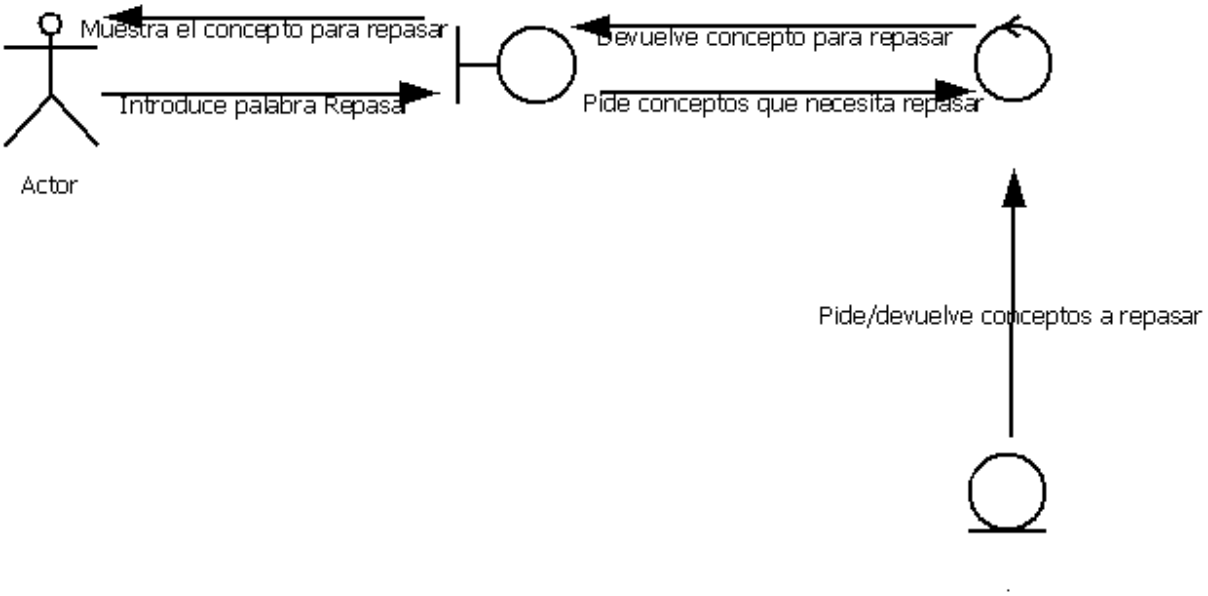


Figura 7 Diagrama interacción casos de uso repasar lenguaje natural

2.4.3.4 Diagrama interacción casos de uso Repasar hipervínculo

En este diagrama se muestra como es la interacción usuario-máquina para una petición de repasar mediante un hipervínculo:

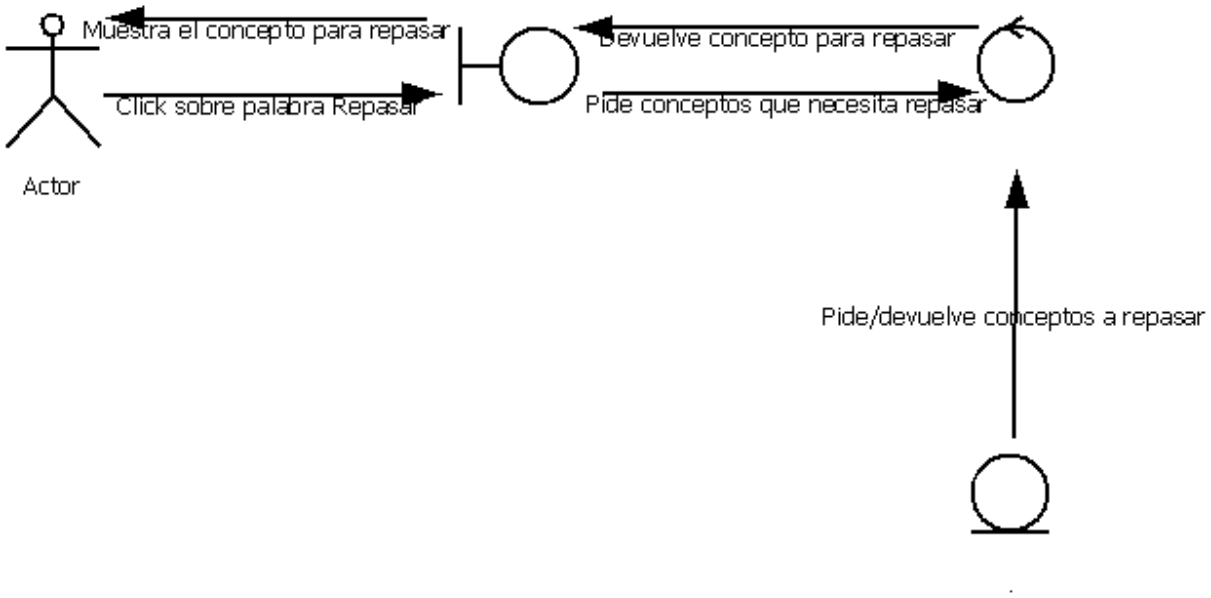


Figura 8 Diagrama interacción casos de uso repasar hipervínculo

3 Descripción informática

3.1 Diseño

En este capitulo se aborda el diseño de la aplicación. Esto se realiza siguiendo un enfoque descendente desde la arquitectura de alto nivel, a un nivel más bajo, según se vaya avanzando hasta la descripción de las clases y módulos, describiendo también la estructura del archivo XML con el modelo conceptual y el esquema de diálogo generado para interactuar con el estudiante usando JARO.

3.2 Arquitectura alto nivel

La arquitectura elegida es la tradicional en 3 capas: (1) la capa del cliente, (2) la capa de aplicación y (3) la capa de datos, como se describe en la Figura 9.

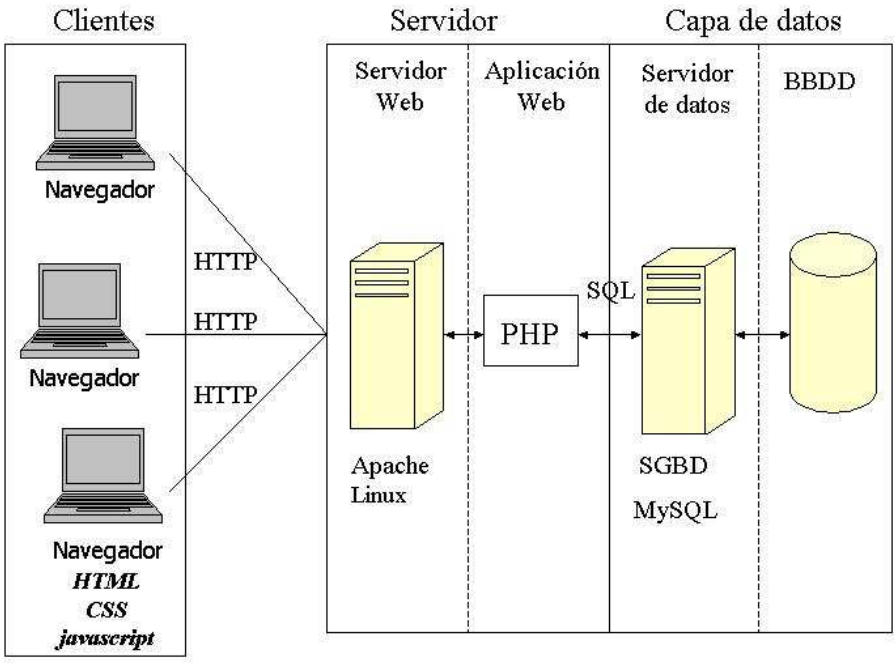


Figura 9 Arquitectura del sistema

- **Capa del cliente**, está el navegador del cliente, que envía y acepta peticiones HTTP a la capa de aplicación, y que además debe interpretar y ejecutar comandos JavaScript y traducir hojas de estilos en cascada (CSS). Las páginas y formularios se mostrarán en esta capa.
- **Capa de aplicación**, consta de la aplicación de servidor Web en este caso un apache y la aplicación de PHP, ambos corriendo bajo un sistema operativo Debian (Linux). El servidor maneja las peticiones HTTP solicitadas por el cliente. Cuando la página solicitada es un script (PHP), el servidor Web envía una petición a la aplicación de PHP. Este genera al vuelo una página dinámica con formato HTML, posiblemente accediendo a la capa de datos mediante el lenguaje SQL, que envía al servidor y, a su vez, éste reenvía dicha página al cliente.
- **Capa de datos**, consiste en un sistema de administración de bases de datos relacionales (RDBMS), que es el responsable de almacenar la información. Este sistema contiene la información en tablas, las cuales están relacionadas mediante identificadores únicos. RDBMS es la opción escogida para la mayoría de de sistemas basados en la Web, ya que las relaciones entre las tablas están definidas por el desarrollador, y no por restricciones propias de la base de datos, como en bases de datos basadas en ficheros. El RDBMS escogido en este proyecto es MySQL.

El propósito del diseño del sistema es entender en profundidad los requisitos no funcionales y restricciones relacionadas con el lenguaje de programación que vamos a utilizar y descomponer el trabajo de implementación en partes manejables.

3.3 Diagrama entidad relación

En el siguiente dibujo mostramos el diagrama E/R para la base de datos, por simplicidad en el dibujo, solo se muestran sin los atributos, que se explicarán conjuntamente con la descripción de las tablas.

Requisitos BBDD

- Un usuario sabe uno o más conceptos.
- Un concepto puede ser conocido por uno o más usuarios.
- Un concepto puede ser conocido por uno o más usuarios.
- Un concepto puede estar relacionado con uno o más conceptos.

Estos requisitos conforman el diagrama Entidad-Relación que se muestra en la Figura 10.

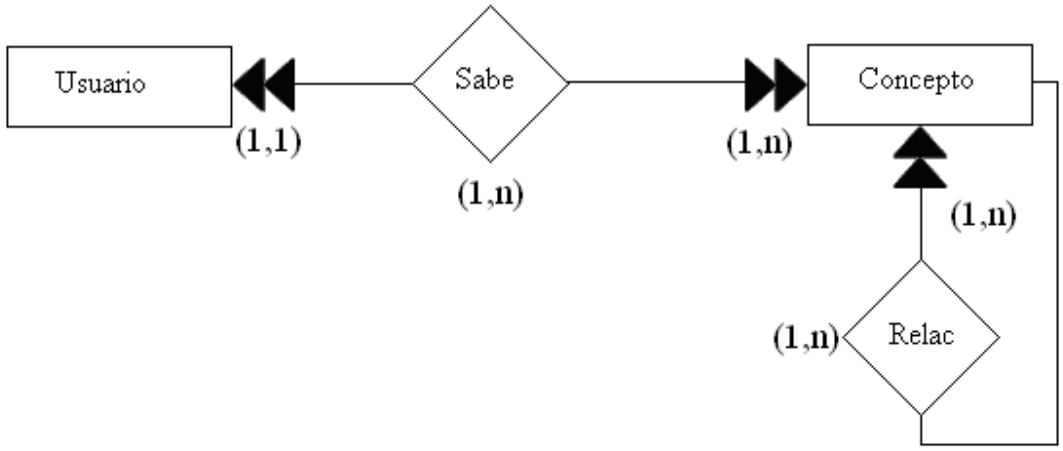


Figura 10. Modelo E/R

La tabla **usuario** está compuesta de los campos:

- Nombre, Apellidos, NIF, sexo y edad
- La clave primaria es el NIF

Estos campos serán los datos del usuario que serán los que nos sirvan para identificarlo.

La tabla **concepto** está compuesta de los campos:

- Nombre, Definición, Img, URL, Extra e Id.
- La clave primaria es el ID

Estos campos, serán los que identificarán un concepto, como son su nombre, su definición, una imagen asociada al mismo y una URL de referencia

La tabla **sabe** está compuesta de los campos:

- Fecha, Hora, CV, NIF e ID.
- Las claves primarias son ID y NIF
- CV, se refiere al valor de confianza de un determinado concepto, que se verá modificado en función de la evolución del usuario

La tabla **Relac** está compuesta de los campos:

- Id1, Id2, IdTipo
- Las claves primarias son Id1 e Id2

Estructura de tablas dentro de Mysql

A continuación, se muestra la distribución de las tablas con los atributos y la identificación de las claves primarias subrayadas:
USUARIO *(nif, nombre, apellidos, sexo, edad)*
SABE *(nif, ID, CV, fecha, hora)*
CONCEPTO *(id, nombre, definición, img, url, extra)*
RELACION *(id1, id2, idtipo)*
RELACIONES *(id_tipo, tipo, extra)*

La Figura 11 muestra la estructura dentro de MySQL.

	Tabla	Acción	Registros	Tipo	Cotejamiento	Tamaño
☐	concepto		12	MyISAM	latin1_swedish_ci	3.3 KB
☐	relaciona		13	MyISAM	latin1_swedish_ci	2.3 KB
☐	sabe		12	MyISAM	latin1_swedish_ci	2.4 KB
☐	usuario		1	MyISAM	latin1_swedish_ci	2.0 KB
	4 tabla(s)	Número de filas	38	MyISAM	latin1_swedish_ci	10.0 KB

Figura 11 Estructura tablas BBDD

Los tipos de datos varchar se corresponden con cualquier carácter de la tabla ASCII y con una longitud de 50 caracteres como máximo. El campo Date, guarda una estructura de fecha en estilo DD-MM-AAAA, siendo obligatorio este formato para la introducción de datos.

concepto

Campo	Tipo	Nulo	Predeterminado
id_con	varchar(50)	No	
nombre	varchar(50)	No	
definicion	varchar(50)	No	
img	varchar(50)	No	
url	varchar(50)	No	
extra	varchar(50)	No	

Tabla 10 concepto

relaciona

Campo	Tipo	Nulo	Predeterminado
id1	varchar(50)	No	
id2	varchar(50)	No	
id_tipo	varchar(50)	No	

sabe

Campo	Tipo	Nulo	Predeterminado
nif	varchar(50)	No	
id_con	varchar(50)	No	
cv	varchar(50)	No	
fecha	date	No	
hora	varchar(50)	No	

usuario

Campo	Tipo	Nulo	Predeterminado
nif	varchar(50)	No	
nombre	varchar(50)	No	
apellidos	varchar(50)	No	
sexo	varchar(50)	No	

Tablas 11 Tablas relaciona, sabe, usuario

Ejemplos reales de datos en las tablas de la base de datos

En este apartado, se muestran datos de ejemplo para cada una de las tablas.

- Ejemplo tabla usuario

		55556677V	Juan	Valdes	Hombre	48

Tabla 12 Ejemplo tabla usuario

- Ejemplo tabla concepto

			2	proceso padre	proceso que inicio a un determinado proceso	esto es una imagen	www.procesopadre.org	Podria ir vacio
			3	proceso hijo	proceso creado por un determinado proceso	esto es una imagen	www.procesohijo.org	Podria ir vacio

Tabla 13 Ejemplo tabla concepto

- Ejemplo tabla relaciona

			3	1	es un			55556677V	1	0.1	0000-00-00	=14:39
			1	4	tiene			55556677V	2	0.7	0000-00-00	=12:00

Tabla 14 Ejemplo tabla relaciona

- Ejemplo tabla sabe

		55556677V	1	0.1	0000-00-00	=14:39
		55556677V	2	0.7	0000-00-00	=12:00

Tabla 115 Ejemplo tabla sabe

3.4 Archivos XML

Los archivos XML y las DTD que se han utilizado han sido los siguientes.

entrada.dtd: Con el contenido de este archivo se comprueba que la estructura que tiene el fichero XML de entrada para el algoritmo sigue las pautas dadas en la especificación del problema.

modelo-conceptual.dtd: Con el contenido de este archivo se comprueba que el fichero XML que genera el algoritmo es valido según las especificaciones del problema.

A continuación, se muestran los archivos mencionados y un ejemplo de cada uno.

Entrada.dtd

En este fichero, se muestran la especificación de los campos que debe tener cada fichero XML.

```
<!ELEMENT modelo_conceptual (conceptos, relaciones)>
<!ATTLIST modelo_conceptual
                          nif       CDATA #REQUIRED
                          nombre    CDATA #REQUIRED
                          apellidos CDATA #REQUIRED
                          sexo      (H|M) #IMPLIED
                          edad      CDATA #IMPLIED
                          fecha     CDATA #REQUIRED
                          hora      CDATA #REQUIRED >
<!ELEMENT conceptos (concepto)+>
<!ELEMENT concepto  (nombre, definicion, img, url, extra)>
<!ATTLIST concepto
          idc       ID    #REQUIRED
          cv        CDATA #REQUIRED >
<!ELEMENT nombre (#PCDATA) >
<!ELEMENT definicion (#PCDATA) >
<!ELEMENT img (#PCDATA) >
<!ELEMENT url (#PCDATA) >
<!ELEMENT extra (#PCDATA) >
<!ELEMENT relaciones (relacion)+>
<!ELEMENT relacion (#PCDATA)>
<!ATTLIST relacion
          origen    IDREF #REQUIRED
          destino   IDREF #REQUIRED >
```

Ejemplo de fichero XML conforme a entrada.dtd

Este fichero muestra un caso de ejemplo de fichero XML de un estudiante conforme a entrada.dtd

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE modelo_conceptual SYSTEM "entrada.dtd">
<modelo_conceptual>
<datos>
 <nif>55556677V</nif>
 <nombre>Juan</nombre>
 <apellidos>Valdés </apellidos>
 <sexo>H</sexo>
 <edad>48</edad>
 <fecha>15/10/2009</fecha>
 <hora>=12:00</hora>
</datos>
      <conceptos>
            <concepto>
                  <id_con>1 </id_con>
                          <cv>0.3</cv>
                  <nombre>proceso</nombre>
                  <definicion>Programa en ejecución.</definicion>
                  <img>esto es una imagen</img>
                  <url>www.aprende.org</url>
                  <extra> Podria ir vacio</extra>
            </concepto>
            <concepto>
                  <id_con>2 </id_con>
                          <cv>0</cv>
                  <nombre>proceso padre</nombre>
                  <definicion>Proceso que ejecuta unusuario.</definicion>
                  <img>esto es una imagen</img>
                  <url>www.aprende.org</url>
                  <extra> Podria ir vacio</extra>
            </concepto>

      </conceptos>

      <relaciones>
            <relacion>
               <origen>2</origen>
               <destino>1</destino>
                     <predicado>es un</predicado>
                  </relacion>
      </relaciones>

</modelo_conceptual>
```

Modelo-conceptual.dtd

Este fichero contiene la estructura que deberá seguir el fichero de salida con el esquema del diálogo generado.

```
<!-- DTD de dialogo.xml (salida) -->

<!ELEMENT dialogo (saludo, segmento+) >
<!ELEMENT saludo (#PCDATA) >
<!ELEMENT segmento (contenido, opcion+)>
<!ATTLIST segmento ids ID #REQUIRED>
<!ELEMENT contenido (#PCDATA)>
<!ELEMENT opcion (accion | aprendizaje | pregunta)+ >
<!ATTLIST opcion disparador CDATA #REQUIRED>
<!ELEMENT accion (#PCDATA)>
<!ATTLIST accion tipo CDATA #REQUIRED>
<!ELEMENT aprendizaje (inicio, definicion, img, url, extra)>
<!ELEMENT inicio (#PCDATA)>
<!ELEMENT definicion (#PCDATA)>
<!ELEMENT img (#PCDATA)>
<!ELEMENT url (#PCDATA)>
<!ELEMENT extra (#PCDATA)>
<!ELEMENT pregunta (enunciado, opcion+)>
<!ATTLIST pregunta idp ID #REQUIRED>
<!ELEMENT enunciado (#PCDATA)>
```

Ejemplo de fichero XML conforme a modelo-conceptual.dtd

En este ejemplo se muestra un ejemplo en XML, conforme a modelo-conceptual.dtd, y base que utilizará JARO para su funcionamiento y la generación de esquemas de diálogo.

Para aclarar la parte de las etiquetas XML decir que la “opción disparador” como etiqueta XML, indica el siguiente salto del esquema de diálogo en función de las respuesta a una determinada pregunta sobre un concepto.

```
<Dialogo>
<Saludo>Bienvenido Usuario</Saludo>

<Segmento>
Segmento de inicio
<identificacion>1</identificacion>

<Contenido>
Describimos la accion que vamos a realizar
```

```
<Opcion_disparador>
Disparador de aprender
<tipo>Aprender</tipo>
<accion_tipo>Ir_seg</accion_tipo>
<accion>2</accion>
</Opcion_disparador>

<Opcion_disparador>
Disparador de repasar
<tipo>Repasar</tipo>
<accion_tipo>Ir_seg</accion_tipo>
<accion>3</accion>
</Opcion_disparador>

<Opcion_disparador>
Disparador de Terminar
<tipo>Finalizar</tipo>
<accion_tipo>Ir_fin</accion_tipo>
<accion>Hasta luego</accion>
</Opcion_disparador>

<Opcion_disparador>
Disparador por defecto
<tipo>Default</tipo>
<accion_tipo>Ir_seg</accion_tipo>
<accion>1</accion>
</Opcion_disparador>
</Contenido>
</Segmento>

<Segmento>
Segmento de aprender conceptos
<identificacion>2</identificacion>

<Contenido>
Muy bien aprendamos nuevos conceptos entonces

<Opcion_disparador_seg_2>
El concepto 1  no tiene que ser repasado
<concepto>1 </concepto>
<tipo>Repasar</tipo>
<accion_tipo>ir_seg</accion_tipo>
<accion>3</accion>
</Opcion_disparador_seg_2>

<Opcion_disparador_seg_2>
El concepto 2  no tiene que ser repasado
<concepto>2 </concepto>
<tipo>Repasar</tipo>
<accion_tipo>ir_seg</accion_tipo>
<accion>3</accion>
</Opcion_disparador_seg_2>
```

```
<Opcion_disparador_seg_2>
El concepto 3 tiene que ser aprendido
<concepto>3</concepto>
<tipo>Aprender</tipo>

<aprendizaje>
A continuación mostramos la secuencia de aprendizaje
<inicio>Veamos que es un proceso residente</inicio>
<definicion>proceso perdido en memoria</definicion>
<img>no tiene</img>
<url>no tiene</url>
<extra>no tiene</extra>
</aprendizaje>
</Opcion_disparador_seg_2>
</Contenido>
</Segmento>

<Segmento>
Segmento de repasar conceptors
<identificacion>3</identificacion>

<Contenido>
Muy bien repasemos conceptos entonces

<Repaso>
Se han repasado menos de dos preguntas

<pregunta>
1
<enuciado>¿Qué es un proceso?</enuciado>

<opcion_disparador>
Respuesta Correcta
<accion_tipo>Incrementar_CV</accion_tipo>
<valor>0.3</valor>
</opcion_disparador>

<opcion_disparador>
Respuesta incorrecta
<accion_tipo>Resetear_CV</accion_tipo>
<valor>0.0</valor>

<aprendizaje>
A continuación mostramos el proceso de aprendizaje
<inicio>¿Que es un proceso?</inicio>
<definicion>Programa en ejecución.</definicion>
<img>esto es una imagen</img>
<url>www.aprende.org</url>
<extra> Podria ir vacion</extra>
</aprendizaje>
</opcion_disparador>
</pregunta>
```

```
<pregunta>
2
<enuciado>¿Es un proceso un Programa en ejecución.?</enuciado>

<opcion_disparador>
Respuesta Correcta
<accion_tipo>Incrementar_CV</accion_tipo>
<valor>0.3</valor>
</opcion_disparador>

<opcion_disparador>
Respuesta incorrecta
<accion_tipo>Resetear_CV</accion_tipo>
<valor>0.0</valor>

<aprendizaje>
A continuación mostramos el proceso de aprendizaje
<inicio>¿Que es un proceso?</inicio>
<definicion>Programa en ejecución.</definicion>
<img>esto es una imagen</img>
<url>www.aprende.org</url>
<extra> Podria ir vacion</extra>
</aprendizaje>
</opcion_disparador>
</pregunta>

<pregunta>
3
<enuciado>¿Un proceso es un proceso padre?</enuciado>

<opcion_disparador>
Respuesta Correcta
<accion_tipo>Incrementar_CV</accion_tipo>
<valor>0.3</valor>
</opcion_disparador>

<opcion_disparador>
Respuesta incorrecta
<accion_tipo>Resetear_CV</accion_tipo>
<valor>0.0</valor>

<aprendizaje>
A continuación mostramos la secuencia de aprendizaje
<inicio>Veamos que es un proceso</inicio>
<definicion>Programa en ejecución.</definicion>
<img>esto es una imagen</img>
<url>www.aprende.org</url>
<extra>www.aprende.org</extra>
</aprendizaje>
</opcion_disparador>
</pregunta>
```

```
<pregunta>
1
<enuciado>¿Qué es un proceso padre?</enuciado>

<opcion_disparador>
Respuesta Correcta
<accion_tipo>Incrementar_CV</accion_tipo>
<valor>0.3</valor>
</opcion_disparador>

<opcion_disparador>
Respuesta incorrecta
<accion_tipo>Resetear_CV</accion_tipo>
<valor>0.0</valor>

<aprendizaje>
A continuación mostramos el proceso de aprendizaje
<inicio>¿Que es un proceso padre?</inicio>
<definicion>Proceso que ejecuta un usuario.</definicion>
<img>esto es una imagen</img>
<url>www.aprende.org</url>
<extra> Podria ir vacion</extra>
</aprendizaje>
</opcion_disparador>
</pregunta>

<pregunta>
2

<enuciado>
¿Es un proceso padre un Proceso que ejecuta un usuario.?
</enuciado>

<opcion_disparador>
Respuesta Correcta
<accion_tipo>Incrementar_CV</accion_tipo>
<valor>0.3</valor>
</opcion_disparador>

<opcion_disparador>
Respuesta incorrecta
<accion_tipo>Resetear_CV</accion_tipo>
<valor>0.0</valor>

<aprendizaje>
A continuación mostramos el proceso de aprendizaje
<inicio>¿Que es un proceso padre?</inicio>
<definicion>Proceso que ejecuta un usuario.</definicion>
<img>esto es una imagen</img>
<url>www.aprende.org</url>
<extra> Podria ir vacion</extra>
</aprendizaje>
```

```
</opcion_disparador>
</pregunta>
</Repaso>
</Contenido>
</Segmento>
</Dialogo>
```

3.5 Algoritmo

En este apartado, se realizará una descripción del algoritmo que será el eje fundamental para el desarrollo de la herramienta JARO y para entender su funcionamiento.

El algoritmo proporcionado por Diana Pérez tiene las siguientes especificaciones:

PRE: A partir de un fichero de entrada XML, válido para la dtd modelo_conceptual.dtd
POST: Almacena información en la base de datos, y genera un modelo conceptual exportable en XML

A continuación se muestra el algoritmo:

1 Generar el primer bloque de saludo con el nombre de la persona cuyo modelo conceptual se recoge en la entrada.

2 Generar el segmento 1 del diálogo con la pregunta inicial: ¿quieres aprender nuevos conceptos, repasar los ya estudiados o terminar la sesión?

3 Generar una opción dentro del segmento 1 para cada posible respuesta a la pregunta id = 1:

3.1 Primera opción: aprender

- Generar salto al segmento 2

3.2 Segunda opción: repasar

- Generar salto al segmento 3

3.3 Tercera opción: terminar

- Salir del programa con un saludo

3.4 Cualquier otra opción, volver a realizar la pregunta id=1

4 Generar el segmento 2 (o bloque de aprendizaje):

4.1 Generar el contenido “Muy bien, aprendamos nuevos conceptos entonces”

4.2 Buscar en la BD los conceptos que tienen CV = 0

4.3 Si no hay conceptos con CV= 0

- Generar el contenido “Vaya, parece que ya has visto todos los conceptos”
- Generar salto al segmento 3

4.4 Si hay conceptos con CV = 0

- Examinar sus relaciones
- Generar el módulo de aprendizaje del concepto jerárquicamente inferior
- Incrementar su CV + 0.1
- Generar salto al segmento 1

5 Generar el segmento 3 (o bloque de repaso):

5.1 Generar el contenido “Muy bien, repasemos entonces.”

5.2 Por cada concepto con CV > 0 y relación entre conceptos generar una pregunta:

- Por cada pregunta, generar dos alternativas:
  * Respuesta correcta
    ** Incrementar CV del concepto(s) + 0.3
    ** Generar “Muy bien”
  * Respuesta incorrecta
    ** Resetear CV = 0
    ** Generar modulo de aprendizaje del concepto(s)

5.3 Por cada bloque de 3 preguntas:

– Generar salto al segmento 1

En esta parte también hay que reseñar la creación de patrones, para luego poder realizar las preguntas correspondientes a cada concepto ya en la aplicación JARO.

Los patrones utilizados han sido los siguientes:

X es un Y: siendo X la definición de un concepto e Y el nombre del concepto, también los hemos usado al revés, para obligar al alumno a ver las cosas de una manera con una lógica mas compleja

X es un Y
X es un Z ⟹ Y=Z

En esta propiedad usamos la propiedad conmutativa, siendo X dos conceptos que pueden tener definiciones equivalentes que el alumno pueda asociar ambas.

3.6 Descripción clases y módulos

En este apartado se detallan los módulos y las librerías que se han usado para la realización del proyecto.

Las librerías usadas son:

- my_sql_functions: Con este módulo de PHP, gestionamos todas las funciones referentes a la bases de datos, conexión, lectura y escritura de la misma

- DOM: Con esta librería lo que conseguimos es realizar todas las llamadas a los archivos XML, leemos, escribimos y modificamos los archivos XML.

La Figura 11 muestra la división de la pantalla de JARO en varios apartados. Esta división se realiza para mostrar el diseño de la interfaz del sistema de repaso web.

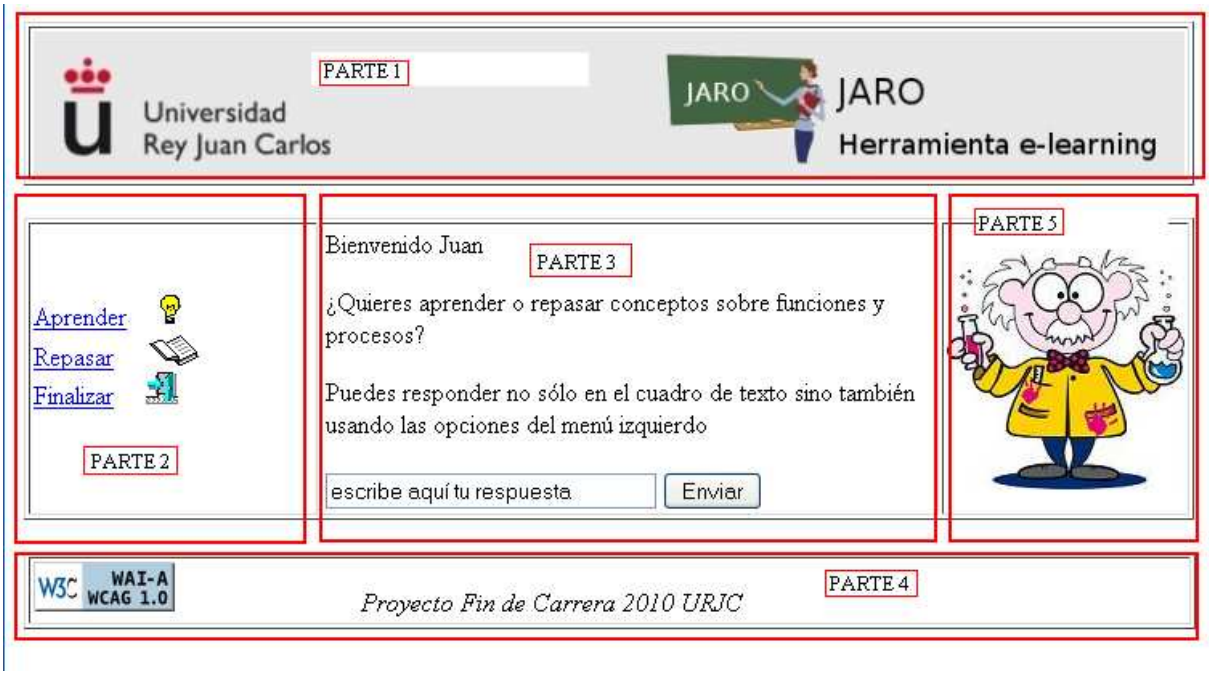


Figura 12 JARO

Parte 1: El encabezado de la página web.

Parte 2: Enlaces a las distintas opciones del menú de JARO.

Parte 3: Interacción entre el alumno y JARO, donde se muestran los esquemas pregunta-respuesta.

Parte 4: Pie de página de la aplicación.

Parte 5: Imagen de JARO. Su objetivo es que ayude a la herramienta a crear un ambiente más amigable.

La Figura 13 muestra un diagrama de estados para ilustrar las situaciones que se pueden dar dentro de JARO para guiar el diálogo con el estudiante.

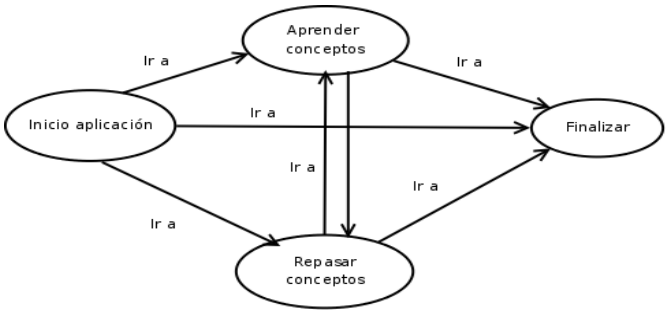


Figura 13 Diagrama de estados de JARO

Siempre que se produce una transición entre estados, todos los progresos que se producen en la herramienta quedan registrados en la base de datos

A continuación pasamos a describir que se hace en cada una de las páginas que componen JARO.

Chat1_1.php

Se da la bienvenida al usuario, y se le muestra las opciones que tiene disponibles. Además, se le muestra un cuadro para poder introducir texto, donde introducirán en lenguaje natural la opción a realizar.

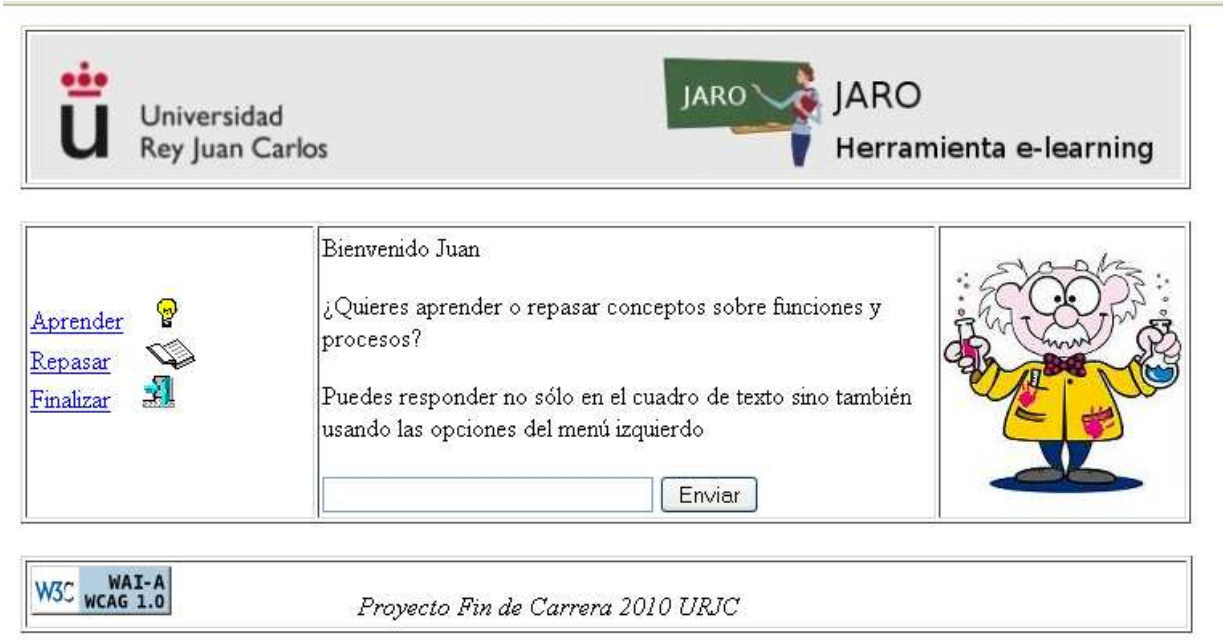


Figura 14. Captura Chat1_1.php

Repasar1_1.php

Este parte del programa se encarga de una vez que se va a repasar un concepto mostrar la pregunta de repaso que se ha creado a través de los patrones de lenguaje natural. Esto solo se hace en caso que el concepto a repasar no genere ninguna pregunta que necesite un patrón.

Figura 15 Captura repasar1_1.php

Repasar2.1.php

En esta parte del programa, se muestra una autoevaluación con valores de 1 a 10 para el alumno en función de cómo de acertada crea que ha sido su respuesta se evalúe. La Figura 16 muestra un ejemplo de pantallazo de JARO para esta funcionalidad.

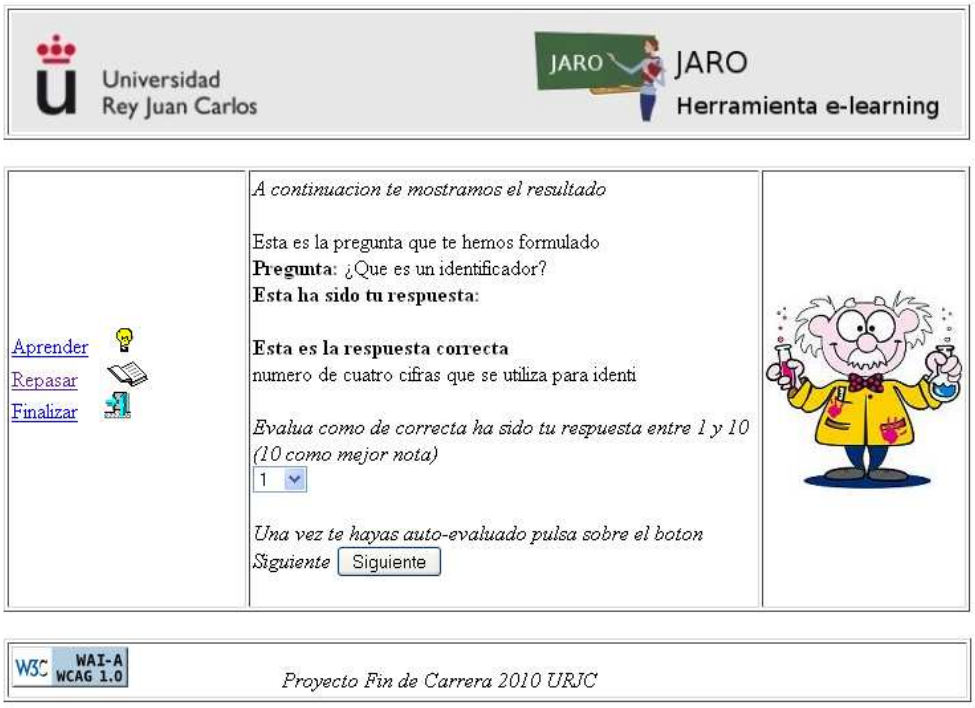


Figura 16 Captura repasar2_1.php

Repasar3_1.php

En esta parte, se pide que introduzca de nuevo la opción a realizar, una vez evaluado el concepto que ha repasado. La Figura 17 muestra un ejemplo de pantallazo de JARO.

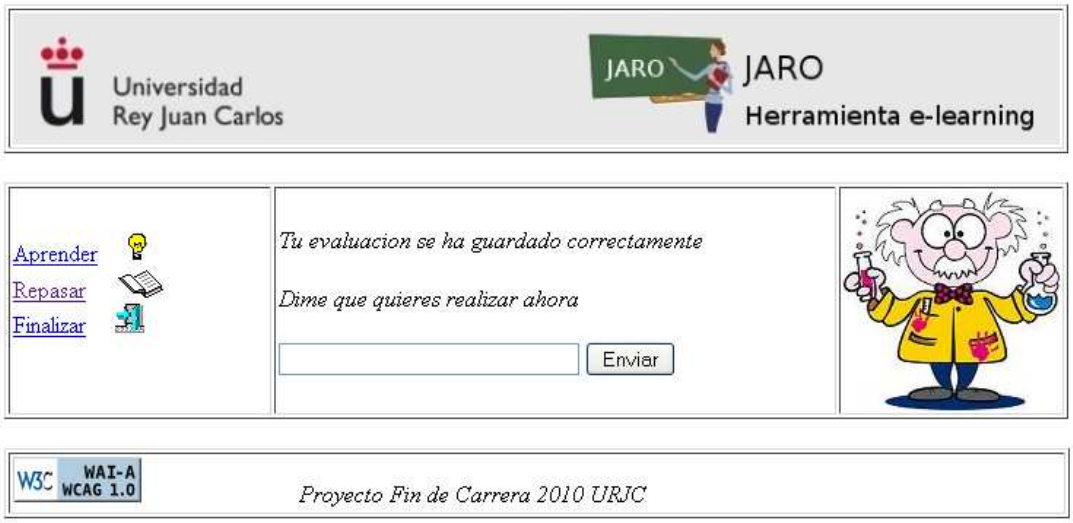


Figura 17 Captura repasar3_1.php

Aprender1_1.php

En esta parte del programa, se muestran el concepto junto con su definición, y relaciones importantes. La Figura 18 muestra un ejemplo de esta funcionalidad en JARO.

Figura 18 Captura aprender1_1.php

Chat2.php

Este módulo se encarga de reconocer que la frase introducida por el estudiante en JARO coincide con una de las opciones esperadas en el diálogo.

Como se puede observar en el diagrama de estados de la Figura 13, el estudiante puede solicitar “aprender” un nuevo concepto, “repasar” un concepto previo o bien abandonar la aplicación. Si el estudiante no introduce ninguna de estas fórmulas, entonces este módulo no reconoce la frase introducida y vuelve a pedir al estudiante que escoja si quiere aprender o repasar un concepto, o bien salir.

4 Pruebas y resultados

Las pruebas se han realizado en dos fases. En primer lugar se ha validado que el programa funciona correctamente y en segundo lugar se ha verificado que el programa cumple los requisitos que previamente se definieron en el proyecto.

La estrategia de pruebas ha sido la siguiente:

1. Pruebas unitarias
2. Pruebas de integración
3. Pruebas de validación
4. Pruebas de aceptación
5. Verificación de requisitos

La estructura de las pruebas realizadas siguen un enfoque ascendente desde las pruebas unitarias de cada módulo del sistema hasta valorar si la aplicación cumple con los requisitos no funcionales, tales como que sea manejable y fácil de usar, y la comprobación del nivel de satisfacción alcanzado por usuarios con y sin conocimientos informáticos.

4.1 Pruebas unitarias

En esta parte de las pruebas nos centraremos en el correcto funcionamiento del algoritmo y de los módulos que implementan JARO. Para ello, se comprobará que a partir del XML se generan esquemas de diálogo adecuados (pruebas de caja blanca) y se comprobará si los resultados de cada módulo coinciden con los valores esperados (pruebas de caja negra).

4.1.1 Caja blanca

La prueba de caja blanca se realizará sobre el algoritmo generador de los esquemas de diálogo a partir de los archivos XML con los modelos conceptuales.

En primer lugar sobre el algoritmo dado distribuiremos cada uno de los bloques.

Bloque 1

1 Generar el primer bloque de saludo con el nombre de la persona cuyo modelo conceptual se recoge en la entrada.

Bloque 2

2 Generar el segmento 1 del diálogo con la pregunta inicial: ¿quieres aprender nuevos conceptos, repasar los ya estudiados o terminar la sesión?

Bloque 3

3 Generar una opción dentro del segmento 1 para cada posible respuesta a la pregunta id = 1:

3.1 Primera opción: aprender

- Generar salto al segmento 2

3.2 Segunda opción: repasar

- Generar salto al segmento 3

3.3 Tercera opción: terminar

- Salir del programa con un saludo

3.4 Cualquier otra opción, volver a realizar la pregunta id=1

Bloque 4

4 Generar el segmento 2 (o bloque de aprendizaje):

4.1 Generar el contenido “Muy bien, aprendamos nuevos conceptos entonces”

4.2 Buscar en la BD los conceptos que tienen CV = 0

Bloque 5

4.3 Si no hay conceptos con CV= 0

- Generar el contenido “Vaya, parece que ya has visto todos los conceptos”
- Generar salto al segmento 3

4.4 Si hay conceptos con CV = 0

Bloque 6

- Examinar sus relaciones
- Generar el módulo de aprendizaje del concepto jerárquicamente inferior
- Incrementar su CV + 0.1
- Generar salto al segmento 1

Bloque 7

5 Generar el segmento 3 (o bloque de repaso):

5.1 Generar el contenido “Muy bien, repasemos entonces.”

Bloque 8

5.2 Por cada concepto con CV > 0 y relación entre conceptos generar una pregunta:

Bloque 9

- Por cada pregunta, generar dos alternativas:
    * Respuesta correcta
        ** Incrementar CV del concepto(s) + 0.3
        ** Generar “Muy bien”
    * Respuesta incorrecta
        ** Resetear CV = 0
        ** Generar modulo de aprendizaje del concepto(s)

Bloque 10

5.3 Por cada bloque de 3 preguntas:

- Generar salto al segmento 1

Una vez definidos los bloques en los que se divide el algoritmo, la Figura 19 muestra el diagrama de cajas blancas que se ha creado.

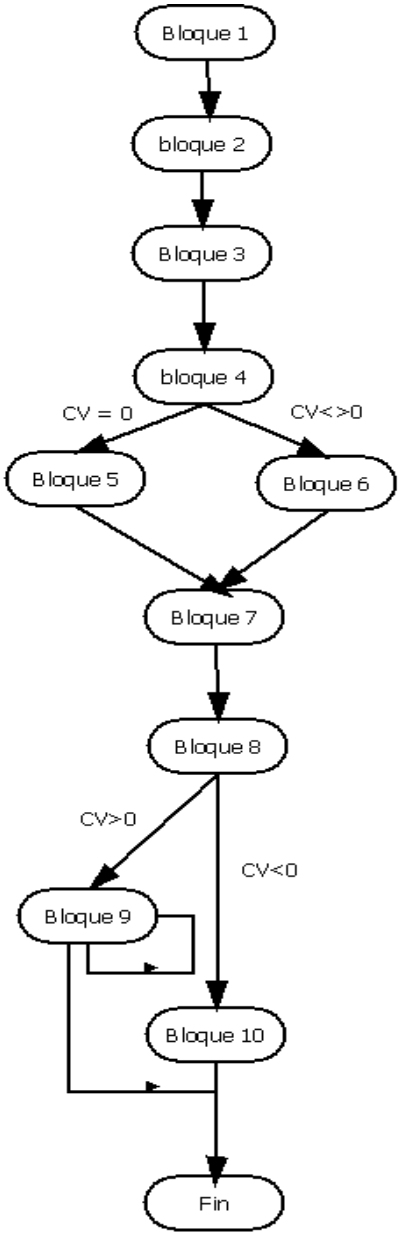


Figura 19 Diagrama de cajas blancas

La Tabla 16 recoge los valores que se han calculado para cada caso de prueba. CV se refiere al valor de confianza que tiene JARO de que el estudiante conozca el concepto a repasar. Este valor oscila entre 0 (falta total de confianza) y 1 (total confianza).

Caso	**Variable**	**Valor**
1	CV	0.7
2	CV	0.5
3	CV	0
4	CV	-0.1
5	CV	-1

Tabla 16 Valores caja blanca

Con los casos de prueba mencionados anteriormente cumplimos que pasamos por todas las partes del grafo comprobando que no hay caminos aislados o bucles de los que no podemos salir, y validando de esta forma el correcto funcionamiento del algoritmo.

4.1.2 Caja negra

Para la realización de las pruebas de caja negra se distinguirán los siguientes valores:

Valores límite

Valores que a la hora de interactuar en lenguaje natural tienen caracteres en mayúsculas y minúsculas, o bien están rodeadas de frases con otros contenidos.

Clases de equivalencia

Estas clases determinan los valores límites a probar para cada caso según se recoge en la Tabla 17

Prueba	**Valor de la prueba**	**Justificación**
1	Espero ApRender	Comprobamos que la aplicación reconoce entre mayúsculas y minúsculas para el caso en que el usuario solicite ir al apartado

		aprender
2	QuieRo RePasar	Comprobamos que la aplicación reconoce entre mayúsculas y minúsculas para el caso en que el usuario solicite ir al apartado repasar
3	Deseo Salir de la aplicación	Comprobamos que la aplicación reconoce entre mayúsculas y minúsculas para el caso en que el usuario solicite ir al apartado salir de la aplicación
6	7	Elegimos este valor para comprobar que funciona con valores intermedios de uso, ya que en los casos máximos y mínimos no se llega a ejecutar la formula

Tabla 17 Valores de las clases de equivalencia

En esta parte de las pruebas, se valida el correcto funcionamiento de la redirección a la página según el estado del algoritmo y el diálogo que se debe generar para el usuario en pantalla.

Prueba 1

Modulo	**Valor introducido**	**Valor esperado**	**Resultado**
Chat1_1.php	Espero ApRender	Ir al modulo aprender1_1.php	Correcto

Tabla 18 Prueba 1

Prueba 2

Modulo	Valor introducido	Valor esperado	Resultado
Chat1_1.php	QuieRo RePasar	Ir al modulo repasar1_1.php	Incorrecto

Tabla 19 Prueba 2

En esta prueba no se ha producido el resultado esperado debido a que en la posterior revisión del código se ha visto que la función para diferenciar entre mayúsculas y minúsculas no estaba bien definida para este parámetro, por lo que se ha ido al modulo repasar1_1.php y se ha corregido dicha parte del módulo.

Prueba 3

Modulo	Valor introducido	Valor esperado	Resultado
Chat1_1.php	Deseo Salir de la aplicación	Ir al modulo finalizar1_1.php	Correcto

Tabla 20 Prueba 3

Prueba 4

Modulo	Valor introducido	Valor esperado	Resultado
Aprender.php	Vacío	Se espera que no salga ningún valor para aprender porque ya todos los valores están aprendidos y solo necesitan ser repasado	Correcto

Tabla 21 Prueba 4

Prueba 5

Modulo	Valor introducido	Valor esperado	Resultado
Repasar.php	Vacío	Se espera que el valor se actualice correctamente en la base de datos, para un valor repasado	Incorrecto.

Tabla 22 Prueba 5

La prueba 5 resultó con un fallo, debido a que el valor de la fórmula, no era pasado correctamente a través del formulario, y siempre resultaba con un valor de 0 por defecto. Este error se ha solucionado correctamente.

Prueba 6

Modulo	Valor introducido	Valor esperado	Resultado
Repasar2_1.php	Introducimos un valor de 7 en el campo autoevaluación	Se espera una actualización correcta del valor de confianza, según el valor introducido	Correcto

Tabla 23 Prueba 6

Prueba 7

Modulo	Valor introducido	Valor esperado	Resultado
Chat1_1.php	Repasar	Se espera una actualización correcta del valor de confianza, según el valor introducido	Correcto

Tabla 24 Prueba 7

Prueba 8

Modulo	Valor introducido	Valor esperado	Resultado
Chat1_1.php	Pulsar sobre el link finalizar	Se espera que nos lleve a la pagina finalizar.php y nos muestre el mensaje de salida	Correcto

Tabla 25 Prueba 8

4.2 Pruebas de integración

En este apartado de las pruebas de integración se comprueba si los módulos funcionan interrelacionados correctamente.

Esta parte se ha ido desarrollando de manera sistemática, desde el inicio del proyecto comprobando que la unión entre los distintos módulos PHP que componen la aplicación eran accesibles desde los links y mediante lenguaje natural, para luego comprobar que las acciones a realizar dentro de cada modulo son realizadas correctamente.

La unión entre módulos se ha comprobado siguiendo el diagrama de estados de la aplicación JARO mostrado en la Figura 13.

4.3 Pruebas de validación

En este apartado se comprueba si el sistema cumple todos los requisitos que se pedían en la fase análisis. También se ha probado que el sistema funciona tanto con archivos XML de pequeño tamaño (menos de diez conceptos) como con archivos XML con más conceptos para asegurar la escalabilidad de la aplicación.

En particular, se muestran a continuación el contenido correcto de las tablas de la base de datos a partir del fichero XML de nombre entrada.xml (anexo con el código en el CD).

Contenido tabla **usuario**

	nif	nombre	apellidos	sexo	edad
	55556677V	Juan	Valdes	Hombre	48

Tabla 26 contenido tabla usuario

Contenido tabla **sabe**

	nif	id_con	cv	fecha	hora
	55556677V	1	0.3	0000-00-00	=14:39
	55556677V	2	0.7	0000-00-00	=12:00
	55556677V	3	0.7	0000-00-00	=12:00
	55556677V	4	0.66	0000-00-00	=12:00
	55556677V	5	0.75	0000-00-00	=12:00
	55556677V	6	0.35	0000-00-00	=12:00
	55556677V	7	0.7	0000-00-00	=12:00
	55556677V	8	0.715	0000-00-00	=12:00
	55556677V	9	0.8	0000-00-00	=12:00
	55556677V	10	0.2	0000-00-00	=12:00
	55556677V	11	0.66	0000-00-00	=12:00
	55556677V	12	0.9	0000-00-00	=12:00

Tabla 27 contenido tabla sabe

Contenido tabla **concepto**

			id_con	nombre	definicion	img	url	extra
☐			1	proceso	programa en ejecuciÃ³n	esto es una imagen	www.proceso.com	campo extra
☐			2	proceso padre	proceso que inicio a un determinado proceso	esto es una imagen	www.procesopadre.org	Podria ir vacio
☐			3	proceso hijo	proceso creado por un determinado proceso	esto es una imagen	www.procesohijo.org	Podria ir vacio
☐			4	Process ID (PID)	identificador de proceso unico	esto es una imagen	www.PID.org	Podria ir vacio
☐			5	User ID (UID)	Indentificador que nos dice quien es el creador de	esto es una imagen	www.UID.org	Podria ir vacio
☐			6	Parent Process ID (PPID)	Es el PID de su proceso padre	esto es una imagen	www.PPID.org	Podria ir vacio
☐			7	Kernel	Parte esencial de un sistema operativo que provee	esto es una imagen	www.kernel.org	Podria ir vacio
☐			8	identificador	numero de cuatro cifras que se utiliza para identi	esto es una imagen	www.identificador.org	Podria ir vacio
☐			9	funcion fork	crea una copia idÃ©ntica del proceso que la invoca	esto es una imagen	www.fork.org	Podria ir vacio
☐			10	funcion wait	Proceso que ejecuta un usuario.	esto es una imagen	www.wait.org	Podria ir vacio
☐			11	funcion exit	Proceso que ejecuta un usuario.	esto es una imagen	www.exit.org	Podria ir vacio
☐			12	funcion	Rutina que tiene el kernel	esto es una imagen	www.funcion.org	Podria ir vacio

Tabla 28 contenido tabla concepto

Contenido tabla **relaciona**

	id1	id2	id_tipo
	2	1	es un
	3	1	es un
	1	4	tiene
	1	5	tiene
	7	1	genera
	7	8	genera
	9	12	es un
	10	12	es un
	11	12	es un
	6	8	es un
	5	8	es un
	4	8	es un
	1	6	tiene

Tabla 29 contenido tabla relaciona

4.4 Pruebas de aceptación

En este apartado se describen las pruebas realizadas por usuarios para la validación de los requisitos enunciados en el análisis.

Los perfiles elegidos han sido al azar, tanto de perfiles técnicos como de perfiles no técnicos para dar así una mayor amplitud a las pruebas realizadas.

La Figura 20 muestra las preguntas iniciales realizadas a los usuarios de JARO y la Tabla 30 recoge el resto de información solicitada (valoración de la facilidad de manejo, rapidez, diseño y opinión general) en la escala de 0 (total disconformidad) a 10 (total conformidad). Estos resultados se preguntaban al término de interaccionar un mínimo de 15 minutos con la herramienta. Se puede observar como la valoración de los usuarios es positiva, tanto en facilidad como en manejo, independientemente del perfil técnico de los usuarios.

También se ha probado JARO en varios navegadores como Internet Explorer, Firefox y Opera.

CUESTIONARIO DE PRUEBAS DE ACEPTACIÓN DE JARO
Muchas gracias por su colaboración

1. Si tiene estudios universitarios, ¿qué carrera cursó?

2. Si no tiene estudios universitarios, ¿cuál es el número de horas que usa al día el ordenador?

3. Indique su rango de edad con un intervalo de 5 años

Figura 20 resultados

Perfil usuario	Edad	Ocupación	Facilidad de manejo	Rapidez	Diseño	Opinión general
No técnico	20-30	Estudiante	7	9	6	7.5
No técnico	55-60	Trabajador	6	10	7	6
No técnico	55-60	Jubilado	5	10	5	6.5
Técnico	30-35	Informático	8	9	6	6
Técnico	25-30	Informático	8	9	6	6
Técnico	30-35	Vendedor	6	9	5	7

Tabla 30 Resultados usuarios

4.5 Planificación proyecto

La Tabla 31 recoge la evolución del proyecto y como se han ido cumpliendo los hitos marcados.

Fecha	**Tareas**
28-10-09 a 12-11-09	- Empezar a organizar el trabajo. - Aprender a usar JabRef como gestor de referencias. - Buscar-empezar a leer las referencias - Registrar el proyecto
13-11-09 a 24-11-09	- Leer las referencias, y al mismo tiempo ir escribiendo un informe en el que se vayan describiendo las tecnologías existentes, y hacer una tabla comparativa.

25-11-09 a 10-12-09	- Escoger una alternativa. - Diseño de la base de datos (modelo E-R). - Diseño a alto nivel de la arquitectura del sistema y sus módulos.
11-12-09 a 07-01-10	- Revisión del diseño de la BD y de la arquitectura. - Reparto de tareas de codificación (interfaces). - Diseño del plan de pruebas.
08-01-10 a 14-01-10	- Revisión de la codificación (I) - Ir escribiendo la descripción informática.
15-01-10 a 21-01-10	- Revisión de la codificación (II) - Ir escribiendo la descripción informática.
22-01-10 a 25-02-10	- Pruebas según el plan redactado previamente. - Ir escribiendo los resultados.
26-02-10 a 11-03-10	- Revisión de las pruebas - Escribir Introducción y Objetivos
12-03-10 a 25-03-10	- Revisar Introducción y Objetivos - Escribir Descripción Informática
26-03-10 a 08-04-10	- Revisar Descripción Informática - Escribir Conclusiones, Bibliografía y Anexos
09-04-10 a mayo	- Revisar Conclusiones, Bibliografía y Anexos - Relectura final - Preparación de la presentación
junio	Defensa del PFC

Tabla 31 Planificación del proyecto

5 Conclusiones

El objetivo de este proyecto era crear una herramienta de repaso web con interacción en lenguaje natural con el estudiante capaz de seguir esquemas de diálogos generados a partir de modelos conceptuales. Este objetivo se ha cumplido mediante la implementación de JARO.

Las pruebas realizadas han verificado el correcto funcionamiento del sistema, y también se han validado los requisitos educidos en el análisis como se describe a continuación.

Requisitos funcionales

- *REQ01general: El alumno tendrá la opción de aprender conceptos*
  Este concepto es básico de la aplicación y se ha realizado con la herramienta creada para el proyecto JARO. En particular, se corresponde con al estado *aprender* desde cualquier parte de la aplicación, bien mediante lenguaje natural o bien mediante un enlace.

- *REQ02general: El alumno tendrá la opción de repasar conceptos*
  Este concepto es básico de la aplicación y se ha realizado con la herramienta creada para el proyecto JARO. En particular, se corresponde con el estado *repasar* desde cualquier parte de la aplicación, bien mediante lenguaje natural o bien mediante un enlace.

- *REQ03general: Las preguntas se generaran a partir del esquema conceptual previamente generado por el algoritmo*
  Mediante el uso de patrón de lenguaje se ha realizado este requisito, que es importante para que la aplicación adquiera un tono más cercano a un interlocutor humano, facilitando así la interacción entre el usuario y la máquina.

- *REQ04general: Las respuestas del alumno serán evaluadas, y en función de la evaluación, el valor de confianza del alumno para un determinado concepto*

variará.

Mediante la lógica interna del programa, se evalúa que el valor de confianza de los conceptos que tiene el alumno varíe en función de sus respuestas y de su aprendizaje.

- *Req05general: El alumno podrá elegir salir del módulo en cualquier momento guardándose los progresos del mismo de manera automática*

  Este apartado es de vital importancia, para la herramienta JARO, ya que cualquier cambio en los valores de confianza de los conceptos se guarda automáticamente para evitar que se repitan preguntas sobre el mismo concepto o sobre conceptos que ya han sido aprendidos.

- *REQ06general: Las peticiones del alumno bien podrán ser mediante interacción con la herramienta JARO a través de teclado y ratón o través de lenguaje natura*l

  En JARO existe la opción de moverse en la aplicación bien mediante el uso de lenguaje natural (escribiendo directamente la operación que se desea realizar dentro de un cuadro de texto), o bien pulsando sobre los enlaces que aparecen a la izquierda de la página.

Requisitos no funcionales

- *REQ01NF: Nuestro sistema tendrá un interfaz amigable al usuario a través de páginas web que sean intuitivas*.

  Este apartado se ha llevado a cabo realizando una interfaz sencilla, centrándose únicamente en los apartados más importantes de la aplicación, y buscando siempre la funcionalidad y la facilidad de uso para el usuario.

- *REQ02NF: El sistema responde en tiempo real a las peticiones del usuario. La velocidad no es un factor primordial, aunque el sistema no debe ser demasiado lento.*

Este apartado se ha llevado a cabo, vigilando como norma, en el código de la aplicación el código que produce penalización en tiempo como puede ser el excesivo uso de llamadas a la base de datos, o la mala implementación de los bucles.

- *REQ03NF: El producto se debe poder utilizar desde cualquier ordenador conectado a Internet.*
  La aplicación ha sido probada satisfactoriamente en distintos navegadores Web dando con en todos ellos un resultado satisfactorio.

Trabajo futuro

Las posibles líneas de trabajo futuro en este proyecto son las siguientes:

- Interacción con otro sistemas de e-learning, como Willow [1].
- Mejora de la gestión y generación de patrones de lenguaje para el diálogo.
- Mejora de la autoevaluación de la aplicación.

Para la primera línea la arquitectura en módulos de JARO y el uso del archivo XML, posibilita la integración con otros sistemas e-learning.

Para la gestión y generación de patrones del lenguaje, se necesitan conocimientos de lingüística, por lo que es un paso previo será realizar un estudio más extenso de la posible generación computacional de patrones.

Finalmente se podría mejora la autoevaluación que realiza JARO a las respuestas del alumno para que en lugar de pedir al estudiante que se asigne una puntuación, sea JARO quien asigné la puntuación. Esto sería posible mediante la integración de JARO en un sistema de evaluación de respuestas en texto libre como Willow.

BIBLIOGRAFÍA

[1] Pérez-Marín, D. (2009). *Adaptive computer assisted assessment of free-text students' answers: an approach to automatically generate students' conceptual models*, tesis doctoral presentada en la Universidad Autónoma de Madrid en 2007 y publicada por VDM en 2009.

[2] Holzinger, A. (2005). Usability engineering methods for software developers, *Commun. ACM* **48**(1), 71-74.

[3] Zadrozny, W.; Budzikowska, M.; Chai, J.; Kambhatla, N.; Levesque, S. & Nicolov, N. (2000), 'Natural language dialogue for personalized interaction', *Commun. ACM* **43**(8), 116--120.

[4] Zadrozny, W.; Budzikowska, M.; Chai, J.; Kambhatla, N.; Levesque, S. & Nicolov, N. (2000), Natural language dialogue for personalized interaction, *Commun. ACM* **43**(8), 116--120.

[5] Hodges, C. (2004), 'Designing to Motivate: Motivational Techniques to Incorporate in E-Learning Experiences', *The Journal of Interactive Online Learning* **2**(3).

[6] Vera, J. A. (2008), *El gran libro de Windows Server 2008*, Marcombo.

[7] Welling, L.; Thomson, L. (2009), *Programación y desarrollo Web con PHP y Mysql*, Anaya.
[8] Freeman, R. G. (2008), *Oracle database 11g new features*, McGraw-Hill.

[9] Prague, C. (2004), *El libro de Microsoft Office Access 2003*, Anaya Multimedia.

[10] Moller, A. (2006), *An introduction to XML and Web technologies*, Addison-Wesley.

[11] Urbaneja, J. (2001), *JSP (Guias Practicas)*, Anaya Multimedia.

ANEXO

Aquí que se adjunta en el CD el código de la aplicación resultado del proyecto realizado.