Universidad de las Ciencias Informáticas

Facultad 4

Trabajo de Diploma para optar por el Título de Ingeniero en Ciencias Informáticas

Herramienta de autor para crear tutores inteligentes basados en la ejemplificación

Autores:

Yasiel Tejeda González

Marlon Cabrera Torres

Tutores:

Ing. Enrique José Altuna Castillo

Ing. Yolanda Mauri Pérez

La Habana, junio del 2014

"Año 56 de la Revolución"

"El único modo de hacer un gran trabajo es amar lo que haces. Si no lo has encontrado todavía, sigue buscando. No te acomodes. Como con todo lo que es propio del corazón, lo sabrás cuando lo encuentres."

Steve Jobs

"Milito en el bando de los impacientes, milito en el bando de los apurados, de los que siempre presionan para que las cosas se hagan y de los que muchas veces tratan de hacer más de lo que se puede."

Fidel Castro

"Nunca consideres el estudio como una obligación, sino como una oportunidad para penetrar en el bello y maravilloso mundo del saber."

Albert Einstein

Declaración de autoría

Declaramos ser autores de este trabajo de diploma y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes _____ del año 2014.

_____ _____

Yasiel Tejeda González
Autor

Marlon Cabrera Torres
Autor

_____ _____

Ing. Enrique José Altuna Castillo
Tutor

Ing. Yolanda Mauri Pérez
Tutor

Dedicatoria

De Yasiel Tejeda González

A mis padres **Vivian** y **Mandy** por el apoyo incondicional a lo largo de toda mi vida.

A mi hermano **Yarmando** al que quiero con la vida.

De Marlon Cabrera Torres

A mis padres **María Isabel** y **Armando** por todo el cariño y la educación que me han dado.

A mi hermana **Mailyn** por ser siempre mi ejemplo a seguir.

Agradecimientos

De Yasiel Tejeda González

A mis padres Vivian y Mandy, que gracias a ellos hoy me convierto en ingeniero. Por ser ejemplos a seguir y guiarme por el buen camino toda la vida. Por dejarme sentir orgulloso cuando hablo de ellos.

A mi hermano Yarmando, del cual espero que se haga un profesional.

A mis abuelas y abuelos, que aunque 2 de ellos no están entre nosotros se sentirían orgulloso en este momento.

A mis tíos y tías Andrés, Alberto, Armindo, Tere, Lidia y Xiomara por siempre estar pendiente de mí.

A todos mis primos en especial Sheila, Norge y Yinet que me han ayudado mucho durante mi carrera.

A mis primos que se encuentran lejos de Cuba pero siempre pendiente de lo que necesitaba.

A mis amigos del IPI Maidelyn, Adriana, Fundora y Norka por compartir tantos momentos juntos.

A mis amigos de la UCI, en especial a Yunier, Maday Cristina, Yuselis, Remberto, Wendy, Micha y Roberto por toda la amistad que me han transmitido durante los 5 años.

A mi compañero de tesis Marlon por tanto apoyo durante la realización de este trabajo de diploma.

A mis tutores, Enrique y Yolanda que nos han apoyado muchísimo en este trabajo de diploma.

A mis compañeros de apartamento y de grupo.

A mis compañeros de la FEU por compartir tantos momentos de trabajo y alegrías en estos 5 años.

A todos los profesores que me han acompañado y guiado en estos años de universidad. Gracias a todos por compartir conmigo más que su conocimiento y ayudarme a ser mejor.

A todo el que me faltó por mencionar y que me ayudó durante mis estudios. Son muchos los que han formado parte de estos 5 años.

A todos ellos, muchas gracias…

De Marlon Cabrera Torres

A mi mamá, por todo el cariño, los años de sacrificio, por mi educación, por enseñarme a vivir, a luchar, a no rendirme, por todo y porque si, gracias.

A mi papá, que donde quiera que este sé que estará orgulloso de sus hijos, por todo lo que aprendí de él en corto tiempo que lo tuve a mi lado y porque su recuerdo me hace querer siempre ser mejor persona.

A mi hermana favorita, gracias por ser y estar, por ser siempre el espejo en el cual mirarme para crecer como persona y como profesional, por estar siempre ahí, y por mimarme y guiarme cuando ando perdido.Mil gracias “chama”.

A mi sobri que tiene a todos locos en la casa, y a el pesimillo de Yander por todo lo que lo jodo siempre, y lo que le falta.

A toda mi familia que también han sido siempre de gran ayuda, muy en especial a mi tío Jesús y a Irma, a mi tía Teresa y mi tío Carlos, a mis primos Bettys, Carmen, Ale, Purringo, a mi abuela, en fin son muchos como para mencionarlos a todos pero a todos gracias.

Al piquete de siempre, que siempre están hasta cuando no están. A mi hermanito Arturo, Yoandri, Dayi, Danne, Robert, Ransel, Claudia, Yoana, Roxana, a Raúl y a Os que me tiró tremendo salve en el diseño. Gracias por ser los fijos.

A Wendy por su paciencia, su comprensión, su cariño y por ser siempre incondicional.

A Misla y a toda su familia, por los buenos ratos compartidos y por quererme y aceptarme como uno más de la familia.

A mi compañero de tesis, “el gordito”, por poner la responsabilidad, la organización y las ganas de trabajar en esta tesis, sin él estaría aún en el capítulo 1.

A mi supertutora Yolanda por guiarnos en el proceso, por su exigencia y su constancia (y por los dulces y meriendas también).

A mi tutor Enrique por darnos una investigación de la que me siento orgulloso y por todo lo que lo jodimos.

A todos mis compañeros de aula, los que están en la escuela y los que no, de todos y cada uno de ellos me llevo un recuerdo enorme de los ratos compartidos, buenos y malos, de todos aprendí algo(no necesariamente algo bueno, fliper y lachy saben de qué hablo), al nerd y a taz por todo

lo que los jodí cada vez que me pasaba algo en la aplicación, al Leo y a Arielito por ser dos grandes amigos, y a Joazel, Lenier, Livany y Adrián por compartir conmigo un primer año de ensueño y miles de historias para contar.

A mis compañeros de cuarto, los del 140101 y los del 135206.

A los buenos amigos del IPI a pesar de que a muchos nos tocó separarnos y casi que ni nos vemos ya, pero que conservo de ahí grandes amigos.

A la gente del futbol, los del sala, los del 11 y a mi equipazo los Coffe Cake.

A todos los profes, los que me dieron clases, y hasta los que no, a la gente del proyecto Español para no hispanohablantes (Dolphin), a todos los que me preguntaban siempre: ¿y la tesis cómo va?, ¿Cuándo te gradúas?, a todo en el que de una forma u otra tuvo que ver con el resultado de esta tesis muchas gracias.

Resumen

Los Sistemas Tutores Inteligentes modelan la enseñanza, el aprendizaje, la comunicación y el dominio del conocimiento del especialista y el entendimiento del estudiante sobre ese dominio. Con el objetivo de simplificar el proceso de construcción de estos sistemas y otras variantes de software educativo han surgido varias herramientas de autor, aplicaciones de software orientadas a la creación de materiales educativos con las que el usuario, sin necesidad de conocer el lenguaje de programación, pueda crear módulos que integren todos los componentes de un curso. Estas herramientas en su mayoría son privativas o atadas a determinadas tecnologías o plataformas y carecen de funcionalidades importantes para la utilización de las mismas en la web actual, como el caso de un módulo multilenguaje, la obtención y ejecución de los tutores de forma independiente y el almacenamiento de las trazas de los usuarios. La esencia de esta investigación es, desarrollar una herramienta de autor para crear y difundir Sistemas Tutores Inteligentes basados en la ejemplificación que apoyen al proceso de enseñanza-aprendizaje. Las herramientas y tecnologías usadas en el desarrollo son libres, por lo que se cumplió con la política de la universidad y del país de potenciar el uso del software libre. El proceso de desarrollo estuvo guiado por la metodología XP y fue generada la documentación correspondiente a las etapas de planificación, diseño, implementación y pruebas, obteniéndose así una herramienta capaz de crear, ejecutar, exportar e importar Sistemas Tutores Inteligentes basados en la ejemplificación, validada por las pruebas unitarias y de aceptación.

Palabras clave: Herramientas de Autor, Sistemas Tutores Inteligentes, Software educativo.

Índice

Introducción

El desarrollo de la educación favorece directamente al progreso social y económico de una región o un país. Para que esta premisa tenga efecto, es necesario un aumento de las capacidades de cada individuo. El objetivo fundamental de la educación es proporcionar a los estudiantes, una formación plena que les ayude a estructurar su identidad y a desarrollar sus conocimientos en aras de participar en la construcción de la sociedad (1).

Las investigaciones relacionadas con la calidad del proceso de enseñanza aprendizaje apuntan a que existe una relación directa entre la cantidad de atención que recibe el estudiante durante la etapa de formación y los resultados generales del proceso. Es por esto que la instrucción uno a uno es considerada como la forma más efectiva para enseñar un dominio del conocimiento.

A partir de esta observación, las organizaciones que a nivel internacional marcan las pautas sobre la calidad de la educación, han establecido como uno de los objetivos del milenio lograr un profesor por cada estudiante (2), sin embargo, en los contextos tradicionales de enseñanza no es posible lograr este objetivo porque la densidad de estudiantes por profesores es muy alta.

La educación a distancia ha intentado resolver esta situación desvinculando el proceso de enseñanza-aprendizaje de las restricciones en tiempo y espacio a las que habitualmente ha estado ligado. En este contexto surgen los software educativos, que se definen como un programa automatizado diseñado con una intencionalidad educativa para ser utilizado en el proceso de aprendizaje, utiliza procedimientos para que el estudiante aprenda, se fomenta el análisis de problemas, facilita el trabajo en grupo, provee soporte en actividades docentes y en el sentido más amplio, mejora las habilidades del pensamiento y la resolución de problemas (3).

En el marco de esta investigación, partiendo de las clasificaciones que propone **Marqués** (3), los software educativos se clasifican en: tutoriales, simuladores, entornos de programación y herramientas de autor. Se caracterizan por ser altamente interactivos, a partir del empleo de recursos multimedia, como videos, sonidos, fotografías, diccionarios especializados, explicaciones de experimentados profesores, ejercicios y juegos instructivos que apoyan las funciones de evaluación y diagnóstico.

Los Sistemas Tutores Inteligentes surgen como una variante de software educativo para brindar un apoyo constante, individualizado e inmediato al estudiante mientras aprenden, de forma similar a como lo haría un profesor humano.

Existen tres tipos de Sistemas Tutores Inteligentes, los cognitivos basados en el modelo, los cognitivos basados en restricciones y los basados en la ejemplificación. Los Sistemas Tutores Inteligentes cognitivos siguen e interpretan el comportamiento de los estudiantes y hacen referencia a un modelo cognitivo, compuesto por reglas de producción para representar las habilidades. En los Sistemas Tutores Inteligentes basados en restricciones el conocimiento está representado en forma de restricciones, en lugar de los caminos de resolución de problemas.

Un tutor cognitivo basado en el modelo ofrece una gran flexibilidad y generalidad de los Sistemas Tutores Inteligentes, aunque son costosos de construir. Ellos requieren una gran cantidad de conocimiento del dominio, como conocimiento en programación con inteligencia artificial y la teoría cognitiva, a menudo a un nivel de doctorado. Se ha demostrado que por lo general toma 200 horas al autor una hora de contenido (4).

Debido a esto se produce un impulso para desarrollar una forma más simple de tutor, que puede coincidir con el mismo comportamiento de un tutor cognitivo. A estos se les llama tutor basado en la ejemplificación (5), porque emula un tutor inteligente sin necesidad de utilizar inteligencia artificial. Los Sistemas Tutores Inteligentes basados en la ejemplificación interpretan los pasos de la solución proporcionada por los estudiantes y los *hints*[1] requeridos y los compara con un grafo de solución para un problema dado, también llamado grafo de comportamiento (directo o acíclico que representa la solución) (6).

En Cuba solamente se han producido unos pocos prototipos de Sistemas Tutores Inteligentes, en todos los casos ligados en extremo a la enseñanza de un dominio del conocimiento y con una estructura monolítica que difícilmente puedan ser extendidos sin elevados esfuerzos de desarrollo (7).

Uno de los elementos que ha limitado la introducción de esta tecnología es sin dudas, el alto nivel de complejidad técnica que encierra ella en sí misma y la necesidad de unir los esfuerzos de equipos de desarrollo multidisciplinarios (especialistas del área de la informática, del dominio del conocimiento que se enseñará y en pedagogía), como plantean los enfoques tradicionales de construcción. Este requisito hace de la construcción de los Sistemas Tutores Inteligentes desde cero, un proceso largo, difícil y costoso, lo que limita el número de tutores que se pueden crear en los diversos dominios. De este modo se obtienen potentes herramientas con limitados costos de explotación e importante utilidad práctica.

[1] hints: mensaje de ayuda o retroalimentación.

Con el objetivo de simplificar el proceso de construcción de los Sistemas Tutores Inteligentes y otras variantes de software educativo han surgido las herramientas de autor, estas no son más que aplicaciones de software orientadas a la creación de materiales educativos con las que el usuario, sin necesidad de conocer el lenguaje de programación, pueda crear módulos que integren todos los componentes de un curso.

El uso de estas herramientas hoy día se ha visto afectado por las leyes de propiedad, que impiden una correcta utilización de estos sistemas en la creación de materiales educativos. Otra limitante que posee la mayor parte de las herramientas disponibles para este fin, es que no permiten a los creadores difundir los tutores realizados en ellas, ya sea, en la propia herramienta o exportándolos en formatos estándares que posibiliten ser visualizados en momentos posteriores y bajo otras condiciones tecnológicas, por ejemplo en los navegadores web.

La comunidad de desarrolladores de aplicaciones educativas en Cuba y específicamente en la UCI, se ha insertado en la construcción de herramientas de autor para desarrollar aplicaciones educativas con tecnología multimedia que apoyen el proceso de enseñanza-aprendizaje; sin embargo, según la bibliografía consultada se comprobó que no permiten la creación ni difusión de sistemas tutores inteligentes.

Para dar solución a la situación problémica existente surge el siguiente **problema de investigación:**

¿Cómo contribuir a la creación y difusión de Sistemas de Tutores Inteligentes basados en la ejemplificación?

Como **objetivo general** se propone, desarrollar una herramienta de autor de Sistemas Tutores Inteligentes basados en la ejemplificación para contribuir a la creación y difusión de estos materiales educativos.

A partir del objetivo general definido se derivan los siguientes **objetivos específicos:**

- Elaborar el marco teórico conceptual a partir del estudio de las principales tendencias y herramientas existentes relacionadas con la creación de Sistemas de Tutores Inteligentes.
- Identificar los requerimientos con los que debe cumplir el software y una metodología de desarrollo de software para aplicar a lo largo de la investigación.
- Implementar las funcionalidades de la herramienta de autor.
- Probar las funcionalidades implementadas.

La investigación se enmarca en el **objeto de estudio:** herramientas de autor para crear Sistemas de Tutores Inteligentes, teniendo como **campo de acción:** herramientas de autor para crear Sistemas de Tutores Inteligentes basados en la ejemplificación.

La **idea a defender** del presente trabajo es: la implementación de una herramienta de autor contribuirá a facilitar el proceso de creación y difusión de Sistemas Tutores Inteligentes basados en la ejemplificación.

Se definieron las siguientes **tareas de investigación:**

- Estudio de la bibliografía actualizada para la elaboración del marco teórico-conceptual referente a los Sistemas Tutores Inteligentes.
- Identificación de las limitaciones de los sistemas similares a la solución que se propone.
- Identificación de los requerimientos con los que debe cumplir la herramienta de autor.
- Generación de los artefactos correspondientes con la metodología de desarrollo de software seleccionada.
- Definición de los elementos que componen la arquitectura de la herramienta de autor a implementar.
- Implementación de las funcionalidades para la herramienta de autor.
- Realización de pruebas a la herramienta de autor para verificar la correspondencia de las funcionalidades con los requerimientos identificados.

Posibles resultados: Herramienta de autor para crear y difundir Sistemas Tutores Inteligentes basados en la ejemplificación.

Para el desarrollo de las tareas de investigación se combinan diferentes **métodos científicos** en la búsqueda y procesamiento de la información, estos son:

A nivel teórico:

Análisis-síntesis: permitió analizar la situación y la problemática existente, la documentación, las características de las metodologías, herramientas y tecnologías, para luego llegar a la conclusión de cuáles son las más adecuadas para el desarrollo.

Análisis histórico-lógico: permitió conocer, con mayor profundidad, los antecedentes y las tendencias actuales referidas a las herramientas de autor y tutores inteligentes existentes, además de conceptos, términos y vocabularios propios del campo como herramienta de autor, tutores, que contribuyen en gran medida al entendimiento del objeto de estudio.

Modelación: la modelación permitió representar los conceptos relacionados con la herramienta y el modelo de datos para un mejor entendimiento de la misma.

A nivel empírico:

Observación: posibilitó tomar las mejores prácticas de sistemas similares con el mismo fin de la herramienta a desarrollar así como las vulnerabilidades para convertirlas en ventajas de la aplicación resultante de esta investigación.

La investigación se estructura en tres capítulos:

Capítulo 1: Fundamentación teórica, aborda los principales conceptos relacionados con el dominio. Se realiza un estudio del estado del arte de las herramientas de autor existentes. Se justifica la selección de las distintas herramientas, tecnologías y metodologías a utilizar.

Capítulo 2: Descripción de la propuesta de solución, se detalla el flujo existente de los procesos involucrados en el objeto de estudio planteado. Se describe la estructura interna del sistema mediante los artefactos generados por la metodología de desarrollo utilizada.

Capítulo 3: Implementación y pruebas, contiene los temas referentes al proceso de implementación y las estrategias de prueba a utilizar.

Capítulo 1 Fundamentación teórica

La idea de aprovechar las herramientas informáticas en la enseñanza se remonta a los años 50. Los primeros sistemas de enseñanza tradicionales desarrollados se conocen con el nombre de EAC (Enseñanza Asistida por Computadora, o CAI: *Computer-Assisted Instruction*). En esa década surgen los software educativos, los que incorporan gran cantidad de contenido, que en su conjunto, conforman un recurso didáctico valioso para cualquier educador. El desarrollo de software educativos tiene particular relevancia en el ámbito pedagógico, su incorporación a los procesos de enseñanza y aprendizaje se considera como uno de los aspectos determinantes en la calidad de su utilización.

1.1. Sistema Tutor Inteligente

Los tutores recobran un especial interés en los años 80 cuando la enseñanza asistida por computadora comienza a utilizar técnicas de la Inteligencia Artificial (IA). En esa época surgen los denominados Sistemas Tutores Inteligentes (en lo adelante STI) con la vocación clara de desarrollar procesos de enseñanza adaptados a los diferentes usuarios/estudiantes (8). Su objetivo es proporcionar una mayor flexibilidad a los tutoriales manejados por computadora y lograr que éstos permitan una mejor interacción con el usuario.

A los STI se les ha dado varias definiciones en las diferentes bibliografías consultadas para esta investigación. Los autores de este trabajo utilizarán la definición de **Wolf** (9) , el cual plantea que son *"sistemas que modelan la enseñanza, el aprendizaje, la comunicación y el dominio del conocimiento del especialista y el entendimiento del estudiante sobre ese dominio".* A estos tutores se les pueden aplicar técnicas de inteligencia artificial o no, dependiendo del tipo de tutor.

Los STI se han desarrollado para mejorar el rendimiento de un estudiante en una amplia gama de dominios. En la escuela secundaria H.B. Beal[2] en Ontario Canadá se realizó una evaluación controlada de una forma interactiva en el sistema de tutorías online para las matemáticas (10). Los resultados mostraron que los estudiantes que recibieron tutoría en línea presentaron una mejoría, mientras que en los estudiantes que recibieron la instrucción de aula regular no hubo mejora después de las pruebas. PAT, un tutor de Álgebra, se utilizó en un experimento a gran

[2] Sitio oficial de la escuela http://www.tvdsb.ca/Beal.cfm

escala donde 470 estudiantes en las clases experimentales que utilizaron el tutor superaron a los estudiantes en clases de comparación en un 15% en las pruebas estandarizadas (11).

Clasificación de los STI

Mediante el uso de la inteligencia artificial de los STI se espera lograr, a través de la utilización de los sistemas informáticos, tutores que sean tan eficaces como los tutores humanos. El término STI cubre una amplia gama de posibles tutores informatizados, de tutores cognitivos basados en el modelo, tutores cognitivos basados en restricciones y tutores basados en la ejemplificación (12).

Tutores cognitivos basados en el modelo: están basados en la teoría y la arquitectura de la cognición de ACT-R[3]. De acuerdo con la teoría de ACT-R, el conocimiento humano puede ser dividido en dos tipos de representaciones, declarativo (consiste en los hechos) y procedural (consiste en las producciones). El conocimiento procedural es formado a través del conocimiento declarativo (13).

El modelo cognitivo o modelo de seguimiento es asociado con un modelo experto que forma parte de la producción de normas que representan el conocimiento del dominio. Los tutores cognitivos usan un proceso llamado modelo de seguimiento, donde el modelo experto es usado para seguir las acciones del estudiante. El modelo de seguimiento puede ofrecer retroalimentación por cada paso que realice el estudiante mientras resuelve un problema. El mismo identifica los errores cuando un paso realizado por el estudiante o bien coincide con una regla de producción que representa un error o cuando no coincide con ninguna regla (13). A través de este modelo, el sistema puede construir una tutoría eficaz.

Tutores cognitivos basados en restricciones: son basados en la teoría de Ohlsson[4] de aprender de los errores del desempeño. El conocimiento está representado en forma de restricciones, en lugar de los caminos de resolución de problemas. Una restricción consiste en tres componentes: una condición de relevancia que describe cuando la restricción es aplicable, una condición de satisfacción que especifica pruebas adicionales y un mensaje de retroalimentación asociada con la restricción. Un tutor basado en restricciones se interesa por la situación actual del estudiante en lugar de lo que el estudiante ha hecho hasta el momento.

[3] Del inglés: Adaptive Character of Thought – Rational.
[4] PhD. Stellan Ohlsson Profesor de la Universidad de Illinois en Chicago.

Siempre y cuando el estudiante no entra en un estado incorrecto, él o ella es libre de hacer lo que desee. Los estados que son pedagógicamente equivalentes se agrupan en clases de equivalencia. Todos los estados de una clase de equivalencia desencadenan las mismas acciones de instrucción por parte del tutor. El modelado basado en restricciones requiere un mayor nivel de abstracción (14). Los tutores basados en restricciones utilizan una serie de limitaciones para comprender un problema, la entrada de un estudiante se compara con estas limitaciones. Una acción se considera correcta hasta que infringe una restricción.

Tutores basados en la ejemplificación: es un modelo cognitivo simplificado basado en un grafo de estado. Los grafos de estado son gráficos finitos y acíclicos donde cada arco representa una acción del estudiante y cada nodo un estado de la interfaz del problema (5). Se encargan de evaluar el comportamiento del estudiante mediante la comparación con ejemplos de comportamiento de resolución de problemas correctos e incorrectos.

Los tutores basados en la ejemplificación (o pseudo-tutores) son capaces de proporcionar una guía paso a paso y en su mayoría son construidos mediante la técnica de programación por demostración. En las ciencias computacionales, la programación por demostración es una técnica para el desarrollo por el usuario final para enseñar a la computadora o a un robot nuevos comportamientos mediante la demostración de una tarea en vez de programar directamente utilizando comandos de la máquina (15).

Pueden tutorar problemas complejos que constan de múltiples estrategias correctas mediante el mantenimiento de múltiples interpretaciones correctas de comportamiento de los estudiantes. Aunque los tutores basados en la ejemplificación son más fáciles de construir que los cognitivos, se ha demostrado que los dos tipos pueden ser conductualmente indistinguibles (16). La gran ventaja de los tutores basados en la ejemplificación es que un autor no necesita conocimiento previo de programación de inteligencia artificial para poder crearlo. Además, el tiempo que un autor necesita para crear tutores basados en la ejemplificación es significativamente menor que para crear tutores cognitivos.

Los tutores cognitivos basados en el modelo son los tutores computacionalmente más costosos de todos. Se requiere una búsqueda completa de un modelo del estudiante. Algunos autores opinan que su caro costo es compensado por sus fortalezas que incluyen generalidad y flexibilidad. Debido a que se basa en un modelo de estudiante escrito por un experto de dominio, es capaz de dar una respuesta muy exacta para cada entrada de los estudiantes. Además como

el tutor es capaz de resolver el problema, es posible brindarle una ayuda al estudiante de cómo debe continuar. Una crítica es que los tutores basados en el modelo obligan al estudiante a seguir un enfoque fijo.

Los tutores basados en restricciones son mucho menos eficientes que los tutores basados en el modelo, están limitados por su conjunto de restricciones. Dado que el modelo no tiene el concepto de un objetivo, no es posible para el tutor proporcionar al estudiante asesoramiento de cómo actuar. La mejor solución que los tutores basados en restricciones pueden ofrecer es una lista de todas las condiciones insatisfechas y las condiciones que pueden ser satisfechas.

Un tutor basado en el modelo o en la ejemplificación marcará una respuesta incorrecta si no se puede encontrar en el modelo o grafo de comportamiento, mientras que un tutor basado en restricciones trata todas las respuestas como correctas hasta que pruebe que son incorrectas. Debido a esto, si el modelo de restricción es incompleto, entonces es posible que las soluciones incorrectas sean tratadas como correctas. Esto va en contra del objetivo de los STI, que es el de imponer métodos correctos para la resolución de problemas y señalando las respuestas incorrectas que un estudiante puede tener.

Para facilitar la creación de los STI son frecuentemente utilizadas las herramientas de autor, estas simplifican el proceso de construcción de los STI y disminuyen el umbral de habilidad para la construcción de los mismos; dichas herramientas serán descritas a continuación.

1.2. Herramienta de autor

En las diferentes bibliografías consultadas se han analizado varias definiciones de herramientas de autor. Para esta investigación se utilizará el concepto que definió **Dabbagh** (17)*: "Las herramientas de autor son herramientas de desarrollo de software que posibilitan a diseñadores instructivos, educadores, maestros y aprendices diseñar un curso multimedia interactivo, y ambientes de aprendizaje en hipermedia sin el conocimiento de lenguajes de programación. Las herramientas de autor tienen como objetivo aplicaciones educativas que contengan generalmente, ya sea implícitamente o explícitamente, un modelo particular de la tarea en el que el usuario final debe estar ocupado, así como un modelo del proceso editorial mismo".*

Las herramientas de autor deben tener una serie de características (18), (19), (20), (21):

- **Alta compatibilidad, genérico y reutilizable:** las herramientas de autor crean los objetos de aprendizaje siguiendo una serie de estándares para que puedan ser utilizados por otras herramientas.
- **Fácil uso y elevada automatización de tareas:** este criterio permite establecer que son herramientas basadas en plantillas prediseñadas por el programador que no requieren por parte del usuario prácticamente ninguna preparación técnica.
- **Facilitan el diseño pedagógico del curso:** cada profesor tiene la posibilidad de adaptar el curso según las características de sus estudiantes y puede enfocarlo a las debilidades que presentan en los contenidos impartidos.
- **Varios niveles de ayuda:** lo mismo en el proceso de creación de los objetos de aprendizaje como en el uso dado por los estudiantes, estas herramientas cuentan con distintos niveles de ayuda que permiten guiar el proceso de aprendizaje y obtener una retroalimentación de las actividades realizadas.
- **Accesibilidad a discapacitados:** a las herramientas se le implementan funcionalidades que permiten que los discapacitados interactúen con ellas, por ejemplo admitir cambios en la configuración de la fuente para personas débiles visuales.
- **Independientes de la plataforma, material en sitios remotos o locales:** en su mayoría son aplicaciones web, aunque también hay de escritorio, que brindan la posibilidad de no estar atadas a un sistema operativo o plataforma determinada.

Las ventajas de la adopción de estos programas como instrumento de trabajo por parte de un profesor son múltiples y evidentes. Además del ahorro de tiempo de desarrollo y la menor formación técnica necesaria, se destacan otras razones como:

- La actualización de los métodos pedagógicos para adaptarnos, tanto individual como institucionalmente, a las necesidades educativas de la sociedad de la información.
- La mayor valoración del profesor con destrezas tecnológicas frente al “analfabeto informático”.
- Las mejoras en la calidad de la enseñanza, ya que permiten un mayor grado de efectividad en la comunicación de contenidos, versatilidad horaria e independencia física del aula o centro formativo y liberación de las funciones más rutinarias relacionadas con la transmisión de información.
- Mayores posibilidades de trabajo en colaboración y posibles nuevas fuentes de ingresos (otras formas de edición, nuevos productos o aplicaciones) (22).

1.3. Herramientas similares

En el mundo se han implementado una serie de herramientas de autor con características y funcionalidades diferentes, ejemplo de ellos son Authorware, Clic, EasyProf, exeLearning, Reload, Lectora y Croda, las cuales permiten elaborar objetos de aprendizaje, crear y gestionar materiales multimedia interactivos, generar contenidos *e-learning*, etc. Específicamente para crear STI han surgido soluciones como IRIS-D, xPST, CTAT, cuyas características serán expuestas a continuación.

IRIS-D

IRIS-D es un entorno para construir Sistemas Tutores Inteligentes cognitivos. Su utilidad reside en facilitar a profesores no expertos en el campo informático, la construcción de sistemas adaptativos de enseñanza-aprendizaje en dominios concretos en los que son expertos. El proceso que sigue IRIS-D para construir un nuevo tutor consiste en modificar, adaptar y completar una arquitectura de tutor genérica que sirve como plantilla para cumplir los requerimientos sobre el tutor final deseado (23). Es una aplicación de escritorio desarrollada en java. Esta aplicación tiene restricciones a la hora de crear los tutores ya que cuenta con una plantilla que es estática. No permite crear tutores basados en la ejemplificación y es de difícil integración con la web.

xPST

El xPST (*Extensible Problem Specific Tutor*), es una herramienta de autor web de STI que ayuda a desarrollar rápidamente tutores basados en la ejemplificación con interfaces existentes, como páginas web (24). Construir tutores con interfaces existentes reduce el tiempo de creación del tutor, y permite que la interfaz sea separable del componente de tutoría. El sistema xPST consta de tres componentes principales: el motor de xPST, el administrador de presentación y la herramienta de creación web. La herramienta ayuda a los autores a crear y desplegar tutores. Su principal deficiencia es que para registrar el comportamiento de los estudiantes utiliza un *plugin* implementado únicamente para Mozilla Firefox, por lo que se convierte en una herramienta dependiente de este navegador.

CTAT

CTAT soporta la creación de tutores cognitivos y tutores basados en la ejemplificación. Es uno de los punteros en cuanto a desarrollo de tutores se refiere (6). Es una aplicación de escritorio, multiplataforma, libre solo para usos investigativos o educativos. Carece de un módulo multilenguaje. Crea los tutores a partir de interfaces desarrolladas en java o flash, estas son tecnologías que con el avance de HTML5 y JavaScript han ido perdiendo popularidad en cuanto

al desarrollo web se refiere, además que dependen de una máquina virtual para ser ejecutadas y poseen una curva de aprendizaje superior.

CRODA

Una experiencia cubana en el desarrollo de herramientas de autor es CRODA. Una aplicación web que proporciona y facilita la creación de objetos de aprendizaje (OA) empleando el estándar SCORM[5]. Al incorporar a los OA este estándar se garantiza la interoperabilidad y reusabilidad de los mismos en diferentes plataformas que utilicen dichos estándares. Esta aplicación presenta una interfaz agradable y de cómodo uso. Facilita el acceso a los diferentes servicios manteniendo una simplicidad en su arquitectura de información. Dicha herramienta brinda además la posibilidad de crear plantillas, que definen una estructura inicial, que posteriormente podrán ser empleadas en la elaboración de algún OA, haciendo posible además que las mismas se encuentren disponibles para todos los usuarios después de que hayan pasado por un proceso de revisión que garantice una correcta estructura de diseño (25).

Conclusiones sobre el estudio de las herramientas similares

Luego del análisis de estas herramientas puede concluirse que, en su mayoría son privativas o atadas a determinadas tecnologías o plataformas y carecen de funcionalidades importantes para la utilización de las mismas en la web actual, como el caso de un módulo multilenguaje y el almacenamiento de las trazas de los usuarios. Además de la obtención y ejecución de los tutores de forma independiente. Se hace necesario entonces desarrollar una herramienta de autor que, solucione estas deficiencias y cumpla con los requerimientos de esta variante de software.

Por otra parte estas herramientas brindan opciones específicas que serán tomadas como base para la propuesta de solución, como es el caso del módulo de demostración que permite al profesor dar distintas soluciones a un tutor determinado y el módulo de edición del grafo de comportamiento que permite adicionar las habilidades y los distintos niveles de ayuda por cada estado, ambos incluidos en CTAT; la posibilidad de exportar un tutor, presente en xPST y las funcionalidades de accesibilidad para los débiles visuales implementadas en CRODA.

[5] Del inglés Sharable Content Object Reference Model.

1.4. Metodología de desarrollo de software

Para cumplir el objetivo de esta investigación es imprescindible el uso de una metodología de desarrollo de software que controle cada uno de los procesos que lleva a cabo el equipo de trabajo.

El desarrollo de software es una tarea compleja y riesgosa que involucra un equipo de personas trabajando en conjunto, por lo que se hace necesario mantener un estricto control sobre los procesos de manera que se garantice la organización y coordinación de todo el trabajo. Un proceso de desarrollo de software es un conjunto de actividades para transformar los requerimientos de un usuario en un software, define quién está haciendo qué, cuándo y cómo para alcanzar un determinado objetivo.

Para garantizar la gestión del desarrollo de un sistema están definidas varias metodologías. Pero no solo se trata de regirse por una metodología sino de seleccionar la adecuada, cada una de ellas está definida para proyectos con características específicas, por lo que se debe analizar bien cuál de las existentes va a resultar más factible emplear, pues esta decisión influye directamente en el éxito de un proyecto.

Las metodologías de desarrollo se pueden enmarcar en dos grandes grupos, las llamadas metodologías tradicionales y las metodologías ágiles. Las tradicionales se centran especialmente en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir, las herramientas y notaciones que se usarán; son recomendadas para los proyectos de grandes dimensiones y con grandes equipos de desarrollo.

En tanto, las metodologías ágiles dan mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas, respondiendo además a los cambios que puedan surgir a lo largo del desarrollo del proyecto. Su selección depende de qué producto se desee desarrollar, de las dimensiones que tendrá el mismo, del tiempo que se disponga y del equipo de trabajo, entre otros factores (26).

Los autores del presente trabajo de diploma deciden utilizar una metodología ágil porque el equipo de desarrollo es pequeño y se trabajará junto a los interesados en el producto final. Se cuenta con poco tiempo para la realización de la herramienta y así mientras se desarrolla se generan solo los documentos necesarios. Las integraciones continuas, las entregas tempranas,

el desarrollo guiado por pruebas, la estandarización del código, las iteraciones cortas son los puntos de Extreme Programming que más satisfacen a las necesidades de la investigación.

Metodología Extreme Programming (XP)

XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en retroalimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios (27).

Kent Beck, el padre de XP, describe la filosofía de XP en (27) sin describir el ciclo de desarrollo. Posteriormente, otras publicaciones de experiencias se han encargado de realizar dicha tarea. A continuación se describe las cuatro fases: planificación, diseño, desarrollo y pruebas, para el desarrollo de la presente investigación.

1. **Planificación:** en esta fase del ciclo de desarrollo el cliente define a grandes rasgos las historias de usuario. Los programadores realizan una estimación del esfuerzo necesario de cada historia de usuario y se establece la prioridad de cada una de ellas. Se realiza el plan de entregas en función de los parámetros: tiempo de desarrollo ideal y grado de importancia para el cliente. En esta etapa el equipo de desarrollo elabora el plan de iteraciones. Todo el trabajo de la iteración es expresado en tareas de ingeniería, cada una de ellas es asignada a un programador como responsable, pero llevadas a cabo por la pareja de programadores. El equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto.
2. **Diseño:** en esta fase se realizan las tarjetas Cargo o Clase, Responsabilidad y Colaboración (CRC) permitiendo desprenderse del método de trabajo basado en procedimientos y trabajar con una metodología basada en objetos. Las tarjetas CRC permiten que el equipo completo contribuya en la tarea del diseño.
3. **Desarrollo:** primeramente se define que el código debe ser desarrollado siguiendo los estándares de desarrollo para facilitar su lectura y modificación por cualquier miembro del equipo de desarrollo. El desarrollo es realizado por dos personas que trabajan de forma conjunta en una computadora. De esta manera, se incrementa la calidad del software desarrollado sin afectar al tiempo de entrega. Este equipo posee unos conocimientos

similares en cuanto a la tarea que van a realizar, es decir, están aproximadamente al mismo nivel. Mientras uno de ellos se encarga de pensar la táctica con la que se va a abordar el problema, el otro se encarga de pensar las estrategias que permiten llevar dichas tácticas a su máximo exponente. Ambos roles son intercambiables. Para una mayor calidad en el código es utilizada las pruebas unitarias mientras se desarrolla la herramienta.

4. **Prueba**: uno de los pilares de XP es el proceso de pruebas, esta metodología anima a probar constantemente tanto como sea posible. Esto permite aumentar la calidad de los sistemas reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección. También permite aumentar la seguridad de evitar efectos colaterales no deseados a la hora de realizar modificaciones y refactorizaciones (28). En esta fase el equipo de desarrollo aplica las pruebas de aceptación.

Buenas prácticas XP

La mayoría de estas prácticas no son nuevas, sino que han sido reconocidas por la industria como las mejores prácticas durante años. En XP, dichas prácticas son llevadas al extremo para que se obtenga algo mejor que la suma de las partes.

1. **Planificación incremental:** la planificación nunca será perfecta, ya que varía en función de cómo varíen las necesidades del negocio. Por tanto, el valor real reside en obtener rápidamente un plan inicial, y contar con mecanismos que permitan conocer con precisión la situación actual del proyecto. Como es lógico, la planificación es iterativa: un representante del negocio decide al comienzo de cada iteración qué características concretas se van a implementar.
2. **Pruebas:** la ejecución automatizada de pruebas es un elemento clave de la XP. Existen tanto pruebas internos (o pruebas de unidad), para garantizar que el mismo es correcto, como pruebas de aceptación, para garantizar que el código hace lo que debe hacer.
3. **Programación en parejas:** nadie programa en solitario, siempre hay dos personas delante del ordenador. Esta es una de las características que más se cuestiona al comienzo de la adopción de la metodología XP dentro de un equipo, pero en la práctica se acepta rápidamente y de forma entusiasta. El hecho de que todas las decisiones las tomen al menos dos personas proporciona un mecanismo de seguridad enormemente valioso.

4. **Refactorización:** la refactorización no sólo sirve para mantener el código legible y sencillo: también se utiliza cuando resulta conveniente modificar código existente para hacer más fácil implementar una nueva funcionalidad.
5. **Diseño simple:** XP exige no tener la ilusión de que un diseño puede resolver todas o gran parte de las situaciones futuras: lo que parece necesario cambia con frecuencia, es difícil acertar a priori. XP define un "diseño tan simple como sea posible" como aquel que: pasa todos las pruebas, no contiene código duplicado, deja clara la intención de los programadores (enfatiza el qué, no el cómo) en cada línea de código y contiene el menor número posible de clases y métodos.
6. **Propiedad colectiva del código:** todo el mundo tiene autoridad para hacer cambios a cualquier código, y es responsable de ellos. Esto permite no tener que estar esperando a otros cuando todo lo que hace falta es algún pequeño cambio.
7. **Integración continua:** la programación extrema hace que la integración sea permanente, con lo que todos los problemas se manifiestan de forma inmediata y no en una fase de integración remota.
8. **Cliente en el equipo:** el cliente siempre está disponible para resolver dudas y para decidir qué se hace en cada momento, en función de los intereses del negocio. Debido a su inmersión dentro del equipo, el cliente obtiene información absolutamente realista del estado del proyecto.
9. ***Releases* pequeñas:** siguiendo la política de XP de dar el máximo valor posible en cada momento, se intenta liberar nuevas versiones de las aplicaciones con frecuencia. Estas deben ser tan pequeñas como sea posible, aunque deben resultar valiosas para el cliente.
10. **Semanas de 40 horas:** la programación extrema lleva a un modo de trabajo en que el equipo está siempre al 100%. Una semana de 40 horas en las que se dedica la mayor parte del tiempo a tareas que suponen un avance puede dar mucho de sí, y hace innecesario recurrir a sobreesfuerzos, excepto en casos extremos.
11. **Estándares de codificación:** para conseguir que el código se encuentre en buen estado y que todas las personas del equipo puedan modificar cualquier parte del código es imprescindible que el estilo de codificación sea consistente. Un estándar de codificación es necesario para soportar otras prácticas de la metodología XP.
12. **Uso de Metáforas:** la comunicación fluida es uno de los valores más importantes de la programación extrema, para conseguir que esto ocurra es imprescindible, entre otras cosas, utilizar el vocabulario del negocio. También es fundamental huir de definiciones abstractas. Dicho de otro modo, la metodología XP no pretende seguir la letra de la ley,

sino su espíritu. Dentro de este enfoque es fundamental buscar continuamente metáforas que comuniquen intenciones y resulten descriptivas, enfatizando el qué por delante del cómo.

Todas estas prácticas interactúan y se refuerzan mutuamente, de manera que el éxito del proceso estará dado por la puesta en marcha de todas en conjunto (27).

1.5. Herramientas y tecnologías para el desarrollo de software

Para la implementación de las funcionalidades el equipo de desarrollo seleccionó herramientas y tecnologías libres utilizadas para la programación web que permiten un desarrollo efectivo y robusto de las aplicaciones, por lo cual pueden ser ejecutadas en los principales sistemas operativos sin ninguna restricción. Por las características de la herramienta a desarrollar se optó por una aplicación web, ya que esta no necesita ser descargada, instalada y configurada en cada uno de los equipos, consumen menos recursos del hardware, pueden ser utilizada por múltiples usuarios al mismo tiempo y para actualizarlas el procedimiento se realiza desde un único servidor para todos los usuarios.

1.5.1. Lenguajes de programación

Un lenguaje de programación puede ser utilizado para controlar el comportamiento de una máquina, particularmente una computadora. Consiste en un conjunto de reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos (29).

En la actualidad existe un gran número de lenguajes de programación que dan soporte a las tendencias actuales para las aplicaciones web, entre los que se encuentran HTML, JavaScript, CSS, PHP, entre otros. Estos lenguajes pueden dividirse en dos grandes grupos, los lenguajes del lado del cliente y los lenguajes del lado del servidor.

Lenguaje del lado del cliente

Los lenguajes del lado del cliente son ejecutados en el navegador del usuario, por lo que su funcionamiento es independiente del servidor. En este caso se encuentran, entre otros, HTML5, CCS 3 y JavaScript, los cuales serán brevemente caracterizados en esta investigación por considerarse necesarios para la implementación de la herramienta.

HTML

HTML5 es la actualización de HTML, el lenguaje en el que es creada la web. Definido formalmente por un cuerpo de normas internacionales conocido como W3C[6], HTML5 consta de más de cien especificaciones que se relacionan con la nueva generación de tecnologías web. Es una plataforma de código abierto desarrollado en términos de derechos de licencia libre (30).

La utilización de HTML5 como uno de los lenguajes para el desarrollo de la herramienta, responde a las necesidades de manipulación y maquetación de los elementos visuales de la aplicación para definir las interfaces de los tutores, utilizando para ello las etiquetas provistas en sus especificaciones que permiten a la aplicación una interacción más potente. Estas etiquetas semánticas permiten a los desarrolladores diseñar bajo estándares y buenas prácticas de programación web.

CSS

CSS 3 es un lenguaje para definir el estilo o la apariencia de las páginas web, escritas con HTML o de los documentos XML[7]. CSS se creó para separar el contenido de la forma, a la vez que permite a los diseñadores mantener un control mucho más preciso sobre la apariencia de las páginas. La novedad más importante que aporta CSS 3 consiste en la incorporación de nuevos mecanismos para mantener un mayor control sobre el estilo con el que se muestran los elementos de las páginas, sin tener que recurrir a trucos, que a menudo complicaban el código de las web (31). En la aplicación se utilizará para la creación de los estilos de los componentes y el posicionamiento de los mismos dentro de la página.

JavaScript

JavaScript es un lenguaje de script multiplataforma, interpretado y que se utiliza para manipular los objetos del DOM[8]. Puede incluirse en cualquier documento y es compatible con HTML, permitiendo el desarrollo de interfaces de usuario mejoradas y páginas web dinámicas (32).

Se decide utilizar este lenguaje, ya que permite validar los campos de los formularios y las acciones de los usuarios antes de enviar las peticiones al servidor; este método no es totalmente seguro, pero agrega un valor de funcionalidad importante al software, evitando las peticiones innecesarias al servidor.

[6] Del inglés World Wide Web Consortium.
[7] Del inglés eXtensible Markup Language.
[8] Del inglés Document Object Model.

Lenguaje del lado del servidor

Los lenguajes del lado del servidor son aquellos que son procesados por el servidor y generan las páginas que son devueltas al cliente en cada petición. Además se encargan de manipular la información que persiste en la base de datos, la seguridad del sistema y en la mayoría de los casos realizan la lógica del negocio de la aplicación.

PHP

PHP[9] es un lenguaje de script interpretado en el lado del servidor utilizado para la generación de páginas web dinámicas, embebidas en páginas HTML y ejecutadas en el servidor. PHP no necesita ser compilado para ejecutarse. Tiene como ventaja que es fácil de aprender, es un lenguaje rápido, tiene licencia de código abierto, es un lenguaje multiplataforma, tiene una alta variedad de funciones y no requiere conocimientos de desarrollo a bajo nivel. Tiene capacidad de conexión con la mayoría de los manejadores de base de datos: MySQL, PostgreSQL, Oracle, MS SQL Server, entre otros. La mayor parte de su sintaxis ha sido tomada de C, Java y Perl con algunas características específicas (33). Para el desarrollo de la herramienta es utilizada la versión 5.4.12 de PHP.

Se decidió utilizar como lenguaje del lado del servidor a PHP 5.4 porque existe abundante documentación sobre él, los desarrolladores tienen experiencia en su uso, es un lenguaje multiplataforma, incluye gran cantidad de funciones y es de fácil acceso ya que es libre.

1.5.2. Framework de desarrollo

En la actualidad es una tendencia en el desarrollo de software la utilización de *frameworks* (o marcos de trabajo). Utilizarlos acelera el proceso de desarrollo al reutilizar el código ya existente, se promueven buenas prácticas de desarrollo con el uso de patrones y mejorar la seguridad de la aplicación. Un *framework* se puede considerar como una aplicación genérica incompleta y configurable (34).

Symfony

Symfony 2.3 es un *framework* HTTP de tipo Petición/Respuesta para desarrollar aplicaciones PHP. Facilita el desarrollo de las aplicaciones web, aumenta exponencialmente la productividad y ayuda a mejorar la calidad. Proporciona varias herramientas y clases encaminadas a reducir el

[9] Del inglés Hipertext Preprocessor.

tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. Cuenta con miles de páginas de documentación distribuidas en varios libros gratuitos y decenas de tutoriales (35).

Es un conjunto de componentes independientes desacoplados, cohesionados que resuelven con mayor facilidad los problemas comunes del desarrollo web. Entre ellos destacan algunos como el administrador de traducciones, administrador de formularios, el inyector de dependencias y el administrador de sesiones que serán usados en el desarrollo de esta aplicación.

Bootstrap

Bootstrap es un framework de código abierto cuyo objetivo es facilitar el desarrollo de aplicaciones o páginas web teniendo una colección de plantillas CSS, HTML y *plugins* JavaScript. Los diseños creados con Bootstrap son simples, limpios e intuitivos. Con sólo agregar algunas clases y las *etiquetas* correctas se pueden lograr casi sin esfuerzo grupos de botones, barras de navegación, *dropdowns*[10], formularios, etc., todo sin tener que escribir una línea de código CSS. Las aplicaciones que utilizan bootstrap adaptan la interfaz dependiendo del tamaño del dispositivo en el que se visualice de forma nativa, esto se denomina diseño adaptativo o *Responsive Design* (36).

En esta investigación es utilizado la versión 3.1 principalmente para el maquetado de la aplicación aunque se utilizan las demás funcionalidades que el mismo proporciona, pensando también en la compatibilidad con los distintos navegadores y dispositivos en que pueda ser utilizada la herramienta.

jQuery

JQuery es una biblioteca para el lenguaje JavaScript. Implementa una serie de clases que permite programar sin preocuparse del navegador, ya que funcionan de forma exacta en todas las plataformas más habituales. La biblioteca tiene licencia de código abierto que permite que siempre cuente con soporte constante y rápido, publicándose actualizaciones de manera periódica (37). La ventaja principal de jQuery es que es mucho más sencillo que sus competidores.

[10] Dropdowns: permite a los usuarios seleccionar uno o varios elementos de una lista predefinida.

Tiene una gran cantidad de *plugins* que pueden ser adicionados fácilmente, traduciéndose esto en un ahorro sustancial de tiempo y esfuerzo, entre ellos el jOrgChart que será usado en el desarrollo de la aplicación para la creación del grafo de comportamiento. Otra ventaja de jQuery sobre sus competidores como Flash es su excelente integración con AJAX[11] que permite cargar contenido sin tener que actualizar la página. Para el desarrollo de la solución es utilizada la versión 1.9 de jQuery.

Doctrine

Doctrine 2 es un ORM[12] para PHP 5.3.0+ que proporciona persistencia transparente de objetos PHP. Se sitúa en la parte superior de una poderosa capa de abstracción de base de datos DBAL[13]. La principal tarea de los ORM es la traducción transparente entre objetos (PHP) y las filas relacionales de la base de datos.

Una característica fundamental de Doctrine es la opción de escribir las consultas de base de datos en un dialecto SQL[14] propio orientado a objetos llamado DQL[15], inspirado en Hibernate HQL. Además DQL difiere ligeramente de SQL en que abstrae considerablemente la asignación entre las filas de la base de datos y objetos, permitiendo a los desarrolladores escribir poderosas consultas de una manera sencilla y flexible (38). En la herramienta este ORM es utilizado en la capa del modelo para persistir en la base de datos los objetos y las relaciones entre ellos.

PHPUnit

PHPUnit es un entorno para realizar pruebas unitarias en el lenguaje de programación PHP. Es un *framework* multiplataforma que se creó con la idea de corregir los errores en cuanto se detecten. Como todos los *frameworks* de pruebas unitarias, utiliza *assert*[16] para verificar que el comportamiento de una unidad de código es el esperado (39). En el desarrollo de la herramienta, fue utilizado PHPUnit en su versión 4.1.0 para realizar las pruebas unitarias, que tienen como objetivo aislar cada parte del programa y demostrar que las partes de forma individual son correctas.

[11] Del inglés Asynchronous JavaScript and XML.
[12] Del inglés Object-Relational Mapping.
[13] Del inglés DataBase Abstraction Layer.
[14] Del inglés Structured Query Language.
[15] Del inglés Doctrine Query Language.
[16] assert: validaciones de PHPUnit.

1.5.3. Sistema Gestor de Base de Datos

Un Sistema Gestor de Base de Datos (SGBD) es un conjunto de programas que permiten crear y mantener una Base de Datos, asegurando su integridad, confidencialidad y seguridad. Existen diferentes tipos de SGBD, pero los más conocidos y utilizados actualmente son PostgreSQL, Oracle y MySQL (40).

PostgreSQL

Cuenta con más de 15 años de desarrollo activo y una arquitectura probada que se ha ganado una sólida reputación de confiabilidad, integridad de datos y corrección. Funciona en los principales sistemas operativos, incluyendo Linux, UNIX y Windows. Tiene soporte completo para claves foráneas, uniones, vistas, disparadores y procedimientos almacenados (en varios idiomas). Se incluye la mayoría de tipos de datos de SQL y es compatible con el almacenamiento de objetos binarios, incluyendo imágenes, sonidos o vídeo.

PostgreSQL tiene como desventaja que la velocidad de gestión de bases de datos baja aunque se mantiene estable incluso si la base de datos aumenta mucho, no incluye herencia entre objetos ya que no existen y carece de un conjunto de herramientas que permitan una fácil gestión de los usuarios y de las bases de datos que contenga el sistema (41).

Oracle

Entre sus principales características se encuentran que el acceso a los datos es según los privilegios concedidos por el administrador y presenta sofisticados procedimientos para hacer copias de seguridad. Se puede asignar espacio en disco para almacenar datos y es posible acceder a estos usando software de otros fabricantes. El mayor inconveniente de Oracle es que si es mal configurado puede ser muy lento. Existen pocos libros buenos sobre asuntos técnicos referentes a la instalación y administración de Oracle (42). Es un producto vendido a nivel mundial, aunque la gran potencia que tiene y su elevado precio hacen que sólo se vea en empresas muy grandes y multinacionales, por norma general.

MySQL

MySQL es un sistema de gestión de bases de datos relacional, esta versión tiene licencia GPL[17] de la GNU[18], creada por la empresa sueca MySQL AB. Las principales características de este SGBD son las siguientes: aprovecha la potencia de sistemas multiprocesadores, gracias a su implementación multihilo, es rápido, fácil de instalar y configurar, cuenta con una infinidad de bibliotecas y herramientas que facilitan su uso, tiene capacidad de gestionar y almacenar grandes cantidades de datos, gran portabilidad entre sistemas y realiza gestión de usuarios y contraseñas, manteniendo un buen nivel de seguridad en los datos (43).

Cualquiera de estos sistemas gestores de bases de datos es apropiado para ser utilizado en la herramienta, pero para almacenar y gestionar los datos se decidió utilizar MySQL 5.6.12 porque puede ser ejecutado en una computadora con escasos recursos, es software libre y es una base de datos rápida.

1.5.4. Servidor web

Un servidor web es un programa que sirve para atender y responder a las diferentes peticiones de los navegadores, proporcionando los recursos que soliciten usando el protocolo HTTP[19] o el protocolo HTTPS (versión cifrada y autenticada) (44). La World Wide Web se ha hecho muy extensa a lo largo del tiempo y con esto se han creado servidores de Internet con licencia libre y propietarios. Para la selección del servidor web a utilizar en la investigación se han valorado los tres más utilizados (ver figura 10 en los anexos), estos son Apache, Nginx y el Internet Information Server (IIS).

IIS

El servidor IIS es un conjunto de servicios para servidores bajo Microsoft Windows, presenta alto rendimiento y fiabilidad pero debe ser usado bajo licencia. A partir de la versión 6 todos los procesos de IIS se ejecutan bajo una cuenta específica que tiene menos privilegios que una cuenta de administrador del sistema pero que aporta mayor seguridad (45).

Nginx

Nginx es un servidor web de código abierto que ha demostrado ser eficaz, ligero y muy potente. Hace uso de *sockets* asincrónicos. Un proceso por núcleo es suficiente para manejar miles de

[17] Del inglés General Public License.
[18] Del inglés GNU is Not Unix.
[19] Del inglés HyperText Transfer Protocol.

conexiones, lo que permite una carga de CPU[20] y de memoria mucho más ligera. Archivos de configuración fáciles de leer y ajustar (46).

Apache

Apache es uno de los servidores web más reconocidos a nivel mundial. Su gran popularidad se debe a múltiples características que brindan una serie de ventajas como:

- Compatible con varios sistemas operativos.
- Es una tecnología gratuita de código abierto.
- Es un servidor altamente configurable de diseño modular. Actualmente existen muchos módulos para Apache y permite su instalación facilitando el trabajo con este servidor web.
- Trabaja con gran cantidad de lenguajes como Perl, PHP, entre otras; teniendo todo el soporte que se necesita para tener páginas dinámicas.
- Permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. Es posible configurar Apache para que ejecute un determinado script cuando ocurra un error en concreto.
- Tiene una alta configurabilidad en la creación y gestión de *logs*. Apache permite la creación de ficheros *log* que facilita tener un mayor control sobre lo que sucede en el servidor (44).

Después de valorar estos aspectos de los servidores web, teniendo en cuenta que la instalación es libre, por su capacidad de configuración, robustez y estabilidad, el lenguaje de programación será PHP y el SGBD MySQL, ha sido seleccionado Apache 2.4.4 como el servidor web soporte de la aplicación. Además que es el más utilizado en el mundo y el equipo de desarrollo tiene experiencia en el uso del mismo.

1.5.5. Entorno de desarrollo integrado

Un IDE[21] es un programa compuesto por una serie de herramientas, tales como editor de texto, compilador, intérprete, depurador, sistema de ayuda para la construcción de interfaces gráficas de usuario, etc., que utilizan los programadores para desarrollar código. Esta herramienta puede

[20] Del inglés Central Processing Unit.
[21] Del inglés Integrated Development Environment.

estar pensada para su utilización con un único lenguaje de programación o bien puede dar cabida a varios de estos (47). Un buen IDE debe incluir las siguientes características:

- Soporte para diversos lenguajes de programación.
- Integración con sistemas de control de versiones.
- Reconocimiento de sintaxis.
- Extensiones y componentes para el IDE.
- Integración con *framework* populares.
- Depurador.
- Importar y exportar proyectos.
- Múltiples idiomas.
- Manual de usuarios y ayuda.

Para el desarrollo de la herramienta se decidió utilizar software libre, por lo que analizaremos dos de los IDE más utilizados en el sistema operativo Linux: Zend Studio y NetBeans.

Zend Studio

Zend Studio es uno de los IDE que agrupa todos los componentes necesarios para el ciclo de desarrollo de aplicaciones PHP, disponible para desarrolladores profesionales. Incluye un conjunto de herramientas de edición, depurado, análisis y optimización, lo que permite simplificar los proyectos complejos. Proporciona una serie de ayudas que pasan desde la creación y gestión de proyectos hasta la depuración de código. Consta de dos partes: las funcionalidades del cliente (contiene interfaz de edición y ayuda) y las del servidor (instala Apache y el módulo PHP o, en caso de que estén instalados, los configura para trabajar juntos en depuración). A pesar de todas estas ventajas se distribuye a través de una licencia comercial (48).

Netbeans

NetBeans IDE es una herramienta escrita en Java, por lo que puede ser utilizado desde cualquier sistema operativo que tenga la máquina virtual de Java. Es un IDE multilenguaje completo y modular que tiene soporte para Java SE, Java EE, Java ME y con vinculación de gran cantidad de módulos (*plugins*). Cuenta con depurador, perfilador, herramientas para refactorizaciones, completamiento de código, editores y herramientas para los lenguajes XML, HTML, CSS, PHP, JavaScript, etc. Cuenta con características visuales para el desarrollo web (49).

Se seleccionó como IDE para el desarrollo de la herramienta, Netbeans en su versión 7.4 debido a que posee un gran soporte para la edición en HTML5, además tiene un excelente balance entre una interfaz con múltiples opciones y completamiento de código del *framework* Symfony. Proporciona un rendimiento superior para los desarrolladores de PHP, proporcionando editores y herramientas integrales para sus tecnologías relacionadas, además de que Zend Studio tiene carácter privativo.

1.5.6. Herramienta CASE

Las herramientas CASE[22] son elementos indispensables para concebir una aplicación informática. El uso de las mismas reduce los gastos y el tiempo en el ciclo de desarrollo del software ya que permiten modelar fácilmente procesos y sistemas, con previo dominio de al menos un lenguaje de modelado soportado por la herramienta (50). Algunas de las herramientas CASE más populares son:

Rational Rose

Es una herramienta basada en modelos. Se integra con las bases de datos y los ambientes integrados de desarrollo. Todos sus productos dan soporte al lenguaje UML[23]. Esta herramienta constituye un entorno de modelado que permite generar código a partir de modelos para múltiples lenguajes (Rational Rose Enterprise). Ofrece un lenguaje de modelado común que agiliza la creación del software. Tiene como desventaja que no es un software libre, su entorno gráfico no es muy amigable para el usuario y necesita alta capacidad de procesamiento (51).

Visual Paradigm

Es una herramienta profesional de diseño que soporta el ciclo de vida completo del desarrollo de software. Considerada como muy completa y fácil de usar, con soporte multiplataforma y que proporciona excelente facilidades de interoperabilidad con otras aplicaciones. Soporta todos los diagramas UML, y los Diagramas Entidad–Relación. Tiene licencia gratuita y comercial. Es fácil de instalar y actualizar, además es compatible entre ediciones (50).

La principal razón por la que fue seleccionada Visual Paradigm 8.0 fue por la necesidad del uso de tecnologías libres durante el desarrollo de la herramienta de autor, además de su excelente integración con MySQL a la hora de generar el modelo de datos. Por otro lado es importante

[22] Del inglés Computer Aided Software Engineering.
[23] Del inglés Unified Modeling Language.

destacar que se integra con el entorno de desarrollo NetBeans 7.4 que es el seleccionado por el equipo de trabajo para la codificación del software.

1.6. Resumen de metodología, herramientas y tecnologías de desarrollo a utilizar

Para la selección de las herramientas a utilizar en el desarrollo de la investigación se tuvieron en cuenta aspectos como: la abundante documentación disponible, el reconocimiento con que cuentan a nivel internacional, su condición de software libre y la experiencia de los desarrolladores en el trabajo con ellas.

Tabla 1 Resumen de metodología, herramientas y tecnologías

Herramienta o tecnología	Nombre	Versión
Metodología de desarrollo	XP	-
Herramienta CASE	Visual Paradigm	8.0
Lenguaje de programación	PHP	5.4.12
Framework del lado del cliente	jQuery	1.9
	Bootstrap	3.1
Framework del lado del servidor	Symfony	2.3
	PHPUnit	4.1.0
Ambiente de desarrollo integrado	Netbeans IDE	7.4
Gestor de base de datos	MySQL	5.6.12
Servidor web	Apache	2.4.4

Conclusiones

- El equipo de desarrollo optó por los STI basados en la ejemplificación debido a que el tiempo de desarrollo de los mismo es considerablemente menor que las otras clasificaciones y son conductualmente indistinguibles, no precisan de conocimientos de técnicas de inteligencia artificial ni de programación y no se requiere de equipos de desarrollo multidisciplinarios.
- Las herramientas de autor estudiadas presentan aspectos que pueden ser mejorados como la internacionalización, la obtención de los tutores de forma independiente y el almacenamiento de las trazas de los usuarios.
- Las herramientas y tecnologías seleccionadas para el desarrollo son libres, cumpliendo así con la política de la universidad y del país de potenciar el uso del software libre.
- Luego de concluir con la fundamentación teórica quedan sentadas las bases para definir las características del sistema a desarrollar.

Capítulo 2 Descripción de la propuesta de solución

En XP la gestión de requisitos es extremadamente simple, el cliente escribe y prioriza las historias de usuario que expresan las necesidades del sistema. Los programadores estiman el esfuerzo asociado y las dependencias entre ellas. Para planificar el trabajo desde el punto de vista técnico las historias de usuario son divididas en tareas para las cuales también se realiza una estimación.

2.1. Descripción del sistema propuesto

La herramienta de autor propuesta tiene como objetivo la creación de STI basados en la ejemplificación mediante la técnica de programación por demostración. Esta herramienta de autor puede ser utilizada en la creación de tutores relacionados con cualquier dominio del conocimiento y nivel de dificultad. Los usuarios podrán desempeñar 3 roles: profesor, estudiante y administrador.

Inicialmente todos los usuarios cuando se registran tienen el rol de estudiante, solo un administrador puede asignar roles de profesor y administrador. El estudiante solo tendrá acceso a las funcionalidades de mostrar, ejecutar y exportar los tutores creados en la herramienta. El profesor es el encargado de la creación, edición, demostración, exportación e importación de los tutores, así como de la revisión de las trazas generadas por el comportamiento de los usuarios en la herramienta y la creación y edición de las materias. Un administrador tendrá acceso a todas las funcionalidades de la herramienta incluyendo la gestión de usuarios.

El sistema permite visualizar los contenidos en dos idiomas, Español e Inglés, exceptuando las plantillas generadas por el profesor e importadas a la herramienta, que mantendrán el idioma con el que fueron creadas. Para mejorar la accesibilidad y usabilidad del sistema se puede aumentar o disminuir el tamaño de la letra según la necesidad de cada usuario.

La herramienta cuenta con un sistema de trazas (ver figura 11 en Anexos) para registrar el comportamiento de los usuarios con la misma. Estas trazas pueden ser consultadas por los profesores o administradores, brindándole una retroalimentación del trabajo realizado con los tutores. Esta retroalimentación puede ser usada por los profesores, para hacer énfasis en los elementos que los estudiantes presentan mayores dificultades o para mejorar los tutores realizados y agregar más niveles de ayuda en los pasos de la solución con mayor dificultad.

Los tutores realizados con esta herramienta estarán compuestos por una interfaz de usuario de tipo página web elaborada previamente en una aplicación externa. La interfaz debe nombrarse index.html, seguir los estándares de HTML5 y estar comprimida en un archivo *zip* junto con todos los elementos que necesite para ejecutarse, como medias, archivos javascript y archivos css. El otro componente de un tutor es la solución del mismo, que no es más que el grafo de comportamiento autogenerado a partir de las soluciones y los *hints* brindados por el profesor en el módulo de demostración de la herramienta.

El módulo de demostración tiene dos secciones: la vista y la solución. En la vista (figura 1), el profesor interactúa con la interfaz definida anteriormente y a partir de esas interacciones la herramienta graba un grafo con las soluciones. La sección de solución (figura 2) brinda un grupo de funcionalidades, mencionadas a continuación, que permiten que la solución sea construida sin programar:

- Muestra el grafo de comportamiento de la solución del tutor.
- Permite añadir los mensajes de retroalimentación de cada nodo correspondientes con acciones correctas, incorrectas y solicitudes de ayuda.
- Define las soluciones alternativas que se pueden generar a partir de un paso.
- Generaliza acciones del estudiante pertenecientes a un paso dentro de la solución.
- Define la habilidad a la que corresponde el paso y el tipo de paso: correcto, incorrecto u opcional.

La solución generada en el módulo de demostración es interpretada en el módulo de ejecución a partir de la secuencia que a continuación se explica:

1. El estudiante realiza una acción sobre la interfaz.
2. Se busca si es posible igualar la acción y el valor introducido con un nodo adyacente al actual.
3. Si no encuentra coincidencias con los nodos adyacentes o coincide con un nodo de tipo incorrecto, la acción ejecutada por el estudiante es señalada como incorrecta.
4. Cada paso puede tener o no asociado niveles de ayuda previamente definidos por el profesor.
5. Cada acción realizada dentro del tutor tendrá una retroalimentación indicándole al estudiante si es correcta o incorrecta.

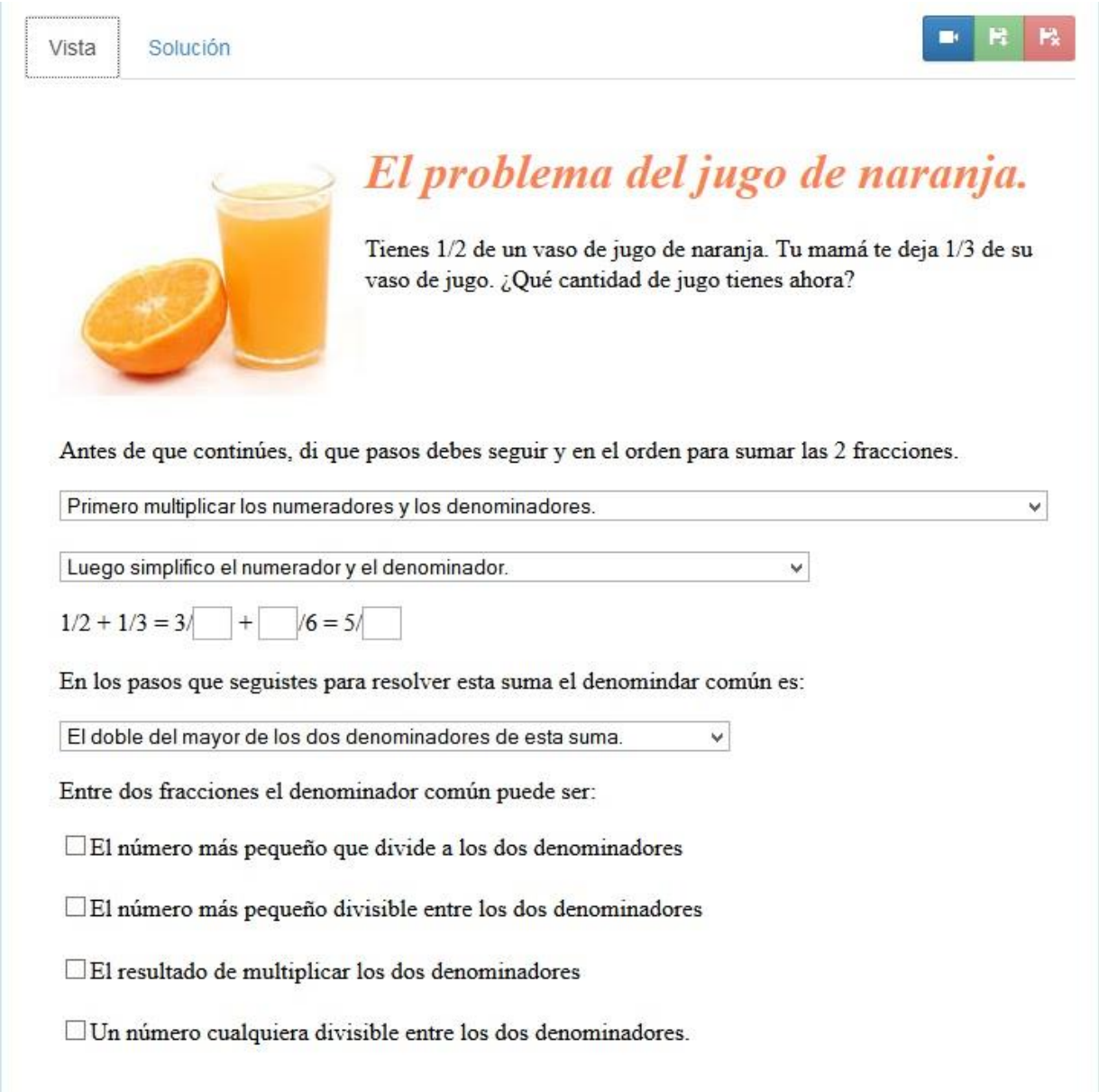


Figura 1 Área de la vista en el módulo de demostración

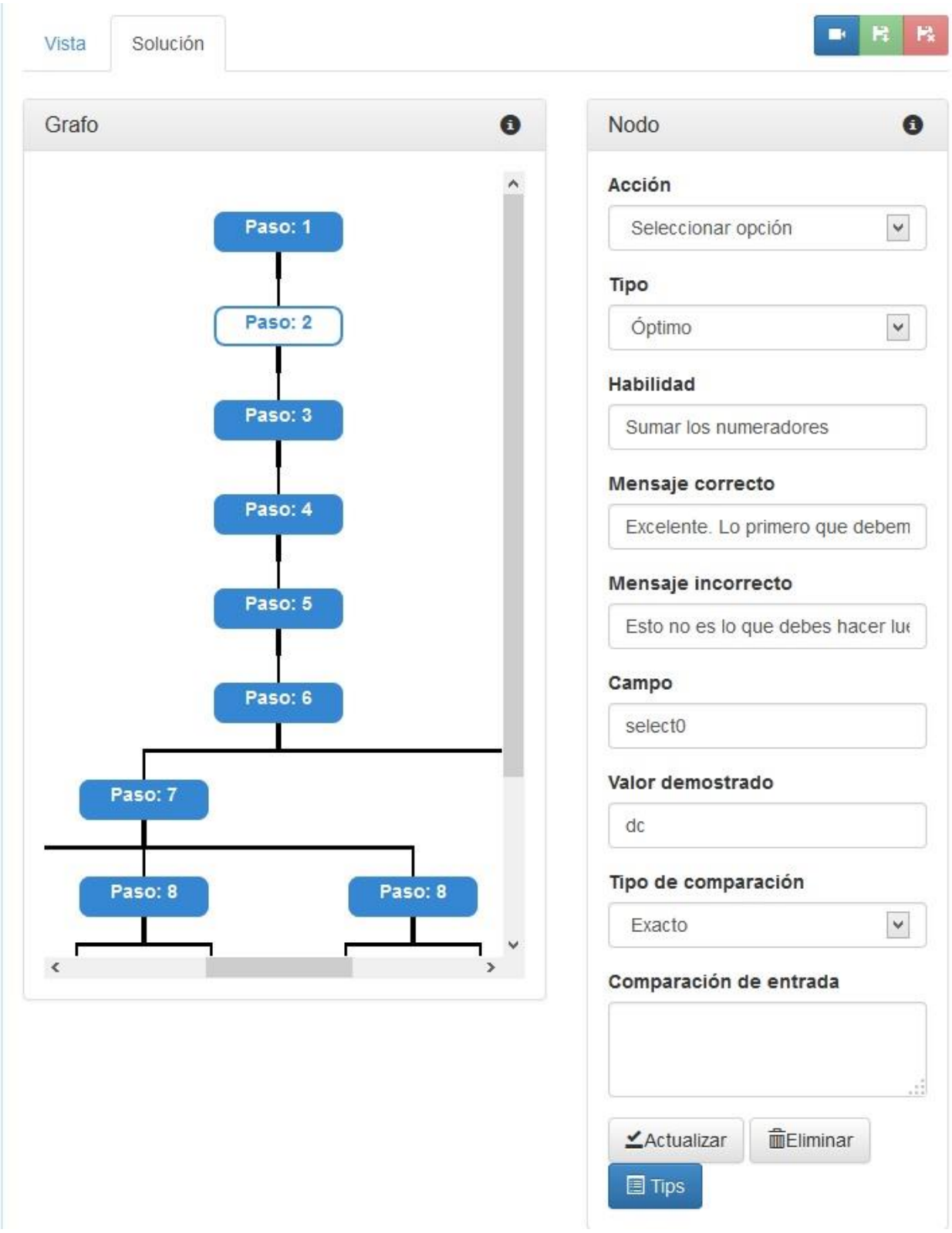


Figura 2 Área de la solución en el módulo de demostración

Los tutores además de ser ejecutados dentro de la herramienta, pueden ser exportados y ejecutados en cualquier entorno que cuente con un navegador web. Cuando un tutor es exportado se genera un archivo *zip* con la siguiente estructura:

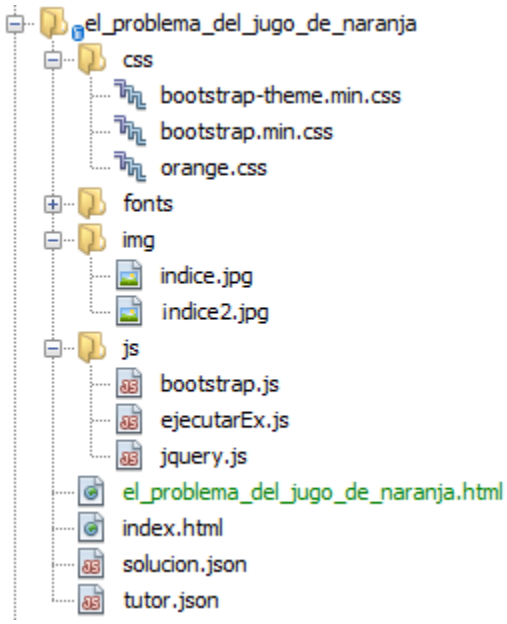


Figura 3 Estructura del tutor exportado

Dentro del tutor exportado se encuentran el bootstrap y el jQuery, elementos necesarios para la correcta ejecución y visualización del mismo, además de los archivos correspondientes con la interfaz del tutor. Se encuentran también dos archivos: tutor.json y solucion.json, el primero guarda los datos referentes al tutor y posibilita que pueda ser importado por la herramienta y el segundo contiene el grafo de solución generado junto con las anotaciones y los mensajes definidos por el profesor. El archivo ejecutarEx.js es el encargado de seguir e interpretar el comportamiento de los estudiantes, compararlo con el grafo de solución y brindar una respuesta.

2.2. Fase de planificación

En la fase de planificación se llevó a cabo la elaboración de las historias de usuario para obtener una descripción de las funcionalidades con que debía cumplir el sistema, se planificó el tiempo que duraría el desarrollo mediante un plan de iteración y finalmente se describieron las tareas de ingeniería.

Modelo conceptual

Para un mejor entendimiento de los conceptos asociados al negocio y sus relaciones, se decidió realizar un modelo conceptual porque a pesar de no exigirse o contemplarse dentro de la metodología XP, resulta de gran ayuda para el equipo de desarrollo.

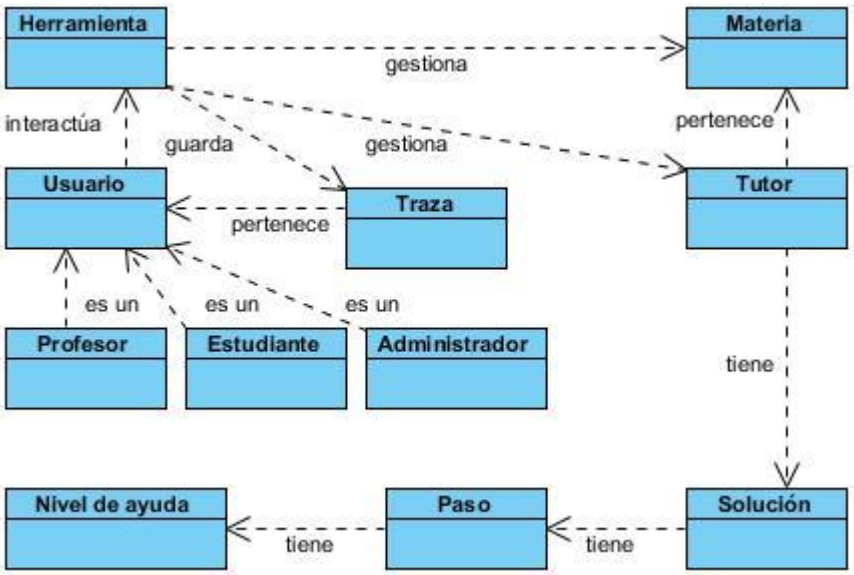


Figura 4 Modelo conceptual

Herramienta: en la herramienta de autor se pueden crear Sistemas Tutores Inteligentes basados en la ejemplificación.

Materia: se encarga de etiquetar y agrupar a los tutores que comparten una misma área del conocimiento.

Tutor: es el producto final del trabajo con la herramienta, es la unión de la vista creada previamente con otras herramientas y la demostración brindada por el profesor.

Solución: contiene todos los pasos correctos e incorrectos demostrados por el profesor, y el comportamiento del estudiante será comparado con ella.

Paso: un paso dentro de la solución es una acción que puede ser realizada en el tutor, se clasifican en tres tipos y contienen la habilidad a la que corresponde el paso y la acción que realizó el usuario.

Niveles de ayuda: un nivel de ayuda es un contenido que se le muestra al usuario para que pueda continuar en la solución del tutor si no sabe qué hacer. Un paso puede tener más de un nivel de ayuda.

Traza: guarda un registro del comportamiento de los estudiantes para revisar su interacción con la herramienta y los tutores.

Usuario: los usuarios relacionados con la herramienta de autor son todas aquellas personas que van a interactuar con la aplicación, tanto para crear los STI como para ejecutarlos. El estudiante dispone solamente los permisos de lectura y ejecución de los tutores, así como la posibilidad de exportar, el profesor es el encargado de la creación, edición, demostración, exportación e importación de los tutores, así como de la revisión de las trazas de los estudiantes y la creación y edición de las materias, y el administrador dispone de todas las funciones administrativas de la herramienta, además de las que pueden hacer un profesor y un estudiante, gestiona los usuarios.

Historias de usuario

Las historias de usuario (HU) son utilizadas en XP para especificar los requerimientos del software. Se trata de tarjetas en las cuales el cliente describe brevemente las características que el sistema debe poseer. El tratamiento de las historias de usuario es muy dinámico y flexible, en cualquier momento las historias de usuario pueden reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas (52).

XP no propone ningún formato específico para las historias de usuario, deja los detalles a consideración del equipo de desarrollo. Teniendo en cuenta los datos necesarios para la planificación y estimación de las historias de usuario y las plantillas utilizadas en desarrollos anteriores (53) (54) , se utilizó la siguiente plantilla:

Tabla 2 Modelo propuesto para una historia de usuario

Historia de usuario	
Número: *número asignado a la HU.*	Nombre: *nombre de la HU.*
Usuario: *persona que realiza las acciones dentro de la HU.*	
Prioridad en negocio: *nivel de prioridad de la HU en el negocio.*	Puntos estimados: *contiene la estimación hecha por el equipo de desarrollo del tiempo de duración de la HU.*
Riesgo en desarrollo: *nivel de riesgo en caso de no realizarse la HU.*	Iteración asignada: *iteración en la que será desarrollada la HU.*

Descripción: *breve descripción de lo que realizará la HU.*
Observaciones: *información extra que se estime agregar para hacer más comprensible la HU.*

Puntos estimados: cuando el valor es 1 equivale a una semana ideal de trabajo. En la metodología XP está definida una semana ideal como 5 días hábiles trabajando 40 horas, es decir, 8 horas diarias. Por lo que cuando el valor de dicho atributo es de 0.5 equivale a 2 días y medio de trabajo, lo que se traduce en 20 horas.

Seguidamente se muestran las historias de usuario de la 6 a la 14 confeccionadas para el desarrollo de la herramienta de autor. Las restantes se pueden encontrar en los anexos del documento.

Tabla 3 HU Mostrar tutor

Historia de usuario	
Número: 6	Nombre: Mostrar tutor
Usuario: estudiante, profesor, administrador	
Prioridad en negocio: alta	Puntos estimados: 0.5
Riesgo en desarrollo: alta	Iteración asignada: 2
Descripción: se selecciona un tutor de la lista visualizada. Se muestra una vista con la descripción y los botones para ejecutar, demostrar, editar, listar, eliminar y exportar el tutor.	
Observaciones:	

Tabla 4 HU Listar tutor

Historia de usuario	
Número: 7	Nombre: Listar tutor
Usuario: estudiante, profesor, administrador	
Prioridad en negocio: alta	Puntos estimados: 0.5
Riesgo en desarrollo: alta	Iteración asignada: 2
Descripción: al hacer clic en la opción tutores se muestra el listado de tutores creados en la herramienta y la descripción de cada uno.	
Observaciones:	

Tabla 5 HU Adicionar tutor

Historia de usuario	
Número: 8	Nombre: Adicionar tutor
Usuario: profesor, administrador	
Prioridad en negocio: alta	Puntos estimados: 1
Riesgo en desarrollo: alta	Iteración asignada: 2

Descripción: al hacer clic en adicionar tutor se abre una vista solicitando los datos necesarios para crear el tutor: - Nombre - Descripción - Archivo - Materia Al concluir se muestra un mensaje confirmando que fue adicionado.
Observaciones:

Tabla 6 HU Modificar tutor.

Historia de usuario	
Número: 9	Nombre: Modificar tutor
Usuario: profesor, administrador	
Prioridad en negocio: alta	Puntos estimados: 1
Riesgo en desarrollo: alta	Iteración asignada: 3
Descripción: cuando se selecciona el tutor de una materia, se debe hacer clic en editar y aparecen los datos del tutor para ser editados: - Nombre - Descripción - Archivo - Materia Al concluir se muestra un mensaje confirmando que fue actualizado.	
Observaciones:	

Tabla 7 HU Eliminar tutor.

Historia de usuario	
Número: 10	Nombre: Eliminar tutor
Usuario: profesor, administrador	
Prioridad en negocio: alta	Puntos estimados: 1
Riesgo en desarrollo: alta	Iteración asignada: 3
Descripción: se selecciona el tutor y al hacer clic en eliminar se pregunta si se desea eliminar el tutor para evitar pérdida de datos accidentales.	
Observaciones:	

Tabla 8 HU Ejecutar tutor

Historia de usuario	
Número: 11	Nombre: Ejecutar tutor
Usuario: estudiante, profesor, administrador.	
Prioridad en negocio: alta	Puntos estimados: 3
Riesgo en desarrollo: alta	Iteración asignada: 4
Descripción: selecciona el tutor que desea ejecutar, se muestra una descripción del mismo y al hacer clic en ejecutar el tutor se muestra en una ventana. Al comenzar a interactuar el usuario con el tutor, son	

guardadas las trazas de su comportamiento en el mismo y cada acción llevada a cabo por el usuario es comparada con la solución brindada por el profesor en el módulo de demostración. Cada acción realizada tiene además una retroalimentación y tiene acceso siempre a los niveles de ayuda correspondiente al estado de la solución actual hasta que el usuario resuelva el tutor.
Observaciones:

Tabla 9 HU Demostrar tutor

Historia de usuario	
Número: 12	Nombre: Demostrar tutor
Usuario: profesor, administrador	
Prioridad en negocio: alta	Puntos estimados: 3
Riesgo en desarrollo: alta	Iteración asignada: 5
Descripción: dentro de la materia se selecciona el tutor que quiere ser demostrado. Al hacer clic en demostrar se muestra en una ventana con 2 pestañas (vista y solución). La pestaña vista contiene la interfaz del tutor para que el usuario haga las demostraciones del mismo y la pestaña solución contiene el grafo de solución generado a partir de la demostración del usuario. En la pestaña solución al seleccionar un nodo se personalizan los pasos de la solución clasificándolos en óptimo, opcional e incorrecto, se le define la habilidad a la que pertenece el paso y se le adicionan los niveles de ayuda. A cada paso se le define un mensaje para cuando es exitosa o errada la acción que realiza el usuario. El campo sobre el que se realiza la acción puede ser editado.	
Observaciones: el grafo de comportamiento que representa la solución es directo o acíclico.	

Tabla 10 HU Exportar tutor

Historia de usuario	
Número: 13	Nombre: Exportar tutor
Usuario: estudiante, profesor, administrador	
Prioridad en negocio: media	Puntos estimados: 1
Riesgo en desarrollo: alta	Iteración asignada: 6
Descripción: dentro de la materia se selecciona el tutor que quiere ser exportado y al hacer clic en exportar se guarda en la ruta especificada por el usuario.	
Observaciones: el tutor es exportado en un archivo comprimido.	

Tabla 11 HU Importar tutor

Historia de usuario	
Número: 14	Nombre: Importar tutor
Usuario: profesor, administrador	
Prioridad en negocio: media	Puntos estimados: 1
Riesgo en desarrollo: alta	Iteración asignada: 6
Descripción: dentro de una materia o en la lista de tutores al hacer clic en el botón de importar se muestra una ventana para seleccionar el tutor que va ser importado. Se muestra un mensaje, confirmando si fue importado el tutor o si hubo algún error.	

Observaciones: el tutor se encuentra en un archivo comprimido.

Plan de iteraciones

En la metodología XP, la creación del sistema se divide en iteraciones. La duración ideal de una iteración está entre una y tres semanas. Para cada una de las iteraciones el cliente establece un conjunto de HU que serán implementadas en cada iteración del sistema. Al final de cada iteración se realizan las pruebas de aceptación y la aplicación tendrá implementadas funcionalidades para dar cumplimiento a los objetivos propuestos (55).

Tabla 12 Plan de iteraciones

Iteración	Orden de las HU	Prioridad
1	Mostrar materias Listar materias Adicionar materias Modificar materias Eliminar materias	Alta
2	Mostar tutores Listar tutores Adicionar tutores	Alta
3	Modificar tutores Eliminar tutores	Alta
4	Ejecutar tutor	Alta
5	Demostrar tutor	Alta
6	Exportar tutor Importar tutor	Media
7	Guardar trazas Consultar trazas	Media
8	Consultar ayudas contextuales Internacionalización	Baja

El plan de duración de las iteraciones se realizó luego de tener el estimado en días que demoraba implementar cada HU. Se tuvo en cuenta además la prioridad que el cliente le asignó a cada historia y el nivel de complejidad que estas poseen.

Tabla 13 Plan de duración de las iteraciones

Iteración	Orden de las HU	Duración total (Semanas)
1	Mostrar materias Listar materias Adicionar materias Modificar materias Eliminar materias	3
2	Mostar tutores Listar tutores Adicionar tutores	2

3	Modificar tutores Eliminar tutores	2
4	Ejecutar tutor	3
5	Demostrar tutor	3
6	Exportar tutor Importar tutor	2
7	Guardar trazas Consultar trazas	2
8	Consultar ayudas contextuales Internacionalización	2

Plan de entregas (*Release* plan)

En el plan de entregas se realiza un cronograma de entregas donde el cliente establece la prioridad de cada HU, cuáles serán agrupadas para conformar una entrega y el orden de las mismas. En correspondencia, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas (55). En la tabla se muestra la versión de la herramienta al concluir cada iteración y si ya fue finalizada esa HU se muestra una F.

Tabla 14 Plan de entregas

Historia de usuario	1	2	3	4	5	6	7	8
Mostrar materia	V 1.0	F	F	F	F	F	F	F
Listar materia	V 1.0	F	F	F	F	F	F	F
Adicionar materia	V 1.0	F	F	F	F	F	F	F
Modificar materia	V 1.0	F	F	F	F	F	F	F
Eliminar materia	V 1.0	F	F	F	F	F	F	F
Mostar tutor	-	V 1.1	F	F	F	F	F	F
Listar tutor	-	V 1.1	F	F	F	F	F	F
Adicionar tutor	-	V 1.1	F	F	F	F	F	F
Modificar tutor	-	-	V 1.2	F	F	F	F	F
Eliminar tutor	-	-	V 1.2	F	F	F	F	F
Ejecutar tutor	-	-	-	V 1.3	F	F	F	F
Demostrar tutor	-	-	-	-	V 1.4	F	F	F
Exportar tutor	-	-	-	-	-	V 1.5	F	F
Importar tutor	-	-	-	-	-	V 1.5	F	F
Guardar trazas	-	-	-	-	-	-	V 1.6	F
Consultar trazas	-	-	-	-	-	-	V 1.6	F
Consultar ayuda contextual	-	-	-	-	-	-	-	V 1.7
Internacionalización	-	-	-	-	-	-	-	V 1.7

Tareas de ingeniería

Todas las HU se dividieron en tareas de ingeniería y se desarrollaron según como fueron planificadas en las iteraciones. A continuación se detallan las tareas de ingeniería propuestas para las HU correspondientes a las iteraciones 2, 3, 4 y 5, las demás pueden ser consultadas en los anexos.

Iteración 2:

Tabla 15 Tarea de ingeniería # 9

Tarea	
Número: 9	Número de HU: 6
Nombre: Implementar las clases, interfaces y formularios necesarias para la visualización de la entidad tutor.	
Tipo de tarea: desarrollo	Estimación: 1,5 días
Fecha de inicio: 13-1-2014	Fecha de fin: 14-1-2014
Programador responsable: Marlon Cabrera Torres – Yasiel Tejeda González	
Descripción: Implementar la clase Tutor y TutorRepository que serán empleadas por el ORM para el mapeo de las entidades y sus relaciones. Además implementar la interfaz show y el formulario de esta entidad. Paquete: Application\HatbeBundle\Entity. Clase: Tutor, TutorRepository, Arbol, Hint, Nodo Interfaz: Tutor\show.html.twig	

Tabla 16 Tarea de ingeniería # 10

Tarea	
Número: 10	Número de HU: 6
Nombre: Crear la ruta para la funcionalidad mostrar tutor.	
Tipo de tarea: desarrollo	Estimación: 1 día
Fecha de inicio: 14-1-2014	Fecha de fin: 15-1-2014
Programador responsable: Marlon Cabrera Torres – Yasiel Tejeda González	
Descripción: crear una ruta en el archivo tutor.yml para acceder a esta funcionalidad.	

Tabla 17 Tarea de ingeniería # 11

Tarea	
Número: 11	Número de HU: 7
Nombre: Construir la interfaz index y la funcionalidad listar tutor.	
Tipo de tarea: desarrollo	Estimación: 1,5 días
Fecha de inicio: 15-1-2014	Fecha de fin: 16-1-2014
Programador responsable: Marlon Cabrera Torres – Yasiel Tejeda González	
Descripción: implementar la funcionalidad listar tutor en el controlador TutorController y la interfaz index para listar los tutores. Interfaz: Tutor\index.html.twig	

Tabla 18 Tarea de ingeniería # 12

Tarea	
Número: 12	Número de HU: 7
Nombre: Crear la ruta para la funcionalidad listar tutor.	
Tipo de tarea: desarrollo	Estimación: 1 día

Fecha de inicio: 17-1-2014	Fecha de fin: 17-1-2014
Programador responsable: Marlon Cabrera Torres – Yasiel Tejeda González	
Descripción: crear una ruta en el archivo tutor.yml para acceder a esta funcionalidad.	

Tabla 19 Tarea de ingeniería # 13

Tarea	
Número: 13	Número de HU: 8
Nombre: Estudio de la libraría simple_html_dom.php	
Tipo de tarea: investigación	Estimación: 1 día
Fecha de inicio: 20-1-2014	Fecha de fin: 20-1-2014
Programador responsable: Marlon Cabrera Torres – Yasiel Tejeda González	
Descripción: estudio de la biblioteca simple_html_dom.php para comprender el funcionamiento de la misma y usarla en la validación del campo vista del tutor.	

Tabla 20 Tarea de ingeniería # 14

Tarea	
Número: 14	Número de HU: 8
Nombre: Implementar la interfaz new y la funcionalidad adicionar tutor.	
Tipo de tarea: desarrollo	Estimación: 3 días
Fecha de inicio: 21-1-2014	Fecha de fin: 23-1-2014
Programador responsable: Marlon Cabrera Torres – Yasiel Tejeda González	
Descripción: implementar la funcionalidad adicionar tutor en el controlador TutorController, la interfaz new para adicionar nuevos tutores y la clase Util.php que será usada en algunas funcionalidades sobre la entidad tutor como lo es adicionar. Interfaz: Tutor\new.html.twig	

Tabla 21 Tarea de ingeniería # 15

Tarea	
Número: 15	Número de HU: 8
Nombre: Crear la ruta para la funcionalidad adicionar tutor.	
Tipo de tarea: desarrollo	Estimación: 1 día
Fecha de inicio: 24-1-2014	Fecha de fin: 24-1-2014
Programador responsable: Marlon Cabrera Torres – Yasiel Tejeda González	
Descripción: crear una ruta en el archivo tutor.yml para acceder a esta funcionalidad.	

Iteración 3:

Tabla 22 Tarea de ingeniería # 16

Tarea	
Número: 16	Número de HU: 9 , 10
Nombre: Implementar la interfaz edit y la funcionalidades para modificar un tutor.	

Tipo de tarea: desarrollo	Estimación: 7 días
Fecha de inicio: 27-1-2014	Fecha de fin: 4-2-2014
Programador responsable: Marlon Cabrera Torres – Yasiel Tejeda González	
Descripción: implementar las funcionalidades editar y eliminar tutor en el controlador TutorController y la interfaz edit para modificar o eliminar un tutor seleccionado. Interfaz: Tutor\edit.html.twig	

Tabla 23 Tarea de ingeniería # 17

Tarea	
Número: 17	Número de HU: 9, 10
Nombre: Crear las rutas para las funcionalidades modificar y eliminar tutor.	
Tipo de tarea: desarrollo	Estimación: 3 días
Fecha de inicio: 5-2-2014	Fecha de fin: 7-2-2014
Programador responsable: Marlon Cabrera Torres – Yasiel Tejeda González	
Descripción: crear las rutas en el archivo tutor.yml para eliminar y modificar un tutor y otra para cuando ya ha sido modificado.	

Iteración 4:

Tabla 24 Tarea de ingeniería # 18

Tarea	
Número: 18	Número de HU: 11
Nombre: Implementar la interfaz ejecutar y la funcionalidad para ejecutar un tutor.	
Tipo de tarea: desarrollo	Estimación: 10 días
Fecha de inicio: 10-2-2014	Fecha de fin: 21-2-2014
Programador responsable: Marlon Cabrera Torres – Yasiel Tejeda González	
Descripción: implementar la funcionalidad ejecutar un tutor en el controlador TutorController y la interfaz ejecutar para resolver un tutor seleccionado. Interfaz: Tutor\ejecutar.html.twig	

Tabla 25 Tarea de ingeniería # 19

Tarea	
Número: 19	Número de HU: 11
Nombre: Implementar los scripts para la ejecución del tutor.	
Tipo de tarea: desarrollo	Estimación: 2 días
Fecha de inicio: 24-2-2014	Fecha de fin: 25-2-2014
Programador responsable: Marlon Cabrera Torres – Yasiel Tejeda González	
Descripción: implementar los dos scripts utilizados para la ejecución del tutor, uno para la ejecución del tutor dentro de la herramienta y otro para cuando el tutor fue exportado y se está ejecutando independiente.	

Tabla 26 Tarea de ingeniería # 20

Tarea

Número: 20	Número de HU: 11
Nombre: Crear las rutas para la funcionalidad ejecutar tutor.	
Tipo de tarea: desarrollo	Estimación: 3 días
Fecha de inicio: 26-2-2014	Fecha de fin: 28-2-2014
Programador responsable: Marlon Cabrera Torres – Yasiel Tejeda González	
Descripción: crear las rutas en el archivo tutor.yml para ejecutar un tutor.	

Iteración 5:

Tabla 27 Tarea de ingeniería # 21

Tarea	
Número: 21	Número de HU: 12
Nombre: Implementar la interfaz demostrar y la funcionalidad para demostrar un tutor.	
Tipo de tarea: desarrollo	Estimación: 5 días
Fecha de inicio: 3-3-2014	Fecha de fin: 7-3-2014
Programador responsable: Marlon Cabrera Torres – Yasiel Tejeda González	
Descripción: implementar la funcionalidad demostrar un tutor en el controlador TutorController y la interfaz demostrar para resolver un tutor seleccionado. Interfaz: Tutor\demostrar.html.twig	

Tabla 28 Tarea de ingeniería # 22

Tarea	
Número: 22	Número de HU: 12
Nombre: Estudio del *plugin* de Jquery jOrgChart.	
Tipo de tarea: investigación	Estimación: 1 día
Fecha de inicio: 10-3-2014	Fecha de fin: 10-3-2014
Programador responsable: Marlon Cabrera Torres – Yasiel Tejeda González	
Descripción: estudio del *plugin* jOrgChart para comprender el funcionamiento del mismo y usarlo para graficar la solución brindada por el profesor de un tutor determinado.	

Tabla 29 Tarea de ingeniería # 23

Tarea	
Número: 23	Número de HU: 12
Nombre: Implementar interfaces y funcionalidades complementarias para demostrar un tutor.	
Tipo de tarea: desarrollo	Estimación: 7 días
Fecha de inicio: 11-3-2014	Fecha de fin: 19-3-2014
Programador responsable: Marlon Cabrera Torres – Yasiel Tejeda González	
Descripción: implementar las funcionalidades de guardar demostración, cancelar demostración, editar nodo, eliminar nodo y gestionar hints. Paquete: Application\HatbeBundle\Controller Clase: ArbolController, NodoController, HintController, TutorController	

Interfaz: Nodo\edit.html.twig, Hint\edit.html.twig, Hint\listar.html.twig, Tutor\includes\grafo.html.twig, Tutor\includes\macroArbol.html.twig

Tabla 30 Tarea de ingeniería # 24

Tarea	
Número: 24	Número de HU: 12
Nombre: crear las rutas para la funcionalidad demostrar tutor.	
Tipo de tarea: desarrollo	Estimación: 2 días
Fecha de inicio: 20-3-2014	Fecha de fin: 21-3-2014
Programador responsable: Marlon Cabrera Torres – Yasiel Tejeda González	
Descripción: crear las rutas en el archivo tutor.yml para demostrar un tutor, en el archivo arbol.yml la ruta notificar, en el archivo nodo.yml las rutas para listar, editar y eliminar nodo y en el archivo hint.yml las rutas necesarias para crear los hints.	

2.3. Fase de diseño

Durante la fase de diseño se confeccionan las tarjetas clase-responsabilidad-colaborador (CRC) para la descripción de cada una de las clases entidades y las controladoras, se define la arquitectura, los patrones de diseño y se realiza el modelo de datos para tener mejor visión de la estructura del sistema.

Tarjetas CRC

Durante la fase de diseño se confeccionan las tarjetas clase-responsabilidad-colaborador (CRC) para la descripción de cada una de las entidades. El uso de este tipo de tarjetas es una técnica de modelado que permite identificar las clases, responsabilidades y colaboraciones. El objetivo es obtener un diseño simple, elegante y fácil de comprender por parte de los programadores (55). Seguidamente se muestran las tarjetas CRC de las clases controladoras, confeccionadas para el desarrollo de la herramienta de autor. Las restantes se pueden encontrar en los anexos del documento.

Tabla 31 Tarjeta CRC Clase SitioController

Nombre de la clase: SitioController	
Responsabilidades	**Colaboraciones**
estaticaAction loginAction loginBoxAction listarTrazasAction guardarTrazaAction buscarAction	Traza, Tutor, Usuario, Util

Tabla 32 Tarjeta CRC ArbolController

Nombre de la clase: ArbolController	
Responsabilidades	**Colaboraciones**
notificarAction createAction createCreateForm newAction showAction editAction createEditForm updateAction deleteAction createDeleteForm	Nodo, Arbol, ArbolType

Tabla 33 Tarjeta CRC MateriaController

Nombre de la clase: MateriaController	
Responsabilidades	**Colaboraciones**
indexAction listarAction createAction createCreateForm newAction showAction editAction createEditForm updateAction deleteAction createDeleteForm	Tutor, Materia, MateriaType

Tabla 34 Tarjeta CRC TutorController

Nombre de la clase: TutorController	
Responsabilidades	**Colaboraciones**
indexAction listarAction createAction createCreateForm newAction showAction editAction createEditForm updateAction deleteAction createDeleteForm demostrarAction ejecutarAction SolucionAction	Tutor, Arbol, Nodo, Hint, Util, Materia, TutorType

Tabla 35 Tarjeta CRC NodoController

Nombre de la clase: NodoController	
Responsabilidades	**Colaboraciones**

editAction createEditForm updateAction deleteAction createDeleteForm	Nodo,NodoType

Tabla 36 Tarjeta CRC HintController

Nombre de la clase: HintController	
Responsabilidades	**Colaboraciones**
listarAction creadoAction createAction createCreateForm editAction createEditForm updateAction deleteAction createDeleteForm	Nodo, Hint, HintType

Modelo de datos

La creación del modelo de datos constituye prioridad en la abstracción para la creación de la base de datos, siendo posible esto a través de un grupo de herramientas, las cuales describen los datos y las relaciones que existen entre ellos. Utilizando la herramienta CASE Visual Paradigm para UML se realizó el siguiente modelo de datos:

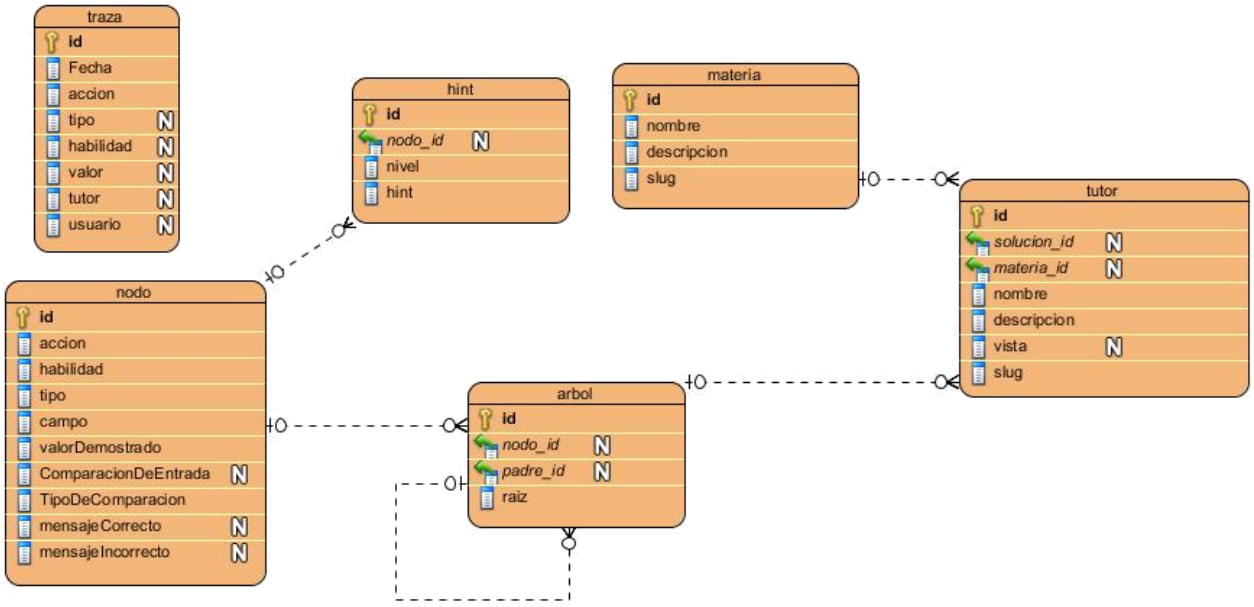


Figura 5 Modelo de datos

A continuación se muestra la descripción de las tablas que conforman el modelo de datos empleadas por la herramienta para su funcionamiento.

- La entidad “árbol” es la encargada de guardar la solución demostrada por el profesor a través del TDA Árbol donde cada nodo representa un paso de la solución del ejercicio. Un árbol tiene una relación de 1 a 1 con tutor, de 1 a mucho con ella misma y de 1 a 1 con nodo.
- La entidad “materia” se encarga de establecer una clasificación para los tutores. Cada materia tiene un nombre y una descripción. Tiene una relación de 1 a mucho con tutor.
- La entidad “tutor” almacena la información que describe a los tutores.
- La entidad “nodo” guarda los valores necesarios por cada paso de la solución como son: tipo de acción, acción, habilidad y los *hints*. Un nodo tiene muchos *hints*.
- La entidad “hint” guarda el texto a mostrar por cada vez que solicite 1 *hint* en el paso de la solución al que corresponde y el nivel que tiene. Tiene una relación de mucho a uno con tutor.
- La entidad traza guarda un registro de las acciones realizadas en la herramienta. Almacena la acción, habilidad, fecha, tipo, valor, el usuario que lo realizo y el tutor sobre el cual interactuó.

Arquitectura

La arquitectura de Symfony2 se centra en la popular arquitectura Modelo-Vista-Controlador (MVC), esta arquitectura es una de las más utilizada por los *frameworks* para PHP. Sin embargo, para su creador Symfony2 no es un *framework* MVC. Symfony2 sólo proporciona herramientas para la parte del Controlador y de la Vista. La parte del Modelo es responsabilidad del programador, aunque existen bibliotecas para integrar fácilmente los ORM[24] más conocidos, como Doctrine y Propel (56). Para el desarrollo de la herramienta de la presente investigación se utiliza Doctrine, debido a que el equipo de desarrollo tiene conocimiento con el trabajo de este ORM. Al ser utilizado el modelo la arquitectura para la investigación es MVC.

Las capas en la herramienta quedan de la siguiente forma:

El modelo: representa la información relacionada con el dominio de la aplicación, es decir, la lógica del negocio. Abstrae la lógica relacionada con los datos, haciendo que la vista y las

[24] Del inglés Object Relational Mapping.

acciones sean independientes del tipo de gestor de base de datos utilizado (34). En la solución se utiliza Doctrine, una biblioteca cuyo objetivo es brindar poderosas herramientas para el acceso a una base de datos relacional. El ORM Doctrine permite asociar objetos a la base de datos creada en MySQL. El modelo está compuesto por 7 entidades Arbol, Hint, Materia, Nodo, Traza, Tutor y Usuario.

La vista: transforma el modelo en interfaces de usuario permitiendo la interacción con el sistema. Solo se encarga de mostrar información. *Twig* es un motor y lenguaje de plantillas para PHP muy rápido y eficiente. Symfony2 recomienda utilizar *Twig* para crear todas las plantillas de la aplicación. La sintaxis de *Twig* se ha diseñado para que las plantillas sean concisas y muy fáciles de leer y de escribir (34). En la herramienta se utiliza la herencia a tres niveles para reutilizar el máximo código posible. En el primer nivel se encuentra una sola plantilla base de toda la aplicación llamada base.html.twig, en el segundo nivel una plantilla para los elementos que son comunes a todas las vistas llamada front.html.twig y en el tercer nivel, plantillas específicas para cada parte de la aplicación.

El controlador: se encarga de procesar las peticiones realizadas desde el navegador realizando los cambios necesarios en el modelo o en la vista. Este se encarga de aislar al modelo y a la vista de los detalles del protocolo utilizado para las peticiones (HTTP, consola de comando, etc.) (34). En la herramienta además del controlador frontal propio del *framework* existen 6 clases controladoras ArbolController, HintController, MateriaController, NodoController, SitioController y TutorController.

Es importante conocer cómo se aplican los principios fundamentales de la arquitectura MVC a las aplicaciones Symfony2 (34).

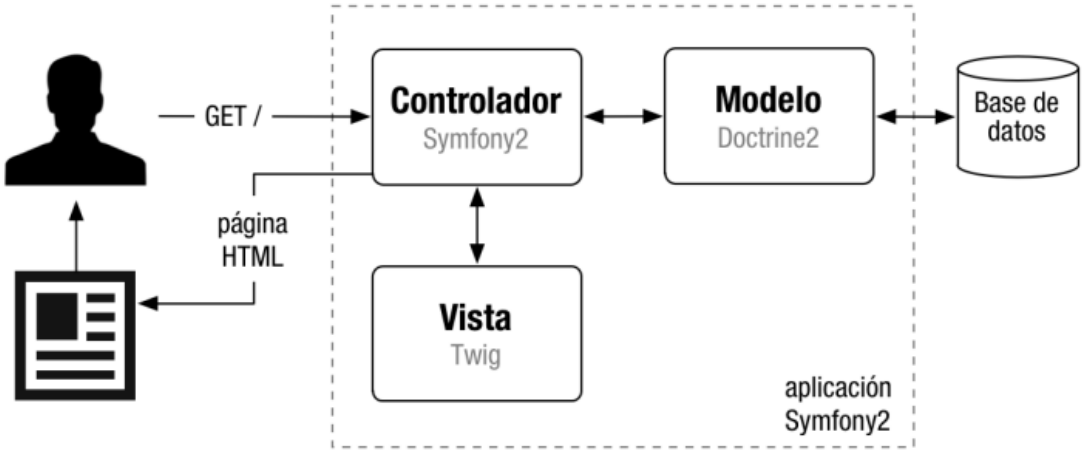


Figura 6 Funcionamiento interno de Symfony2

Cuando el usuario solicita ver una página de la herramienta, internamente sucede lo siguiente (34):

- El sistema de enrutamiento determina qué Controlador está asociado con la página solicitada.
- Symfony2 ejecuta el Controlador asociado. Un controlador no es más que una clase PHP en la que puedes ejecutar cualquier código que quieras.
- El Controlador solicita al Modelo los datos necesarios para mostrar. El modelo no es más que una clase PHP especializada en obtener información, normalmente de una base de datos.
- Con los datos devueltos por el Modelo, el Controlador solicita a la Vista que cree una página mediante una plantilla y que inserte los datos del Modelo.
- El Controlador entrega al servidor la página creada por la Vista.

A pesar de llegar a hacer cosas muy complejas con Symfony2, el funcionamiento interno siempre es el mismo: el Controlador manda y ordena, el Modelo busca la información que se le pide, la Vista crea páginas con plantillas y datos (34).

Patrones de diseño

Un patrón de diseño provee un diseño para refinar los subsistemas o componentes de un software, o la relación que existe entre ellos. Estos describen estructuras comúnmente recurrentes de componentes de comunicación que solucionan un problema general de diseño en

un contexto particular (57). El *framework* Symfony2 utiliza en su implementación un conjunto de patrones de diseño para dar solución a problemas específicos del diseño orientado a objetos durante el flujo de ejecución de una petición.

Alta cohesión:

La información que almacena una clase debe ser coherente y debe estar relacionada con la clase, asignando responsabilidades con una alta cohesión (58). La alta cohesión dice que cada clase debe contener toda la información (atributos) y funcionalidades (métodos) asociadas al objeto que representan. Siempre se ve en concordancia con el bajo acoplamiento porque si las clases modelan toda la información asociada a un objeto, significa que hay poca dependencia de otras clases externas para su correcto funcionamiento. En symfony2 se pone de manifiesto en las clases entidad.

Bajo acoplamiento:

El bajo acoplamiento es un principio que debemos recordar durante las decisiones de diseño: es la meta principal que es preciso tener siempre. Es un patrón evaluativo que el diseñador aplica al juzgar sus decisiones de diseño. Asigna una responsabilidad a una clase para mantener bajo acoplamiento, o sea, disminuir la dependencia entre clases, evitando que una modificación en alguna de ellas repercuta en gran medida en el resto, posibilitando además una mayor reutilización (58).

En la herramienta las clases TutorController y MateriaController heredan solamente de Controller, por lo que se garantiza que exista un grado moderado de acoplamiento entre las clases. Además, al no asociar las clases del modelo con las de la vista o el controlador, la dependencia entre las clases, en este caso, se mantiene baja.

Creador:

El patrón creador guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte al bajo acoplamiento (58).

En Symfony2 en la clase Controller se definen y ejecutan todas las acciones. En las acciones se crean los objetos de las clases que representan las entidades, evidenciando de este modo que la clase Controller es "creador" de dichas entidades.

En Symfony2 todas las peticiones web son manejadas por el controlador frontal, que es el punto de entrada único de toda la aplicación en un entorno determinado.

Decorador:

El propósito de este patrón es añadir responsabilidades adicionales a un objeto dinámicamente, proporcionando una alternativa flexible a la especialización mediante herencia, cuando se trata de añadir funcionalidades (59).

Este patrón añade funcionalidad a una clase de forma dinámica y transparente. En la aplicación el archivo base.html.twig o plantilla global almacena el código HTML que es común a todas las páginas de la herramienta, de ella hereda el archivo front.html.twig que es el que contiene el diseño general de la herramienta para no tener que repetirlo en cada página.

Patrón observador:

Un efecto muy frecuente en aquellos sistemas que se fragmentan en un conjunto de clases que cooperan, es la necesidad de mantener la consistencia entre los distintos objetos interrelacionados. Para no recurrir a soluciones fuertemente acopladas (que reducen la posibilidad de reutilización), este patrón define una dependencia “uno-a-muchos” entre objetos, para que, cuando uno de ellos cambie su estado, todos los que dependan de él sean avisados y puedan actualizarse convenientemente (59).

La utilización de este patrón se aprecia en el archivo jQuery.jOrgChart.js, al marcar un nodo el estado se cambia de normal ha seleccionado y se muestran sus atributos en el panel de edición del nodo.

Patrón fachada:

El propósito de este patrón es proporcionar una interfaz unificada de alto nivel que, representando a todo un subsistema, facilite su uso. La “fachada” satisface a la mayoría de los clientes, sin ocultar las funciones de menor nivel a aquellos que necesiten acceder a ellas (59).

El uso de este patrón GoF se puede observar en la utilización de la función de jQuery $post() del archivo edit.html.twig para eliminar el árbol del nodo seleccionado.

Conclusiones

- El estudio de la audiencia a la que va dirigida la solución delimitó que existen tres tipos de usuarios que interactúan con la herramienta con diferentes niveles de acceso, el estudiante, el profesor y el administrador.
- Se definieron ocho iteraciones que abarcan un total de 18 HU, que describen los aspectos principales a tener en cuenta para el desarrollo de la solución.
- Se construyó el plan de entregas que enmarcó el tiempo de desarrollo de las HU en 19 semanas, determinando un cronograma que especifica las entregas que deben hacerse.
- Las HU fueron divididas en un total de 37 tareas de ingeniería las cuales dieron paso a la implementación de la solución.
- Se realizó una descripción del sistema a través de las tarjetas CRC para un mejor entendimiento del mismo, se definió el modelo de datos, la arquitectura y los patrones de diseño.

Capítulo 3 Implementación y pruebas

Las pruebas de software forman parte de la última fase que propone XP, con el objetivo de lograr una herramienta que cumpla con cada uno de los requisitos previamente identificados. XP divide las pruebas de software o de sistema en dos grupos, las pruebas unitarias y las pruebas de aceptación. Las pruebas unitarias son las encargadas de verificar el código y son diseñadas por los programadores en la fase de codificación. Las pruebas de aceptación están destinadas a evaluar si al final de una iteración se consiguió la funcionalidad que se esperaba, estas pruebas usualmente son diseñadas por el usuario o cliente final (60). En el presente capítulo se describe la fase de codificación y la validación de la herramienta desarrollada.

3.1. Fase de codificación

En esta fase de codificación se definen los estilos de código y cómo se realizan las pruebas unitarias a la herramienta.

Pautas de codificación

XP enfatiza la comunicación de los programadores a través del código, por lo cual es indispensable que se sigan ciertas pautas de programación, estas mantienen el código legible para los miembros del equipo, facilitando los cambios. Para el desarrollo de la herramienta de autor objeto de esta investigación se definieron las siguientes pautas:

- Estilo de código lowerCamelCase en nombres de variables, funciones, métodos y argumentos.
- Utilizar espacios de nombres para todas las clases.
- Utilizar el sufijo *Interface* en las interfaces.
- Utilizar el sufijo *Exception* en las excepciones.
- Utiliza caracteres alfanuméricos y guiones bajos para los nombres de archivo.
- En la declaración de clases y métodos las llaves deben estar debajo.
- La indentación debe ser con un tabulador establecido a 4 espacios.
- Las constantes *true*, *false* y *null* deben ser escritos en minúsculas.
- El número de caracteres por línea deben ser de 80 columnas aunque se permiten hasta 120.
- Debe haber un espacio después de cada estructura de control (*if, for, foreach, while, switch, try...catch,* etc.).

Para llevar a cabo una buena implementación además de definir las pautas de codificación para un mejor entendimiento del código, es necesario realizar pruebas para tratar de minimizar errores en la aplicación. Estas pruebas son las conocidas pruebas unitarias y XP propone su realización durante la fase de codificación.

Pruebas unitarias

Las pruebas unitarias son una de las piedras angulares de XP. Todos los módulos del sistema deben de pasar las pruebas unitarias antes de ser liberados o publicados. Por otra parte, las pruebas deben ser definidas antes de realizar el código (*Test-driven programming*). Que todo código liberado pase correctamente las pruebas unitarias es lo que habilita la propiedad colectiva del código. En este sentido, el sistema y el conjunto de pruebas debe ser guardado junto con el código, para que pueda ser utilizado por otros desarrolladores, en caso de tener que corregir, cambiar o recodificar parte del mismo. Cuando se encuentra un error (*bug*), este debe ser corregido inmediatamente, y se deben tener precauciones para que errores similares no vuelvan a ocurrir. Asimismo, se generan nuevas pruebas para verificar que el error haya sido resuelto (61).

Uno de los errores que se suele cometer es pensar que se puede dejar la construcción de las pruebas para los últimos meses en la realización de un proyecto. Descubrir todos los errores que pueden aparecer lleva tiempo, y más si se deja la depuración de todos para el final. Las unidades de prueba están directamente relacionadas con el concepto de posesión del código.

Symfony2 ha optado por utilizar el *framework* PHPUnit, que prácticamente se ha convertido en un estándar en el mundo PHP. De esta forma, las pruebas unitarias de Symfony2 combinan la potencia de PHPUnit con las utilidades y facilidades proporcionadas por Symfony2 (34).

En la herramienta desarrollada como resultado de la presente investigación las pruebas unitarias se definen en la carpeta Test del *bundle*. Por convención de Symfony2, cada prueba unitaria se define en una clase cuyo nombre acaba en Test. Se utiliza la misma estructura de directorios del elemento que se quiere probar dentro de la carpeta Test. Fueron creadas 23 pruebas al comenzar el desarrollo y realizadas al finalizar cada funcionalidad.

A continuación se muestra un ejemplo del código de cómo fueron implementadas estas pruebas unitarias.

```
public function testValidarAccionNula() {
    $traza = new Traza();
    $traza->setFecha(new \DateTime());
    $traza->setAccion(null);
    $traza->setUsuario("user");

    $listaErrores = $this->validator->validate($traza);
    $this->assertEquals(1, $listaErrores->count(), 'Error validando la acción, no debe ser nula');
}
```

Figura 7 Implementación de la prueba unitaria del método testValidarAccionNula()

Al ejecutar las pruebas se muestra en consola los resultados de las mismas.

```
←[30;42mOK (21 tests, 21 assertions)←[0m

C:\wamp>php phpunit.phar -c www\Tesis\Applicacion\Hatbe\app www\Tesis\Applicacio
n\Hatbe\src\Application\HatbeBundle\Tests\Entity
PHPUnit 4.1.0 by Sebastian Bergmann.

Configuration read from C:\wamp\www\Tesis\Applicacion\Hatbe\app\phpunit.xml.dist


.←[41;37mF←[0m.................

Time: 310 ms, Memory: 9.25Mb

There was 1 failure:

1) Application\HatbeBundle\Tests\Entity\HintTest::testValidarHintNulo
Error validando el hint, no debe ser nulo
Failed asserting that 1 matches expected 0.

C:\wamp\www\Tesis\Applicacion\Hatbe\src\Application\HatbeBundle\Tests\Entity\Hin
tTest.php:37
←[37;41m                                                    ←[0m
←[37;41mFAILURES!                                           ←[0m
←[37;41mTests: 21, Assertions: 21, Failures: 1.←[0m
```

Figura 8 Resultado de ejecutar una prueba unitaria

3.2. Fase de pruebas

Una de las principales fortalezas de la metodología de desarrollo XP es el proceso de prueba, el cual se realiza de manera continua para asegurar durante todo el proceso de desarrollo, el éxito del producto que se está elaborando. Esto permite elaborar un software de más calidad donde los errores son detectados en un plazo de tiempo corto para que sean corregidos.

Pruebas de aceptación

Las pruebas de aceptación se crearán a partir de las historias de usuario. Una historia puede tener una o varias pruebas, según sea la funcionalidad a probar. El cliente es responsable de definir las pruebas de aceptación, estas son del tipo “caja negra”, en el sentido de que únicamente definen el resultado que debe tener el sistema ante unas entradas concretas. Las pruebas de aceptación que no tengan éxito generarán nuevas tareas para la próxima iteración, afectando así a la velocidad del proyecto, y proporcionando además una puntuación del éxito o fracaso de cada historia de usuario, o de cada equipo de trabajo (62). Para especificar dichos casos de prueba se propone utilizar la siguiente plantilla elaborada por el equipo de desarrollo, tomando como referencia la utilizada en (53):

Tabla 37 Modelo propuesto para una prueba de aceptación

Caso de prueba	
Número: *número que identifica el caso de prueba.*	Número HU: *número de historia de usuario correspondiente al caso de prueba.*
Nombre: *nombre de la HU a la que pertenece el caso de prueba.*	
Condiciones de ejecución: *condiciones necesarias para ejecutar la prueba.*	
Pasos de ejecución/entradas: *valores de entrada.*	
Resultado esperado: *salida de la ejecución.*	

Para validar la solución implementada se definen los siguientes casos de prueba de aceptación:

Tabla 38 CP # 1

Caso de prueba	
Número: 1	Número HU: 1
Nombre: Mostrar materia.	
Condiciones de ejecución:	
Pasos de ejecución/entradas: al hacer clic en la opción **materias** en el menú superior de la herramienta se visualiza el listado de materias y al hacer clic en la **materia** se muestra en una vista.	
Resultado esperado: se muestra en una vista la materia con su descripción y la lista de tutores con que cuenta.	

Tabla 39 CP # 2

Caso de prueba	
Número: 2	Número HU: 2
Nombre: Listar materia.	
Condiciones de ejecución:	
Pasos de ejecución/entradas: existen dos alternativas para listar las materias. En la primera, el usuario debe seleccionar la opción **materias** en el menú superior de la herramienta. En la segunda, si el usuario se encuentra en una materia específica, debe hacer clic en el botón **listar materias** en el menú inferior.	
Resultado esperado: se muestra en una vista el listado de las materias.	

Tabla 40 CP # 3

Caso de prueba

Número: 3	Número HU: 3
Nombre: Adicionar materia.	
Condiciones de ejecución: el usuario que tiene iniciada la sesión debe ser profesor o administrador.	
Pasos de ejecución/entradas: al hacer clic en la opción **materias** en el menú superior de la herramienta, se muestra el listado de materias, en la parte inferior, se accede al botón **nueva materia.**	
Resultado esperado: se muestra una vista que permite introducir los datos necesarios para adicionar una nueva materia: - Nombre - Descripción Al hacer clic en **crear** se muestra un mensaje confirmando que fue adicionada y si se hace clic en **cancelar** se regresa a la vista anterior.	

Tabla 41 CP # 4

Caso de prueba	
Número: 4	Número HU: 4
Nombre: Modificar materia.	
Condiciones de ejecución: el usuario que tiene iniciada la sesión debe ser profesor o administrador.	
Pasos de ejecución/entradas: al hacer clic en la opción **materias** en el menú superior de la herramienta se muestra el listado de materias, se debe hacer clic en el botón **editar** de la materia.	
Resultado esperado: se muestra una vista que permite modificar los datos de la materia: - Nombre - Descripción Al hacer clic en actualizar se muestra un mensaje confirmando que esta materia fue modificada.	

Tabla 42 CP # 5

Caso de prueba	
Número: 5	Número HU: 5
Nombre: Eliminar materia.	
Condiciones de ejecución: el usuario que tiene iniciada la sesión debe ser profesor o administrador.	
Pasos de ejecución/entradas: existen dos alternativas para eliminar una materia. En la primera, el usuario debe seleccionar la opción **materias** en el menú superior de la herramienta, se muestra el listado de materias, se accede al botón **editar** de la materia y en esta vista hacer clic en el botón **eliminar**. En la segunda, el usuario accede a una materia y debe hacer clic en el botón **eliminar** que se encuentra en la parte inferior.	
Resultado esperado: se muestra un mensaje de confirmación preguntando si desea eliminar la materia, si se hace clic en **aceptar** se elimina la materia y si hace clic en **cancelar** no es eliminada.	

Tabla 43 CP # 6

Caso de prueba	
Número: 6	Número HU: 6
Nombre: Mostrar tutor.	
Condiciones de ejecución:	
Pasos de ejecución/entradas: existen dos alternativas para mostrar los datos de un tutor. En la primera, el usuario debe seleccionar la opción **tutores** en el menú superior de la herramienta, se muestra el listado de tutores y de ellos hacer clic en un tutor. En la	

segunda, el usuario hace clic en una materia, se muestra en una vista la materia con los tutores que le corresponden y se selecciona un tutor.
Resultado esperado: se muestra en una vista el tutor con su id, nombre, descripción, la dirección de la vista, la solución y la materia. Se muestra un menú inferior con los botones de ejecutar, listar tutores y exportar

Tabla 44 CP # 7

Caso de prueba	
Número: 7	Número HU: 7
Nombre: Listar tutor.	
Condiciones de ejecución:	
Pasos de ejecución/entradas: existen dos alternativas para listar los tutores. En la primera, el usuario debe hacer clic en la opción **tutores**, en el menú superior de la herramienta. En la segunda, si el usuario se encuentra en un tutor en específico, hacer clic en el botón **listar tutores** que se encuentra en el menú inferior.	
Resultado esperado: se muestra en una vista el listado de tutores.	

Tabla 45 CP # 8

Caso de prueba	
Número: 8	Número HU: 8
Nombre: Adicionar tutor.	
Condiciones de ejecución: el usuario que tiene iniciada la sesión debe ser profesor o administrador.	
Pasos de ejecución/entradas: existen dos alternativas para adicionar los tutores. En la primera, debe seleccionar la opción **tutores** en el menú superior de la herramienta, se muestra una vista con el listado de tutores, en el menú inferior, se accede al botón **nuevo tutor**. En la segunda, si se encuentra en una materia, se puede añadir un tutor al hacer clic en el botón **nuevo tutor** que se encuentra en el menú inferior.	
Resultado esperado: se muestra una vista solicitando los datos necesarios para crear el tutor: - Nombre - Descripción - Archivo - Materia Al hacer clic en **crear** se muestra un mensaje confirmando que fue adicionado y si se hace clic en **cancelar** se regresa a la vista anterior.	

Tabla 46 CP # 9

Caso de prueba	
Número: 9	Número HU: 9
Nombre: Modificar tutor.	
Condiciones de ejecución: el usuario que tiene iniciada la sesión debe ser profesor o administrador.	
Pasos de ejecución/entradas: existen dos alternativas para modificar los tutores. En la primera, el usuario debe hacer clic en la opción **tutores** en el menú superior de la herramienta, se muestra el listado de tutores y se accede al botón **editar** del tutor. En la segunda, si se encuentra en una materia, se debe hacer clic en el botón **editar** del tutor.	
Resultado esperado: se muestra una vista que permite modificar los datos del tutor: - Nombre - Descripción - Archivo - Materia	

Al hacer clic en **actualizar** se muestra un mensaje confirmando que fue modificado y si se hace clic en **cancelar** regresa a la vista del tutor.

Tabla 47 CP # 10

Caso de prueba	
Número: 10	Número HU: 10
Nombre: Eliminar tutor.	
Condiciones de ejecución: el usuario que tiene iniciada la sesión debe ser profesor o administrador.	
Pasos de ejecución/entradas: existen dos alternativas para eliminar los tutores. En la primera, hacer clic en la opción **tutores** en el menú superior de la herramienta, se muestra el listado de tutores, se accede a un tutor y en esta vista hacer clic en el botón **eliminar**. En la segunda si se encuentra en una materia, muestra el listado de tutores y al acceder a un tutor hacer clic en el botón **eliminar** que se encuentra en el menú inferior.	
Resultado esperado: se muestra un mensaje de confirmación preguntando si desea eliminar el tutor, si hace clic en el botón **aceptar**, se elimina el tutor y si hace clic en el botón **cancelar**, no es eliminado.	

Tabla 48 CP # 11

Caso de prueba	
Número: 11	Número HU: 11
Nombre: Ejecutar tutor.	
Condiciones de ejecución: el usuario tiene que tener iniciada la sesión.	
Pasos de ejecución/entradas: existen dos alternativas para ejecutar un tutor. En la primera, el usuario al hacer clic en la opción **tutores** en el menú superior de la herramienta, se muestra el listado de tutores, se debe hacer clic en un tutor y en la vista del tutor hacer clic en el botón **ejecutar**. En la segunda, si se encuentra en una materia, se muestra el listado de tutores y al acceder a un tutor hacer clic en el botón **ejecutar** que se encuentra en el menú inferior.	
Resultado esperado: se muestra una vista con la interfaz del tutor lista para ser ejecutada por el estudiante, proporcionándole la retroalimentación y los niveles de ayuda proporcionados por el profesor en el módulo de demostración en la manera que el estudiante valla interactuando con el tutor.	

Tabla 49 CP # 12

Caso de prueba	
Número: 12	Número HU: 12
Nombre: Demostrar tutor.	
Condiciones de ejecución: el usuario que tiene iniciada la sesión debe ser profesor o administrador.	
Pasos de ejecución/entradas: existen dos alternativas para demostrar un tutor. En la primera, al seleccionar la opción **tutores** en el menú superior de la herramienta, se muestra el listado de tutores, se accede a un tutor y en esta vista dar clic en el botón **demostrar**. En la segunda, cuando se encuentra en una materia, muestra el listado de tutores y al acceder a un tutor en el menú inferior hacer clic en el botón **demostrar**.	
Resultado esperado: se muestra una vista con las pestañas vista y solución. La vista es donde se realiza el tutor y en la pestaña solución se guarda la información de cada nodo del árbol (la acción, el tipo, la habilidad, el mensaje correcto, el mensaje incorrecto, el campo, el valor demostrado, el tipo de comparación y la comparación de entrada). En la parte superior derecha aparecen los botones de demostrar el tutor, guardar y resetear la demostración.	

Tabla 50 CP # 13

Caso de prueba	
Número: 13	Número HU: 13
Nombre: Exportar tutor.	
Condiciones de ejecución: el usuario tiene que tener iniciada la sesión.	
Pasos de ejecución/entradas: existen dos alternativas para exportar un tutor. En la primera, al seleccionar la opción **tutores** en el menú superior de la herramienta, se muestra el listado de tutores, se accede a un tutor y en esta vista hacer clic en el botón **exportar**. En la segunda, si se encuentra en una materia, muestra el listado de tutores y al acceder a un tutor, hacer clic en el botón **exportar** que se encuentra en el menú inferior.	
Resultado esperado: en dependencia de la configuración del navegador, se guarda el tutor o se pregunta la dirección donde quiere guardarlo.	

Tabla 51 CP # 14

Caso de prueba	
Número: 14	Número HU: 14
Nombre: Importar tutor.	
Condiciones de ejecución: el usuario que tiene iniciada la sesión debe ser profesor o administrador.	
Pasos de ejecución/entradas: existen dos alternativas para importar un tutor. En la primera, al hacer clic en la opción **tutores** en el menú superior de la herramienta, se muestra el listado de tutores, hacer clic en el botón **importar** que se encuentra en el menú inferior. En la segunda, si se encuentra en una materia, se muestra el listado de tutores y al acceder a un tutor hacer clic en el botón **importar** que se encuentra en el menú inferior.	
Resultado esperado: se muestra un cuadro de diálogo para seleccionar el tutor que será importado. Se crea el tutor en la base de datos y se redirige a la vista de mostrar tutor.	

Tabla 52 CP # 15

Caso de prueba	
Número: 15	Número HU: 16
Nombre: Consultar trazas.	
Condiciones de ejecución: el usuario que tiene iniciada la sesión debe ser profesor o administrador.	
Pasos de ejecución/entradas: en el botón que contiene el nombre del usuario, hacer clic en **consultar trazas**.	
Resultado esperado: se muestra una vista con una tabla que contiene la fecha y hora, el usuario, el tutor, la acción, la habilidad y el valor que es guardado en la traza. Permite hacer filtros en las trazas.	

Tabla 53 CP # 16

Caso de prueba	
Número: 16	Número HU: 17
Nombre: Consultar ayuda contextual.	
Condiciones de ejecución: el usuario que tiene iniciada la sesión debe ser profesor o administrador.	
Pasos de ejecución/entradas: cuando se accede a la demostración de un tutor, en las distintas áreas de la vista se le pasa el mouse por encima.	
Resultado esperado: muestra la ayuda asociada al área de trabajo correspondiente o el campo a rellenar.	

Tabla 54 CP # 17

Caso de prueba	
Número: 17	Número HU: 18
Nombre: Internacionalización	
Condiciones de ejecución:	
Pasos de ejecución/entradas: en el banner de la herramienta hacer clic en el botón que aparece a la derecha con las siglas del que prefiera (inglés o español).	
Resultado esperado: todos los elementos, títulos y secciones en la herramienta aparecerán en el idioma seleccionado	

Resultados de las pruebas

La figura 6 presenta los resultados de las 8 iteraciones de pruebas. Se obtuvo un total de 17 no conformidades significativas y 16 no significativas.

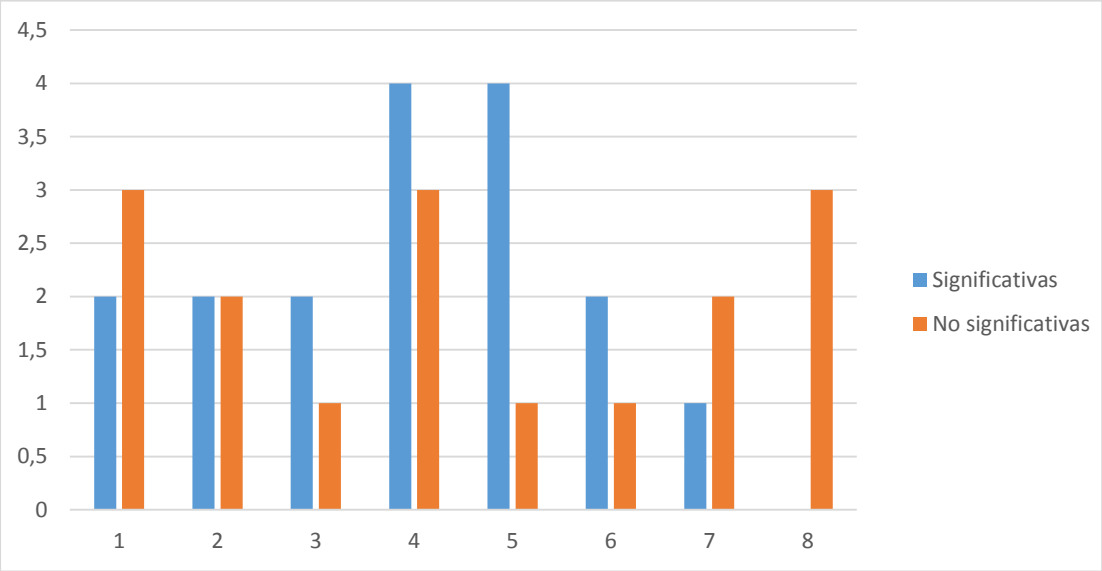


Figura 9 Resultados de las pruebas de aceptación

Las no conformidades (NC) no significativas, se centraron en errores ortográficos como: omisiones de tildes y cambio de mayúscula por minúscula y mensajes sin traducir al inglés. Las significativas consistieron en errores de validación, aceptar letras donde se esperaban valores numéricos, errores con la ejecución de tutores que incluyeran pasos opcionales y en la generación del tutor exportado. Una vez concluida cada iteración de pruebas el equipo de desarrollo analizó las NC obtenidas para solucionar los errores y mejorar la calidad del producto.

Conclusiones

- Se definieron las pautas de codificación y las pruebas unitarias para llevar a cabo el proceso de implementación del sistema.

- Al finalizar cada iteración fueron realizadas las pruebas de aceptación para fomentar la conformidad y seguridad del cliente con respecto al producto, se detectaron las no conformidades significativas y no significativas las cuales fueron resueltas por el equipo de desarrollo.
- Las pruebas aplicadas al sistema permitieron corregir a tiempo todas las no conformidades y garantizar un completamiento adecuado de cada uno de los componentes concebidos para la aplicación.

Conclusiones generales

- A partir de las herramientas de autor estudiadas se decidió incluir en la solución aspectos como la internacionalización, la obtención de los tutores de forma independiente y el almacenamiento de las trazas de los usuarios.
- La utilización de la metodología XP garantizó el éxito en el desarrollo del producto, generando la documentación necesaria para ser utilizada como consulta en la realización de nuevas versiones de la herramienta.
- El uso de las herramientas y tecnologías seleccionadas brindaron las prestaciones necesarias para la elaboración de un sistema robusto. También permitió cumplir con la política de soberanía tecnológica por la que proclama el país.
- Se obtuvo como resultado de esta investigación una herramienta de autor multilenguaje que permite la creación y ejecución de Sistemas Tutores Inteligentes basados en la ejemplificación, además de su exportación para ser difundidos y reutilizados en otros entornos.
- Las pruebas unitarias y de aceptación realizadas permitieron obtener un producto fiable y de calidad que cumple con las exigencias del cliente.

Recomendaciones

- Extender la herramienta para que sea capaz de construir Sistemas Tutores Inteligentes cognitivos basados en el modelo.
- Exportar el tutor de manera tal que pueda ser ejecutado sin el uso de un navegador.
- Agregar a la herramienta funcionalidades para construir y editar las interfaces del tutor de manera visual.

Bibliografía

1. **Rodríguez, José Palos.** Organización de Estados Iberoamericanos. [En línea] 2011. [Citado el: 1 de 11 de 2013.] http://www.oei.es/valores2/palos2.htm.

2. **Machinea, José Luis, Bárcena, Alicia y León, Arturo.** *Objetivos de Desarrollo del Milenio: una mirada desde América Latina y el Caribe.* 2005. ISBN 92-1-322741-8.

3. **MARQUÈS, P.** *METODOLOGÍA PARA LA ELABORACIÓN DE SOFTWARE EDUCATIVO.* 1995.

4. **Murray, T.** *Authoring Intelligent Tutoring Systems: An Analysis of the State of the Art International Journal of Artificial Intelligence in Education.*

5. **Koedinger, K.R., Aleven, V., Heffernan. T., Mclaren, B. & Hockernberry, M.** *Opening the Doors to Non-Programmers: Authoring Intelligent Tutor Behavior by Demonstration.* Proceedings of 7th Annual Intelligent Tutoring Systems Conference, moaceio, Brazil. : s.n., 2004.

6. **Aleven, Vincent, McLaren, Bruce M y Sewall, Jonathan .** *A New Paradigm for Intelligent Tutoring Systems: Example-Tracing Tutors.* Carnegie Mellon University, Pittsburgh, PA, USA : Human-Computer Interaction Institute, 2009.

7. *Los Sistemas Tutores Inteligentes y su impacto en la enseñanza de la programación.* **Altuna Castillo, Enrique .** Memorias del XV Congreso Internacional de Informática en la Educación, Informática 2013. : s.n., 2013. ISBN: 978-959-7213-02-4.

8. **Urretavizcaya Loinaz, M.** *Sistemas Inteligentes en el ámbito de la Educación.* s.l. : Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial. No.12 pp. 5-12, 2001. 1137-3601.

9. *Context Dependent Planning in a Machine Tutor.* **WOLF, B.** University of Massachusetts, Amherst, Massachusetts. : s.n., 1984.

10. **Beal, C. R., y otros.** *On-line tutoring for math achievement testing: A controlled evaluation.* Journal of Interactive Online Learning : s.n., 2007.

11. **Koedinger, K. R., Hadley, W. H. y Anderson, J. R.** *Intelligent tutoring goes to school in the big city.* s.l. : International Journal of Artificial Intelligence in Education., 1997.

12. **Corbett, A. T., Koedinger, K. R., & Anderson, J. R.** *Intelligent tutoring systems.* Amsterdam, The Netherlands: Elsevier Science B. V. : s.n., 1997.

13. **Anderson, J. R. y Lebiere, C.** *The atomic components of thought: Erlbaum.* 1998.

14. **Ohlsson, S.** *Learning from performance errors.* 1996.

15. **Shneiderman, Ben , Kaufmann, Morgan y Lieberman, Henry.** *Your Wish is My Command: Programming By Example.* 2001. 1-55860-688-2.

16. **McLaren, B. M., y otros.** *A new paradigm for intelligent tutoring systems: Example-tracing tutors.* s.l. : International Journal of Artificial Intelligence in Education., 2009.

17. **Dabbagh, N. H.** *Authoring tools and Learning Systems: A Historical Perspective.* 2001.

18. *Authoring tools for advanced technology learning enviroments. Toward cost-effective adaptive, interactive and intelligent educacional software.* **Murray, Tom.** Dordrecht, Netherlands : Kluwer Academic Publishers, 2003.

19. **DE LEEUWE, M.** Elearningsite.com. [En línea] 2002. [Citado el: 6 de 12 de 2013.] http://www.elearningsite.com/index.html.

20. **Hall, Brandon .** Authoring Tool Strategies. [En línea] 2001. www.brandonhall.com/public/execsums/authoring_tool_strategies.pdf. 09.

21. *GuiónEditor y HamWeb. Centro de Tecnología e Innovación (Teledomedia).* **CATALINA, C.** 2002.

22. **Zazpe, Pedro Razquin.** *Los Sistemas de Autor Multimedia.* Escuela Universitaria de Biblioteconornía y Documentación (EUBD) Universidad Complutense de Madrid. : s.n., 1998.

23. *HERRAMIENTAS DE AUTOR PARA ENSEÑANZA Y DIAGNÓSTICO: IRIS-D.* **Ferrero, B. y Arruarte, A. .** 12, s.l. : Inteligencia artificial: Revista Iberoamericana de Inteligencia Artificial, 2001, Vol. 5.

24. *The Extensible Problem-Specific Tutor (xPST): Evaluation of an API for Tutoring on Existing Interfaces.* **Stephen, GILBERT, Stephen , B. BLESSING y Sateesh , KODAVALI.** Estados Unidos : Virtual Reality Applications Center, Iowa State University, University of Tampa, 2009.

25. **Preval Cobas, Dianelys y Iturria Pozo, Jorge .** *Creación colaborativa de objetos de aprendizaje a partir de una herramienta de autor Web.* La Habana : s.n., 2010.

26. **Pressman, R. S.** *Ingeniería del Software. Un enfoque práctico.* 2002. 5ta parte Vol. 1..

27. **Beck, Kent.** *Extreme Programming Explained: Embrace Change.* s.l. : Addison-Wesley, 1999. ISBN 978-0321278654.

28. **ESCRIBANO, G. F.** *eXtreme Programming / Programación Extrema.* 2002.

29. **LOUDEN, Kenneth C.** *Lenguajes de programación: Principios y práctica. .* s.l. : Cengage Learning Editores, 2004.

30. **Mauri Pérez, Yolanda y de la Soledad García, Ernesto.** *Desarrollo del sistema de gestión de contenidos de la colección El Navegante en su versión multiplataforma.* La Habana : s.n., 2012.

31. **Alvarez, Miguel Angel .** *Manual de CSS 3.* 2011.

32. **Eguíluz Pérez, Javier.** *Introducción a JavaScript.* 2009.

33. **Sæther Bakken, Stig, y otros.** *Manual de PHP.* s.l. : Free Software Foundation, 2001.

34. **Eguiluz, Javier .** *Desarrollo web ágil con Symfony2.* 2011.

35. **Sitio oficial de Symfony.** http://www.symfony.es/. [En línea] 2010. [Citado el: 15 de 12 de 2013.]

36. **LERNER, Reuven M.** *At the forge: Twitter bootstrap.* Linux Journal, 2012 . vol. 2012, no 218, p. 6.

37. **Alvarez, Miguel Angel .** *Manual de jQuery.*

38. **SensioLabs.** *Doctrine ORM for PHP.* 2009.

39. **Sitio oficial de PHPUnit.** [En línea] http://www.phpunit.de/.

40. **CONNOLLY, Thomas M. y BEGG, Carolyn E.** *Sistemas de bases de datos. Un enfoque práctico.* 2005.

41. **Group., Postgres Global Development.** *Manual del usuario de PostgreSQL.* 1999.

42. **Sitio oficial de Microsoft Office Online.** [En línea] [Citado el: 30 de 11 de 2013.] http://office.microsoft.com..

43. **Sun Microsystems .** *MySQL 5.0 Reference Manual.* 2010.

44. **J. Kabir, Mohamemed.** *La biblia del Servidor Apache.*

45. **Domínguez-Dorado, M.** *Introducción a las aplicaciones Web con ASP e IIS.* España : Editorial Iberprensa (Madrid)., Julio, 2004.

46. **Nedelcu, Clément .** *Nginx HTTP Server.* Francia : s.n., 2013.

47. **Entornos de Desarrollo Integrado.** [En línea] [Citado el: 15 de 12 de 2013.] http://petra.euitio.uniovi.es/~i1667065/HD/documentos/Entornos%20de%20Desarrollo%20Integrado.pd.

48. **Crusat, Marc Vaquer.** *Zend Studio: el entorno de desarrollo para PHP.* 2004.

49. **Sitio oficial NetBeans .** [En línea] [Citado el: 15 de 12 de 2013.] https://netbeans.org/community/releases/74/.

50. **Sierra, María.** *Trabajando con Visual Paradigm for UML.* Cantabria : Universidad de Cantabria – Facultad de Ciencias. : s.n.

51. **Canales, Roberto.** *Arquitectura SOA e Integración de aplicaciones.* 2008.

52. **Jeffries, R., Anderson, A., Hendrickson, C.** *Extreme Programming Installed.* Addison-Wesley : s.n., 2001.

53. **Lara Rodriguez, Jose Ernesto y Hernandez Bernal, Yurisbel .** *Análisis, diseño e implementación de un IDE Online.* Universidad de las Ciencias Informáticas, La Habana : s.n., 2010.

54. **Rodríguez Corbea, Maite y Ordóñez Pérez, Meylin.** *La metodología XP aplicable al desarrollo del software educativo en Cuba.* Universidad de las Ciencias Informáticas, La Habana : s.n., 2007.

55. **Letelier, P. y Penadés, Mª C.** *Métodologías ágiles para el desarrollo de software:eXtreme Programming (XP).* Valencia : Universidad de Valencia : s.n., 2008. Vol. 05, 26. ISSN 1666-1680..

56. **Potencier, Fabien.** *What is Symfony2?* 2011.

57. **Buschmann, F., y otros.** *Pattern - Oriented Software Architecture.* 1996.

58. **Larman, Craig.** *Applying UML and patterns: an introduction to object-oriented analysis and design and iterative development.* New Jersey : Prentice Hall PTR Upper Saddle River, 2004.

59. **Facultad de informática, Universidad Politécnica de Madrid.** *Patrones del "Gang of Four".* 2005.

60. **Gutiérrez, JJ , y otros.** *Pruebas del sistema en Programación Extrema.* Deparatamento de Lenguajes y Sistemas Infromáticos. Universidad de Sevilla : s.n., 2006.

61. **Joskowicz, José.** *Reglas y Prácticas en eXtreme Programming.* 2008.

62. **Mas, Jordi , Aycart Pérez, David y Ginestà, Marc Gibert .** *Ingeniería del software en entorno de SL.* Barcelona, España : Eureca Media, SL, 2007.

63. **Estadísticas para la web.** [En línea] [Citado el: 1 de 12 de 2013.] http://w3techs.com/technologies/overview/web_server/all.

Anexos

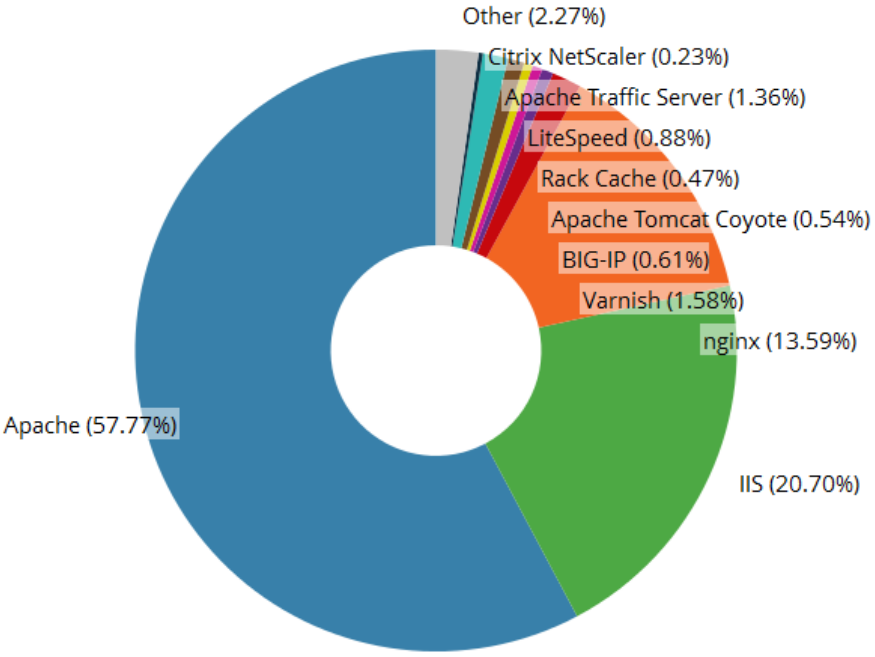
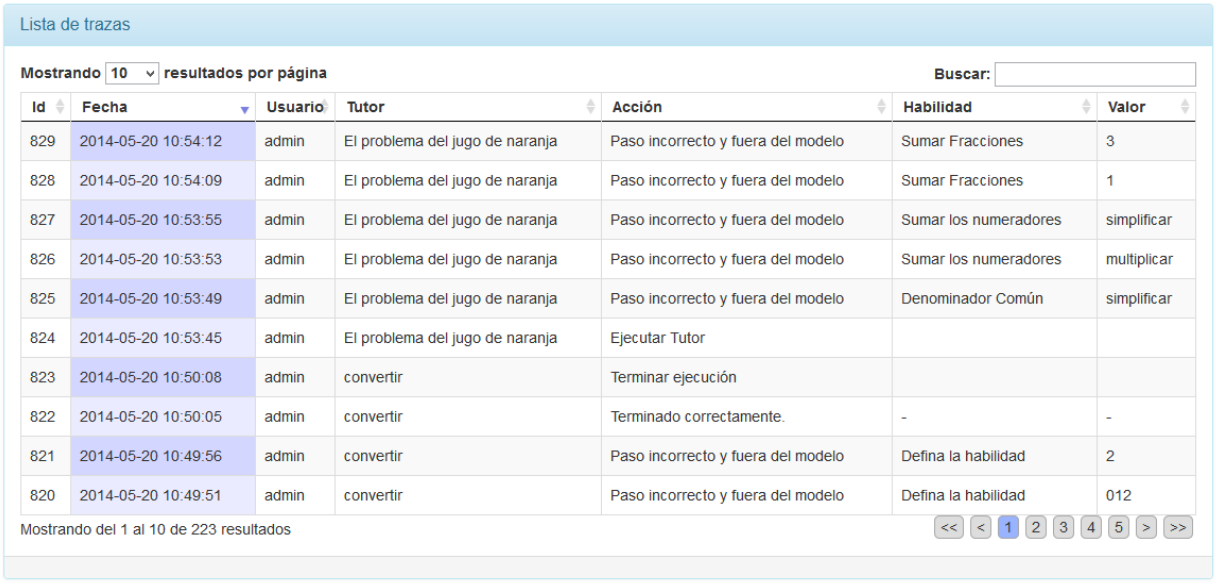


Figura 10 Estadísticas uso servidores web 2013 *(63)*



Lista de trazas

Mostrando 10 resultados por página — Buscar:

Id	Fecha	Usuario	Tutor	Acción	Habilidad	Valor
829	2014-05-20 10:54:12	admin	El problema del jugo de naranja	Paso incorrecto y fuera del modelo	Sumar Fracciones	3
828	2014-05-20 10:54:09	admin	El problema del jugo de naranja	Paso incorrecto y fuera del modelo	Sumar Fracciones	1
827	2014-05-20 10:53:55	admin	El problema del jugo de naranja	Paso incorrecto y fuera del modelo	Sumar los numeradores	simplificar
826	2014-05-20 10:53:53	admin	El problema del jugo de naranja	Paso incorrecto y fuera del modelo	Sumar los numeradores	multiplicar
825	2014-05-20 10:53:49	admin	El problema del jugo de naranja	Paso incorrecto y fuera del modelo	Denominador Común	simplificar
824	2014-05-20 10:53:45	admin	El problema del jugo de naranja	Ejecutar Tutor		
823	2014-05-20 10:50:08	admin	convertir	Terminar ejecución		
822	2014-05-20 10:50:05	admin	convertir	Terminado correctamente.	-	-
821	2014-05-20 10:49:56	admin	convertir	Paso incorrecto y fuera del modelo	Defina la habilidad	2
820	2014-05-20 10:49:51	admin	convertir	Paso incorrecto y fuera del modelo	Defina la habilidad	012

Mostrando del 1 al 10 de 223 resultados — << < 1 2 3 4 5 > >>

Figura 11 Interfaz de consultar trazas

Historias de usuario

Tabla 55 HU Mostrar materia

Historia de usuario

Número: 1	Nombre: Mostrar materia
Usuario: estudiante, profesor, administrador	
Prioridad en negocio: alta	Puntos estimados: 0.75
Riesgo en desarrollo: alta	Iteración asignada: 1
Descripción: se selecciona una materia de la lista visualizada. Se muestra una vista con la descripción y los tutores con que cuenta.	
Observaciones:	

Tabla 56 HU Listar materia

Historia de usuario	
Número: 2	Nombre: Listar materia
Usuario: estudiante, profesor, administrador	
Prioridad en negocio: alta	Puntos estimados: 0.25
Riesgo en desarrollo: alta	Iteración asignada: 1
Descripción: a hacer clic en la opción materias se muestra el listado de las que están creadas en la herramienta y la descripción de cada una.	
Observaciones:	

Tabla 57 HU Adicionar materia

Historia de usuario	
Número: 3	Nombre: Adicionar materia
Usuario: profesor, administrador	
Prioridad en negocio: alta	Puntos estimados: 1
Riesgo en desarrollo: alta	Iteración asignada: 1
Descripción: al hacer clic en adicionar materias se muestra una vista que permite introducir los datos necesarios para adicionar una nueva materia: - Nombre - Descripción Al ser adicionada se muestra un mensaje confirmando que esta materia fue insertada.	
Observaciones:	

Tabla 58 HU Modificar materia

Historia de usuario	
Número: 4	Nombre: Modificar materia
Usuario: profesor, administrador	
Prioridad en negocio: alta	Puntos estimados: 0.5
Riesgo en desarrollo: alta	Iteración asignada: 1
Descripción: cuando se selecciona la materia a modificar se debe hacer clic en editar y aparece una vista con los datos de la materia para ser actualizados:	

- Nombre - Descripción Al editarlos se muestra un mensaje de confirmación.
Observaciones:

Tabla 59 HU Eliminar materia

Historia de usuario	
Número: 5	Nombre: Eliminar materia
Usuario: profesor, administrador	
Prioridad en negocio: alta	Puntos estimados: 0.5
Riesgo en desarrollo: alta	Iteración asignada: 1
Descripción: se selecciona la materia y se hace clic en eliminar, se pregunta si se desea eliminar la materia con sus tutores para evitar pérdida de datos accidentales.	
Observaciones:	

Tabla 60 HU Guardar trazas

Historia de usuario	
Número: 15	Nombre: Guardar trazas
Usuario: estudiante, profesor, administrador	
Prioridad en negocio: media	Puntos estimados: 1
Riesgo en desarrollo: media	Iteración asignada: 7
Descripción: todas las acciones realizadas por el usuario en su interacción con el tutor son guardadas en la base de datos.	
Observaciones:	

Tabla 61 HU Consultar trazas

Historia de usuario	
Número: 16	Nombre: Consultar trazas
Usuario: profesor, administrador	
Prioridad en negocio: media	Puntos estimados: 1
Riesgo en desarrollo: media	Iteración asignada: 7
Descripción: al hacer clic en consultar trazas se muestra la lista de trazas.	
Observaciones: en las trazas son guardadas la fecha y hora, la acción, el tutor con que interactúa, la habilidad que corresponde a cada acción, el usuario y el valor.	

Tabla 62 HU Consultar ayuda contextual

Historia de usuario	
Número: 17	Nombre: Consultar ayuda contextual

Usuario: profesor, administrador	
Prioridad en negocio: baja	Puntos estimados: 1
Riesgo en desarrollo: baja	Iteración asignada: 8
Descripción: en distintas áreas de la funcionalidad demostrar tutor, al mover el mouse por encima aparecerán los mensajes para facilitar el proceso de demostración del tutor por parte del usuario.	
Observaciones:	

Tabla 63 HU Internacionalización

Historia de usuario	
Número: 18	Nombre: Internacionalización
Usuario: estudiante, profesor, administrador	
Prioridad en negocio: baja	Puntos estimados: 1
Riesgo en desarrollo: baja	Iteración asignada: 8
Descripción: en la vista principal del sistema aparecerá un botón que contendrá las siglas del idioma contrario al que se está utilizando para que sea cambiado por el usuario (inglés o español).	
Observaciones:	

Tareas de ingeniería

Iteración 1:

Tabla 64 Tarea de ingeniería # 1

Tarea	
Número: 1	Número de HU: 1
Nombre: Implementar las clases, interfaces y formularios necesarias para la visualización de la entidad materia.	
Tipo de tarea: desarrollo	Estimación: 1 día
Fecha de inicio: 19-11-2013	Fecha de fin: 19-11-2013
Programador responsable: Marlon Cabrera Torres – Yasiel Tejeda González	
Descripción: implementar la clase Materia y MateriaRepository que serán empleadas por el ORM para el mapeo de las entidades y sus relaciones y por HU como esta. Además implementar la interfaz show y el formulario de esta entidad. Paquete: Application\HatbeBundle\Entity. Clase: Materia, MateriaRepository Interfaz: Materia\show.html.twig	

Tabla 65 Tarea de ingeniería # 2

Tarea	
Número: 2	Número de HU: 1
Nombre: Crear la ruta para la funcionalidad mostrar materia.	
Tipo de tarea: desarrollo	Estimación: 1 día

Fecha de inicio: 20-11-2013	Fecha de fin: 20-11-2013
Programador responsable: Marlon Cabrera Torres – Yasiel Tejeda González	
Descripción: crear una ruta en el archivo materia.yml para acceder a esta funcionalidad.	

Tabla 66 Tarea de ingeniería # 3

Tarea	
Número: 3	Número de HU: 2
Nombre: Construir la interfaz index y la funcionalidad listar materia.	
Tipo de tarea: desarrollo	Estimación: 1 día
Fecha de inicio: 21-11-2013	Fecha de fin: 21-11-2013
Programador responsable: Marlon Cabrera Torres – Yasiel Tejeda González	
Descripción: implementar la funcionalidad listar materia en el controlador MateriaController y la interfaz index para listar las materias. Interfaz: Materia\index.html.twig	

Tabla 67 Tarea de ingeniería # 4

Tarea	
Número: 4	Número de HU: 2
Nombre: Crear la ruta para la funcionalidad listar materia.	
Tipo de tarea: desarrollo	Estimación: 1 día
Fecha de inicio: 22-11-2013	Fecha de fin: 22-11-2013
Programador responsable: Marlon Cabrera Torres – Yasiel Tejeda González	
Descripción: crear una ruta en el archivo materia.yml para acceder a esta funcionalidad.	

Tabla 68 Tarea de ingeniería # 5

Tarea	
Número: 5	Número de HU: 3
Nombre: Implementar la interfaz new y la funcionalidad adicionar materia.	
Tipo de tarea: desarrollo	Estimación: 3 días
Fecha de inicio: 25-11-2013	Fecha de fin: 27-11-2013
Programador responsable: Marlon Cabrera Torres – Yasiel Tejeda González	
Descripción: implementar la funcionalidad adicionar materia en el controlador MateriaController y la interfaz new para adicionar nuevas materias. Interfaz: Materia\new.html.twig	

Tabla 69 Tarea de ingeniería # 6

Tarea	
Número: 6	Número de HU: 3
Nombre: Crear la ruta para la funcionalidad adicionar materia.	
Tipo de tarea: desarrollo	Estimación: 2 días

Fecha de inicio: 28-11-2013	Fecha de fin: 29-11-2013
Programador responsable: Marlon Cabrera Torres – Yasiel Tejeda González	
Descripción: crear una ruta en el archivo materia.yml para acceder a esta funcionalidad y otra para cuando la materia está creada.	

Tabla 70 Tarea de ingeniería # 7

Tarea	
Número: 7	Número de HU: 4 , 5
Nombre: Implementar la interfaz edit y la funcionalidades para modificar una materia.	
Tipo de tarea: desarrollo	Estimación: 2,5 días
Fecha de inicio: 2-12-2013	Fecha de fin: 4-12-2013
Programador responsable: Marlon Cabrera Torres – Yasiel Tejeda González	
Descripción: implementar las funcionalidades editar y eliminar materia en el controlador MateriaController y la interfaz edit para modificar o eliminar una materia seleccionada. Interfaz: Materia\edit.html.twig	

Tabla 71 Tarea de ingeniería # 8

Tarea	
Número: 8	Número de HU: 4, 5
Nombre: Crear las rutas para las funcionalidades modificar y eliminar materia.	
Tipo de tarea: desarrollo	Estimación: 2,5 días
Fecha de inicio: 4-12-2013	Fecha de fin: 6-12-2013
Programador responsable: Marlon Cabrera Torres – Yasiel Tejeda González	
Descripción: crear las rutas en el archivo materia.yml para eliminar y modificar una materia y otra para cuando se modificó.	

Iteración 6:

Tabla 72 Tarea de ingeniería # 25

Tarea	
Número: 25	Número de HU: 13
Nombre: Definir la estructura del fichero que se generará al exportar un tutor.	
Tipo de tarea: desarrollo	Estimación: 1 día
Fecha de inicio: 24-3-2014	Fecha de fin: 24-3-2014
Programador responsable: Marlon Cabrera Torres – Yasiel Tejeda González	
Descripción: definir la estructura del fichero que se generara al exportar un tutor para que pueda ser ejecutado fuera de la herramienta o ser importado posteriormente por la misma.	

Tabla 73 Tarea de ingeniería # 26

Tarea	
Número: 26	Número de HU: 13

Nombre: Implementar la funcionalidad para exportar un tutor.	
Tipo de tarea: desarrollo	Estimación: 2 días
Fecha de inicio: 25-3-2014	Fecha de fin: 26-3-2014
Programador responsable: Marlon Cabrera Torres – Yasiel Tejeda González	
Descripción: implementar la funcionalidad exportar tutor en el controlador TutorController.	

Tabla 74 Tarea de ingeniería # 27

Tarea	
Número: 27	Número de HU: 13
Nombre: Crear las rutas para la funcionalidad exportar tutor.	
Tipo de tarea: desarrollo	Estimación: 2 días
Fecha de inicio: 27-3-2014	Fecha de fin: 28-3-2014
Programador responsable: Marlon Cabrera Torres – Yasiel Tejeda González	
Descripción: crear las rutas en el archivo tutor.yml para exportar un tutor.	

Tabla 75 Tarea de ingeniería # 28

Tarea	
Número: 28	Número de HU: 14
Nombre: Implementar la funcionalidad para importar un tutor.	
Tipo de tarea: desarrollo	Estimación: 3 días
Fecha de inicio: 31-3-2014	Fecha de fin: 2-4-2014
Programador responsable: Marlon Cabrera Torres – Yasiel Tejeda González	
Descripción: implementar la funcionalidad importar tutor en el controlador TutorController.	

Tabla 76 Tarea de ingeniería # 29

Tarea	
Número: 29	Número de HU: 14
Nombre: Crear las rutas para la funcionalidad importar tutor.	
Tipo de tarea: desarrollo	Estimación: 2 días
Fecha de inicio: 3-4-2014	Fecha de fin: 4-4-2014
Programador responsable: Marlon Cabrera Torres – Yasiel Tejeda González	
Descripción: crear las rutas en el archivo tutor.yml para importar un tutor.	

Iteración 7:

Tabla 77 Tarea de ingeniería # 30

Tarea	
Número: 30	Número de HU: 15
Nombre: Definir los elementos sobre los cuales se van a guardar las trazas de los estudiantes mientras interactúan con la herramienta.	

Tipo de tarea: investigación	Estimación: 1 día
Fecha de inicio: 7-4-2014	Fecha de fin: 7-4-2014
Programador responsable: Marlon Cabrera Torres – Yasiel Tejeda González	
Descripción: estudiar otras herramientas que guarden trazas y definir los elementos sobre los cuales se van a guardar las trazas de los estudiantes mientras interactúan con la herramienta.	

Tabla 78 Tarea de ingeniería # 31

Tarea	
Número: 31	Número de HU: 15
Nombre: Implementar el servicio guardar trazas.	
Tipo de tarea: desarrollo	Estimación: 4 días
Fecha de inicio: 8-4-2014	Fecha de fin: 11-4-2014
Programador responsable: Marlon Cabrera Torres – Yasiel Tejeda González	
Descripción: implementar el servicio guardar trazas y usarlo en las acciones que se definieron que se iban a guardar las trazas de los estudiantes.	

Tabla 79 Tarea de ingeniería # 32

Tarea	
Número: 32	Número de HU: 16
Nombre: Implementar la interfaz trazas y la funcionalidades para consultar las trazas de un estudiante.	
Tipo de tarea: desarrollo	Estimación: 3 días
Fecha de inicio: 21-4-2014	Fecha de fin: 23-4-2014
Programador responsable: Marlon Cabrera Torres – Yasiel Tejeda González	
Descripción: implementar la interfaz trazas y la funcionalidad para consultar las trazas de un estudiante en el controlador UsuarioController.	

Tabla 80 Tarea de ingeniería # 33

Tarea	
Número: 33	Número de HU: 16
Nombre: Crear las rutas para la funcionalidad consultar trazas.	
Tipo de tarea: desarrollo	Estimación: 2 días
Fecha de inicio: 24-4-2014	Fecha de fin: 25-4-2014
Programador responsable: Marlon Cabrera Torres – Yasiel Tejeda González	
Descripción: crear las rutas en el archivo usuario.yml para consultar trazas.	

Iteración 8:

Tabla 81 Tarea de ingeniería # 34

Tarea

Número: 34	Número de HU: 17
Nombre: Definir las áreas de la herramienta que contarán con ayudas contextuales.	
Tipo de tarea: desarrollo	Estimación: 3 días
Fecha de inicio: 28-4-2014	Fecha de fin: 30-4-2014
Programador responsable: Marlon Cabrera Torres – Yasiel Tejeda González	
Descripción: definir las áreas de la herramienta que contarán con ayuda contextual y los textos que mostrarán.	

Tabla 82 Tarea de ingeniería # 35

Tarea	
Número: 35	Número de HU: 17
Nombre: Editar las interfaces que contarán con ayudas contextuales.	
Tipo de tarea: desarrollo	Estimación: 2 días
Fecha de inicio: 2-5-2014	Fecha de fin: 3-5-2014
Programador responsable: Marlon Cabrera Torres – Yasiel Tejeda González	
Descripción: agregar a las interfaces que contarán con ayuda contextual el icono de ayuda en el área con su respectivo contenido.	

Tabla 83 Tarea de ingeniería # 36

Tarea	
Número: 36	Número de HU: 18
Nombre: Estudio del componente de Internacionalización de Symfony.	
Tipo de tarea: investigación	Estimación: 1 día
Fecha de inicio: 5-5-2014	Fecha de fin: 5-5-2014
Programador responsable: Marlon Cabrera Torres – Yasiel Tejeda González	
Descripción: estudio del componente de Internacionalización de Symfony para comprender el funcionamiento del mismo y usarlo en la aplicación.	

Tabla 84 Tarea de ingeniería # 37

Tarea	
Número: 37	Número de HU: 18
Nombre: Implementación de la internacionalización del sitio.	
Tipo de tarea: desarrollo	Estimación: 4 días
Fecha de inicio: 6-5-2014	Fecha de fin: 9-5-2014
Programador responsable: Marlon Cabrera Torres – Yasiel Tejeda González	
Descripción: definir los idiomas en los que podrá ser visualizado el sitio con la creación del fichero con los mensajes en cada idioma (inglés y español). Edición de todas las rutas definidas en la aplicación para que admitan ser internacionalizadas agregándole la variable _locale al inicio de cada ruta.	

Tarjetas CRC

Tabla 85 Tarjeta CRC Clase Arbol

Nombre de la clase: Arbol	
Responsabilidades	**Colaboraciones**
getId getNodo getHijos getPadre getRaiz setId setNodo setHijos addHijo setPadre setRaiz	Nodo Arbol

Tabla 86 Tarjeta CRC Clase Hint

Nombre de la clase: Hint	
Responsabilidades	**Colaboraciones**
getId setNivel getNivel setHint getHint getNodo setNodo	Nodo

Tabla 87 Tarjeta CRC Clase Materia

Nombre de la clase: Materia	
Responsabilidades	**Colaboraciones**
getSlug getTutores setNombre getNombre setDescripcion getDescripcion	Tutor

Tabla 88 Tarjeta CRC Clase Tutor

Nombre de la clase: Tutor	
Responsabilidades	**Colaboraciones**
getSlug getId setNombre getNombre setDescripcion getDescripcion setVista getVista setSolucion getSolucion setMateria getMateria getUploadRootDir	Materia Arbol

getUploadDir getUploadCompleteDir getUploadTempDir setArchivo getArchivo upload removeUpload preUpdate	

Tabla 89 Tarjeta CRC Clase Nodo

Nombre de la clase: Nodo	
Responsabilidades	**Colaboraciones**
getCampo getValorDemostrado getComparacionDeEntrada getTipoDeComparacion getMensajeCorrecto getMensajeIncorrecto setCampo setValorDemostrado setComparacionDeEntrada setTipoDeComparacion setMensajeCorrecto setMensajeIncorrecto getId setAccion getAccion setHabilidad getHabilidad setTipo getTipo getHints addHint	Hint

Tabla 90 Tarjeta CRC Clase Traza

Nombre de la clase: Traza	
Responsabilidades	**Colaboraciones**
getId setFecha getFecha setAccion getAccion getUsuario setUsuario setTipo getTipo setTutor getTutor setHabilidad getHabilidad setValor getValor	