Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación
Ingeniería de Software y Sistemas (ISYS)

Desarrollo del módulo de elaboración, presentación en línea y corrección automática de la evaluación instrumental de idiomas a los estudiantes de postgrado por parte de la Escuela de Idiomas Modernos de la UCV

Trabajo Especial de Grado
presentado ante la Ilustre
Universidad Central de Venezuela
Por el Bachiller

Escobar Lugo Nestor Leonel

para optar al título de

Licenciado en Computación

Tutor: Prof. Sergio Rivas.

Caracas, Julio del 2013

Acta

Quienes suscriben, miembros del jurado designado por el Consejo de la Escuela de Computación de la Facultad de Ciencias de la Universidad Central de Venezuela, para examinar el Trabajo Especial de Grado titulado: Desarrollo del módulo de elaboración, presentación en línea y corrección automática de la evaluación instrumental de idiomas a los estudiantes de Postgrado por parte de la Escuela de Idiomas Modernos de la UCV, presentado por el bachiller Nestor L. Escobar L., C.I.: 20.116.113, a los fines de optar por el título de Licenciado en Computación, dejan constancia de lo siguiente:

Dicho trabajo, leído por cada uno de los miembros del jurado, se fijó el día Lunes 29 de Julio de 2013, a las 11:00 am, para que el autor lo defendiera en forma pública en la Escuela de Computación, mediante una presentación oral de su contenido, luego de lo cual respondió a las preguntas formuladas. Finalizada la defensa pública del Trabajo Especial de Grado, el jurado decidió aprobarlo con la nota de _____ puntos.

En fe de lo cual se levanta la presente Acta, en Caracas a los veintinueve (29) días del mes de Julio del año dos mil trece (2013), dejando constancia de que actuó como coordinador del jurado el Profesor Sergio Rivas

_____ _____

Profa. Jossie Zambrano (Jurado) Profa. Karima Velasquez (Jurado)

_____

Prof. Sergio Rivas (Tutor)

Dedicatoria

A Dios

Quien siempre me ha acompañado en este largo recorrido.

A ustedes Nena y Fabio

Por servirme de inspiración. Les debo todo lo que soy.

A ti Belitzabeth

Que siempre estuviste allí para mí.

A ustedes Abuela y Tony

Que donde quiera que estén, sé que siempre me cuidan.

Agradecimientos

A Dios

Que en los momentos de dificultad me ha llenado de fuerza necesaria para levantarme y continuar.

A ustedes Nena y Fabio

Que me dieron la vida y siempre han estado conmigo en todo momento dando lo máximo para ofrecerme todo lo necesario para una vida feliz, gracias por haberme guiado por el mejor camino, y aunque hemos pasado momentos difíciles siempre he contado con su apoyo y amor. Los amo.

A ti Belitzabeth

Gracias por el apoyo que me has dado, cuando lo he necesitado. Eres partícipe de esto. Te amo.

A ti Armando

Gracias por tu amistad y tus buenos consejos en el momento oportuno.

A mis familiares

Que de alguna manera contribuyeron en mi formación como persona y me apoyaron durante toda la carrera.

A ti Sergio Rivas

Gracias por tus enseñanzas, por dedicar parte de tu tiempo a instruirme como profesional en el área. Gracias por depositar tu confianza en mí y ser mi mentor durante gran parte de mi carrera.

A ti Jossie

Por estar siempre atenta a cualquier inquietud, y por tus sabios consejos.

A ti Adrián Bottini

Gracias por todo lo que aprendí de ti.

A ti Daniela

Por ser mi compañera durante gran parte de este camino ya culminado.

A ti Lilo

Por esos momentos de alegría cuando más lo necesitaba.

A la ilustre Universidad Central de Venezuela

Que me ha dado la oportunidad de realizarme en esta etapa profesional de mi vida.

Al personal de CIOMMA

Gracias por permitirme desempeñar mis labores en el Centro. Nuevamente a Sergio y a Adrián, Adelis, Jaime, Eliezer y Otilio.

A mis profesores

Gracias a todos esos profesores que fueron excelentes en cada asignatura que cursé y a aquellos que, sin darme clases, se portaron excelentes conmigo. Gracias nuevamente a Sergio, a Jossie, a Adrián, a Adelis y a Jaime, Marlliny, Joalí, Néstor, Lucía, Wilfredo, Tina, Radhamés, Zenaida, Eugenio, Eleonora, Héctor y muchos otros. Personas como ustedes hacen posible que la Universidad sea lo que es. Gracias a todos.

A mis amigos y compañeros

Gracias a Felix, Fernando, Johan, Manuela, Miguel, Oswaldo, César, Juan, Beto y muchos otros. Gracias a ustedes la universidad ha sido una gran etapa como estudiante, fueron cinco grandes años.

Resumen

El objetivo del presente Trabajo Especial de Grado consiste en el desarrollo de una aplicación Web que automatice y apoye el conjunto de procesos relacionados a la gestión, elaboración, presentación y corrección de la evaluación instrumental que realiza la Escuela de Idiomas Modernos; dando soporte al personal involucrado, para aumentar así su satisfacción, y al mismo tiempo optimizar los costos y los tiempos de respuesta. Para ello, se aplicó una adaptación del método de desarrollo Programación Extrema, donde se realizó la fase de planificación de Iteraciones cada una conformada por historias de usuario, el diseño y la implementación para la aplicación de pruebas. Como producto final, se obtuvo una integración de estos procesos con la aplicación Web denominada *aTesT*, que actualmente representa el sistema de gestión de exámenes de los cursos de idiomas que ofrece la Coordinación de Extensión de la Escuela de Idiomas Modernos de la Facultad de Humanidades y Educación en la Universidad Central de Venezuela.

Palabras Clave: Rails, *aTesT*, metodologías ágiles, aplicaciones Web, evaluación instrumental, postgrado, idiomas.

Índice de Contenido

Índice de Ilustraciones

Introducción

Internet ha tenido un gran auge en las últimas décadas. Esto ha provocado que empresas y organizaciones sustituyan sus programas de escritorio por aplicaciones Web, permitiendo de esta manera que los usuarios finales se vean favorecidos por estos servicios.

Una de las organizaciones que se ha sumado al desarrollo de aplicaciones Web, es la Escuela de Idiomas Modernos de la Facultad de Humanidades y Educación de la Universidad Central de Venezuela (UCV). Los procesos de inscripción y administración de los cursos de idiomas suministrados por esta Escuela, son algunos de los procedimientos que fueron implementados bajo este tipo de aplicación. No obstante, aún existen procesos que se realizan de forma manual y con poco uso de tecnología, como es el caso de los procesos que intervienen en la evaluación instrumental de idiomas que presentan los estudiantes de postgrado, que consiste en la comprensión lectora y traducción de un idioma extranjero.

El objetivo de este Trabajo Especial de Grado consiste en desarrollar una una aplicación Web, específicamente un módulo que permita la gestión, elaboración y presentación de la evaluación instrumental ingresando las preguntas y sus respectivas respuestas en la aplicación como documentos digitales. La gestión abarca todo lo referente a observar y editar el contenido, eliminar, activar o desactivar evaluaciones. La elaboración incluye ingresar el contenido del examen junto al idioma y postgrado al que pertenece, y seleccionar cuál de los párrafos del texto será el de traducción. La presentación permitirá a los estudiantes responder el examen en línea. Por otro lado, la aplicación contendrá un prototipo para corregir la sección de traducción de manera automática empleando la adaptación de un algoritmo, en principio utilizado para detectar plagio de documentos, que calcula la similitud entre dos textos.

El presente documento se encuentra estructurado de la siguiente manera:

El Capítulo I desarrolla el contexto del problema, el cual se encuentra relacionado con la gestión, elaboración, presentación y corrección de la evaluación instrumental. Asimismo, se plantea la solución para esta problemática, la cual incluye el objetivo general y los objetivos específicos, la importancia y justificación de automatizar los procesos involucrados en el problema, mediante el desarrollo de una aplicación Web y los beneficios que generaría a la comunidad involucrada. Posteriormente, se tiene el alcance de la aplicación que expone las funcionalidades que el sistema estaría en la capacidad de ofrecer.

En el Capítulo II se presentan todas las bases para el Marco Conceptual. Se presentan las tecnologías Web donde se muestran las bases conceptuales de las mismas, en las que se fundamenta el desarrollo de este trabajo. Cabe destacar que todas ellas son herramientas de software libre. También se describen los documentos digitales y se plantea una técnica utilizada para la detección de similitudes entre documentos digitales que será empleada para la corrección automática del examen.

El Capítulo III presenta el método de desarrollo de software a ser utilizado, el cual es *Programación Extrema* que está clasificado como un método ágil de desarrollo. También se presenta el desarrollo de la aplicación compuesto por cada una de las iteraciones necesarias.

Finalmente se presentan las conclusiones, recomendaciones y referencias bibliográficas utilizadas durante el desarrollo de este trabajo.

Capítulo I Problema de Investigación

Este capítulo tiene como finalidad presentar el contexto del problema, relacionado con la gestión, elaboración, presentación y corrección de la evaluación instrumental de la Escuela de Idiomas Modernos de la UCV, la cual consiste en la compresión lectora de un idioma extranjero para estudiantes de postgrado. Igualmente, se destaca el objetivo general y los objetivos específicos de este trabajo, así como la importancia y justificación de automatizar los procesos involucrados con el problema, mediante el desarrollo de una aplicación Web y los beneficios que generaría a la comunidad involucrada. Por último, se muestra el alcance de la aplicación que expone las funcionalidades que el sistema estaría en la capacidad de ofrecer.

1.1 Situación Actual

La Escuela de Idiomas Modernos de la Facultad de Humanidades y Educación de la Universidad Central de Venezuela, se encuentra automatizando diferentes procesos para obtener resultados óptimos sobre las actividades llevadas a cabo por ella. Entre estas actividades se puede señalar, la inscripción y administración de los cursos de idiomas extranjeros suministrados por esta Escuela. No obstante, aún existen actividades que se siguen realizando de forma manual, con ningún o poco uso de la tecnología, como es el caso de los procesos que intervienen en el desarrollo de la evaluación instrumental (ejecutados semanalmente), lo que conlleva a que exista un gran porcentaje de error humano, un cuantioso tiempo de respuesta y costos elevados, lo cual origina grandes desventajas a la organización.

La evaluación instrumental sirve para medir los conocimientos suficientes de un aspirante para cursar un postgrado acerca de un idioma extranjero, y ésta consiste en la comprensión lectora y traducción de un texto relacionado con el área de dicho postgrado.

Esta evaluación instrumental se aplica a estudiantes de postgrado (en todas las áreas) de la Universidad Central de Venezuela, donde el idioma en el que está basada la prueba es seleccionado por cada persona que opta para aprobar dicho examen. Los idiomas disponibles son: Inglés, Alemán, Francés, Italiano y Portugués. Cada uno de ellos se aplica en esta evaluación, dependiendo de la disponibilidad, así como de los diferentes contextos según el área correspondiente al postgrado.

Entre las actividades relacionadas con el módulo en cuestión que se llevan a cabo en la Coordinación de Extensión de dicha Escuela se encuentran las siguientes:

1. Elaboración del examen.
2. Presentación del examen.
3. Corrección del examen.

A continuación se hace una descripción de cada una de las tareas antes mencionadas que intervienen en el examen instrumental de la Escuela de Idiomas Modernos.

1.1.1 Elaboración del examen

El examen es elaborado manualmente por personal especializado de la Escuela de Idiomas Modernos. El contenido de estas pruebas se transcribe y semanalmente es reproducido para su presentación cada día sábado.

1.1.2 Presentación del examen

El día estipulado para presentar la prueba instrumental las personas inscritas deben presentarse en el salón y horario acordado, en donde se encontrará

un Supervisor, que confirmará la asistencia y recibirá el comprobante del depósito realizado. El examen consiste en una comprensión lectora de un texto en el idioma extranjero seleccionado anteriormente, además posee una duración de una hora y media y se puede hacer uso de un diccionario. Luego estos exámenes se entregan a los profesores correspondientes para que realicen la corrección. Se refleja este proceso a través de la Ilustración 1.

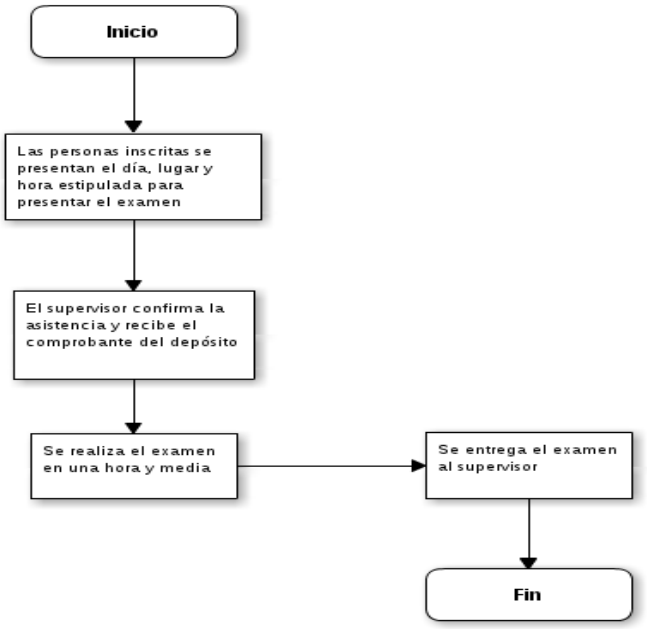


Ilustración 1: Presentación del examen.

1.1.3 Corrección del examen

Una vez terminados los exámenes, los mismos se entregan a los profesores correspondientes para que realicen su corrección, calificando el examen como aprobado o reprobado. Ésta es completamente manual y si el profesor lo requiere puede tener una segunda revisión para corroborar la nota. Posteriormente, se escriben las notas en una lista y el martes siguiente de la presentación del examen se les envía un mensaje de texto con su calificación, o las personas inscritas podrían llamar por teléfono a las instalaciones de la Coordinación de Extensión de la Escuela de Idiomas Modernos para conocer su nota. En la Ilustración 2 se muestra este proceso.

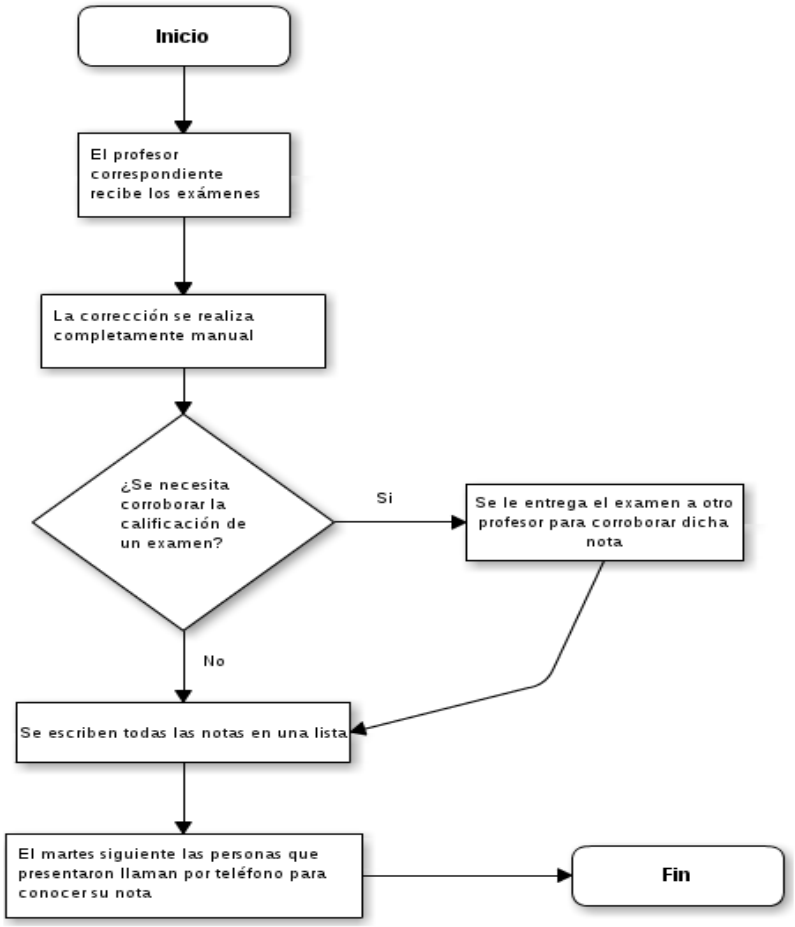


Ilustración 2: Corrección del examen.

1.2 Planteamiento del Problema

Cada una de las actividades descritas en la sección anterior se llevan a cabo de forma manual, siendo este el principal problema que se plantea. Esto, además de aumentar el número de errores en el flujo de trabajo de las tareas, aumenta los costos en cuanto a recursos físicos y los tiempos de respuestas. Otro problema que se presenta es que los procesos duran más tiempo en ser terminados y en su mayoría la cantidad de material utilizado es elevado. En este sentido se busca optimizar los procedimientos, tomando en cuenta aquellos que realmente son necesarios realizar.

El contenido del examen es transcrito en un editor de texto y es guardado como un archivo en la computadora y su reproducción los días sábados genera un alto costo en material impreso.

Para la presentación, la escuela de idiomas tiene que coordinar un aula de alguna facultad de la universidad para que sea posible que los estudiantes presenten la evaluación manuscrita.

La corrección de la evaluación también es completamente manual, lo que implica que, además de gastar tiempo leyendo la respuesta del resumen del texto por parte del estudiante, también invertir tiempo y esfuerzo en leer tanto el párrafo de traducción como la respuesta palabra por palabra para poder dar una calificación al examen. Esto no sucedería si existiera una funcionalidad que comparara dos textos y cuantificara de alguna manera la similitud existente entre dichos textos.

En síntesis, se puede observar que como todos estos procedimientos pueden presentar diferentes inconvenientes a los miembros de la Coordinación de Extensión de la Escuela de Idiomas Modernos a lo largo de toda su ejecución, se propone desarrollar módulos e integrarlos a *aTesT* (una aplicación Web que utiliza el personal de la escuela de idiomas para la elaboración automatizada del diseño de los exámenes escritos de los cursos de extensión) para que se automatice y estandarice las actividades de elaboración, presentación y corrección de la evaluación instrumental, permitiendo así facilitar y agilizar dichos procesos.

Para la elaboración se propone una interfaz de usuario con un editor de texto enriquecido integrado para ingresar el contenido del examen, el cual esté catalogado en el sistema de acuerdo al idioma y al postgrado; además, que con tan sólo un clic se seleccione el párrafo a ser traducido y posteriormente sea ingresada la traducción correcta en el idioma español para este texto (actividad realizada sólo una vez) que servirá para compararla con las traducciones de los estudiantes mediante un algoritmo que detecta similitud entre textos. Todo esto permite tener

las evaluaciones almacenadas digitalmente en una base de datos y la reutilización de su información cada vez que sea necesario.

Adicionalmente se proporciona un módulo en el cual el administrador del sistema podrá observar, editar el contenido, activar o desactivar y eliminar un examen instrumental, lo que genera que el personal indicado de la Escuela de Idiomas tenga un mayor control sobre estas evaluaciones.

Con el uso de esta aplicación los estudiantes pueden presentar la evaluación en línea almacenando las respuestas, al igual que los datos del examen, como documentos digitales. De esta manera, la presentación se llevaría a cabo en los laboratorios de la Escuela de Idiomas frente a un computador sin depender de la opción de gestionar un aula para el día del examen y la impresión del mismo no sería necesario, reduciendo así considerablemente los costos en material impreso.

En el módulo de corrección, la finalidad es proveer una interfaz en la cual se observe el contenido del examen, el párrafo o los párrafos a ser traducidos y las respuestas ingresadas por los estudiantes en el módulo de presentación. Se propone emplear la adaptación de un algoritmo, que en principio es usado para detectar plagio de documentos, para calcular el porcentaje de similitud entre el texto del estudiante y el texto ingresado como traducción correcta en el módulo de elaboración, y de esta manera calificar como aprobada o reprobada la traducción de manera automática. Es importante resaltar el tiempo que se ahorra contando con una aplicación que provea automáticamente la corrección de la traducción mediante un algoritmo que recibe como entrada dos textos y cuya resultado es el porcentaje de similitud entre dichos textos.

En la Ilustración 3 se muestra gráficamente cada una de las partes que constituyen la aplicación Web propuesta.

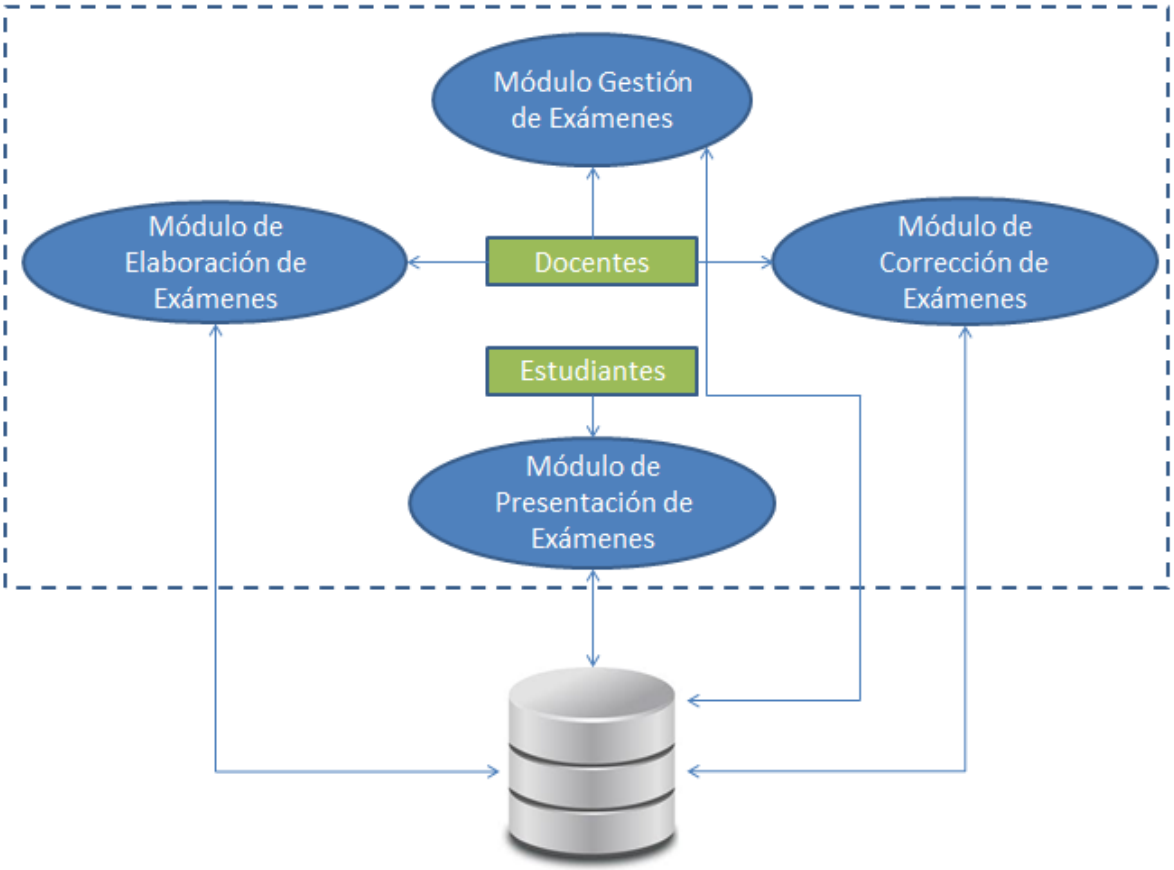


Ilustración 3: Componentes de la aplicación.

En esta imagen se presentan los distintos componentes del sistema, los cuales son los módulos de gestión, elaboración, presentación y corrección del examen. Los docentes o el personal especializado de la Escuela de Idiomas interactuarán con los módulos de gestión, elaboración y corrección, mientras que los estudiantes lo harán con el módulo de presentación del examen.

1.3 Objetivo General

Desarrollar una aplicación Web que automatice y apoye el conjunto de procesos relacionados a la gestión, elaboración, presentación y corrección de la evaluación instrumental en línea que realiza la Escuela de Idiomas Modernos de la Universidad Central de Venezuela.

1.4 Objetivos Específicos

- Desarrollar un módulo que permita la gestión de exámenes instrumentales, específicamente que permita editar, eliminar, activar o desactivar y observar el detalle de un examen.

- Desarrollar un módulo que permita la elaboración de la evaluación instrumental, así como el almacenamiento de las respuestas correctas y datos adicionales de dicho examen.

- Desarrollar un módulo que permita a los estudiantes de postgrado de la Universidad Central de Venezuela la presentación de la evaluación instrumental, almacenando las respuestas como documentos digitales.

- Desarrollar un algoritmo que realice la corrección automática de la prueba mediante una técnica de detección de similitudes entre documentos.

- Diseñar e implementar la Base de Datos en la cual se almacene la información relacionada al problema planteado.

- Aplicar el método de Programación Extrema (XP) para desarrollar los módulos.

1.5 Justificación

La Escuela de Idiomas Modernos de la Universidad Central de Venezuela es la encargada de desarrollar la evaluación instrumental. La Coordinación de Extensión de esta Escuela es la encargada de efectuar todas las actividades que conllevan a la presentación del examen; además, mantiene una comunicación constante con los estudiantes de postgrado de dicha Universidad. Sin embargo, como estos procesos son totalmente manuales, pueden generar errores e

inconvenientes, además de generar una gran cantidad de documentos impresos innecesarios.

Se presenta un conjunto de soluciones a estas dificultades, con el objeto de aumentar el grado de satisfacción en los diferentes usuarios, tales como: el público que opta por esta evaluación, docentes, directivos y personal administrativo. De esta manera se busca minimizar el porcentaje de error que pudiese existir a la hora de ejecutar estas tareas y/o actividades, apoyándonos en las bondades que ofrecen las aplicaciones Web.

Entre los beneficios que aporta la automatización de estos procesos, se pueden mencionar:

- Mejorar los tiempos de respuesta y validar los datos.
- Disminuir el esfuerzo y el tiempo invertido.
- Disminuir los costos en recursos físicos, como el material impreso.

Capítulo II Marco Conceptual

En este capítulo se presentan las bases conceptuales acerca de las tecnologías Web empleadas en la plataforma de la aplicación *aTesT*, de la cual formarán parte los módulos a desarrollar en este trabajo y la comparación de documentos digitales que sirven de fundamento teórico para el desarrollo de este Trabajo Especial de Grado.

En la primera sección, se explica brevemente las Aplicaciones Web, la arquitectura Cliente/Servidor, los Servicios Web y el patrón de diseño *MVC* (Modelo Vista Controlador), que se encarga de separar los datos de una aplicación, la interfaz de usuario, y la lógica de negocio en tres componentes distintos. También se muestran las herramientas tecnológicas necesarias para el desarrollo de la aplicación Web propuesta. Las tecnologías del lado del cliente son *HTML 5*, *CSS 3*, *JavaScript*, *AJAX,* el framework *jQuery* y diversas extensiones a partir de este; las del lado del servidor son *Apache*, *Unicorn*, *Ruby on Rails*; y las del servidor de bases de datos *MySQL*.

En la segunda sección se presentan diversos conceptos y técnicas relacionados con la comparación de documentos digitales, información que será utilizada para el módulo de corrección automática del examen instrumental.

2.1 Tecnologías Web

Las tecnologías Web son utilizadas para acceder a los recursos de conocimiento disponibles en Internet utilizando un navegador. Están muy extendidas por muchas razones: facilitan el desarrollo de sistemas de Gestión del Conocimiento, su flexibilidad en términos de escalabilidad (es decir, a la hora de expandir el sistema), su sencillez de uso y que imitan la forma de relacionarse de las personas, al poner a disposición de todos el conocimiento de los demás, por encima de jerarquías, barreras formales u otras cuestiones. Estas tecnologías

pueden llegar a proporcionar recursos estratégicos, pero, evidentemente, no por la tecnología en sí misma, que está disponible ampliamente, sino por lo fácil que es personalizarla y construir con ella sistemas de Gestión del Conocimiento propietarios de la empresa. Las tecnologías Web utilizadas en este proyecto se describen a lo largo de esta sección.

2.1.1 Aplicaciones Web

Se denominan aplicaciones Web aquellas aplicaciones que los usuarios pueden utilizar accediendo a un servidor Web a través de Internet o de una Intranet mediante un navegador. En otras palabras, es una aplicación de software que se codifica en un lenguaje soportado por los navegadores Web en la que se confía la ejecución al navegador *(Aplicaciones Web, 2012)*.

Una ventaja significativa es que las aplicaciones Web funcionan de la misma manera, independientemente de la versión del sistema operativo instalado en el cliente. En vez de crear clientes para *Windows, Mac OS, GNU/Linux* y otros sistemas operativos, la aplicación Web se escribe una vez y se ejecuta igual en todas partes. Sin embargo, hay aplicaciones inconsistentes escritas con *HTML*, *CSS*, y otras especificaciones para navegadores Web que pueden causar problemas en el desarrollo y soporte de estas aplicaciones, principalmente debido a la falta de adicción de los navegadores a los estándares Web (especialmente versiones de Internet Explorer anteriores a la 7.0). Adicionalmente, la posibilidad de los usuarios de personalizar muchas de las características de la interfaz (tamaño y color de fuentes, tipos de fuentes, inhabilitar *JavaScript*) pueden interferir con la consistencia de la aplicación Web. Otras de las ventajas que poseen las aplicaciones Web se listan a continuación:

- Inmediatez de acceso: las aplicaciones Web no necesitan ser descargadas, instaladas y configuradas.

- Menos *Bugs*: Las aplicaciones Web deberán ser menos propensas a colgarse y crear problemas técnicos debido a *software* o conflictos de *hardware* con otras aplicaciones existentes, protocolos o software personal interno. Con aplicaciones basadas en Web, todos utilizan la misma versión, y todos los bugs pueden ser corregidos tan pronto como son descubiertos.

- Múltiples usuarios concurrentes. Las aplicaciones basadas en Web pueden realmente ser utilizada por múltiples usuarios al mismo tiempo.

Las aplicaciones Web funcionan bajo una arquitectura llamada Cliente/Servidor, la cual es abarcada a continuación.

2.1.1.1 Arquitectura Cliente/Servidor

La arquitectura Cliente/Servidor es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. Un cliente realiza peticiones a otro programa, el servidor, que le da respuesta. Esta idea también se puede aplicar a programas que se ejecutan sobre una sola computadora, aunque es más ventajosa en un sistema operativo multiusuario distribuido a través de una red de computadoras *(Arquitectura Cliente/Servidor, 2012)*.

En esta arquitectura, la capacidad de proceso está repartida entre los clientes y los servidores, aunque son más importantes las ventajas de tipo organizativo debidas a la centralización de la gestión de la información y la separación de responsabilidades, lo que facilita y clarifica el diseño del sistema.

La separación entre cliente y servidor es una separación de tipo lógico, donde el servidor no se ejecuta necesariamente sobre una sola máquina ni es necesariamente un solo programa. Los tipos específicos de servidores incluyen los servidores Web, los servidores de archivo, los servidores de correo, etc. Mientras

que sus propósitos varían de unos servicios a otros, la arquitectura básica seguirá siendo la misma.

La red Cliente/Servidor es aquella red de comunicaciones en la que todos los clientes están conectados a un servidor, en el que se centralizan los diversos recursos y aplicaciones con que se cuenta; y que los pone a disposición de los clientes cada vez que estos son solicitados. Esto significa que todas las gestiones que se realizan se concentran en el servidor, de manera que en él se disponen los requerimientos provenientes de los clientes que tienen prioridad, los archivos que son de uso público y los que son de uso restringido, los archivos que son de sólo lectura y los que, por el contrario, pueden ser modificados, etc. Este tipo de red puede utilizarse conjuntamente en caso de que se esté utilizando en una red mixta *(Arquitectura Cliente/Servidor, 2012)*.

Una de las maneras de estructurar los componentes de software en este modelo de aplicación distribuida es siguiendo el patrón de arquitectura *MVC* (Modelo Vista Controlador).

2.1.1.2 Patrón de Diseño MVC (Modelo Vista Controlador)

La arquitectura de tres capas se refleja en el patrón de arquitectura *MVC* (Modelo Vista Controlador), desarrollado con el propósito reducir el acoplamiento, o grado de dependencia entre módulos, entre la lógica de negocios y la de presentación.

Es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de negocio en tres componentes distintos. La finalidad de este modelo es mejorar la reusabilidad por medio del desacoplo entre la *vista* y el *modelo*. El *modelo* es la representación específica de la información con la cual el sistema opera. La *vista* presenta el *modelo* en un formato adecuado para interactuar, usualmente la interfaz de usuario. Finalmente el *controlador* responde a eventos, usualmente acciones del usuario, e invoca

peticiones al *modelo* y, probablemente, a la *vista (MVC, 2012)*.

Una posible descripción gráfica del patrón se presenta en la Ilustración 4:

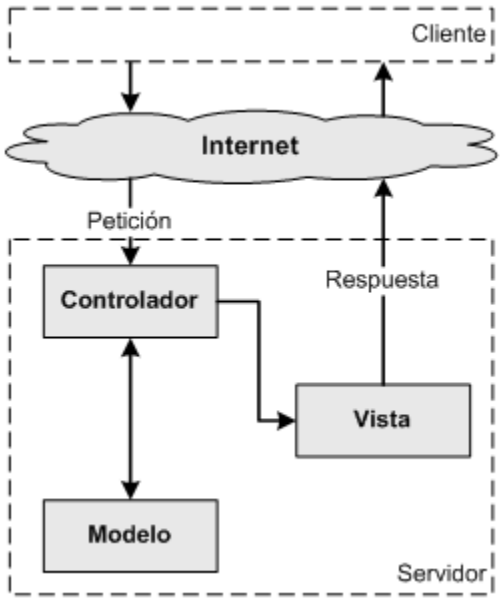


Ilustración 4: Modelo Vista Controlador (*Patrón MVC, 2013*)

Este modelo de arquitectura presenta varias ventajas:

- Hay una separación entre los componentes de un programa; lo cual nos permite implementarlos por separado.

- Al incorporar el modelo de arquitectura MVC a un diseño, los módulos de un programa se pueden construir por separado y luego unirlos en tiempo de ejecución.

- Facilidad de desarrollo y acortamiento del tiempo de respuesta gracias a las tareas paralelizadas.

- Aumenta en gran medida el nivel de reusabilidad de código. Facilita una evolución continua de los sistemas, sin puntos de ruptura, ya que un cambio

en un sistema afectará a uno o más componentes pero nunca afectará significativamente al núcleo de la aplicación.

Aunque se pueden encontrar diferentes implementaciones de *MVC*, el flujo que sigue el control generalmente es el siguiente:

1. El usuario interactúa con la interfaz de usuario de alguna forma (por ejemplo, el usuario pulsa un botón, enlace)

2. El controlador recibe (por parte de los objetos de la interfaz-vista) la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega, frecuentemente a través de un gestor de eventos (handler) o callback.

3. El controlador accede al modelo, actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario (por ejemplo, el controlador actualiza el carro de la compra del usuario). Los controladores complejos están a menudo estructurados usando un patrón de comando que encapsula las acciones y simplifica su extensión.

4. El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se reflejan los cambios en el modelo (por ejemplo, produce un listado del contenido del carro de la compra.

5. La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.

A continuación se presentan las herramientas tecnológicas utilizadas para trabajar con este patrón.

2.1.2 Herramientas Tecnológicas para el desarrollo de Aplicaciones Web

En los primeros tiempos de la computación cliente/servidor, cada aplicación tenía su propio programa cliente y su interfaz de usuario, éstos tenían que ser instalados separadamente en cada estación de trabajo de los usuarios. Una mejora al servidor, como parte de la aplicación, requería típicamente una mejora de los clientes instalados en cada una de las estaciones de trabajo, añadiendo un costo de soporte técnico y disminuyendo la eficiencia del personal *(Aplicación Web, 2012)*.

En contraste, las aplicaciones Web generan dinámicamente páginas en un formato estándar, soportado por navegadores Web comunes, como *HTML* o *XHTML*, y actualmente *HTML 5,* la cual es la quinta revisión importante del lenguaje básico de la Red Informática Mundial o *World Wide Web* (*WWW*), *HTML*. Adicionalmente se utilizan lenguajes interpretados del lado del cliente, tales como JavaScript, para añadir elementos dinámicos a la interfaz de usuario. También se han desarrollado tecnologías para coordinar estos lenguajes con tecnologías del lado del servidor, como por ejemplo *PHP*, *Ruby on Rails*, *JSP*, entre otros.

2.1.2.1 Tecnologías del lado del Cliente

Las tecnologías del lado del cliente, es decir, las que se ejecutan en el navegador del usuario, son las páginas dinámicas que se procesan en el cliente. En estas páginas toda la carga de procesamiento de los efectos y funcionalidades la soporta el navegador. Estas tecnologías se detallan a continuación.

2.1.2.1.1 Lenguaje de Marcado de Hipertexto, versión 5 (HTML 5)

HTML 5 es una colección de estándares para el diseño y desarrollo de páginas Web. Esta colección representa la manera en que se presenta la

información en el navegador de Internet y la manera de interactuar con ella. *HTML 5* es una nueva versión de diversas especificaciones como *HTML 4* y *XHTML 1 (Hogan, 2012. a)*.

A la par, *HTML 5* pretende proporcionar una plataforma con la que desarrollar aplicaciones Web más parecidas a las aplicaciones de escritorio, donde su ejecución dentro de un navegador no implique falta de recursos o facilidades para resolver las necesidades reales de los desarrolladores. Para ello se están creando unas interfaces de programación de aplicaciones o *Application Programming Interfaces* (APIs) que permitan trabajar con cualquiera de los elementos de la página y realizar acciones que antes de la creación de *HTML 5* era necesario realizar por medio de tecnologías adicionales.

HTML 5 incluye novedades significativas en diversos ámbitos:

- Estructura del cuerpo: la mayoría de los sitios Web tienen un formato común, formado por elementos como cabecera, pie, navegadores, entre otros. HTML 5 permite agrupar todas estas partes de una Web en nuevas etiquetas que representarán cada una de las partes típicas de una página.

- Etiquetas para contenido específico: se utilizaba una única etiqueta para incorporar diversos tipos de contenido enriquecido, como animaciones Flash o vídeo. Ahora se utilizan etiquetas específicas para cada tipo de contenido en particular, como audio, vídeo, entre otros.

- *Canvas*: es un nuevo componente permite dibujar en la página todo tipo de formas, por medio de las funciones de un API. Estas formas podrán estar animadas y responder a interacción del usuario sin la necesidad de tener instalado ningún complemento en el navegador.

- Bases de datos locales: el navegador permite el uso de una base de datos local, con la que se podrá trabajar en una página Web por medio del cliente y a través de un API. Es algo análogo a las *Cookies*, pero pensadas para almacenar grandes cantidades de información, lo que permitirá la creación de aplicaciones Web que funcionen sin necesidad de estar conectadas a Internet.

- *Web Workers*: son procesos que requieren bastante tiempo de procesamiento por parte del navegador, pero que se podrán realizar en un segundo plano, para que el usuario no tenga que esperar que se terminen para empezar a usar la página.

- Aplicaciones Web Offline: existe otro API para el trabajo con aplicaciones Web, que se podrán desarrollar de modo que funcionen también de forma local y sin estar conectados a Internet.

- Geolocalización: las páginas Web se pueden localizar geográficamente.

- Fin de las etiquetas de presentación: todas las etiquetas que tienen que ver con la presentación del documento, que modifican estilos de la página, son eliminadas. Por ejemplo las etiquetas , <left> y <right> para posicionar algún elemento de la página. La responsabilidad de definir el aspecto de una Web está a cargo únicamente de las hojas de estilo en cascada CSS o CSS 3, de lo cual se hace referencia a continuación.

2.1.2.1.2 Hojas de Estilo en Cascada, versión 3 (CSS 3)

Las Hojas de Estilo en Cascada se crearon 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. Una de las características principales de *CSS* es su flexibilidad y las diferentes opciones que ofrece para realizar una misma

tarea *(Eguíluz, 2012. a)*.

Desde que *CSS* comenzó han pasado muchos años y ya se encuentra por la especificación de *CSS 3*, la cual incorpora una serie de novedades cuyo objetivo principal es que los desarrolladores tengan un mayor control sobre el estilo con el que se muestran los elementos de las páginas.

Las ventajas principales en esta nueva versión son la inclusión de nuevas propiedades especialmente en cuanto al aspecto gráfico. Incluye bordes redondeados, textos con sombras, la capacidad de asignar múltiples fondos, un mejor manejo de tabla, entre otros. Con las nuevas propiedades, la carga de la página disminuye pues el hecho de que muchos efectos estén bajo el control del navegador hace que diversos recursos visuales e imágenes usadas anteriormente ya no tengan razón de seguir siendo utilizados *(Hogan, 2012. b)*.

2.1.2.1.3 JavaScript

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas Web dinámicas. Una página Web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario (*Eguíluz, 2012. b*).

Técnicamente, *JavaScript* es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. Este lenguaje se define como orientado a objetos.

Entre las acciones típicas que se pueden realizar en *JavaScript* se tienen dos vertientes. Por un lado los efectos especiales sobre páginas Web, para crear contenidos dinámicos y elementos de la página que tengan movimiento, cambien

de color o cualquier otro dinamismo. Por el otro, *JavaScript* nos permite ejecutar instrucciones como respuesta a las acciones del usuario, con lo que se puede crear páginas interactivas con programas como calculadoras, agendas, o tablas de cálculo.

A pesar de su nombre, *JavaScript* no guarda ninguna relación directa con el lenguaje de programación Java, y actualmente se encuentra en su versión 1.8.5 (*Eguíluz, 2012. b*).

2.1.2.1.4 jQuery

jQuery es una biblioteca de *JavaScript,* actualmente en su versión 1.7.2, que permite a los usuarios aplicar funcionalidades dinámicas a las páginas Web con gran facilidad. *jQuery* ofrece varias características de gran alcance, incluyendo la posibilidad de acceder a una parte de la página, modificar el contenido de la misma, añadir animaciones, entre otros *(Harwani, 2012)*.

Esta biblioteca es rápida, concisa y simplifica la manera de interactuar con los documentos *HTML*, así como también el manejo de eventos, animación, y las interacciones para JavaScript asíncrono y *XML* para el desarrollo Web rápido. *jQuery* consiste en un único archivo *JavaScript* que contiene las funcionalidades comunes de manejo de los elementos del *Modelo de Objetos del Documento* o *Document Object Model* (*DOM*), eventos y efectos. Adicionalmente una de las características consideradas más importante de *jQuery* es que es no intrusivo, es decir, que está totalmente separado de la información y estructura sintáctica del sitio**.** *jQuery* está diseñado para cambiar la forma en que se escribe *JavaScript*.

La forma de interactuar con la página es mediante la función *$()*, un alias de *jQuery()*, que recibe como parámetro una expresión *CSS* o el nombre de una etiqueta del documento *HTML* y devuelve todos los nodos (elementos) que concuerden con la expresión.

A continuación se listan de forma concreta las características de esta biblioteca:

- Selección de elementos, interactividad y modificaciones del árbol DOM.
- Manejo de eventos.
- Manipulación de la hoja de estilos CSS.
- Efectos y animaciones.
- Animaciones personalizadas.
- Funcionalidades de interacciones para JavaScript asíncrono y XML.
- Soporte de extensiones.
- Utilidades varias como obtener información del navegador, operar con objetos y vectores, funciones como trim() (elimina los espacios en blanco del principio y final de una cadena de caracteres), entre otros.
- Compatible con los navegadores Mozilla Firefox 2.0+, Internet Explorer 6+, Safari 3+, Opera 10.6+ y Google Chrome 8+.

2.1.2.1.5 jQuery User Interface (jQuery UI)

jQuery UI es una biblioteca de componentes para *jQuery* que le añade un conjunto de complementos, *widgets* y efectos visuales para la creación de aplicaciones Web, actualmente en su versión 1.8.20 (*jQuery UI, 2012*).

La biblioteca se divide en cuatro módulos:

- Núcleo: contiene las funciones básicas para el resto de módulos como lo son:

    - *Core*: requerido para todas las interacciones y *widgets*.
    - *Widget*: la fábrica de *widget*, base para todos los *widgets*.
    - *Mouse*: el *widget* de ratón, una clase base para todas las interacciones y *widgets* con la interacción a través del ratón.
    - *Position*: una extensión de utilidad para los elementos de posicionamiento en relación con otros elementos.

- Interacciones: añade comportamientos complejos a los elementos, por ejemplo:

    - *Draggable*: hace que cualquier elemento de la página se pueda arrastrar.
    - *Droppable*: habilita elementos donde los elementos que fueron arrastrados son liberados.
    - *Resizable*: hace que a cualquier elemento de la página se le pueda modificar su tamaño.
    - *Selectable*: hace una lista de elementos seleccionables a través del ratón arrastrando un cuadro o haciendo clic en ellos.
    - *Sortable*: hace una lista de elementos que se pueden ordenar.

- *Widgets*: es un conjunto completo de controles de interfaz. Cada control tiene un conjunto de opciones configurables y se les puede aplicar estilos *CSS*, por ejemplo:

    - *Accordion*: crea un *widget* de navegación en forma de acordeón.
    - *Autocomplete*: crea un *widget* de autocompletado.
    - *Button*: crea un *widget* de botón.

- *Dialog:* abre un cuadro de diálogo que puede arrastrarse y redimensionarse.
- *Slider*: crea un control deslizable con rangos y accesible a través del teclado.
- *Tabs*: transforma un conjunto de contenedores en una estructura de pestañas.
- *Datepicker*: un selector de fechas que puede ser activado a través de un elemento de entrada.
- *Progressbar*: un indicador de estado que puede ser utilizado para un estado de carga.

- Efectos: una API para añadir transiciones animadas y facilidades para interacciones.

2.1.2.1.6 JavaScript y XML (AJAX) Asíncrono

Una vez descrito *JavaScript* y la biblioteca *jQuery*, se presenta ahora una técnica de desarrollo que tiene mucho que ver con estas tecnologías.

AJAX significa *JavaScript* asíncrono y *XML*. Aunque en sentido estricto *AJAX* no es en sí una tecnología, mezcla técnicas de programación bien conocidas de una manera poco común para que los desarrolladores Web creen aplicaciones con interfaces de usuario mucho más atractivas (*Ballard & Moncur, 2012*).

Es una técnica de desarrollo Web para crear aplicaciones interactivas mediante la combinación de tres tecnologías ya existentes: *HTML* y Hojas de Estilo en Cascada (*CSS*) para presentar la información; *JavaScript*, para interactuar dinámicamente con los datos, y lenguaje de marcado extensible o *eXtensible Markup Language* (*XML*) y para intercambiar y manipular datos con un servidor Web (aunque las aplicaciones que usan *AJAX* pueden usar otro tipo de formato, incluyendo texto plano, para realizar esta labor).

AJAX es una tecnología asíncrona, en el sentido de que los datos adicionales se solicitan al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página. *JavaScript* es el lenguaje interpretado en el que normalmente se efectúan las funciones de llamada de *AJAX* mientras que el acceso a los datos se realiza mediante *XMLHttpRequest*, objeto disponible en los navegadores actuales.

En las aplicaciones Web tradicionales los usuarios interactúan mediante formularios, que al enviarse, realizan una petición al servidor Web. El servidor se comporta según lo enviado en el formulario y contesta enviando una nueva página Web. Se desperdicia mucho ancho de banda, ya que gran parte del *HTML* enviado en la segunda página Web, ya estaba presente en la primera. Además, de esta manera no es posible crear aplicaciones con un alto grado de interacción. En la Ilustración 5 se ve el funcionamiento de una aplicación Web tradicional.

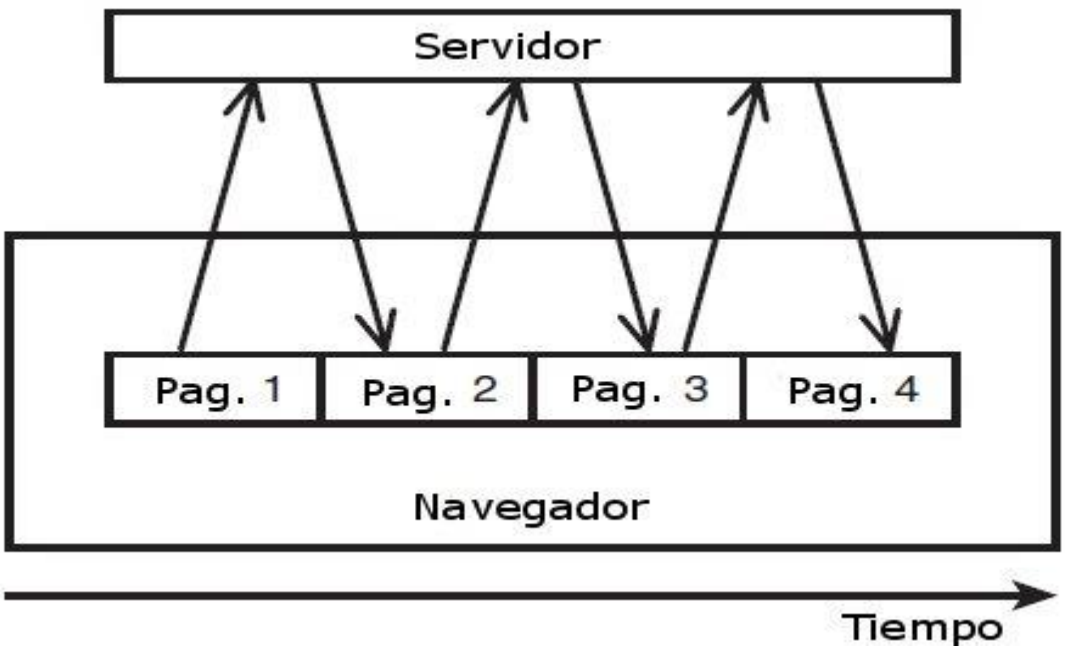


Ilustración 5: Funcionamiento de una aplicación Web tradicional *(Ballard & Moncur, 2012)*.

En aplicaciones *AJAX* se pueden enviar peticiones al servidor Web para

obtener únicamente la información necesaria, empleando *Simple Object Access Protocol (SOAP)* o algún otro protocolo para servicios Web basado en *XML*, y usando *JavaScript* en el cliente para procesar la respuesta del servidor Web. Esto resulta en una mayor interacción gracias a la reducción de información intercambiada entre servidor y cliente, además de que parte del proceso de la información lo hace el propio cliente, liberando al servidor de ese trabajo. Lo negativo de este proceso es que la descarga inicial de la página es más lenta al tener que obtener todo el código *JavaScript*. En la Ilustración 6 se ve el funcionamiento de una aplicación Web con *AJAX*.

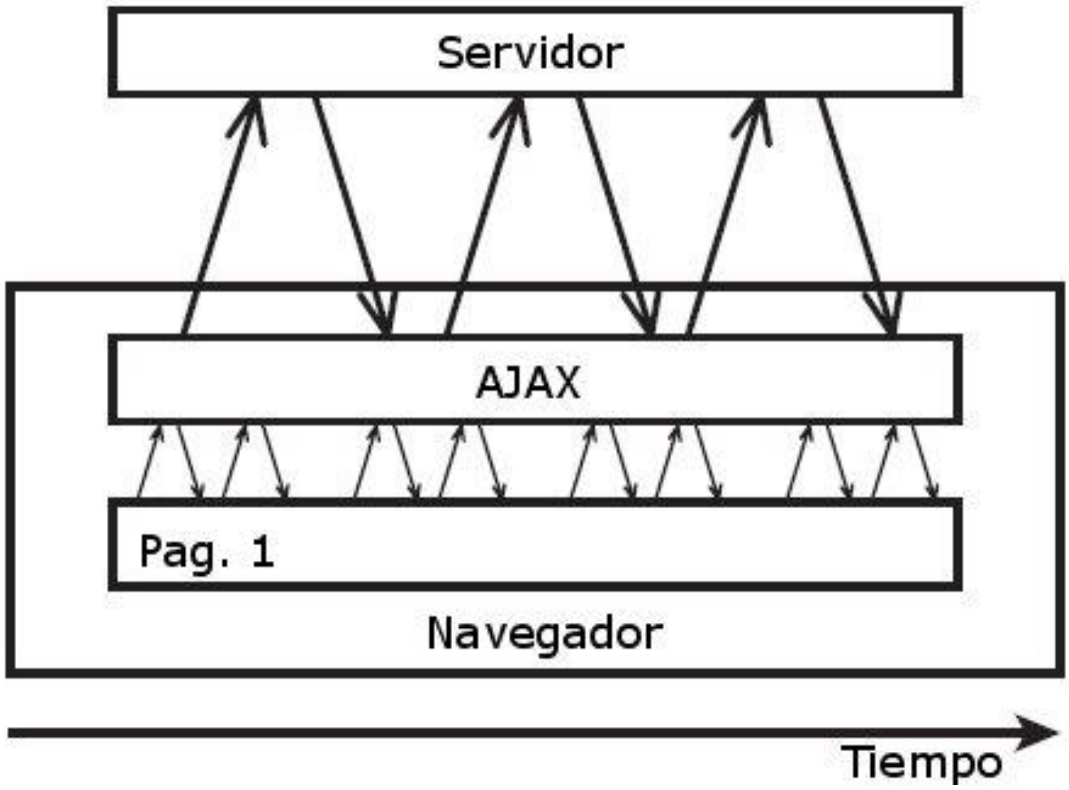


Ilustración 6: Funcionamiento de una aplicación Web con *AJAX (Ballard & Moncur, 2012).*

2.1.2.2 Tecnologías del lado del Servidor

Para complementar las tecnologías que serán usadas, se debe mencionar las del lado del servidor. En el cual posee la habilidad de correr programas que

interactúan con los clientes o las páginas de los sitios Web. Estos pequeños programas residen en los servidores Web y permiten tener interactividad en los sitios Web. Por ejemplo, si en un sitio Web se necesita que todos los usuarios entren con contraseña, la base de datos que contiene los nombres de usuario y contraseñas para dicho sitio Web, y el programa que ofrece la autenticación de acceso se basará en el servidor Web y no se encuentra en la página Web.

2.1.2.2.1 Plataforma Debian GNU/Linux

Para empezar con estas tecnologías se debe hablar de la elección del sistema operativo, que en este caso se desarrolló gracias al *Proyecto Debian*, el cual es una asociación de personas que han hecho causa común para crear un sistema operativo libre, y se llama *Debian GNU/Linux*.

Un sistema operativo es un programa o conjunto de programas que en un sistema informático gestiona los recursos de *hardware* y provee servicios a los programas de aplicación, ejecutándose en modo privilegiado respecto de los restantes. El centro de un sistema operativo es el núcleo. El núcleo es el programa más importante en la computadora, realiza todo el trabajo básico y le permite ejecutar otros programas. Los sistemas *Debian* actualmente usa el núcleo de *Linux*. *Linux* es una pieza de software creada en un principio por Linus Torvalds y soportada por miles de programadores a lo largo del mundo *(Debian, 2012)*.

Debian se caracteriza por:

- La disponibilidad en varias arquitecturas.
- Una amplia colección de software disponible.
- Un grupo de herramientas para facilitar el proceso de instalación y actualización del *software (apt, aptitude, dpkg, synaptic, dselect,* entre

otras). Todas ellas obtienen información de donde descargar *software* desde */etc/apt/sources.list*, que contiene los repositorios.

- Su compromiso con los principios y valores involucrados en el movimiento del *Software* Libre.

2.1.2.2.2 Apache

El software servidor que se va a utilizar en el sistema operativo antes mencionado será el *HTTP* Apache de código abierto para plataformas *Unix*, *Windows, Macintosh* y otras, que implementa el protocolo *HTTP* y la noción de sitio virtual. Cuando comenzó su desarrollo en 1995 se basó inicialmente en código del popular *NCSA HTTPd* 1.3, pero más tarde fue reescrito por completo. *(Apache, 2012)*

Apache presenta entre otras características mensajes de error altamente configurables, bases de datos de autenticación y negociado de contenido. La a rquitectura del servidor Apache es modular, lo cual implica que puede ser adaptado a diferentes entornos y necesidades, con los diferentes módulos de apoyo que proporciona, y con la Interfaz de Programación de Aplicaciones necesaria. El servidor consta de una sección núcleo y diversos módulos que aportan muchas funcionalidades que podrían considerarse básicas para un servidor Web. Entre las ventajas de utilizar apache se encuentran que es multiplataforma, extensible, fácil para conseguir ayuda y/o documentación y es gratuito.

2.1.2.2.3 Unicorn

Otro servidor que se utiliza es Unicorn, el cual es un servidor *HTTP* para aplicaciones *Rack*, cuyas peticiones y respuestas *HTTP* son envueltas de la manera más simple posible, unificando la API para los servidores web, los *frameworks* y el

software entre ellos, en una única llamada. Este servidor es diseñado para servir rápidamente a los clientes en baja latencia, las conexiones de gran ancho de banda y aprovechar las características de *Unix*. Los clientes lentos sólo deben ser atendidos por la colocación de un Proxy reverso (recibe todo el tráfico procente de Internet y con destino en el servidor) capaz de amortiguar tanto la solicitud como la respuesta en medio de Unicorn y los clientes lentos *(Unicorn, 2012)*.

Unicorn es compatible con Ruby versión 1.8 y 1.9.

2.1.2.2.4 Ruby on Rails

Para implementar la aplicación Web, se usa un *framework* llamado *Ruby on Rails*, ya que permite facilidad en el desarrollo, implementación y en el mantenimiento de la misma. Durante los meses que siguieron a su lanzamiento inicial, *Rails* pasó de ser un "juguete desconocido" a ser un fenómeno en todo el mundo, y más importante, se ha convertido en el marco de elección para la aplicación de una amplia gama de los llamados "aplicaciones Web 2.0" *(Ruby & Thomas & Hansson, 2012)*.

Todas las aplicaciones Rails se implementan utilizando el Modelo Vista Controlador (*MVC*). Sin embargo, Rails tiene *MVC* más allá: cuando se desarrolla en Rails, se comienza con una solicitud de trabajo, hay un lugar de cada pieza de código, y todas las partes de la aplicación interactúan en una forma estándar. Los programadores profesionales pueden escribir pruebas. A medida que se agregan funcionalidades al código, *Rails* crea automáticamente talones de prueba para esa funcionalidad.

Las aplicaciones de *Rails* están escritas en *Ruby*, un lenguaje moderno y orientado a objetos de secuencias de comandos. *Ruby* es conciso, puede expresar las ideas de manera natural. Esto conduce a que los programas sean fáciles de escribir y (tan importante) sean fáciles de leer meses después. *Rails* tiene *Ruby* para el límite, que se extiende en nuevas formas que hacen que la vida de

programador sea más sencilla. Lo que implica que los programas puedan ser más cortos y más legibles *(Ruby & Thomas & Hansson, 2012)*.

Dos fundamentos filosóficos para mantener el código de *Rails* corto y legible: DRY (*Don´t Repeat Yourself*) y la convención sobre la configuración. DRY se refiere a que no repitas: cada pieza de conocimiento en un sistema se debe expresar en un solo lugar. *Rails* utiliza la potencia de Ruby para traer esto a la vida. Se puede encontrar una duplicación muy poco en una aplicación *Rails*. Para los programadores que utilizan otros entornos Web, donde un simple cambio en el esquema podría involucrarlos en una media docena o más cambios en el código se creó lo que se llama convención sobre la configuración. Esto quiere decir que *Rails* tiene parámetros por defecto para casi todos los aspectos de género de punto, junto a su solicitud.

Si se siguen las convenciones, se puede escribir una aplicación *Rails* con menos código que una típica aplicación Web *Java*, la cual utiliza una configuración de *XML*. *Rails* permite facilidad para los desarrolladores al momento de integrar características tales como *Ajax* y *REST* en su código, ya que el apoyo está construido adentro.

2.1.2.3 Tecnología del lado del Servidor de Bases de Datos

Debido a que la aplicación Web a desarrollar va a manejar grandes y complejos volúmenes de datos, es necesario hacer uso de los servidores de bases de datos, para así poder compartir la información con un conjunto de clientes de manera segura. Ante este enfoque, un sistema manejador de base de datos deberá ofrecer soluciones de forma fiable, rentable y de alto rendimiento.

2.1.2.3.1 MySQL

Se va a utilizar el sistema manejador de base de datos relacional (SMBDR)

llamado *MySQL*, que es multihilo y multiusuario. *MySQL AB* es una compañía comercial de software libre, que distribuye y soporta *MySQL*, y además lo desarrolla bajo un esquema de licencia dual. Por un lado se tiene la licencia *GNU GPL* (General Public License GNU) y por otro, aquellas empresas que desean utilizar este sistema en productos privados, pueden comprar la licencia que les permita su uso.

MySQL es ampliamente utilizado en aplicaciones Web, en plataformas (*Linux/Windows-Apache-MySQL- PHP/Perl/Python*), y por herramientas de seguimiento de errores. Su popularidad en las aplicaciones Web está muy ligada a PHP, que a menudo aparece en combinación con *MySQL*, sin embargo existen varias APIs que permiten a las aplicaciones escritas en diversos lenguajes de programación, acceder a las bases de datos *MySQL*, como son *C, C++, C#, Pascal, Delphi* (vía *dbExpress*), *Eiffel, Smalltalk, Java* (con una implementación nativa del driver de *Java*), entre otros.

Algunas características de *MySQL* son las siguientes *(MySQL, 2012)*:

- Interioridad y portabilidad

    - Escrito en C y C++.
    - Es multiplataforma. Funciona en plataformas como AIX, BSD, FreeBSD, HP- UX, GNU/Linux, Mac OS X, NetBSD, Windows 95/98/NT/2000/XP/Vista y otras versiones de Windows, entre otros.
    - Es multihilo. Utiliza hilos del kernel, aprovechando así la ventaja de usar multiprocesamiento con varios procesadores.
    - Proporciona sistemas de almacenamiento transaccional y no transaccional.
    - Múltiples motores de almacenamiento (MyISAM, Merge, InnoDB, Memory/heap, entre otros), permitiendo al usuario escoger la que sea más adecuada para cada tabla de la base de datos.

- Tipos de Datos

    - Posee diversos tipos de datos: enteros con/sin signo de 1, 2, 3, 4, y 8 *bytes* de longitud*, float, double, char, varchar, text, blob, date, time, datetime, timestamp, year, set, enum*, y tipos espaciales *OpenGIS*.
    - Tiene registros de longitud fija y longitud variable.

- Sentencias y funciones

    - Soporte completo para operadores y funciones en las cláusulas de consultas *Select* y *Where*.
    - Soporte completo para las cláusulas sql group by y *order by*. Soporte de funciones de agrupación (*count(), count(distinct ...), avg(), std(), sum(), max(), min(), y group_concat())*.

    - Puede mezclar tablas de distintas bases de datos en la misma consulta.

- Seguridad

    - Tiene un sistema de privilegios y contraseñas que es muy flexible y seguro, y que permite verificación basada en el *Host*. Las contraseñas son seguras porque todo el tráfico de contraseñas está encriptado cuando se conecta con un servidor.

2.2 Comparación de Documentos Digitales

En esta sección se presentan diversos conceptos y técnicas relacionados con la comparación de documentos digitales, específicamente se introduce el concepto

de documento digital, y se propone la utilización de una técnica, que mediante el desarrollo de un algoritmo, permita la corrección automática de la evaluación instrumental.

2.2.1 Documentos digitales

Gracias al desarrollo y a los avances en las tecnologías informáticas, se ha introducido el concepto de documento digital. Se define como documento a "cualquier unidad significativa de información que haya sido registrada en un soporte que permita su almacenamiento y posterior recuperación" (MSINFO Sistemas de Información). Es un "diploma, carta, relación u otro escrito que ilustra acerca de un hecho, principalmente histórico" (LAROUSSE, 1997), en general, "cualquier otra cosa que sirve para ilustrar o comprobar algo" (LAROUSSE, 1997). Son documentos, por ejemplo, los libros, las revistas, los informes, las facturas, entre otros.

La información digital se puede definir como "todo aquello que está representado mediante ceros y unos dentro de una computadora" (Biblioteca Digital Universitaria de la DGSCA). Algunos ejemplos de información digital incluyen textos, videos, imágenes, sonidos, entre otros, los cuales están representados y codificados en distintos formatos.

Uniendo estas definiciones, se puede decir que un documento digital es un documento cuya información está representada mediante ceros y unos, y que puede ser visualizada a través de un dispositivo informático. Estos son almacenados en diversos formatos, dependiendo de la aplicación con la que fueron creados. Dentro de los más comunes, se encuentra el formato *PDF*, acrónimo de Formato de Documento Portable (Portable Document Format), creado por *Adobe Systems*, el cual "permite obtener y visualizar información desde cualquier aplicación y en cualquier sistema informático, así como compartirla con prácticamente cualquier persona en cualquier sitio" (Adobe). Es un formato de almacenamiento de documentos que conserva todas las fuentes, formatos, colores y gráficos con los que

fueron creados, manteniendo la presentación final del mismo.

2.2.2 Técnicas para Detectar la Copia de Documentos Digitales

En el trabajo de seminario "Estudio de Técnicas de Medición de Similitud entre Documentos Digitales" (Klopp N, 2008) se plantearon distintas técnicas para la detección de copia de documentos, las cuales se explican a continuación.

2.2.2.1 Técnica: Detección de Documentos Similares usando Términos Importantes

En esta técnica se extraen los términos de los documentos a comparar y se le asignan un nivel de importancia el cual se denomina como *IQ*. Un término que aparece en apenas unos cuantos documentos es altamente selectivo y tendrá un alto *IQ*. Si, por lo contrario, aparece en muchos documentos, es menos selectivo y tendrá un *IQ* bajo. Se buscan los términos más significantes de todos los documentos a comparar (los que tienen un *IQ* alto) y se procede a hacer la comparación entre cada par de documentos (por ejemplo, sean d1, d2 y d3 los documentos a comprar, la comparación se realiza entre d1 con d2, d1 con d3, y d2 con d3). Para ello, se consultan aquellos términos que se encuentren en ambos documentos y, si la cantidad de términos resultantes de esta consulta es mayor que cierto umbral preestablecido, se identifica los documentos como similares (*Cooper, Coden, & Brown, 2002*).

El esquema general se muestra en la Ilustración 7. En primer lugar se realiza la extracción de términos y se les asignan los respectivos *IQ*. Luego, para generar el reporte, se procede a comparar los términos importantes para comprobar si se produce la superación del umbral preestablecido.

Ilustración 7: Detección de documentos similares usando términos importantes (Flores & klopp, 2008).

2.2.2.2 Técnica: Mecanismos de Detección de Copia para Documentos Digitales

En esta técnica los documentos son llevados a texto plano y son divididos en secuencias de unidades consecutivas llamados *chunk*, donde las unidades pueden ser secciones, párrafos, sentencias, palabras y caracteres. Por ejemplo, se puede establecer como *chunk* una secuencia de cinco palabras, de dos sentencias, una palabra individual, entre otros (*Brin, Davis & García-Molina, 1995*).

A cada *chunk* se le asigna un código hash, los cuales son almacenados en la base de datos junto al documento en los que aparecen.

Cuando un documento va a ser comparado, se lleva igualmente a texto plano y se procede realizar la extracción de *chunks*. De la misma manera, se asigna un código hash y se busca en la base de datos. Si existe, es contada como una similitud. Si la cantidad de similitudes es mayor a un umbral preestablecido, los documentos se consideran copiados.

En la Ilustración 8 se muestra el esquema general de esta técnica.

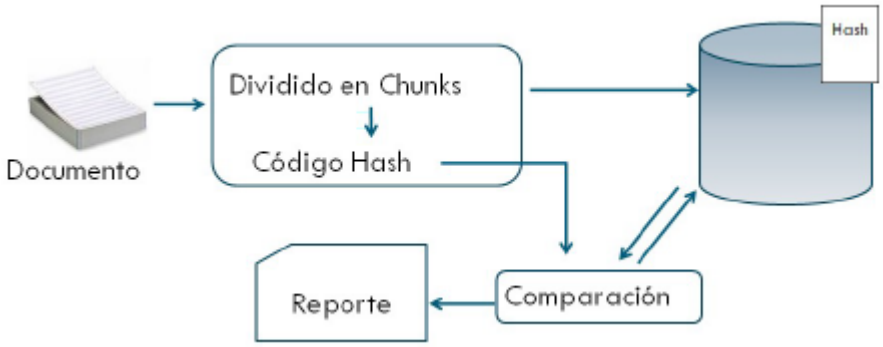


Ilustración 8: Mecanismo de detección de copia entre documentos digitales (Flores & klopp, 2008).

2.2.2.3 Técnica Check: Sistema de Detección de Plagio de Documentos

Esta técnica se basa en una comparación de términos presentes en los documentos para determinar la similitud entre estos. Por cada documento, sección, subsección o párrafo, se extraen los términos relevantes (es decir, aquellos que por lo general dan significado al documento como sustantivos, verbos, adjetivos y adverbios) que lo conforman y se les asigna un peso, correspondiente a la frecuencia con la que este aparece, dividido entre la cantidad total de términos. Se crean dos vectores, uno de términos y otro de pesos, los cuales serán usados como parámetros en una función que retorna un estimador de similitud (*Si, Leong, & Lau, 1997*).

A continuación en la Ilustración 9 se presenta un cuadro comparativo con las técnicas mencionadas anteriormente en cuanto a las acciones que cumple cada una.

Técnica/Acción	Compara secciones del documento	Compara documentos de distintos tamaños	Detecta similitudes	Detecta copias
Detección de	NO	NO	SÍ	NO

Documentos Similares usando Términos Importantes.				
Mecanismos de detección de Copia para documentos digitales.	SÍ	SÍ	NO	SÍ
Check: Sistema de Detección de Plagio de Documentos	SÍ	SÍ	SÍ	SÍ

Ilustración 9: Cuadro comparativo de técnicas para detección de copias de documentos digitales.

De diversas técnicas estudiadas, la elegida para elaborar el algoritmo de corrección automática es la técnica *Check*, ya que permite detectar, además de copias, similitudes y también es posible comparar documentos de distintos tamaños, las cuales son las condiciones del problema planteado, ya que los textos de las traducciones pueden ser de distintos tamaños, y lo más importante, es que el algoritmo cuantifica la similitud con un valor comprendido en el intervalo [0,1] (al igual que la calificación para la traducción otorgada por el personal que corrige la evaluación). Para ello se tomará en cuenta una adaptación de esta técnica, la cual es abarcada en el siguiente punto.

2.2.3 Adaptación de la Técnica Check

Para la elaboración del módulo de corrección automática de la evaluación instrumental se propone una adaptación de la técnica Check. Para ello se tomará en cuenta (*Flores & klopp, 2008*):

- Extracción de palabras clave: sólo se toman en cuenta las palabras clave que son las que, por lo general, dan significado al texto, como sustantivos verbos, adjetivos y adverbios.

- Asignación de pesos para cada palabra clave: por cada palabra clave, se asigna un peso correspondiente a la frecuencia en la que esta aparece dentro de la sección, entre la cantidad total de palabras de la misma. Se consideró como sección a los párrafos.

- Generación de los vectores normalizados y de referencia: se construye un vector de referencia uniendo todas las palabras clave de dos párrafos a comparar, sin repetición. Una vez construido este vector, se procede a elaborar un vector normalizado para cada párrafo. Este vector se crea asignando, por cada palabra del vector de referencia, el peso que esta tiene dentro del párrafo. Si la palabra no existe en el párrafo, se asigna el valor 0. Más adelante, esto será explicado mediante un ejemplo.

- Cálculo del coseno del ángulo entre vectores para determinar la similitud: se calcula la similitud de los dos vectores normalizados correspondientes a cada párrafo mediante el cálculo del coseno del ángulo entre estos vectores. La fórmula para calcular el coseno del ángulo entre vectores se muestra a continuación en la Ilustración 10:

$$S(V_A, V_B) = \frac{\sum_{i=1}^{|R|} xA_i xB_i}{\sqrt{\sum_{i=1}^{|R|} xA_i{}^2 \sum_{i=1}^{|R|} xB_i{}^2}}$$

Ilustración 10: Fórmula para el cálculo del coseno del ángulo entre dos vectores

Donde:

S(VA, VB) es un valor comprendido entre 0 y 1, el cual corresponde a la similitud existente entre dos párrafos denotados por A y B, siendo 0 el valor que toma cuando no existen coincidencias entre ambos párrafos y 1, cuando ambos párrafos son iguales.

|R| es el tamaño dado por el vector de referencia.

Xai y Xbi son los pesos, determinados por la posición i de cada vector normalizado.

A continuación se explica, mediante un ejemplo, el funcionamiento de la adaptación que se hizo de la técnica *Check* para ser utilizada en el desarrollo de la del módulo de corrección de la aplicación Web planteada en este trabajo, detallando cada uno de los pasos que se llevaron a cabo.

Dado un documento a comparar, el cual se denotará de aquí en adelante como dA, y un documento contra el que se desea realizar la comparación, el cual se denotará como dB, se llevan a cabo los siguientes pasos:

Paso 1 - Extracción de palabras clave de los párrafos

Para todos los párrafos de dA y dB, se suprimen las palabras que no son clave (aquellas que por lo general no agregan significado al documento, como artículos, preposiciones, conectores, etc.).

Sean pA y pB dos párrafos en particular de dA y dB respectivamente, los cuales tienen el siguiente texto:

pA = “La metodología AGILUS es una metodología de desarrollo de aplicaciones que provee una gran flexibilidad en el desarrollo de software”

pB = “La metodología de desarrollo de aplicaciones AGILUS es una metodología que brinda una gran adaptabilidad en la construcción de éstas”

Se extraen las palabras clave, denotadas por VA y VB, y se le asignan los pesos correspondientes, denotados como WA y WB, para los párrafos pA y pB respectivamente. Cada peso corresponde a la cantidad de veces que aparece una palabra en el texto dividida entre la cantidad de palabras clave del mismo:

VA = [metodología, agilus, es, desarrollo, aplicaciones, provee, gran, flexibilidad, software]

WA = [2/11, 1/11, 1/11, 2/11, 1/11, 1/11, 1/11, 1/11, 1/11]

VB = [metodología, desarrollo, aplicaciones, agilus, es, brinda, gran, adaptabilidad, construcción]

WB = [2/10, 1/10, 1/10, 1/10, 1/10, 1/10, 1/10, 1/10, 1/10]

Paso 2 - Comparación de los párrafos

Se genera el vector de referencia, dentado como R, uniendo las palabras claves de ambos párrafos:

R = [metodología, agilus, es, desarrollo, aplicaciones, provee, gran, flexibilidad, software, brinda, adaptabilidad, construcción]

Se generan los vectores normalizados, denotados por XA y XB. Para ello, se buscan las palabras del vector R dentro del vector de palabras clave de ambos párrafos, y se asigna el peso asociado dentro del mismo. Por ejemplo, la palabra metodología aparece en VA con un peso igual a 2/11 y en VB, con un peso de 2/10. Ambos valores se asignan en la misma posición (en este caso, la primera posición del vector) de los vectores normalizados correspondientes. Continuando con las

palabras del vector de referencia, agilus aparece en ambos párrafos con un peso de 1/11 y 1/10. Igualmente, esta entrada se almacena en ambos vectores normalizados, que en este caso, corresponden a la segunda posición. Si la palabra del vector de referencia no aparece en alguno de los vectores, en su vector normalizado se coloca el valor 0. Estos vectores se muestran a continuación con todos los valores establecidos.

XA = [2/11, 1/11, 1/11, 2/11, 1/11, 1/11, 1/11, 1/11, 1/11, 0, 0, 0]

XB = [2/10, 1/10, 1/10, 1/10, 1/10, 0, 1/10, 0, 0, 1/10, 1/10, 1/10]

Una vez construidos los vectores normalizados, se lleva a cabo el cálculo del coseno del ángulo entre vectores utilizando la formula expuesta en la Ilustración 10.

En la Ilustración 11 se muestra el cálculo del coseno del ángulo entre vectores para los vectores XA y XB.

$$\frac{\frac{1}{11}}{\sqrt{\frac{15}{121}\frac{3}{25}}}$$

Ilustración 11: Cálculo del coseno del ángulo para los vectores XA y XB.

Al comparar estos párrafos, la coincidencia obtenida fue de 0,74536 o 74,536% de similitud.

Capítulo III Marco Aplicativo

El presente capítulo trata sobre la programación *XP* y la aplicación.

Se presenta una adaptación del proceso de desarrollo de software al caso particular de estudio, basado en el método de *Programación Extrema (XP)* para la construcción de los módulos de la aplicación Web. En este sentido se describe el contexto de desarrollo, y cada una de las fases del método desarrollo utilizado para la automatización y estandarización de los procesos relacionados con la gestión, elaboración, presentación y corrección de la evaluación instrumental de idiomas.

3.1 Programación Extrema (Extreme Programming)

XP (*eXtreme Programming*) se trata de un método ágil. Se basa en la simplicidad, la comunicación y la retroalimentación o reutilización del código desarrollado. Se enfoca en los requerimientos comunicados por el cliente. El objetivo principal que se persigue es la satisfacción del cliente, por eso tiene mucha importancia la comunicación con los usuarios o clientes. Esta comunicación se va a soportar principalmente en las historias de usuario (*UserStories*) cuando proviene desde el cliente, y de las entregas y versiones parciales del sistema cuando la comunicación es hacia el cliente *(Beck, 1997)*. Una posible adaptación de este método incluye fases como planificación, diseño, codificación y pruebas (Ilustración 12), las cuales serán detalladas en la sección de adaptación de las actividades en *XP*.

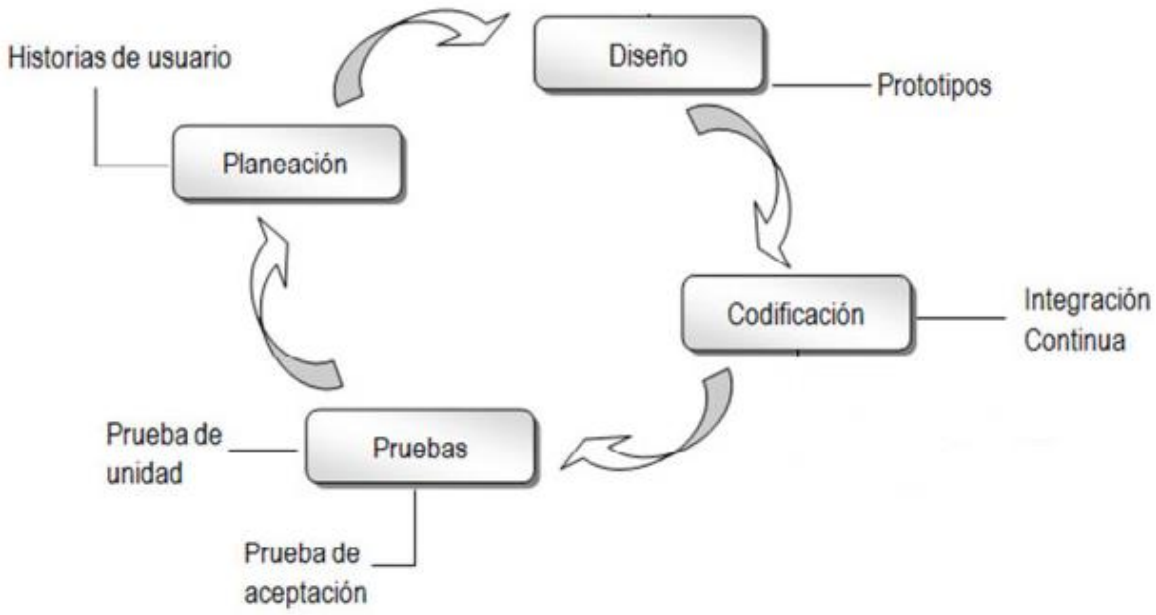


Ilustración 12: Programación Extrema *(Metodología ágil, 2011*).

3.1.1 Iteraciones

El desarrollo está basado en iteraciones. La idea de las iteraciones es ir trabajando sobre versiones pequeñas o parciales del sistema hasta llegar al producto final. En el desarrollo de la aplicación se reciben los requerimientos y la retroalimentación progresivamente. En éste sistema en las primeras iteraciones se trata de realizar una arquitectura de sistema que pueda ser utilizada durante el resto del proyecto.

Las iteraciones pueden ser de dos tipos principalmente: por objetivos o por lapsos de tiempo. En el desarrollo de este sistema las iteraciones están basadas en lapsos de tiempo estimados a una semana. Durante el tiempo fijado para cada iteración se realizan las implementaciones indicadas en las historias de usuario que abarque.

3.1.2 Historias de Usuario

Las historias de usuario son la técnica utilizada para especificar los requisitos del software, éstas tienen el propósito de describir los requerimientos de los clientes. El contenido de las historias de usuario proviene del cliente.

En cada iteración se lleva a cabo un grupo de historias, es decir, en cada iteración se trabaja sobre uno o más requerimientos, tal como se puede apreciar en la Ilustración 13.

Iteración 0	Historia de usuario 1 Historia de usuario 2 . .
Iteración 1	. . Historia de usuario 5 Historia de usuario 6 . .
. .	. .

Ilustración 13: Iteraciones e historias de usuario.

El tratamiento de las historias de usuario es muy dinámico y flexible, en cualquier momento las historias de usuario pueden romperse, reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en un tiempo corto.

Cabe destacar que se tienen tres variantes de historia de usuario para los requerimientos: nueva, corrección y mejora.

Con respecto a la información contenida en la historia de usuario, existen varias plantillas sugeridas pero no un único formato a seguir.

En esta aplicación se utilizará el formato mostrado en la Ilustración 14:

Número	Descripción	Tipo	Tiempo estimado

Ilustración 14: Formato de historias de usuario.

3.1.3 Actividades en *XP*

- Codificar: en *XP* se considera que el único elemento importante en el proceso de desarrollo es el código (instrucciones de software que un computador pueda interpretar). Esta fase de codificación expresa la interpretación del problema. La codificación también pueden ser usada para determinar la solución más adecuada.

- Hacer pruebas: en esta fase se realizan pruebas unitarias y pruebas de aceptación. Las pruebas unitarias sirven para determinar si una determinada característica funciona según lo previsto. Las pruebas de aceptación verifican que los requisitos fueron comprendidos con exactitud por los programadores para satisfacer las necesidades reales del cliente.

- Escuchar: Los programadores tienen que escuchar lo que los clientes necesitan que el sistema realice y la "lógica de negocio" que se necesita. Ellos deben entender estas necesidades lo suficientemente bien para dar la retroalimentación a los usuarios sobre los aspectos técnicos de cómo el problema puede ser resuelto.

- Diseñar: el diseño crea una estructura que organiza la lógica del sistema. Desde el punto de vista de la simplicidad se puede decir que el desarrollo del

sistema no necesita más que la codificación, la prueba y escuchar al usuario. Si esas actividades se llevan a cabo así, el resultado debe ser siempre un sistema que funciona. En la práctica, esto no funcionará. Se puede llegar muy lejos sin el diseño, pero el sistema puede tornarse muy complejo y las dependencias del sistema dejan de ser claras. Se puede evitar esto mediante la creación de una estructura de diseño que organice la lógica del sistema.

3.1.4 Adaptación de las actividades en XP

La aplicación de las tareas *XP* busca lograr con éxito la alta comunicación con el cliente y la agilidad en el proceso de desarrollo.

En este proyecto se tienen cuatro tareas fundamentales durante las iteraciones:

- Planificación: en esta fase se utilizan las historias de usuario con el formato mostrado anteriormente.

- Diseño: se hacen las definiciones y diagramas correspondientes a cada iteración, de manera que se pueda comprender y solucionar el problema.

- Codificación: el código de implementación del sistema se registra y sincroniza constantemente en un servidor de control de versiones (*git*). Se muestran porciones de código fuente que sean esenciales en la comprensión del sistema y la solución a los requerimientos de las historias de usuario.

- Pruebas: las pruebas serán de aceptación, en las cuales el usuario o cliente pone a prueba el sistema y verifica que hayan quedado cubiertos los requerimientos. También se realizan pruebas unitarias.

3.2 Desarrollo de la Aplicación

En esta sección se presenta una adaptación del proceso de desarrollo de software al caso particular de estudio, basado en el método de Programación Extrema. En este sentido se describe el contexto de desarrollo, y cada una de las iteraciones (por objetivos) realizadas para la automatización de los procesos relacionados a la elaboración, presentación y corrección de la evaluación instrumental que realiza la Escuela de Idiomas Modernos de la Universidad Central de Venezuela.

Se exponen las actividades de desarrollo e implementación de cada iteración, siguiendo el esquema propuesto correspondiente al método de desarrollo, documentando las actividades de diseño e implementación y las estrategias de pruebas.

3.2.1 Iteración 0: Requerimientos del sistema

Mar 01/Ene/2013 - Viernes 11/Ene/2013

En esta iteración se definen los requerimientos generales del sistema tomando como base la propuesta elaborada. Adicionalmente, los componentes de dicho sistema son representados a través de un esquema gráfico (metáfora), a partir del cual se desarrollan las demás iteraciones.

3.2.1.1 Diseño

El sistema debe contar con un módulo que permita la elaboración del examen instrumental en línea, ingresando el contenido en un editor de texto enriquecido integrad a la aplicación, luego eligiendo el párrafo o los párrafos a ser traducidos y posteriormente ingresando una traducción correcta para la sección del texto a traducir. También debe contar con otro módulo que permita presentar dicho examen almacenando las respuestas al mismo por parte de los estudiantes como documentos digitales, y controlando las características de la presentación

como la verificación de identidad para acceder a la presentación y el tiempo estipulado para realizar las actividades. Además debe tener un módulo que se encargue de proveer la corrección automática de la parte de traducción de la evaluación mediante la adaptación de un algoritmo, que en principio es utilizado para la detección de plagios, el cual arroja un número entre 0 y 1 que determina el porcentaje de similitud entre la traducción correcta y la del estudiante. Este sistema está destinado a ser usado por el personal de la Escuela de Idiomas Modernos y por los aspirantes del título de algunos de los diferentes postgrados que ofrece la Universidad Central de Venezuela.

- Requerimientos funcionales

Los requerimientos funcionales describen el comportamiento, funciones o servicios del sistema, y realizan los objetivos, tareas o actividades solicitadas por el usuario. Después de dialogar con el usuario y escuchar los procesos que son necesarios que el sistema ejecute, se definieron los siguientes requerimientos funcionales, destacando la importancia de contar con un componente de software que calcule un valor para calificar de manera automática la parte traducción de una evaluación instrumental:

- Elaborar la evaluación instrumental en línea, catalogando cada uno de los exámenes creados según el postgrado al que éste esté asociado, y almacenando su contenido como un documento digital.

- Controlar el acceso de cada estudiante a la presentación de un examen en específico para el día estipulado de la evaluación.

- Mantener automáticamente activa la presentación del examen sólo en el tiempo estipulado para la resolución del mismo.

- Almacenar las respuestas del estudiante en la presentación como documentos digitales.

- Ingresar la nota correspondiente a la pregunta de comprensión lectora del examen.

- Obtener automáticamente una calificación para la parte de traducción de la evaluación.

- Calificar a un estudiante como aprobado o reprobado.

- Requerimientos no funcionales

Los requerimientos no funcionales abarcan aspectos del sistema visibles para el usuario, que no están relacionados de forma directa con el comportamiento funcional del sistema. Se definieron los siguientes requerimientos no funcionales:

- Restringir todo tipo de accesos no autorizados, y permitir acciones a los usuarios sólo según sus privilegios en el sistema.

- Mostrar gran eficiencia y velocidad de respuesta.

- Implementar el software únicamente con herramientas de software libre.

- Metáfora del Sistema.

Para cumplir con los requerimientos establecidos se realizó un esquema general del sistema, representado en la Ilustración 15, en donde se identifican cada uno de los componentes que lo conforman y la interacción entre cada uno de ellos.

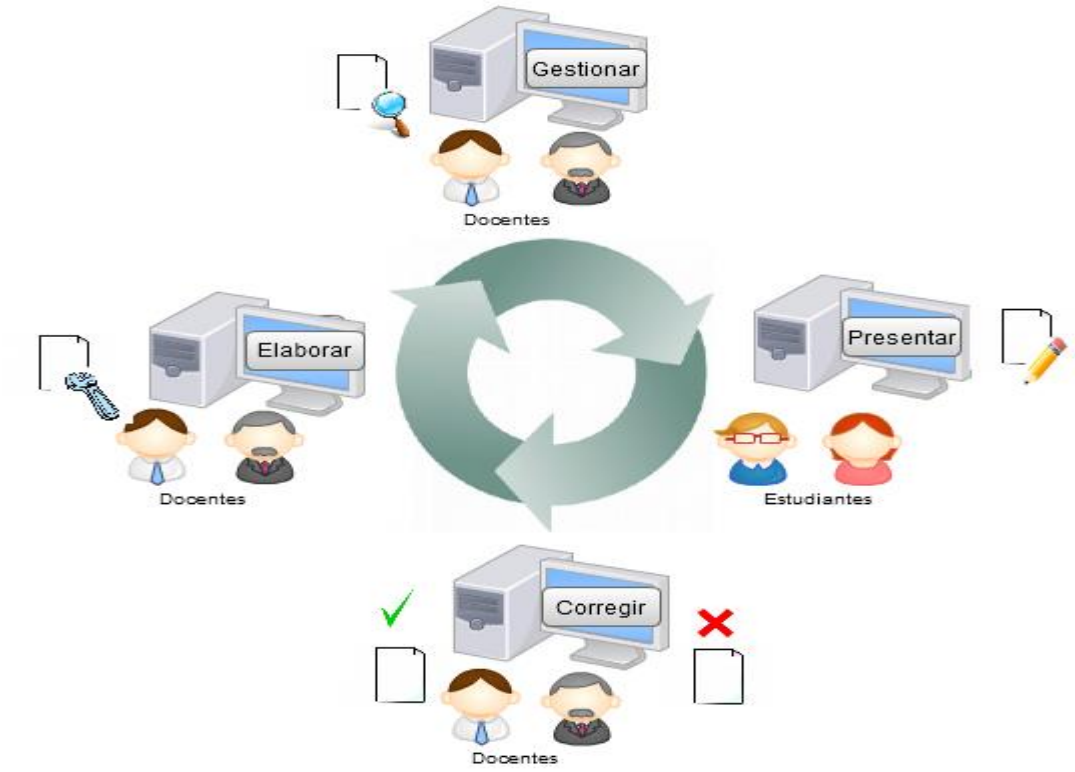


Ilustración 15: Metáfora del sistema.

En esta imagen se presentan los distintos componentes de la aplicación, así como los distintos actores involucrados en la misma. Está el módulo de gestión de exámenes donde los docentes podrán editar, eliminar, activar o desactivar, y ver el detalle de las evaluaciones.

También se presenta el módulo de elaboración del examen en el cual el personal docente podrá conformar el contenido de la evaluación ingresando las preguntas y las respectivas respuestas.

Por otro lado está la interfaz de presentación del examen en la que los estudiantes pueden ver el contenido del mismo y además pueden ingresar las respuestas que posteriormente se almacenan en la base de datos para su posterior corrección. Cabe destacar que esta interfaz está activa sólo durante el tiempo estipulado para la resolución de la prueba, y luego de haberse consumado este tiempo automáticamente la aplicación lo notifica y deshabilita la posibilidad de

seguir respondiendo la evaluación.

Se cuenta con otro componente, mediante el cual los docentes o el personal autorizado tienen la opción de corregir automáticamente las evaluaciones. Es aquí donde tiene lugar el algoritmo de corrección automática que se va a implementar y que recibirá como entrada la respuesta elaborada por los docentes en el módulo de elaboración, y cada una de las respuestas de los estudiantes para así arrojar un resultado para cada examen, reduciendo considerablemente el esfuerzo y el tiempo que se puede invertir corrigiendo la evaluación manualmente.

3.2.2 Iteración 1: Instalación y creación de objetos en la base de datos

Lun 14/Ene/2013 - Vie 25/Ene/2013

En esta iteración en primer lugar se instalan todos los componentes de software necesarios para conformar el ambiente de trabajo, luego se crearán los nuevos objetos en la base de datos del sistema *aTesT* y se determinarán cuales serán usados de los ya existentes.

3.2.2.1 Planificación

En la Ilustración 16 se muestran las historias de usuario desarrolladas en esta iteración:

Número	Descripción	Tipo	Fecha
1	Instalar el ambiente en el equipo de trabajo.	Nueva	14/01/2013
2	Crear nuevos objetos en la base de datos.	Nueva	14/01/2013

Ilustración 16: Historia de usuario de la Iteración 2.

3.2.2.2 Diseño

Para construir el ambiente de trabajo, se utiliza *Ruby Version Manager*

(RVM) para la instalación del lenguaje de programación *Ruby* en su versión 1.9.3, y para la creación del *gemset* donde estarán contenidas todas las gemas utilizadas para el desarrollo de la aplicación. Posteriormente se implanta la aplicación *aTesT* haciendo uso del controlador de versiones *Git*.

Con respecto a la información que se desea almacenar en base de datos para el desarrollo de esta aplicación, se deben tener datos acerca de los postgrados que ofrece la Universidad Central de Venezuela, el área de conocimiento y la facultad a la cual pertenecen, esto con el objetivo de catalogar los exámenes instrumentales según el postgrado al que esté asociado. Para cada evaluación se almacenarán los siguientes datos: idioma, postgrado, contenido, párrafo seleccionado para la traducción, y la respuesta correcta a esa traducción. Se crearán las tablas "*examen_instrumental*", "*postgrado*", "*area_conocimiento*" y "*facultad*", las cuales se ven reflejadas en la Ilustración 17:

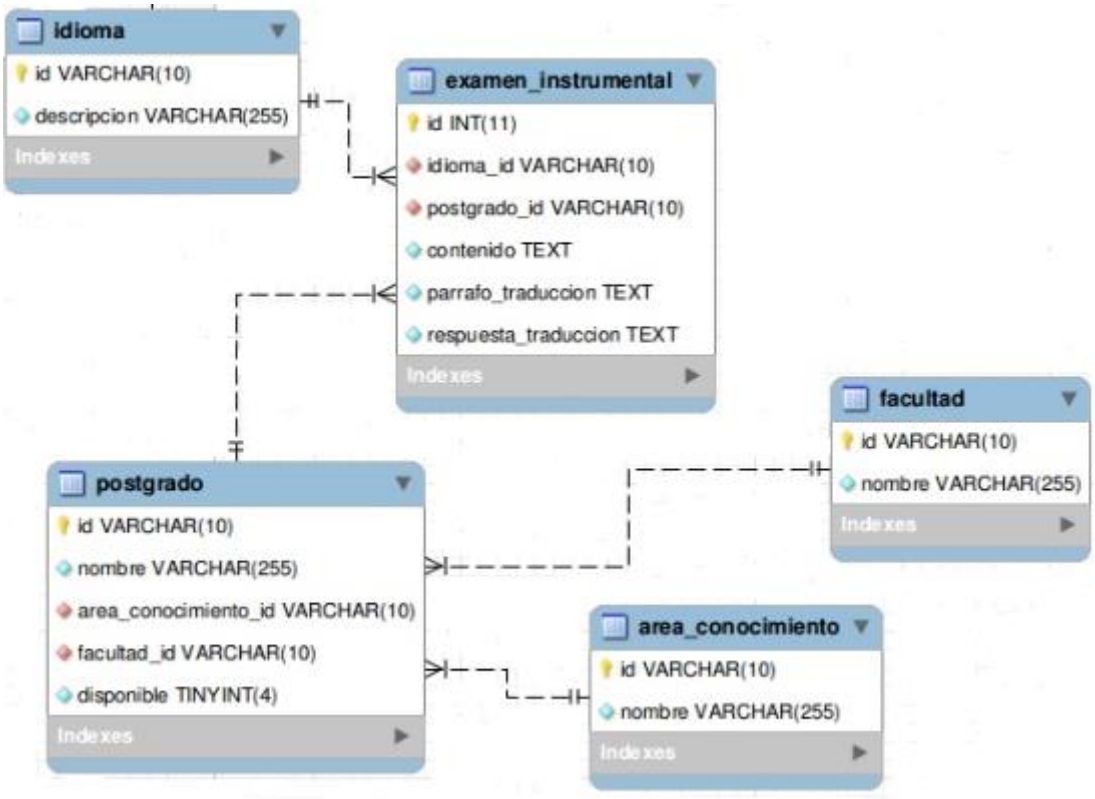


Ilustración 17: Diagrama de tablas para la Iteración 1.

La información de los postgrados será obtenida del sitio oficial de la UCV http://www.postgrado.ucv.ve/.

3.2.2.3 Codificación

Para instalar *aTesT* en la máquina de trabajo se debe ejecutar el siguiente comando en el terminal del sistema:

```
git clone http://<usuario>@fundeim.human.ucv.ve/git/atest.git
```

De esa manera se tendrá el código fuente de la aplicación localmente.

Para la instalación de *Ruby Version Manager* se debe ejecutar el siguiente comando en el terminal del sistema:

```
$ curl https://raw.github.com/wayneeseguin/rvm/master/binscripts/rvm-installer | bash -s stable
```

Luego haciendo uso de *RVM*, se instalará el lenguaje *Ruby* en su versión 1.9.3, y se creará el respectivo *gemset*:

```
$ rvm install 1.9.3
$ rvm gemset create tesis
$ rvm use 1.9.3@tesis
```

Para terminar de implantar el ambiente se instalará *aTesT*, ejecutando este comando en el directorio principal de la aplicación:

```
$ bundle install
```

El código para la creación de los nuevos objetos en base de datos se presenta en la Ilustración 18:

```
CREATE  TABLE IF NOT EXISTS `atest_development`.`facultad`
(  `id` VARCHAR(10) NOT NULL ,  `nombre` VARCHAR(255) NOT
NULL ,  PRIMARY KEY (`id`) )ENGINE = InnoDBDEFAULT
CHARACTER SET = latin1 CREATE  TABLE IF NOT EXISTS
`atest_development`.`area_conocimiento` (  `id` VARCHAR(10)
NOT NULL ,  `nombre` VARCHAR(255) NOT NULL ,  PRIMARY KEY
(`id`) )ENGINE = InnoDBDEFAULT CHARACTER SET =
latin1 CREATE  TABLE IF NOT EXISTS
`atest_development`.`postgrado` (  `id` VARCHAR(10) NOT
NULL ,  `nombre` VARCHAR(255) NOT NULL ,
`area_conocimiento_id` VARCHAR(10) NOT NULL ,
`facultad_id` VARCHAR(10) NOT NULL ,  `disponible`
TINYINT(4) NOT NULL DEFAULT '1' ,  PRIMARY KEY (`id`) ,
INDEX `fk_postgrado_facultad` (`facultad_id` ASC) ,  INDEX
`fk_postgrado_area_conocimiento` (`area_conocimiento_id`
ASC) ,  CONSTRAINT `fk_postgrado_area_conocimiento`
FOREIGN KEY (`area_conocimiento_id` )    REFERENCES
`atest_development`.`area_conocimiento` (`id` )    ON
DELETE CASCADE    ON UPDATE CASCADE,  CONSTRAINT
`fk_postgrado_facultad`    FOREIGN KEY (`facultad_id` )
REFERENCES `atest_development`.`facultad` (`id` )    ON
DELETE CASCADE    ON UPDATE CASCADE)ENGINE = InnoDBDEFAULT
CHARACTER SET = latin1
```

Ilustración 18: Código para la creación de los nuevos objetos en base de datos.

3.2.1.4 Pruebas

El ambiente de trabajo para realizar el desarrollo fue instalado exitosamente. Luego de la ejecución del comando para la instalación de ruby en su versión 1.9.3, la tecnología se instaló correctamente como se muestra en la Ilustración 19:

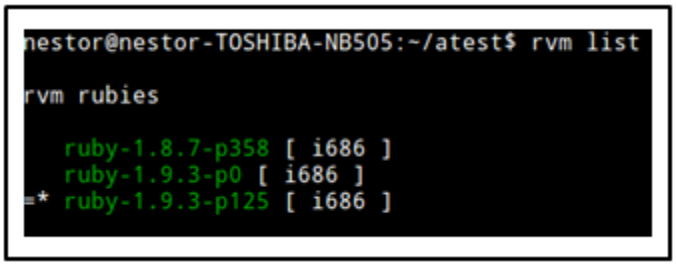


Ilustración 19: Versiones de *Ruby - Ruby Version Manager*.

De igual manera la aplicación *aTesT* fue instalada como observa en la Ilustración 20:

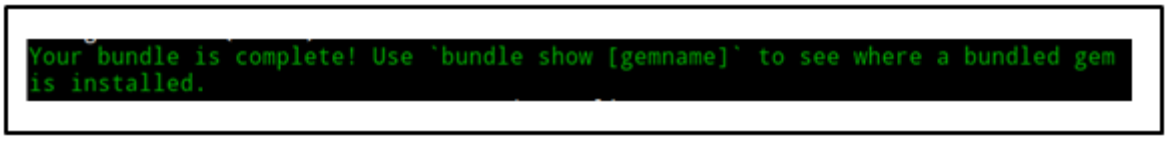


Ilustración 20: Instalación de *aTesT*.

3.2.3 Iteración 2: Módulo de elaboración de la evaluación

Lun 28/Ene/2013 - Vie 08/Feb/2013

En esta iteración en primer lugar se incorpora al sistema *aTesT* el módulo de elaboración de la evaluación, se crean nuevos objetos en la base de datos del y se determinan cuales serán usados de los ya existentes.

3.2.3.1 Planificación

En la Ilustración 21 se muestran las historias de usuario desarrolladas en esta iteración:

Número	Descripción	Tipo	Fecha
3	Crear las interfaces para el módulo de elaboración del examen.	Nueva	28/01/2013
4	Almacenar en base de datos el párrafo de traducción del examen como cadena de caracteres.	Nueva	28/01/2013
5	Almacenar los datos del examen creado.	Nueva	28/01/2013
6	Realizar las validaciones correspondientes en los formularios de las vistas correspondientes al módulo de elaboración.	Nueva	28/01/2013

Ilustración 21: Historia de usuario de la Iteración 2.

3.2.3.2 Diseño

Los postgrados de la Universidad Central de Venezuela están clasificados por área de conocimiento y por facultad a la que pertenece. Debido a esto, el usuario podrá seleccionar el postgrado al cual está asociada la evaluación, bien sea categorizándolos por área de conocimiento o por facultad. En la siguiente interfaz el contenido de la evaluación será ingresada en un editor de texto enriquecido. Posteriormente se podrá elegir el párrafo que será utilizado para la parte de traducción del examen y finalmente se ingresará una respuesta correcta a dicha traducción. Toda esta información será almacenada en la base de datos una vez terminada la creación del examen. La secuencia que seguirá el usuario en las interfaces del módulo de elaboración se muestra gráficamente en la Ilustración 22.

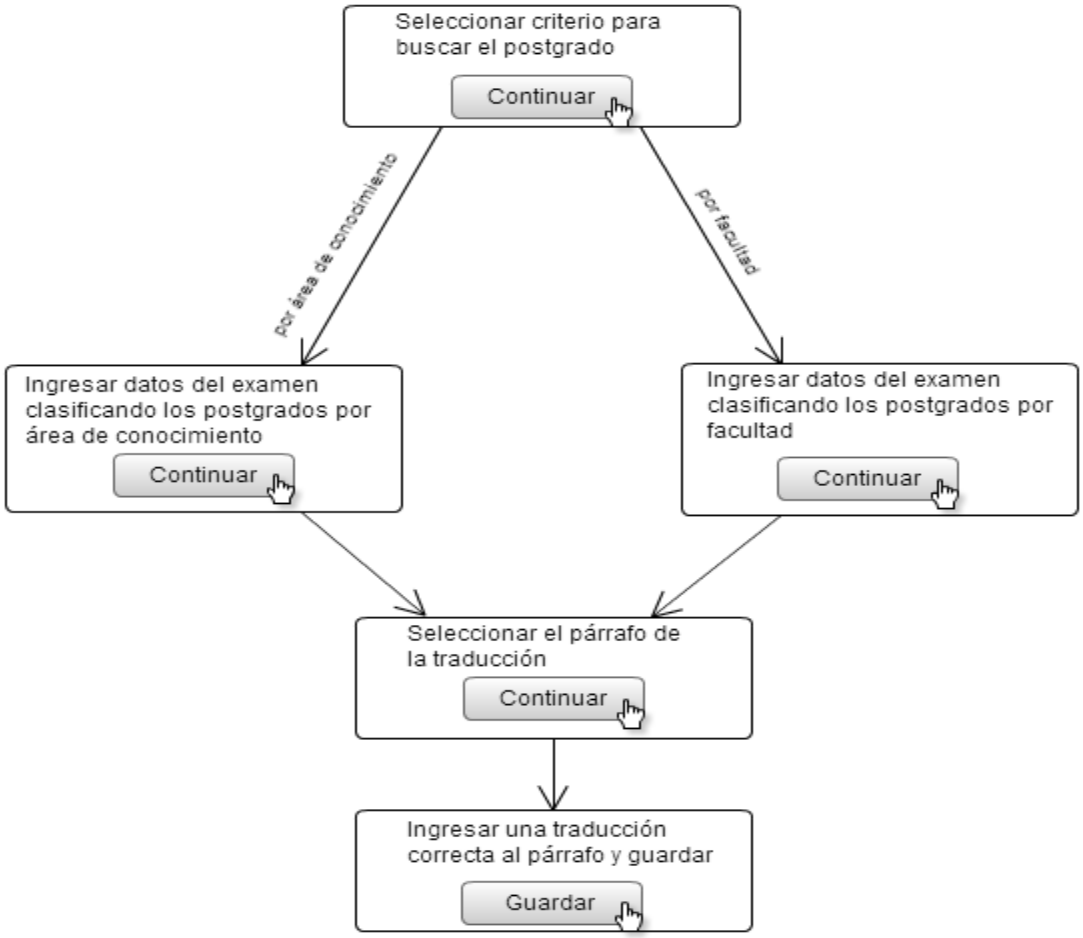


Ilustración 22: Secuencia del usuario en las interfaces del módulo de elaboración.

Consideraciones técnicas: el contenido de la evaluación será ingresada sobre un elemento *HTML textarea* convertido en un editor de texto enriquecido mediante *elRTE,* el cual es un complemento escrito en JavaScript usando *jQuery UI*. Para obtener el contenido del párrafo de traducción como cadena de caracteres, se eliminará el texto las etiquetas y los caracteres especiales *HTML* haciendo uso de la función *gsub* la cual retorna una copia de la cadena a la cual se aplica esta función, sustituyendo las ocurrencias de un patrón (primer parámetro), por una cadena de caracteres (segundo parámetro). Generalmente el primer parámetro es una expresión regular, la cual será construida de manera que coincidan todas las etiquetas *HTML* que son de la forma /<[a-z]{1,n}>/.

3.2.3.3 Codificación

Creación de las interfaces:

- Seleccionar criterio de clasificación de los postgrados en la Ilustración 23:

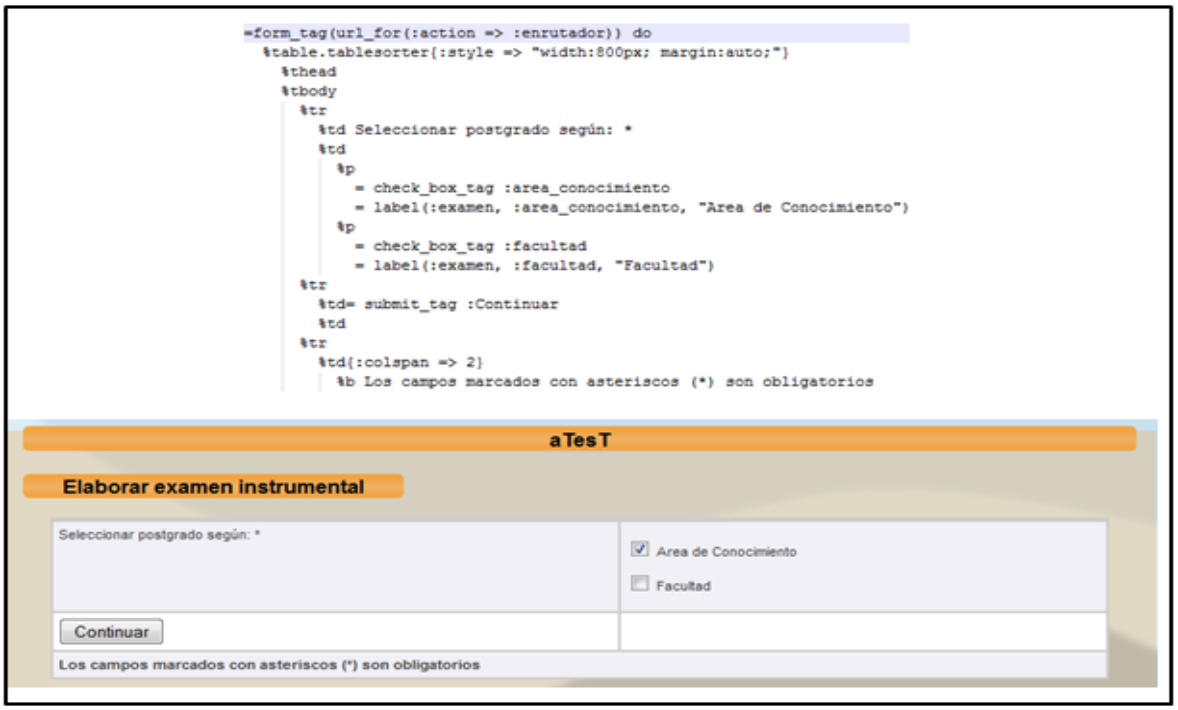


Ilustración 23: Código e interfaz para seleccionar el criterio de clasificación de postgrados.

- Ingresar datos del examen en la Ilustración 24:

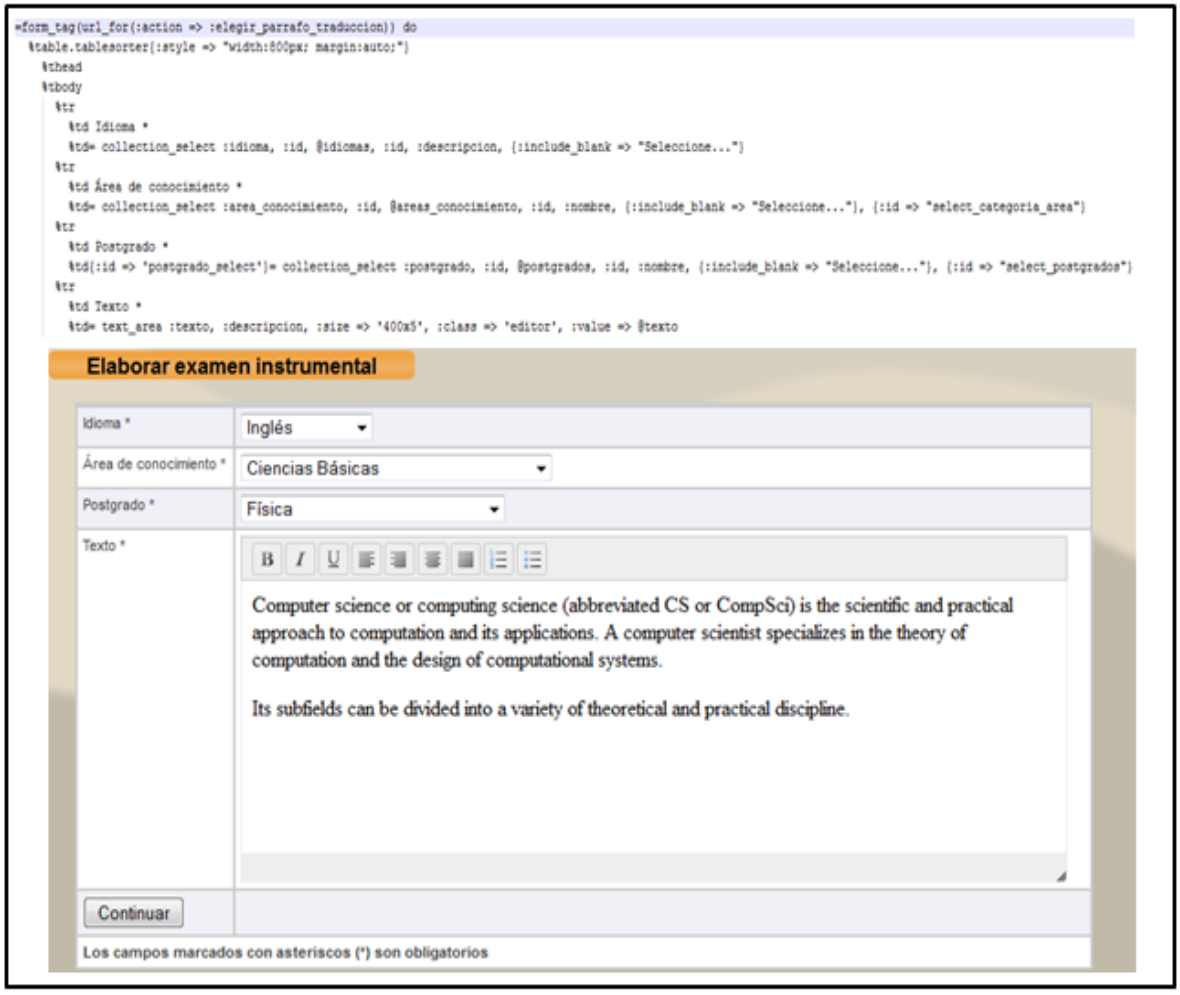


Ilustración 24: Código e interfaz de ingreso de datos del examen.

- Seleccionar el párrafo de traducción en la Ilustración 25:

```
=form_tag(url_for(:action => :ingresar_respuesta_traduccion, :examen => @examen, :parrafos => @parrafos)) do
  %table.tablesorter{:style => "width:800px; margin:auto;"}
    %thead
    %tbody
      %tr
        %td{:colspan => 2}
          %b Elegir el párrafo a ser traducido en el examen *
      -i = 1
      -j = 1
      -@parrafos.each do |p|
        -if (p.length > 0)
          %tr
            %td
              = check_box_tag :"#{i}"
            %td
              %b (#{i})
              =p.html_safe
```

aTesT

Elaborar examen instrumental

Elegir el párrafo a ser traducido en el examen *	
☑	(1) Computer science or computing science (abbreviated CS or CompSci) is the scientific and practical approach to computation and its applications. A computer scientist specializes in the theory of computation and the design of computational systems.
☐	(2) Its subfields can be divided into a variety of theoretical and practical disciplines.
Continuar	

Los campos marcados con asteriscos (*) son obligatorios

Ilustración 25: Código e interfaz de selección del párrafo de traducción

- Ingresar respuesta para la traducción y guardar en la Ilustración 26:

```
=form_tag(url_for(:action => :elaboracion_examen_procesar, :examen => @examen, :texto_traduccion => @texto_traduccion)) do
  %table.tablesorter{:style => "width:800px; margin:auto;"}
    %thead
    %tbody
      %tr
        %td Texto:
        %td= simple_format(h(@texto_traduccion))
      %tr
        %td Traducción: *
        %td= text_area :respuesta, :contenido, :size => '80x5', :class => 'editor', :id => "traduccion"
      %tr
        %td= submit_tag :Guardar
        %td
```

Elaborar examen instrumental

Texto:	Computer science or computing science (abbreviated CS or CompSci) is the scientific and practical approach to computation and its applications. A computer scientist specializes in the theory of computation and the design of computational systems.
Traducción: *	Ciencias de la computación o ciencias de la computación (abreviado CS o CompSci) es el enfoque científico y práctico para la computación y sus aplicaciones. Un informático se especializa en la teoría de la computación y el diseño de sistemas computacionales.
Guardar	

Los campos marcados con asteriscos (*) son obligatorios

Ilustración 26: Código e interfaz de ingreso de la respuesta para la traducción.

Para la transformación del párrafo de traducción como cadena de caracteres se implementó el siguiente método mostrado en la Ilustración 27:

```
def eliminar_tags(p)
  p = p.gsub(/ /," ")
  p = p.gsub(/<[a-z]{1,10}>/,"")
  p = p.gsub(/<\/[a-z]{1,10}>/,"")
  return p.strip
end
```

Ilustración 27: Método para transformar un texto en cadena de caracteres.

3.2.3.4 Pruebas

Las pruebas unitarias realizadas en la Iteración 2 se muestran en la Ilustración 28:

Historia de Usuario	Descripción del caso de prueba	Resultado esperado	Resultado Obtenido
4	Texto con formato HTML = "Examen instrumental cuyo texto está basado en Gerencia Estratégica de Sistemas Agroalimentarios. En este examen usted deberá dar un resumen en español del texto y traducir el párrafo indicado igualmente al idioma español. "	Texto plano como cadena de caracteres = "Examen instrumental cuyo texto está basado en Gerencia Estratégica de Sistemas Agroalimentarios. En este examen usted deberá dar un resumen en español del texto y traducir el párrafo indicado igualmente al idioma español."	Texto plano como cadena de caracteres = "Examen instrumental cuyo texto está basado en Gerencia Estratégica de Sistemas Agroalimentarios. En este examen usted deberá dar un resumen en español del texto y traducir el párrafo indicado igualmente al idioma español."
4	Texto con formato HTML = "<p> In Dealing with monetary policy, it is	Texto Plano como cadena de caracteres = "In Dealing with monetary policy, it is	Texto Plano como cadena de caracteres = "In Dealing with monetary

	Necessary to take into account That It Adopts Characteristics particularly in the case of Cuba, since there is not a market economy but a planning center, mainly, of a financial type. </ p> <p> In keeping With These considerations, the instruments of monetary policy Carried out by the work of the Central bank up to date are the Following: controls over exchange rates and statutory reserve ratios, Among other Provisions. </ p>"	Necessary to take into account That It Adopts Characteristics particularly in the case of Cuba, since there is not a market economy but a planning center, mainly, of a financial type. In keeping With These considerations, the instruments of monetary policy Carried out by the work of the Central bank up to date are the Following: controls over exchange rates and statutory reserve ratios, Among other Provisions."	policy, it is Necessary to take into account That It Adopts Characteristics particularly in the case of Cuba, since there is not a market economy but a planning center, mainly, of a financial type. In keeping With These considerations, the instruments of monetary policy Carried out by the work of the Central bank up to date are the Following: controls over exchange rates and statutory reserve ratios, Among other Provisions."

Ilustración 28: Pruebas para la Iteración 2.

Pruebas de aceptación: el día martes 13 de febrero del año 2013, se programó una reunión con el Prof. Lucius Daniel Director de la Escuela de Idioma y Administrador del sistema en la cual hizo uso de la aplicación interactuando con el módulo de elaboración.

Las validaciones en las interfaces del módulo de elaboración fueron codificadas exitosamente, ya que cualquier campo del formulario con un valor errado, era detectado y el flujo del sistema retornaba de nuevo a la vista de dicho formulario.

Los párrafos del texto del contenido de la evaluación fueron separados correctamente con la utilización del método *Split* de tal manera que se mostró cada párrafo por separado para poder elegir mediante un elemento *select* cuál sería elegido para traducir.

Finalmente, toda la información referente de la evaluación fue guardada con éxito en la tabla "*examen_instrumental*".

Se realizó la observación de que el sistema no permite seleccionar más de un párrafo para la traducción, motivo por el cual éste módulo necesita ser modificado en una iteración posterior.

3.2.4 Iteración 3: Gestión de exámenes

Lun 11/Feb/2013 - Vie 22/Feb/2013

En esta iteración se tiene planificado construir las interfaces para el módulo de administración de los exámenes, con sus respectivas validaciones en cada uno de sus formularios. Este módulo permitirá editar, eliminar, activar (o desactivar) y ver el contenido de una evaluación instrumental.

3.2.4.1 Planificación

En la Ilustración 29 se muestran las historias de usuario desarrolladas en esta iteración:

Número	Descripción	Tipo	Fecha
7	Crear las interfaces para el módulo de administración de exámenes.	Nueva	11/02/2013
8	Crear interfaces que permitan editar el contenido de un examen instrumental como parte del módulo de administración.	Nueva	11/02/2013
9	Realizar las validaciones correspondientes en los formularios de las vistas correspondientes al módulo de administración.	Nueva	11/02/2013

Ilustración 29: Historias de usuario de la Iteración 3.

3.2.4.2 Diseño

El usuario con rol de administrador puede observar en una tabla todos los exámenes creados hasta la fecha en una tabla que tiene la funcionalidad de filtrar contenido según un patrón de búsqueda determinado por lo que se escriba en un campo de texto ubicado en la parte superior a la tabla. Esta tabla tiene el postgrado al cual pertenece el examen, la fecha de creación, un enlace que activa una ventana modal la cual contiene el contenido del examen, un enlace que lleva a las interfaces para la edición de cada evaluación, otro para eliminar un examen del repositorio y finalmente el último enlace para activar o desactivar un examen. Lo último será tomado en cuenta para la asignación de exámenes a cada estudiante; si un examen está desactivado no puede ser asignado a un estudiante para su presentación.

La interfaz principal tiene un botón que llevará a la creación de un nuevo examen (módulo de elaboración).

Consideraciones técnicas: la funcionalidad de la tabla para filtrar contenido será implementada utilizando el complemento de *jQuery uiTableFilter* recibiendo como parámetro lo que sea introducido en un campo de texto.

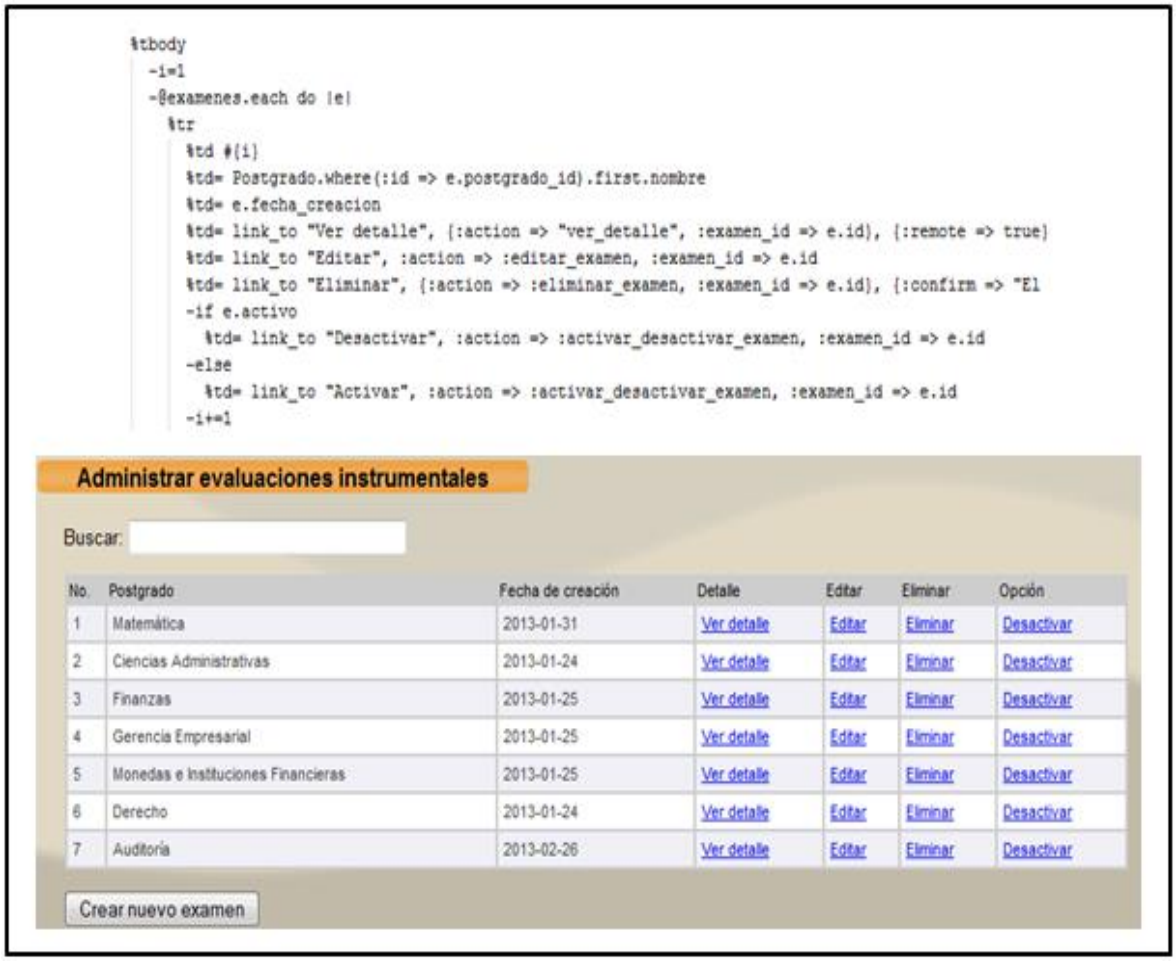
La ventana modal que se activará en el enlace "Ver Detalle" de la tabla y será implementada mediante la funcionalidad *Dialog* que ofrece *jQueryUI;* el contenido de dicha ventana será el texto correspondiente al examen.

Con respecto a las interfaces para la edición de exámenes, estas se implementaran de igual manera que se desarrollaron las interfaces del módulo de elaboración del examen. La parte editable del examen será su contenido, la elección de los párrafos para la sección de traducción, y la propia traducción de los párrafos elegidos de dicha evaluación.

3.2.4.3 Codificación

Creación de las interfaces:

- La interfaz principal del módulo de administración de exámenes se muestra en la Ilustración 30:



```
%tbody
  -i=1
  -@examenes.each do |e|
    %tr
      %td #{i}
      %td= Postgrado.where(:id => e.postgrado_id).first.nombre
      %td= e.fecha_creacion
      %td= link_to "Ver detalle", {:action => "ver_detalle", :examen_id => e.id}, {:remote => true}
      %td= link_to "Editar", :action => :editar_examen, :examen_id => e.id
      %td= link_to "Eliminar", {:action => :eliminar_examen, :examen_id => e.id}, {:confirm => "El
      -if e.activo
        %td= link_to "Desactivar", :action => :activar_desactivar_examen, :examen_id => e.id
      -else
        %td= link_to "Activar", :action => :activar_desactivar_examen, :examen_id => e.id
      -i+=1
```


Ilustración 30: Código e interfaz del módulo de administración de exámenes.

- El script para la invocación de la funcionalidad para hacer filtrar la tabla se observa en la Ilustración 31:

Ilustración 31: Código y funcionalidad para filtrar contenido en la tabla.

- El script para activar la ventana modal se refleja en la ilustración 32:

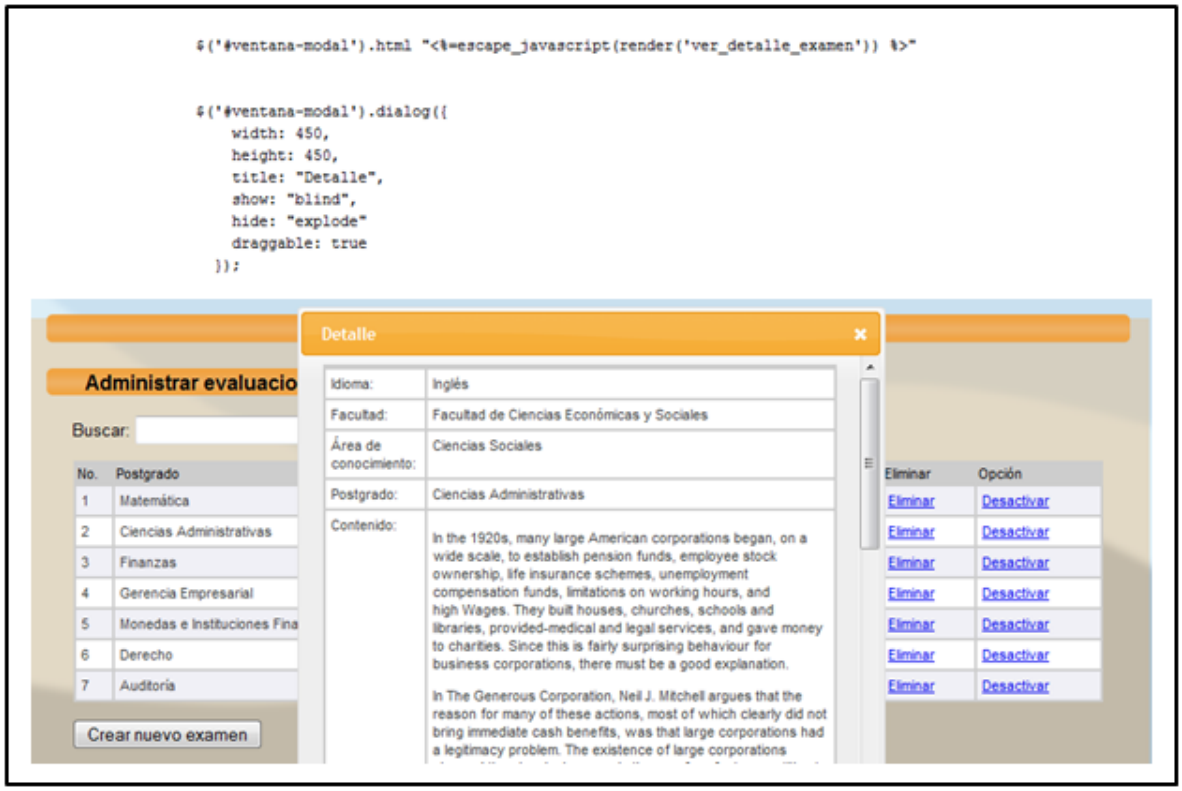


Ilustración 32: Código y funcionalidad para mostrar el contenido de un examen en una ventana modal.

3.2.4.4 Pruebas

Pruebas de aceptación: el día martes 26 de marzo del año 2013, se estableció una reunión con el Prof. Lucius Daniel Director de la Escuela de Idioma y Administrador del sistema en la cual hizo uso de la aplicación interactuando con el módulo de gestión de exámenes.

Las validaciones en el módulo de gestión de examen fueron probadas y los resultados fueron los esperados. En estas interfaces el flujo del sistema fue el correcto y cada una de las acciones aplicadas a una evaluación (eliminar, activar o desactivar y editar) ejecutaban las operaciones pertinentes en la base datos.

Al igual que en el módulo de elaboración del examen, la edición de una evaluación registraban en la base de datos con éxito cualquier información modificada de ésta. En esta edición ya fue posible seleccionar más de un párrafo para su traducción.

Este desarrollo recibió total aprobación del usuario.

3.2.5 Iteración 4: Mejorar módulo de elaboración de exámenes

Lun 25/Feb/2013 - Vie 08/Mar/2013

En esta iteración en primer lugar se realizan mejoras en la interfaces del módulo de elaboración para satisfacer el requerimiento del usuario que consiste en poder seleccionar varios párrafos para la sección de traducción del examen, y no un único párrafo como se había implementado. Adicionalmente se crearán nuevos objetos en la base de datos.

3.2.5.1 Planificación

En la Ilustración 33 se muestran las historias de usuario desarrolladas en esta iteración:

Número	Descripción	Tipo	Fecha
3	Crear las interfaces para el módulo de elaboración del examen.	Mejora	25/02/2013
2	Crear nuevos objetos en la base de datos.	Mejora	25/02/2013

Ilustración 33: Historias de usuario de la Iteración 4.

3.2.5.2 Diseño

Se realizan modificaciones en las interfaces del módulo de elaboración del examen, específicamente en la fase de elección del párrafo para la traducción, para poder elegir más de un párrafo para la traducción.

Con respecto a las modificaciones en los objetos en base de datos, la clave primaria de la tabla postgrado, la cual es el identificador del postgrado, pasa de ser una cadena de caracteres (Varchar) que denotaba una abreviación del nombre del postgrado, a ser un numero entero (Int) autoincremental. Además a la tabla "*examen_instrumental*", la cual contiene la información sobre los exámenes instrumentales, se le agregan los campos "*fecha_creacion*" y "*activo*", el último indica si un examen está activado para su presentación o de lo contrario no se toma en cuenta para su presentación, cuya decisión el compete al administrador del sistema.

Entre los nuevos objetos que se crean en la base de datos están las tablas "*examen_presentacion*" la cual contiene una referencia a la tabla "*examen_instrumental*" y tiene un campo fecha que indica la fecha de presentación de dicho examen, es decir, en esta tabla se almacenan las jornadas de presentación de diversos exámenes en un día específico. Adicionalmente se tiene la tabla "*examen_presentacion*" la cual mantiene la relación entre un estudiante y la presentación de un examen y en ella se almacenan, además de los identificadores (estudiante, examen y postgrado), el estatus de la presentación (en espera para

presentar, examen presentado, en espera de calificación, aprobado o reprobado), y las respectivas respuestas al resumen y a la traducción del estudiante en el examen.

En la Ilustración 34 se muestran los nuevos objetos en la base de datos.

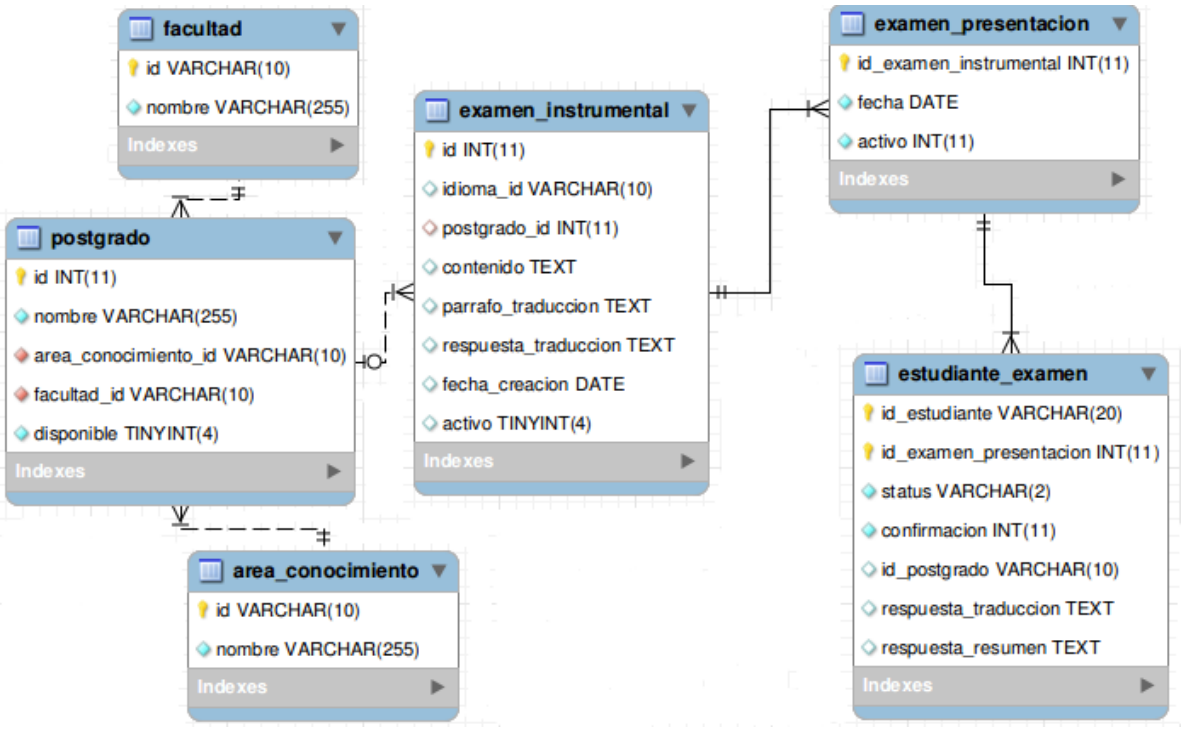


Ilustración 34: Diagrama de tablas para la Iteración 4.

Consideraciones técnicas: anteriormente esta elección era a través de un elemento *select* y sólo podía ser elegido un párrafo. Ahora esta selección será haciendo uso de elementos *checkbox* y de esta manera será posible seleccionar más de un párrafo para que sea traducido en la presentación del examen.

3.2.5.3 Codificación

El código para la creación de los nuevos objetos en base de datos es el que sigue a continuación en la Ilustración 35:

```
CREATE TABLE `atest_development`.`examen_instrumental` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `idioma_id` varchar(10) DEFAULT NULL,
  `postgrado_id` int(11) DEFAULT NULL,
  `contenido` text,
  `parrafo_traduccion` text,
  `respuesta_traduccion` text,
  `fecha_creacion` date DEFAULT NULL,
  `activo` tinyint(4) DEFAULT '1',
  PRIMARY KEY (`id`),
  KEY `fk_examen_instrumental_postgrado` (`postgrado_id`),
  CONSTRAINT `fk_examen_instrumental_postgrado` FOREIGN KEY
(`postgrado_id`) REFERENCES `postgrado` (`id`) ON DELETE CASCADE ON
UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=11 DEFAULT CHARSET=latin1

CREATE TABLE `atest_development`.`postgrado` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `nombre` varchar(255) NOT NULL,
  `area_conocimiento_id` varchar(10) NOT NULL,
  `facultad_id` varchar(10) NOT NULL,
  `disponible` tinyint(4) NOT NULL DEFAULT '1',
  PRIMARY KEY (`id`),
  KEY `fk_postgrado_facultad` (`facultad_id`),
  KEY `fk_postgrado_area_conocimiento` (`area_conocimiento_id`),
  CONSTRAINT `fk_postgrado_area_conocimiento` FOREIGN KEY
(`area_conocimiento_id`) REFERENCES `area_conocimiento` (`id`) ON DELETE
CASCADE ON UPDATE CASCADE,
  CONSTRAINT `fk_postgrado_facultad` FOREIGN KEY (`facultad_id`)
REFERENCES `facultad` (`id`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=184 DEFAULT CHARSET=latin1
```

Ilustración 35: Código para la creación de los nuevos objetos en base de datos.

El código para de la interfaz modificada en el módulo de elaboración del examen se muestra en la Ilustración 36 y 37:

```
:javascript
  $(document).ready(function() {
    $('input[type="submit"]').attr('disabled','disabled');
    $('input[type="checkbox"]').change(function(){
      if ($('input[type="checkbox"]').is(':checked')){
        $('input[type="submit"]').removeAttr('disabled');
      }else{
        $('input[type="submit"]').attr('disabled','disabled');
      }
    });
  });

=form_tag(url_for(:action => :ingresar_respuesta_traduccion, :examen => @examen, :parrafos => @parrafos)) do
  %table.tablesorter{:style => "width:800px; margin:auto;"}
    %thead
    %tbody
      %tr
        %td{:colspan => 2}
          %b Elegir el párrafo a ser traducido en el examen *
      -i = 1
      -j = 1
      -@parrafos.each do |p|
        -if (p.length > 0)
          %tr
            %td
              = check_box_tag :"#{i}"
            %td
              %b (#{i})
              =p.html_safe
```

Ilustración 36: Código de la interfaz modificada de selección del párrafo.

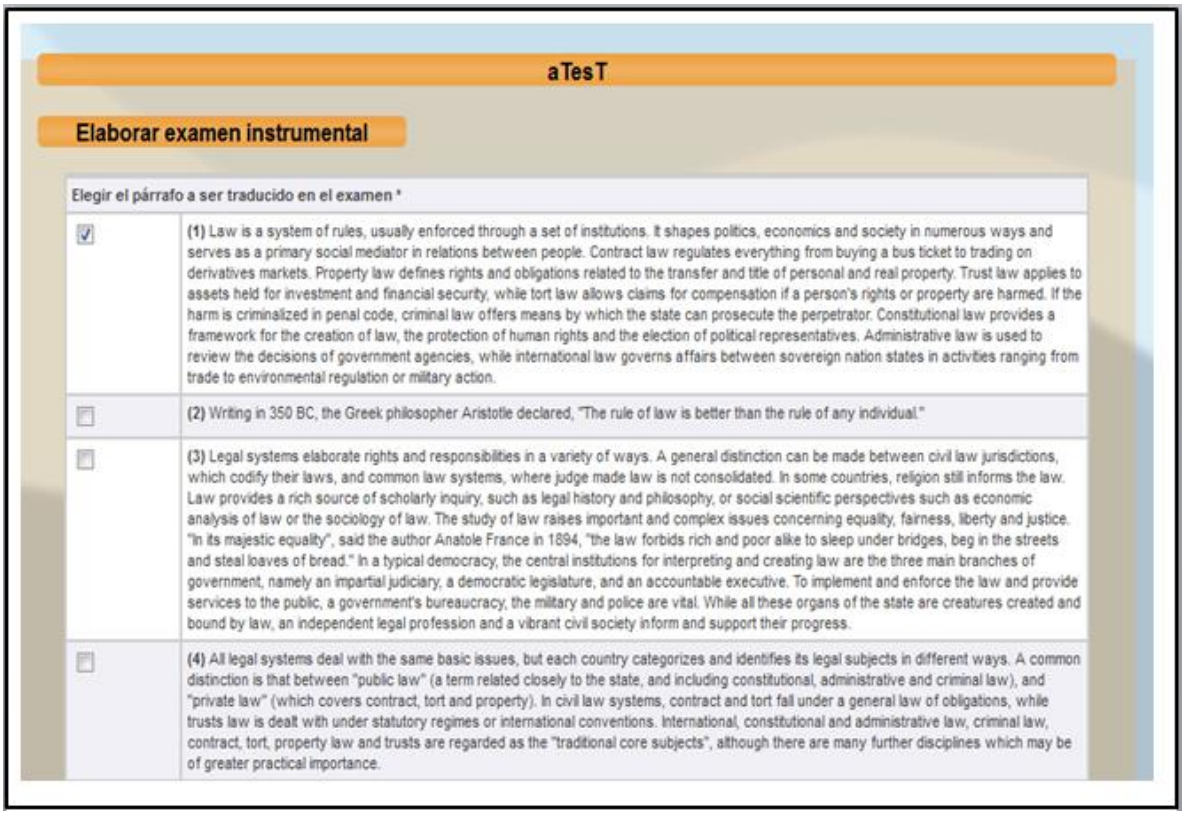


Ilustración 37: Interfaz modificada de selección del párrafo de traducción.

3.2.5.4 Pruebas

Pruebas de aceptación: el día martes 26 de marzo del año 2013, se estableció una reunión con el Prof. Lucius Daniel Director de la Escuela de Idioma y Administrador del sistema en la cual hizo uso de la aplicación interactuando con el módulo de gestión de exámenes y pudo verificar el cambio que se hizo en las interfaces del módulo de la elaboración con respecto a la Iteración 2.

El cambio en el módulo de presentación del examen fue realizado de manera exitosa, ya que se pudo elegir varios párrafos en lugar de sólo uno mediante los elementos de tipo *checkbox*.

3.2.6 Iteración 5: Presentación del examen

Lun 11/Mar/2013 - Vie 22/Mar/2013

En esta iteración se crean las interfaces del módulo de presentación de la evaluación instrumental, incluyendo la implementación para almacenar las respuestas de los estudiantes como documentos digitales y manejar el tiempo en la presentación del examen el cual consiste en 2 horas.

3.2.6.1 Planificación

En la Ilustración 38 se muestran las historias de usuario desarrolladas en esta iteración:

Número	Descripción	Tipo	Fecha
10	Crear las interfaces para el módulo de presentación del examen.	Nueva	11/03/2013
11	Almacenar las respuestas de los exámenes como cadena de caracteres.	Nueva	11/03/2013
12	Manejar el tiempo de presentación del	Nueva	11/03/2013

	examen.		

Ilustración 38: Historias de usuario para la Iteración 5.

3.2.6.2 Diseño

Durante el día destinado para la presentación de la evaluación, una vez el estudiante ingrese con su cuenta a la aplicación, se le mostrará un enlace, el cual llevará hacia la interfaz de presentación. Esta interfaz consistirá en un formulario con dos áreas de texto convertidas en un editor de texto enriquecido. La primera área de texto será para responder la parte de comprensión lectora y la segunda para responder la sección de traducción; cada una estará ubicada en la parte inferior del contenido de cada pregunta.

Para obtener el contenido de las respuestas como cadena de caracteres, se aplicará la misma solución empleada con el contenido del párrafo de traducción en el módulo de elaboración del examen.

En la interfaz de presentación del examen se debe llevar el control del tiempo, ya que el estudiante dispone de 2 horas para responder la evaluación. Cuando el estudiante se dirija a la interfaz de presentación, el sistema almacenará la hora actual como la hora en la cual comenzó el examen, esto ocurrirá una sólo una vez, es decir, si el estudiante sale de la aplicación e inicia sesión de nuevo con la intención de reiniciar el tiempo el sistema verificará y tomará como tiempo de comienzo el primer tiempo el cual fue almacenado en la base de datos. El sistema informará al estudiante cuando reste 1 hora, 30 minutos y 10 minutos para que finalice el examen, y finalmente si se cumplen las 2 horas y el estudiante no ha finalizado manualmente, se hará *submit* automáticamente guardando las respuestas en la base de datos para su posterior corrección.

Consideraciones técnicas: Las áreas de texto son convertidas en un editor de texto enriquecido mediante el complemento escrito en *JavaScript* llamado *elRTE.*

Para obtener el contenido de las respuestas como cadena de caracteres, se eliminarán las etiquetas y los caracteres especiales *HTML* haciendo uso de la función *gsub* la cual retorna una copia de la cadena a la cual se aplica esta función, sustituyendo las ocurrencias de un patrón (primer parámetro), por una cadena de caracteres (segundo parámetro). Generalmente el primer parámetro es una expresión regular, la cual será construida de manera que coincidan todas las etiquetas *HTML* que son de la forma /<[a-z]{1,n}>/.

Por otro lado, en el lado del cliente cada 5 segundos se calculará la hora actual mediante la función *toLocaleTimeString()* de la clase *Date* y se calculará la diferencia entre la hora actual y la hora de comienzo (en segundos), que será obtenida asíncronamente mediante *AJAX* yendo al servidor para consultar la base de datos. La ejecución de una acción cada cierto tiempo se realiza mediante la función *setInterval* la cual llama a una función o evalúa una expresión a intervalos especificados (en milisegundos). El sistema informará al estudiante cuando reste 1 hora, 30 minutos y 10 minutos para que finalice el examen.

3.2.6.3 Codificación

El código para la interfaz de presentación del examen se observa en la Ilustración 39:

```
=form_tag(url_for(:action => :procesar_respuesta_examen), :id => 'formulario') do
  %table.tablesorter{:style => "width:800px; margin:auto;"}
    %thead
    %tbody
      %tr
        %td{:colspan => 2}
          %b 1. Summarize the main ideas of the text in Spanish
      %tr
        %td=@contenido.html_safe
      %tr
        %td= text_area :examen_instrumental, :respuesta, :size => '100x5', :class => 'editor', :id => "traduccion"
      %tr
        %td{:colspan => 2}
          %b 2. Translate these paragraphs of the text
      %tr
        %td=@parrafo_traduccion.html_safe
      %tr
        %td= text_area :examen_instrumental, :traduccion, :size => '100x5', :class => 'editor', :id => "traduccion"
      %tr
        %td= submit_tag "Finalizar", :confirm => "¿Está seguro de que desea finalizar el examen?"
```

Ilustración 39: Código de la interfaz de presentación del examen.

La interfaz de presentación del examen se muestra en la Ilustración 40 y 41:

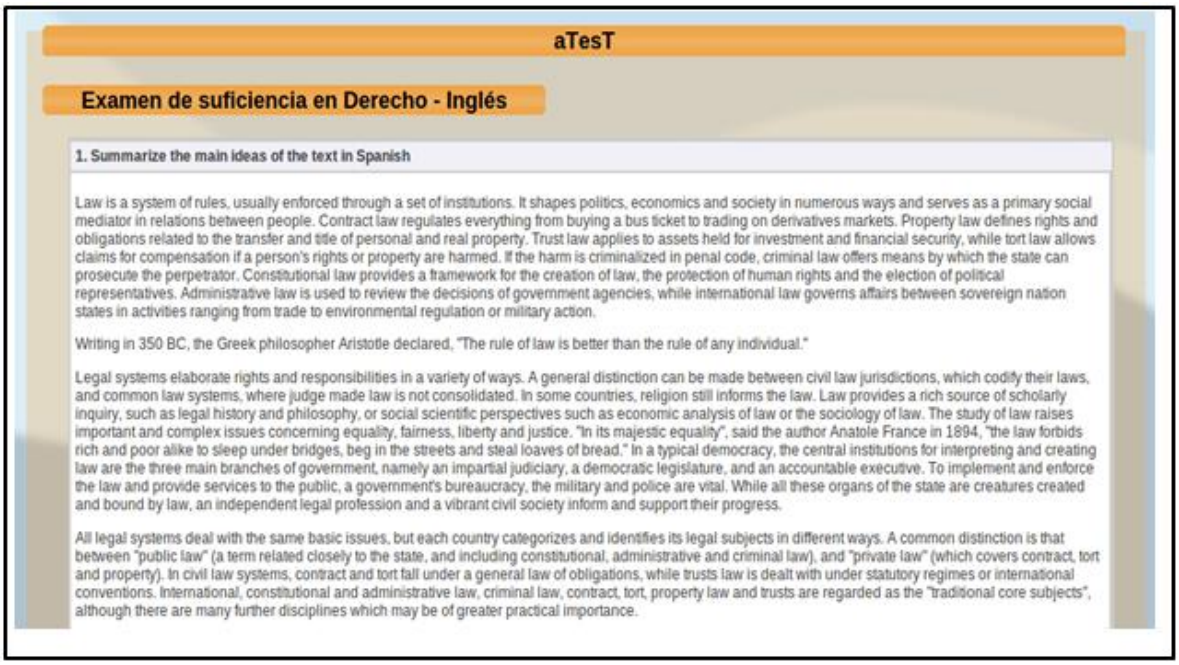


Ilustración 40: Primera parte de la interfaz de presentación del examen.

Ilustración 41: Segunda parte de la interfaz de presentación del examen.

El código para procesar las respuestas de los estudiantes se observa en la Ilustración 42:

```
def procesar_respuesta_examen
  @estudiante_examen = EstudianteExamen.find(session[:usuario].ci,session[:examen_presentacion_id])
  @estudiante_examen.respuesta_resumen = params[:examen][:respuesta_resumen]
  @estudiante_examen.respuesta_traduccion = eliminar_tags(params[:examen][:respuesta_traduccion])
  @estudiante_examen.status_id = 'P'
  @estudiante_examen.save
  flash[:mensaje] = "Sus respuestas fueron almacenadas con éxito"
  redirect_to :action => :index, :controller => :estudiante
end
```

Ilustración 42: Código para procesar las respuestas del examen.

El código para guardar el tiempo de comienzo del examen se refleja en la Ilustración 43:

```
def guardar_tiempo
  @estudiante_examen = EstudianteExamen.find(session[:usuario].ci,session[:examen_presentacion_id])
  unless @estudiante_examen.tiempo_comienzo
    @estudiante_examen.tiempo_comienzo = Time.now.strftime("%H:%M:S")
    @estudiante_examen.save
  end
  redirect_to :action => :index
end
```

Ilustración 43: Código para guardar el tiempo de comienzo del examen.

El código para obtener el tiempo de comienzo del examen en el lado del servidor se despliega en la Ilustración 44:

```
def calcular_tiempo
  @tiempo_comienzo = Time.parse(EstudianteExamen.find(session[:usuario].ci,session[:examen_presentacion_id]).tiempo_comienzo.to_s).strftime("%H:%M:%S")
  render :text => @tiempo_comienzo, :layout => false
end
```

Ilustración 44: Código para obtener el tiempo de comienzo del examen.

El código del lado del cliente para manejar el tiempo de presentación se muestra en la Ilustración 45:

```
setInterval (->
  $.ajax
    url: 'presentar_examen_instrumental/calcular_tiempo',
    success: (data) ->
      hora_comienzo = data
      arr = hora_comienzo.split(':')
      segundos_comienzo = (+arr[0]) * 60 * 60 + (+arr[1]) * 60 + (+arr[2]);
      #segundos_final = segundos_comienzo + 7200
      fecha = new Date()
      hora_actual = fecha.toLocaleTimeString()
      arr = hora_actual.split(':')
      segundos_actuales = (+arr[0]) * 60 * 60 + (+arr[1]) * 60 + (+arr[2]);

      if (segundos_actuales - segundos_comienzo) > 3600 and (segundos_actuales - segundos_comienzo) < 5400 and !parada_hora
        alert "Restan 60 minutos para la culminación del examen"
        parada_hora = true
      else
        if (segundos_actuales - segundos_comienzo) > 5400 and (segundos_actuales - segundos_comienzo) < 5600 and !parada_media_hora
          alert "Restan 30 minutos para la culminación del examen"
          parada_media_hora = true
        else
          if (segundos_actuales - segundos_comienzo) > 5600 and (segundos_actuales - segundos_comienzo) < 7200 and !parada_diez_minutos
            alert "Restan 10 minutos para la culminación del examen"
            parada_diez_minutos = true
          else
            if (segundos_actuales - segundos_comienzo) > 7200
              alert "El examen ha culminado, sus respuestas serán almacenadas"
              $("#formulario").submit()
), 5000
```

Ilustración 45: Código para manejar el tiempo de presentación del examen.

3.2.6.4 Pruebas

Pruebas de aceptación: el día martes 26 de marzo del año 2013, en la misma reunión con el Prof. Lucius Daniel donde se probó el módulo de gestión de exámenes, se realizaron las pruebas para el módulo de presentación del examen. Se registró para la fecha un estudiante para presentar una prueba del postgrado de Derecho en el idioma Inglés.

Las respuestas ingresadas en la interfaz principal del módulo de presentación del examen por parte de los estudiantes fueron almacenadas de manera exitosa.

Las pruebas correspondientes al algoritmo utilizado para convertir en cadena de caracteres las respuestas de los estudiantes fueron hechas para la Historia de Usuario 4 en la Iteración 2.

Con respecto a la Historia de Usuario 12 referente a manejar el tiempo de presentación del examen, se realizó una prueba unitaria con una evaluación cuya presentación inició a las 22:27 (22 horas con 27 minutos). Para las 23:27 cumpliéndose la primera hora del examen el sistema generó una alerta como se observa en la Ilustración 46:

Ilustración 46: Alerta de 60 minutos restantes en la interfaz de presentación del examen

Para las 23:57 cumpliéndose hora y media del examen el sistema generó una alerta como se observa en la Ilustración 47:

Ilustración 47: Alerta de 30 minutos restantes en la interfaz de presentación del examen.

Para las 24:17 cumpliéndose hora y 50 minutos del examen el sistema

generó una alerta como se observa en la Ilustración 48:

Ilustración 48: Alerta de 10 minutos restantes en la interfaz de presentación del examen.

Para las 24:27 cumpliéndose el tiempo para la finalización de la evaluación (2 horas) el sistema generó una alerta como se observa en la Ilustración 49 y posteriormente automáticamente se realizó *submit* del formulario con las respuestas del estudiante en ese momento.

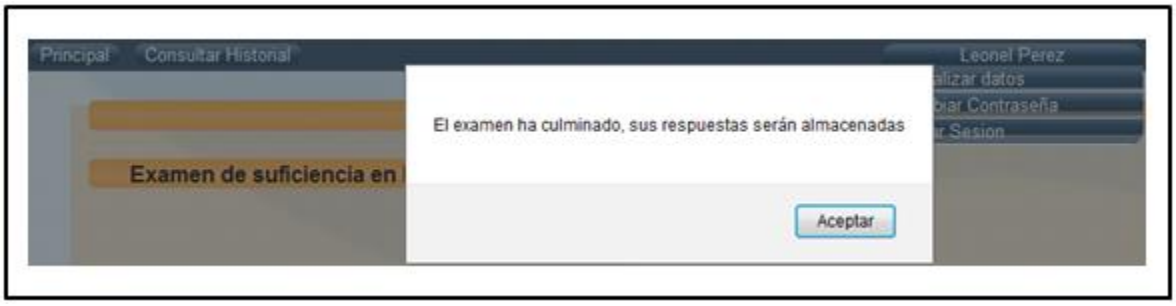


Ilustración 49: Alerta de finalización del examen.

De esta manera los resultados de las pruebas fueron los esperados, ya que cada alerta se produjo en el tiempo esperado.

3.2.7 Iteración 6: Algoritmo para corrección automática

Lun 25/Mar/2013 - Vie 05/Abr/2013

En esta iteración se implementará un algoritmo para calcular el porcentaje de similitud entre dos texto, el cual será empleado en el módulo de corrección del examen para corregir automáticamente la parte de traducción de la evaluación.

3.2.7.1 Planificación

En la Ilustración 50 se muestran las historias de usuario desarrolladas en esta iteración:

Número	Descripción	Tipo	Fecha
13	Implementar un algoritmo que calcule el porcentaje de similitud entre dos textos.	Nueva	25/03/2013

Ilustración 50: Historias de usuario para la Iteración 6.

3.2.7.2 Diseño

El código en pseudoformal del algoritmo para calcular el porcentaje de similitud entre dos textos se observa en la Ilustración 51:

```
extraer_palabras(texto)
{
    palabras = separar_palabras(texto)
    eliminar_articulos(palabras)
    eliminar_preposiciones(palabras)
    eliminar_conectores(palabras)
    .
    .
    .
    return palabras
}
check(palabras_a, palabras_b)
{
    para palabra_i en palabras_a hacer
        pesos_a << apariciones(palabra_i)/cantidad_palabras_a
    fin
    para palabra_i en palabras_b hacer
        pesos_b << apariciones(palabra_i)/cantidad_palabras_b
    fin
    vector_referencia = union(palabras_a, palabras_b)
    para palabra_i en vector_referencia hacer
        vector_normalizado_a << pesos_a(i)
        vector_normalizado_b << pesos_b(i)
    fin
    para i en 0..tamaño(vector_normalizado_a)
        sumatoria_numerador += vector_normalizado_a(i)*vector_normalizado_b(i)
        sumatoria_denominador_a += vector_normalizado_a(i)^2
        sumatoria_denominador_b += vector_normalizado_b(i)^2
    fin
    return sumatoria_numerador/sqrt(sumatoria_denominador_a*sumatoria_denominador_b)
}
palabras_texto_a = extraer_palabras(texto_a)
palabras_texto_b = extraer_palabras(texto_b)
similitud = check(palabras_a, palabras_b)
```

Ilustración 51: Código pseudoformal del algoritmo para el cálculo de similitud entre dos textos.

3.2.7.3 Codificación

El código para determinar el porcentaje de similitud entre dos textos basado en la técnica *Check* se muestra en la Ilustración 52:

```
def check(palabras_a,palabras_b)
    palabras_a_uniq.each do |pal|
      pesos_a << palabras_a.count(pal).to_f/size_a
    end
    palabras_b_uniq.each do |pal|
      pesos_b << palabras_b.count(pal).to_f/size_b
    end
    vector_referencia = palabras_a.concat(palabras_b).uniq
    vector_referencia.each do |v|
      index_a = palabras_a_uniq.index(v)
      if index_a == nil then
        vector_normalizado_a << 0
      else
        vector_normalizado_a << pesos_a[index_a]
      end
      index_b = palabras_b_uniq.index(v)
      if index_b == nil then
        vector_normalizado_b << 0
      else
        vector_normalizado_b << pesos_b[index_b]
      end
    end
    for i in 0..vector_normalizado_a.size - 1
      sumatoria_numerador += vector_normalizado_a[i]*vector_normalizado_b[i]
      sumatoria_denominador_a += vector_normalizado_a[i]**2
      sumatoria_denominador_b += vector_normalizado_b[i]**2
    end
    similitud = sumatoria_numerador/Math.sqrt(sumatoria_denominador_a*sumatoria_denominador_b)
    return similitud
  end
```

Ilustración 52: Algoritmo para calcular el porcentaje de similitud entre dos textos.

3.2.7.4 Pruebas

En la Ilustración 53 se muestran las pruebas unitarias realizadas en para esta Iteración:

Historia de Usuario	Descripción del caso de prueba	Resultado esperado	Resultado Obtenido
13	Texto A = “La metodología AGILUS es una metodología de desarrollo de aplicaciones que provee una gran flexibilidad en el desarrollo de software” Texto B = “La metodología de desarrollo de aplicaciones AGILUS es una metodología que brinda una gran adaptabilidad en la construcción de éstas”	Porcentaje de similitud = 74,536%	Porcentaje de similitud = 74,536%
13	Texto A = “Escribiendo en el año 350 aC, el filósofo griego Aristóteles, declaró: "El estado de derecho es mejor que el estado de cualquier persona."” Texto B = “Escribiendo en el año 350 aC, el filósofo griego Aristóteles, declaró: "El estado de derecho es mejor que el estado de cualquier persona."”	Porcentaje de similitud = 100%	Porcentaje de similitud = 100%
13	Texto A = “La ley es un sistema de reglas, por lo general aplicado a través de un conjunto de instituciones. Da forma a la política, la economía y la sociedad de muchas maneras y sirve como un mediador social primordial en las relaciones entre las personas. Derecho contractual regula todo, desde la compra de un billete de autobús a negociación en los mercados de derivados. El derecho de propiedad se definen los derechos y obligaciones relacionados con la transferencia y el título de los bienes muebles e inmuebles. Ley de Fideicomiso se aplica a los activos mantenidos para la inversión y la seguridad financiera, mientras que la ley de responsabilidad civil permite a las reclamaciones de indemnización si se dañan los derechos o la propiedad de una persona. Si el daño está tipificada como delito en el código penal, derecho penal ofrece medios por los cuales el Estado puede procesar al perpetrador. Derecho constitucional proporciona un marco para la creación de la ley, la protección de los derechos humanos y la elección de los representantes políticos. El derecho administrativo se utiliza para revisar las decisiones de los organismos gubernamentales, mientras que los asuntos internacionales rige la ley entre los Estados	Porcentaje de similitud = 98,78%	Porcentaje de similitud = 98,78%

	nacionales soberanos en actividades que van desde el comercio a la regulación ambiental o la acción militar." Texto B = "La ley es un sistema de reglas, por lo general aplicado a través de un conjunto de instituciones. Da forma a la política, la economía y la sociedad de muchas maneras y sirve como un mediador social primordial en las relaciones entre las personas. Derecho contractual regula todo, desde la compra de un billete de autobús a negociación en los mercados de derivados. El derecho de propiedad define los derechos y obligaciones relacionados con la transferencia y el título de los bienes muebles e inmuebles. Ley de Fideicomiso se aplica a los activos mantenidos para la inversión y la seguridad financiera, mientras que la ley de responsabilidad civil permite las reclamaciones de indemnización si se dañan los derechos o la propiedad de una persona. Si el daño está tipificado como delito en el código penal, derecho penal ofrece medios por los cuales el Estado puede procesar al perpetrador. Derecho constitucional proporciona un marco para la creación de la ley, la protección de los derechos humanos y la elección de los representantes políticos. El derecho administrativo se utiliza para revisar las decisiones de los organismos gubernamentales, mientras que los asuntos internacionales rige la ley entre los Estados nacionales soberanos en actividades que van desde el comercio a la regulación ambiental o la acción militar."		
13	Texto A = "Aplicaciones con Tecnología Internet" Texto B = "Modelos Matemáticos Aplicados"	Porcentaje de similitud = 0%	Porcentaje de similitud = 0%

Ilustración 53: Pruebas para la Iteración 6.

Cabe destacar que estas pruebas demuestran la eficacia del método desde el punto de vista algorítmico, sin embargo se podrían realizar más pruebas con datos reales de respuestas de estudiantes para verificar si en algún caso la cuantificación

que realiza el algoritmo no coincide con la decisión tomada por el profesor.

3.2.8 Iteración 7: Corrección del examen

Lun 08/Abr/2013 - Vie 19/Abr/2013

En esta iteración se crean las interfaces del módulo de corrección de la evaluación instrumental, incluyendo a dicho módulo el algoritmo para determinar el porcentaje de similitud entre dos textos desarrollado en la Iteración anterior.

3.2.8.1 Planificación

En la Ilustración 54 se muestran las historias de usuario desarrolladas en esta iteración:

Número	Descripción	Tipo	Fecha
14	Crear las interfaces para el módulo de corrección del examen.	Nueva	08/04/2013
15	Integrar el algoritmo de corrección al módulo.	Nueva	08/04/2013

Ilustración 54: Historias de usuario para la Iteración 7

3.2.8.2 Diseño

El usuario con rol de administrador podrá observar en una tabla todos los exámenes de estudiantes calificados o por calificar en una tabla que tiene la funcionalidad de filtrar contenido según un patrón de búsqueda determinado por lo que se escriba en un campo de texto ubicado en la parte superior a la tabla. Esta tabla tiene la fecha de presentación del examen, el nombre completo y la cédula de identidad del estudiante, el postgrado al cual pertenece, el idioma del examen, el resultado de la corrección y las acciones que se pueden ejecutar con respecto a la evaluación, las cuales son *corregir* y *revisar* (en caso de ya estar corregida la

evaluación); en la columna de acciones se tendrán los enlaces correspondientes a cada interfaz (*corregir_examen* y *revisar_examen*). El usuario puede listar todos los exámenes que se han presentado o simplemente podrá ver sólo los del mes actual.

La interfaz para corregir el examen consiste en una tabla con un formulario compuesto por un elemento *select* que servirá para calificar la parte de comprensión lectora del examen con 0, 0.25, 0.5, 0.75 ó 1. Además tiene un campo de texto que tendrá la calificación de la parte de traducción cuyo valor aparecerá automáticamente como resultado de invocar al algoritmo implementado para calcular el porcentaje de similitud entre dos textos (este valor es editable). Cabe destacar que esta interfaz contendrá el contenido de cada una de las preguntas del examen, así como las respectivas respuestas dadas por el estudiante.

Finalmente la interfaz para revisar el examen tendrá el mismo contenido que la interfaz anterior, y en el elemento *select* estará seleccionado el valor de la calificación anterior.

Consideraciones técnicas: La funcionalidad de la tabla para filtrar contenido será implementada utilizando el complemento de *jQuery uiTableFilter* recibiendo como parámetro lo que sea introducido en un campo de texto.

3.2.8.3 Codificación

El código e interfaz de la vista principal del módulo de corrección de exámenes se observa en la Ilustración 55:

```
-@estudiante_examen.each do |e|
  %tr
    -if e.status_id.to_s == 'P' or e.status_id.to_s == 'A' or e.status_id.to_s == 'R' or e.status_id.to_s == 'E' or e.status_id.to_s == 'F'
      %td= ExamenPresentacion.where(:id => e.examen_presentacion_id).first.fecha
      %td= Usuario.where(:ci => e.estudiante_instrumental_ci).first.nombre_corto
      %td= e.estudiante_instrumental_ci
      %td= Postgrado.where(:id => ExamenInstrumental.where(:id => e.examen_instrumental_id).first.postgrado_id).first.nombre
      %td= Idioma.where(:id => ExamenInstrumental.where(:id => e.examen_instrumental_id).first.idioma_id).first.descripcion
      -if e.status_id.to_s == 'P'
        %td Sin corregir
        %td= link_to "Corregir", {:action => "corregir_examen", :cedula => e.estudiante_instrumental_ci, :examen_id => e.examen_instrumental_id,
      -else
        -if e.status_id.to_s == 'A'
          %td Aprobado
          %td= link_to "Revisar", {:action => "revisar_examen", :cedula => e.estudiante_instrumental_ci, :examen_id => e.examen_instrumental_id,
        -else
          -if e.status_id.to_s == 'R'
            %td Reprobado
            %td= link_to "Revisar", {:action => "revisar_examen", :cedula => e.estudiante_instrumental_ci, :examen_id => e.examen_instrumental_id
          -else
            -if e.status_id.to_s == 'E'
              %td No ha presentado
              %td
```

Ilustración 55: Código e interfaz principal del módulo de corregir exámenes.

El código y la interfaz de la vista donde se listan todas las evaluaciones del módulo de corrección de exámenes se despliegan en la Ilustración 56:

```
=text_field :filter, :filter, :id => 'filter'
%p
  =link_to "Ver #{@mes}", :action => :index
%p
  %table.tablesorter{:style => "width:800px; margin:auto;"}
    %thead
      %td Fecha
      %td Nombre
      %td Cédula
      %td Postgrado
      %td Idioma
      %td Resultado
      %td Acciones
```

Ilustración 56: Código e interfaz de la vista donde se listan todas las evaluaciones del módulo de corrección de exámenes.

El código de la interfaz para corregir una evaluación se refleja en la Ilustración 57:

```
=form_tag(url_for(:action => :correccion_procesar, :cedula => @cedula, :examen_presentacion_id => @examen_presentacion_id)) do
  %table.tablesorter{:style => "width:800px; margin:auto;"}
    %thead
    %tbody
      %tr
        %td
          %b Contenido:
        %td=@contenido_examen.html_safe
      %tr
        %td
          %b Respuesta - resumen:
        %td
          %b=@respuesta_resumen.html_safe
      %tr
        %td
          %b Calificación:
        %td
          %select{:name => :calificacion_resumen, :id => :calificacion_resumen}
            %option{:value => 0} 0
            %option{:value => 0.25} 0.25
            %option{:value => 0.5} 0.5
            %option{:value => 0.75} 0.75
            %option{:value => 1} 1
      %tr
        %td
          %b Párrafo traducción:
        %td=@parrafo_traduccion
      %tr
        %td
          %b Respuesta - traducción:
        %td
          %b=@respuesta_traduccion.html_safe
      %tr
        %td
          %b Calificación:
```

Ilustración 57: Código de la interfaz para corregir una evaluación.

La primera y segunda parte de la interfaz de corrección de examen se muestran en la Ilustración 58:

Ilustración 58: Primera y segunda parte de la interfaz de corrección de examen.

El código y la interfaz para la interfaz de revisión del examen se observan en la Ilustración 59:

```
%td
  %b Calificación - resumen:
%td
  Nota en corrección anterior: #{@calificacion_resumen}
  %select{:name => :calificacion_resumen, :id => :calificacion_resumen}
    -if @calificacion_resumen == 0
      %option{:value => 0, :selected => true} 0
      %option{:value => 0.25} 0.25
      %option{:value => 0.5} 0.5
      %option{:value => 0.75} 0.75
      %option{:value => 1} 1
    -else
      -if @calificacion_resumen == 0.25
        %option{:value => 0} 0
        %option{:value => 0.25, :selected => true} 0.25
        %option{:value => 0.5} 0.5
        %option{:value => 0.75} 0.75
        %option{:value => 1} 1
```

	This usually entails exploring case-law reports, legal periodicals and legislation. Law practice also involves drafting documents such as court pleadings, persuasive briefs, contracts, or wills and trusts. Negotiation and dispute resolution skills (including ADR techniques) are also important to legal practice, depending on the field.
Respuesta - resumen:	**Respuesta resumen**
Calificación - resumen:	Nota en corrección anterior: 0.5 0
Párrafo traducción:	Law is a system of rules, usually enforced through a set of institutions. It shapes politics, economics and society in numerous ways and serves as a primary social mediator in relations between people. Contract law regulates everything from buying a bus ticket to trading on derivatives markets. Property law defines rights and obligations related to the transfer and title of personal and real property. Trust law applies to assets held for investment and financial security, while tort law allows claims for compensation if a person's rights or property are harmed. If the harm is criminalized in penal code, criminal law offers means by which the state can prosecute the perpetrator. Constitutional law provides a framework for the creation of law, the protection of human rights and the election of political representatives. Administrative law is used to review the decisions of government agencies, while international law governs affairs between sovereign nation states in activities ranging from trade to environmental regulation or military action.
Respuesta - traducción:	**Respuesta traduccion**
Calificación - traducción:	1.0
TOTAL:	0.75
Finalizar	

Ilustración 59: Código e interfaz para la interfaz de revisión del examen.

3.2.8.4 Pruebas

Se realizaron pruebas unitarias para verificar las validaciones en el módulo de gestión de examen y los resultados fueron los esperados. En estas interfaces el flujo del sistema fue el correcto y cada una de las acciones aplicadas a una evaluación (Corregir o Revisar) ejecutaban las operaciones pertinentes en la base datos.

Los datos con los que se trabajaron en las secciones de corrección y revisión de la evaluación fueron las respuestas obtenidas en las pruebas del módulo de presentación.

El algoritmo para el cálculo del porcentaje de similitud entre textos fue

integrado exitosamente, ya que cargaba el porcentaje automáticamente en un elemento *textfield* del formulario de la vista de corrección de un examen.

3.2.9 Iteración 8: Asignación de exámenes

Lun 22/Abr/2013 - Vie 03/May/2013

En esta iteración, en primer lugar, se realizan cambios necesarios en la base de datos, tales como crear nuevas tablas y agregar y/o cambiar algunas referencias entre tablas. También se desarrolla un método que sirve para seleccionar el examen que presenta un estudiante en específico.

3.2.9.1 Planificación

En la Ilustración 60 se muestran las historias de usuario desarrolladas en esta iteración:

Número	Descripción	Tipo	Fecha
2	Crear nuevos objetos en la base de datos.	Mejora	22/04/2013
16	Crear un método que asigne automática y aleatoriamente un examen a un estudiante según el postgrado y el idioma.	Nueva	22/04/2013

Ilustración 60: Historias de usuario para la Iteración 8.

3.2.9.2 Diseño

Las modificaciones en la base de datos serán las siguientes:

- Una tabla llamada *status_examen* para almacenar los diferentes estados en los cuales puede encontrarse una evaluación (A: Aprobado, R: Reprobado, P: Presentado, E: En espera).

- Se agrega una referencia en la tabla *estudiante_examen* a la tabla creada *status_examen* para conocer el estado de la presentación de una evaluación.

- Se agrega una referencia en la tabla *estudiante_examen* a la tabla *examen_instrumental* y se elimina la que anteriormente tenía a *examen_presentación*. Esto debido a que la tabla *examen_presentacion* sirve para almacenar las fechas que representan las jornadas de presentación de un examen así que no necesita la referencia a la tabla *examen_instrumental* la cual contiene los datos de las evaluaciones.

- Se eliminará en la tabla *estudiante_examen* la referencia a la tabla *postgrado,* ya que por medio de la referencia a la tabla se puede consultar el postgrado.

- Se agrega un nuevo campo en la tabla *estudiante_examen* llamado *tiempo_comienzo* que sirve para almacenar la hora de comienzo del examen de cada estudiante para así manejar el tiempo de presentación (2 horas).

En la Ilustración 61 puede verse una representación gráfica y parcial de la base de datos en la esta Iteración mediante un diagrama de tablas:

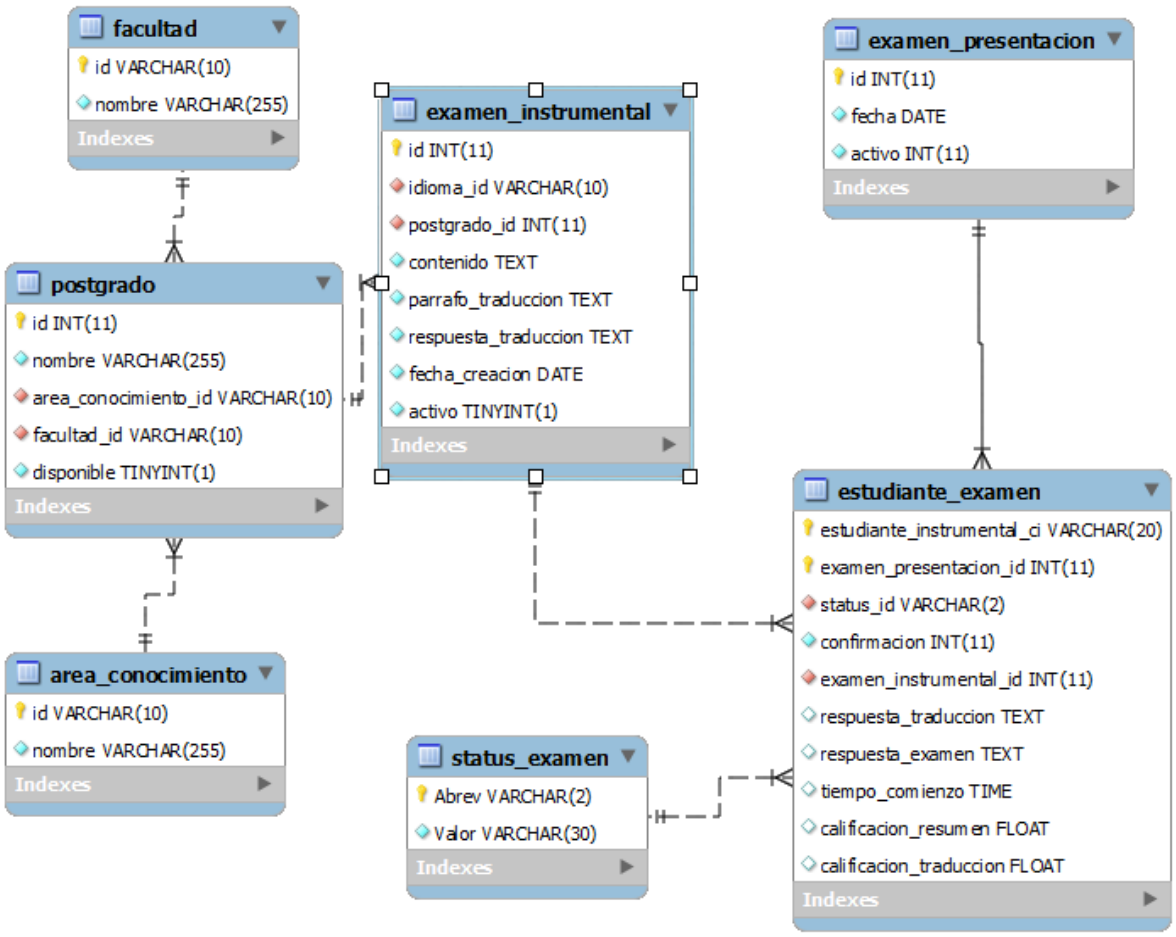


Ilustración 61: Diagrama de tablas para la Iteración 8.

Con respecto al método desarrollado, éste es parte del modelo *EstudianteExamen,* recibe como parámetros la cédula de identidad del estudiante y la identificación de su postgrado y seleccionará un examen aleatoriamente (de los exámenes activos) para asignarlo para su presentación. Se valida el caso en el cual el estudiante ya haya presentado alguno de los exámenes y se asigna otro (en caso de estar disponible).

3.2.9.3 Codificación

El código para la creación de los nuevos objetos en base de datos es el que sigue a continuación en la Ilustración 62:

```
CREATE  TABLE IF NOT EXISTS `atest_development`.`estudiante_examen` (
  `estudiante_instrumental_ci` VARCHAR(20) NOT NULL ,
  `examen_presentacion_id` INT(11) NOT NULL ,
  `status_id` VARCHAR(2) NOT NULL ,
  `confirmacion` INT(11) NOT NULL ,
  `examen_instrumental_id` INT(11) NOT NULL ,
  `respuesta_traduccion` TEXT NULL DEFAULT NULL ,
  `respuesta_examen` TEXT NULL DEFAULT NULL ,
  `tiempo_comienzo` TIME NULL DEFAULT NULL ,
  `calificacion_resumen` FLOAT NULL DEFAULT NULL ,
  `calificacion_traduccion` FLOAT NULL DEFAULT NULL ,
  PRIMARY KEY USING BTREE (`estudiante_instrumental_ci`, `examen_presentacion_id`) ,
  INDEX `fk_examen_v` (`examen_presentacion_id` ASC) ,
  INDEX `fk_status` (`status_id` ASC) ,
  INDEX `fk_examen_instrumental` (`examen_instrumental_id` ASC) ,
  CONSTRAINT `fk_constraint_examen_presentacion`
    FOREIGN KEY (`examen_presentacion_id` )
    REFERENCES `atest_provisional`.`examen_presentacion` (`id` )
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `fk_estudiante_instrumental`
    FOREIGN KEY (`estudiante_instrumental_ci` )
    REFERENCES `aceim_development`.`estudiante_instrumental` (`usuario_ci` )
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `fk_examen_instrumental`
    FOREIGN KEY (`examen_instrumental_id` )
    REFERENCES `atest_provisional`.`examen_instrumental` (`id` )
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `fk_status`
    FOREIGN KEY (`status_id` )
    REFERENCES `atest_provisional`.`status_examen` (`Abrev` )
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8
```

Ilustración 62: Código para la creación de nuevos objetos en base de datos para la Iteración 8.

El código del método para seleccionar el examen se presenta en la Ilustración 63:

```
def examen_instrumental_id_presentar(ci, postgrado_id)
  examen_seleccionado = nil
  i = 0
  presento_examen = false
  examenes = ExamenInstrumental.where(:postgrado_id => postgrado_id)
  cantidad_examenes = examenes.size
  examenes_desordenados = examenes.sort_by{rand}
  examenes_desordenados.each  do |e|
    if EstudianteExamen.where(:estudiante_instrumental_ci => ci, :examen_instrumental_id => e.id) == [] then
      presento_examen = false
    else
      presento_examen = true
    end
    i += 1
    if i >= cantidad_examenes or !presento_examen then
      examen_seleccionado = e
      break
    end
  end
  return examen_seleccionado.id
end
```

Ilustración 63: Código del método para seleccionar el examen a presentar.

3.2.9.4 Pruebas

Las pruebas de unidad de esta Iteración se listan en la Ilustración 64:

Historia de Usuario	Descripción del caso de prueba	Resultado esperado	Resultado Obtenido
16	Se registró en base de datos un estudiante del postgrado de derecho en el idioma Inglés que aún no ha presentado ninguna evaluación. Además se crearon dos exámenes de dicho postgrado en el idioma Inglés con id 4 y 5 respectivamente.	4 o 5	1° ejecución: **4** 2° ejecución: **4** 3° ejecución: **5** 4° ejecución: **4** 5° ejecución: **5**

	Se realizaron 5 ejecuciones del método.		
16	Se registró en base de datos un estudiante del postgrado de derecho en el idioma Inglés que ya ha presentado el examen con id 4. Además se crearon dos exámenes de dicho postgrado en el idioma Inglés con id 4 y 5 respectivamente. Se realizaron 5 ejecuciones del método.	5	1° ejecución: **5** 2° ejecución: **5** 3° ejecución: **5** 4° ejecución: **5** 5° ejecución: **5**
16	Se registró en base de datos un estudiante del postgrado de derecho en el idioma Inglés que ya ha presentado todos los exámenes disponibles del postgrado en el idioma. Además se crearon dos exámenes de dicho postgrado en el idioma Inglés con id 4 y 5 respectivamente. Se realizaron 5 ejecuciones del método.	4 o 5	1° ejecución: **5** 2° ejecución: **4** 3° ejecución: **5** 4° ejecución: **5** 5° ejecución: **4**

Ilustración 64: Pruebas para la Iteración 8.

3.2.1 Iteración 9: Controlar acceso a la presentación del examen

Lun 06/May/2013 - Vie 17/May/2013

En esta iteración, se creará una funcionalidad que permita generar claves de acceso para los estudiantes que presentarán la evaluación en un día específico para restringir el acceso a la presentación.

3.2.1.1 Planificación

En la Ilustración 65 se muestran las historias de usuario desarrolladas en esta iteración:

Número	Descripción	Tipo	Fecha
17	Generar claves de acceso para la presentación de la evaluación.	Nueva	06/05/2013

Ilustración 65: Historias de usuario para la Iteración 9.

3.2.1.2 Diseño

Se restringe el acceso a la presentación del examen, con el fin de que la evaluación se presente sólo en las máquinas del laboratorio de la Escuela de Idiomas Modernos a la hora que el personal encargado lo decida. Las claves estarán conformadas por 4 caracteres que pueden ser letras de la a 'a' la 'z' y números del '0' al '9'; se irá añadiendo caracter a caracter aleatoriamente para formar la cadena.

Estas claves estarán contenidas en un archivo con formato *pdf* junto a la cédula de identidad y nombre del estudiante, el idioma y el postgrado del examen, y un espacio para firmar. Este archivo, que servirá como lista para verificar asistencia, podrá ser generado desde el sistema con tan un sólo un clic desde el módulo de corrección de exámenes.

En la página principal estará un enlace los días en los que tenga lugar la presentación de una evaluación para que el encargado con permisos de administrador pueda generar el documento.

Consideraciones técnicas: la generación del archivo con formato *pdf* se realizará a través de la gema *prawn*. Se creará una clase que heredará de la clase

Prawn::Document con un método llamado *to_pdf* donde se codificará el diseño del documento con los datos correspondientes.

3.2.1.3 Codificación

El código del diseño del documento en formato *pdf* se muestra en la Ilustración 66:

```
def to_pdf(estudiante_examen)
  tiempo = Time.now
  fecha = tiempo.strftime("%Y-%m-%d")
  Prawn::Document.new(:page_layout => :portrait,:page_size => "LETTER") do |pdf|
    pdf.image "#{Rails.root}/app/assets/images/eim.png", :at => [0,730],:width => 90, :height => 70
    pdf.image "#{Rails.root}/app/assets/images/ucv.png", :at => [250,730],:width => 50, :height => 50
    pdf.image "#{Rails.root}/app/assets/images/fhe.png", :at => [470,730],:width => 60, :height => 60
    pdf.move_down 60
    pdf.text "UNIVERSIDAD CENTRAL DE VENEZUELA ", :align => :center, :size => 12
    pdf.text "Facultad de Humanidades y Educación", :align => :center, :size => 12
    pdf.formatted_text [{:text => "Coordinación de Extensión",:styles => [:italic, :bold],:size => 12}], :align => :center
    pdf.move_down 60
    pdf.formatted_text [{:text => "Listado de estudiantes - #{fecha.to_s}",:styles => [:bold],:size => 12}]
    pdf.move_down 30
    data = [["C.I.","NOMBRE","POSTGRADO","IDIOMA","CLAVE","FIRMA"]]
    estudiante_examen.each do |e|
      data << [e.estudiante_instrumental_ci, Usuario.where(:ci => e.estudiante_instrumental_ci).first.nombre_corto,
          Postgrado.where(:id => ExamenInstrumental.where(:id => e.examen_instrumental_id).first.postgrado_id).first.nombre,
          Idioma.where(:id => ExamenInstrumental.where(:id => e.examen_instrumental_id).first.idioma_id).first.descripcion , e.token_acceso, " "]
    end
    pdf.table(data, :header => true)

  end.render
```

Ilustración 66: Código para el diseño del documento en formato *pdf*.

El código para generar las claves desde el controlador se muestra en la Ilustración 67:

```
def generar_claves_acceso
   caracteres = ['a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r',
   's','t','u','v','w','x','y','z','0','1','2','3','4','5','6','7','8','9']
   tiempo = Time.now
   fecha = tiempo.strftime("%Y-%m-%d")
   @examen_presentacion = ExamenPresentacion.where(:fecha => fecha.to_date).collect{|x| x.id}
   @estudiante_examen = EstudianteExamen.where(:examen_presentacion_id  => @examen_presentacion)
   @estudiante_examen.each do |e|
     e.token_acceso = generar_clave(caracteres)
     e.save
   end
   @estudiante_examen = EstudianteExamen.where(:examen_presentacion_id  => @examen_presentacion)
   output = Report.new.to_pdf(@estudiante_examen)
   respond_to do |format|
     format.pdf do
       send_data output, :filename => "listado-estudiantes-#{fecha.to_s}.pdf",
                         :type => "application/pdf"
     end
   end
 end

 def generar_clave(caracteres)
   clave = ""
   for i in 0..3
     caracteres = caracteres.sort_by{rand}
     clave = clave + caracteres[0]
   end
   return clave
 end
```

Ilustración 67: Código en el controlador para generar las claves de acceso para la presentación del examen.

El código para colocar el enlace para generar la lista con las claves de acceso en la vista principal del módulo de corrección de exámenes se muestra en la Ilustración 68:

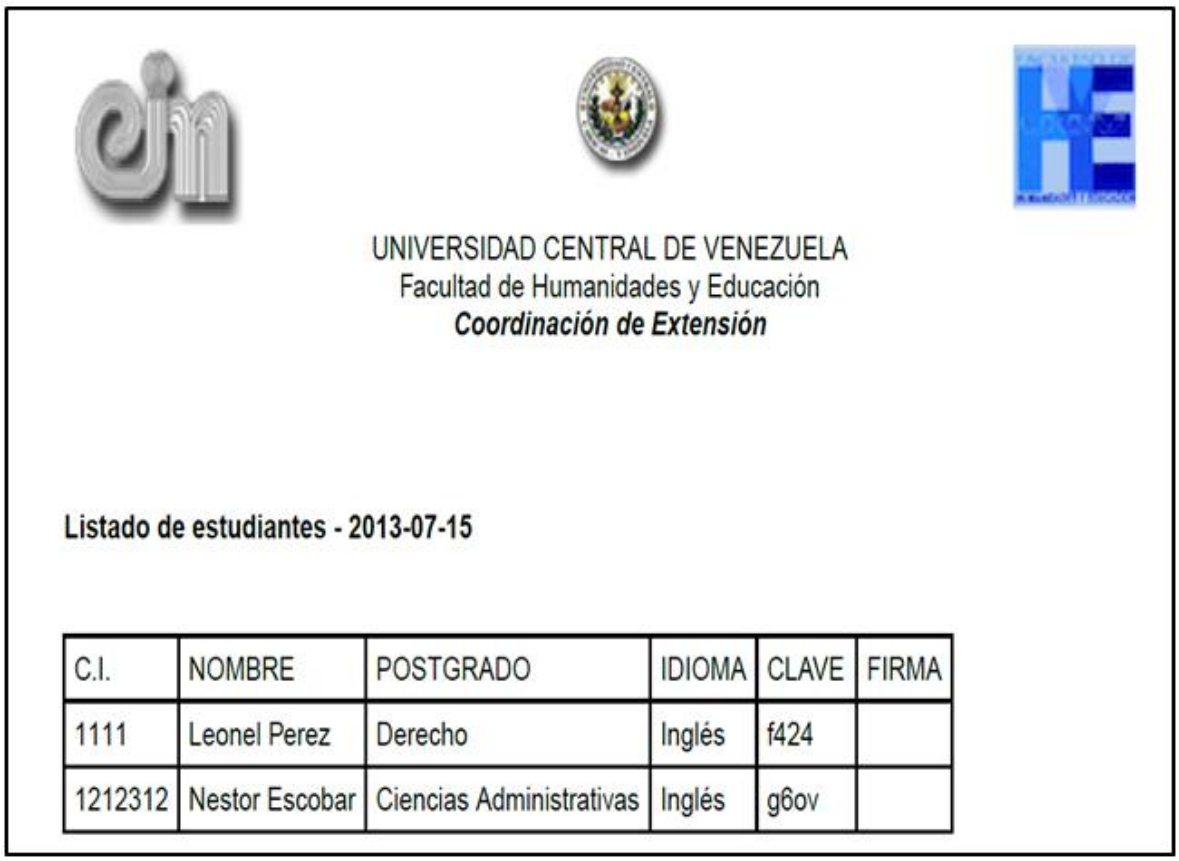
```
%p
  -if @presentar
    =link_to "Generar claves de acceso", :action => :generar_claves_acceso, :format => "pdf"
```

Ilustración 68: Código para colocar el enlace para generar la lista con las claves de acceso y la vista principal del módulo de corrección de exámenes.

3.2.1.4 Pruebas

Para realizar pruebas unitarias se registró en la base de datos la fecha 15 de julio del 2013 como fecha de presentación, también se registraron los datos de dos estudiantes, uno de ellos para presentar la evaluación en Inglés del postgrado de *Derecho* y el otro para el postgrado *Ciencias Administrativas*, ambas en el idioma Inglés.

El documento generado se puede observar en la Ilustración 69:



UNIVERSIDAD CENTRAL DE VENEZUELA
Facultad de Humanidades y Educación
Coordinación de Extensión

Listado de estudiantes - 2013-07-15

C.I.	NOMBRE	POSTGRADO	IDIOMA	CLAVE	FIRMA
1111	Leonel Perez	Derecho	Inglés	f424	
1212312	Nestor Escobar	Ciencias Administrativas	Inglés	g6ov	

Ilustración 69: Documento en formato *pdf* generado con las claves de acceso para la presentación del examen.

En efecto, las claves fueron generadas aleatoriamente de manera exitosa.

Conclusiones

El objetivo de este TEG consistió en la creación de un módulo para la gestión, elaboración y presentación en línea de la evaluación instrumental de idiomas, además que posibilite automatizar la corrección de la misma, la cual consiste en la comprensión lectora y actividades relacionadas con el idioma en cuestión, para los estudiantes de postgrado de la Universidad Central de Venezuela. Se logró la obtención de un producto que cumple con los requerimientos planteados por los usuarios al principio de este trabajo. Asimismo, se logró una integración exitosa entre las nuevas funcionalidades desarrolladas con las existentes en el sistema *aTesT*, haciéndolo cada vez más completo y contribuyendo con la automatización de un mayor número de tareas altamente demandadas e indispensables dentro del entorno académico de la Escuela de Idiomas Modernos.

Aún cuando el proceso de levantamiento de información fue difícil desde un comienzo, se logró captar todos los requerimientos, debido al buen diálogo entre el desarrollador y los usuarios. Los procesos y actividades tomados en cuenta en este TEG involucran a un grupo de personas que forman parte del personal de la Escuela de Idiomas y, en diversas ocasiones, contactarlos es una tarea ardua, sin embargo, algunas de ellas facilitaron el proceso, puesto que nos brindaron la información necesaria para realizar el análisis de requerimientos.

La elección de la técnica *Check* para realizar el algoritmo para obtener el porcentaje de similitud entre dos textos, se debió a que, las otras técnicas de detección de copias o similitudes en documentos revisadas durante la investigación no son adaptables para resolver el problema presentado, ya que algunas no cuantifican la similitud existente entre los texto, otras sólo detectan copias y no similitudes, y en la mayoría de los casos solamente podían ser aplicadas a textos de igual tamaño.

Se empleó una adaptación de esta técnica, ya que no se tomaron en cuenta

aspectos como: la característica estructural del texto, la verificación de originalidad y la abstracción de concebir el texto por párrafos o secciones.

Con respecto al impacto que puede tener emplear un algoritmo que corrija automáticamente la traducción de un examen de idiomas en el desarrollo de las actividades de la Escuela de Idiomas Modernos referentes a la evaluación instrumental, se destaca principalmente el ahorro de tiempo que puede estar presente en la corrección de las evaluaciones (realizada semanalmente) y lo novedoso de utilizar un algoritmo informático para resolver un problema de idiomas.

El personal de la Escuela de Idiomas Modernos se mostró abierto a nuevas ideas provenientes de la Facultad de Ciencias, y permitió poner en marcha un proyecto que tiene como uno de sus objetivos principales emplear una nueva manera de resolver un problema humanístico mediante la aplicación de ciencias de la computación.

El algoritmo fue integrado a la aplicación y arrojó los resultados esperados en las pruebas unitarias que se realizaron comparando dos traducciones de un determinado texto. Esto demuestra que una pieza de software, que en principio funciona para detectar plagio de documentos, puede ser adaptado para construir prototipos que tengan funcionalidades, como por ejemplo, la corrección automática de un examen de idiomas. Sin embargo como trabajo futuro se plantea tomar exámenes pasados ya corregidos y usarlos como entrada para probar el algoritmo, de manera que se compruebe la exactitud del factor de corrección utilizado para el algoritmo.

Cabe destacar que la corrección de la primera parte de las evaluaciones instrumentales, la cual consiste en hacer un resumen de todo el texto, es manual (en la aplicación), por lo tanto pudiésemos estar hablando en términos generales de corrección semiautomática de la evaluación instrumental.

El logro de la realización de los requerimientos se debe a las bases del marco conceptual y a la adaptación de la *programación extrema*. Una relativa libertad en el uso de los artefactos permitió un desarrollo objetivo, desapegado de dogmas y sobrecarga informativa. Algunas de las prácticas no pudieron realizarse al pie de la letra, por ejemplo la programación en parejas no se pudo realizar exactamente como lo dictan los cánones ya que un sólo desarrollador estuvo involucrado. Es un tipo de aplicación en la cual se requería en cierta medida la presencia del "cliente", de manera de que su características fueran bien conocidas y definidas en el trabajo de investigación, y las particularidades del sistema se iban revisando y corrigiendo con cada entrega.

El uso de *RubyOnRails* como tecnología del lado del servidor agilizó considerablemente el proceso de desarrollo, ya que se aprovecharon las diferentes estructuras y métodos predefinidos que posee, los cuales minimizan el tiempo y esfuerzo de codificación. Asimismo, el paradigma *MVC* que posee *Rails* brindó una organización al trabajar en el desarrollo, ya que permite tener una separación lógica y física de los componentes de la aplicación.

La realización de este Trabajo Especial de Grado ha aportado una experiencia muy enriquecedora acerca de cómo mantener el equilibrio en los aspectos de la programación, la flexibilidad del sistema frente a la correcta definición de las estructuras y métodos básicos y que no deberían cambiar mucho, la teoría y prácticas exitosas frente a la innovación. Muchas de las soluciones innovadoras o espontáneas del desarrollo de este sistema resultaron en aciertos recompensados, así como también en algunas dificultades técnicas.

En el desarrollo de la aplicación se utilizaron una diversas tecnologías, cada una de ellas muy beneficiosa pero que bajo una buena integración, aporta muchos más beneficios tanto al usuario como al desarrollador y a la aplicación misma. Una aplicación con buenas bases, que sea incrementable y que sea mantenible,

posiblemente le espera una vida larga.

Recomendaciones

A raíz de la implementación del módulo de corrección, en donde se abarcan funcionalidades de otorgar una calificación para el resumen del examen y de aplicar el algoritmo para obtener un porcentaje de similitud entre la traducción correcta y la del estudiante, se podría plantear la posible variación del factor de corrección de la traducción la cual es calificada como aprobada con un porcentaje de similitud mayor o igual a 75%. Se podrían generar datos estadísticos resultados de comparar, mediante el algoritmo de corrección, un examen modelo con una gran cantidad de evaluaciones clasificadas como correctas o incorrectas por personal especializado de la Escuela de Idiomas, con el fin de establecer umbrales determinados por números entre 0 y 1 que permitan catalogar como aprobado o reprobado un examen. En otras palabras, se habla de crear un módulo de afinación que le permita al módulo de corrección ser cada vez más exacto en la calificación otorgada.

Adicionalmente, se recomienda desarrollar una funcionalidad donde se puedan crear nuevos postgrados que en un futuro la Universidad Central de Venezuela pueda agregar a su catálogo de especializaciones, maestrías y/o doctorados.

Referencias Bibliográficas

(AJAX, 2012) Asynchronous JavaScript And XML. Consultado el 5 de abril del 2012 en http://www.webtaller.com/maletin/articulos/que-es-ajax.php

(Apache, 2012). Servidor HTTP Apache. Consultado el 6 de abril de 2012 en http://es.wikipedia.org/wiki/Servidor_HTTP_Apache

(Aplicaciones Web, 2012). Consultado el 6 de abril del 2012 en http://es.wikipedia.org/wiki/Aplicaci%C3%B3n_Web http://www.maestrosdelweb.com/editorial/css-3-mas-social-que-nunca/

(Arquitectura Cliente/Servidor, 2012) Consultado el 5 de abril del 2012 en http://www.tucodigofuente.com/2011/11/24/acceso-a-datos-programacion-por-capas/

(*Ballard & Moncur, 2012*) AJAX, JavaScript y PHP. Consultado el 5 de abril del 2012.

(Beck, 1997) Extreme Programming Explained. Consultado el 3 de enero del 2013

(Biblioteca Digital Universitaria de la DGSCA) Consultado el 31 de abril del 2012 en http://www.bibliodgsca.unam.mx/tesis/tes7cllg/sec_16.htm

(Brin, Davis & García-Molina, 1995) Copy detection mechanisms for digital documents. New York, NY, USA. Consultado el 33 de abril del 2012.

(Debian, 2012) El proyecto Debian. Consultado el 4 de abril del 2012 en http://www.debian.org/intro/about

(Eguíluz, 2012 .a) Introducción a JavaScript. Consultado el 5 de abril del 2012.

(Eguíluz, 2012 .b) Introducción a CSS. Consultado el 4 de abril del 2012.

(Flores & klopp, 2008). Desarrollo de una aplicación web que permite detectar similitudes entre documentos digitales. Consultado el 28 de julio del 2012 y el 26 de marzo del 2013.

(Harwani, 2012) jQuery Recipes. Consultado el 5 de abril del 2012.

(Hogan, 2012 .a) HTML5 & CSS3. Consultado el 4 de abril del 2012.

(Hogan, 2012. b) HTML5 & CSS3. Consultado el 4 de abril del 2012.

(*jQuery UI, 2012*). Consultado el 5 de abril del 2012 en http://jqueryui.com/home

(LAROUSSE, 1997) El pequeño Larousse ilustrado. México: Larousse. Consultado el 31 de abril de 2012.

(Metodología ágil, 2011) Consultado el 2 de enero del 2013 en http://www.scielo.br/scielo.php?pid=S1807-17752011000200008&script=sci_arttext

(MVC, 2012) Modelo Vista Controlar. Consultado el 4 de abril del 2012 en http://www.comusoft.com/modelo-vista-controlador-definicion-y-caracteristicas

(MySQL, 2012) MySQL. Consultado el 19 de abril del 2012 en http://www.webestilo.com/mysql/intro.phtml

(MSINFO Sistemas de Información) Consultado el 22 de abril del 2012 en http://www.msinfo.info/propuestas/documentos/documentos_digitales.html

(Patrón MVC, 2013) Modelo Vista Controlador. Consultado el 5 de marzo del 2013 http://librosweb.es/symfony_1_2/capitulo_2/el_patron_mvc.html

(Ruby & Thomas & Hansson, 2012) Agile Web Development with Rails Fourth Edition. Consultado el 6 de abril del 2012 y el 2 de enero del 2013.

(Si, Leong, & Lau, 1997) Check. A Document Plagiarism Detection System. Consultado el 28 de julio del 2012 y el 22 de marzo del 2013.

(Unicorn, 2012) Servidor HTTP Unicorn. Consultado el 8 de abril del 2012 en http://unicorn.bogomips.org/