Módulo Superación para la gestión de cursos de idiomas en la Universidad de las Ciencias Informáticas

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor:

Pedro Iván Fernández González

Tutores:

Msc. Roberkis Terrero Galano

Ing. José Alejandro García Calderón

La Habana, Cuba

Junio, 2019

“Año 61 de la Revolución”

Si no es correcto, no lo hagas.
Si no es verdad, no lo digas.

Proverbio japonés.

Declaración de autoría

Declaro por este medio que yo: Pedro Iván Fernández González, con carné de identidad 95070746665, soy el autor principal del trabajo final de tesis de pregrado que se titula: “Módulo Superación para la gestión de cursos de idiomas en la Universidad de las Ciencias Informáticas” y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

Y para que así conste, se firma la presente declaración jurada de autoría en La Habana, a los _____ días del mes de _____ del año 2019.

_____

Pedro Iván Fernández González

Autor

_____	_____
Msc. Roberkis Terrero Galano	**Ing. José Alejandro García Calderón**
Tutor	Tutor

Agradecimientos

Agradezco a mis padres por el apoyo incondicional que durante toda mi vida me han dado, a ellos se lo debo todo.

A mi tía Gilmi por haberme "aguantado" durante estos años en La Habana y ayudarme siempre que lo necesité.

A mis amigos: Guillermo y José Carlos, por la motivación y ayuda dada en momentos difíciles, y sobretodo los inolvidables recuerdos que me regalaron durante estos 5 años de la carrera.

A mis tutores, por el apoyo, el tiempo y los consejos brindados.

A mis profesores por los conocimientos y experiencias transmitidos aportando a mi desempeño como profesional y ciudadano cubano.

A Cuba, la Revolución y la Universidad de las Ciencias Informáticas.

Dedicatoria

A mis padres, mis amigos, mis profesores, mis tutores y todas las personas que contribuyeron de alguna manera a la realización de esta tesis.

Resumen

Resumen

La Universidad de las Ciencias Informáticas promueve la formación de profesionales capacitados en el dominio de idiomas extranjeros a través de las labores del Centro de Idiomas (CENID) de dicha institución. En este, se manejan grandes volúmenes de información para el cumplimiento de sus funciones, lo que en ocasiones dificulta la consulta inmediata y la actualización de la misma, así como la emisión de reportes. La presente investigación tiene como objetivo el desarrollo de un módulo integrado al Sistema de Gestión Académica que permita mejorar la eficiencia de los procesos vinculados a la gestión de los cursos impartidos por el centro. Para el desarrollo de la aplicación se utilizó la metodología Scrum y el entorno tecnológico definido por el Departamento de Desarrollo de la Dirección de Informatización, el cual es una adaptación del marco de trabajo CodeIgniter. Para la implementación del módulo se emplearon herramientas y tecnologías libres. Como resultado final se obtiene un módulo que apoya la informatización de la gestión de los cursos en dicho centro. También, facilita la búsqueda de información para la confección y administración de grupos docentes, asimismo contribuye a una mejor organización de los registros de evaluaciones y asistencia de los estudiantes.

Palabras clave: cursos, idiomas, aprendizaje semipresencial, sistemas de gestión.

Índice

Índice

Índice

Índice

Índice

Índice de figuras

Índice

Índice de tablas

Introducción

Introducción

El idioma es la lengua propia de una nación, un pueblo u otro grupo social, es el conjunto de reglas compartido por los individuos que se están comunicando permitiéndoles intercambiar pensamientos, ideas y emociones. En el mundo de hoy, las lenguas o idiomas forman parte clave de la cultura moderna, porque ayudan a ampliar el conocimiento e interactuar con personas de diferentes partes del mundo.

El creciente aumento de la investigación en las diferentes áreas del saber ha llevado a la necesidad de una actualización constante, en pro de los avances de la ciencia y la técnica. Para sobrevivir en este mundo competitivo, es importante asumir nuevos retos y oportunidades. A través del uso de las nuevas tecnologías de la información y las comunicaciones, el mundo se ha convertido en una pequeña aldea, donde el primero que recibe la información tendrá ventaja sobre aquellos que no la reciban o la reciban muy tarde. Es recomendable publicar los resultados de investigaciones en revistas internacionales accesibles y en la medida de lo posible, en una lengua que llegue a más lectores [1]. Resulta indispensable que un profesional domine su idioma materno, así como varios idiomas para competir en el mundo profesional actual. De esa manera favorecer el acceso a la cultura, al intercambio y la búsqueda del conocimiento [2].

En Cuba, como parte de la política de la enseñanza de la Educación Superior comenzó a implementarse a partir del año 2016 el requisito de demostrar el dominio del idioma extranjero inglés para graduarse de una carrera universitaria. La política de perfeccionamiento de la enseñanza del idioma está orientada a facilitar el proceso de formación integral de los estudiantes, como parte del ejercicio futuro de la profesión [2].

En la Universidad de las Ciencias Informáticas (UCI), la política se ha implementado en todas sus carreras (Ingeniería en Ciencias Informáticas, Ingeniería en Bioinformática, curso de ciclo corto de Administración de Redes y Seguridad Informática), teniendo como soporte el Centro de Idiomas (CENID). Cuya misión es facilitar, a la comunidad universitaria y extra-universitaria, el aprendizaje y uso de competencias comunicativas en idiomas, a través de servicios especializados de formación, evaluación, certificación traducción e interpretación [3]. Por consiguiente, diseña e implementa programas para el aprendizaje de idiomas que impulsan el proceso de internacionalización de la Universidad, también, propicia la formación de profesionales integrales con un alto nivel científico-productivo y el desarrollo de investigaciones. El centro fue inaugurado el 2 de noviembre del 2015 en el salón “Me dicen Cuba” de la institución. Inicia con la evaluación del idioma inglés, extendiéndose en lo adelante hacia el portugués, ruso, francés, italiano y alemán, cuenta con la bibliografía, materiales audiovisuales, información digital, laboratorios de audición y

una minisala de computación, como complemento de la tecnología digital necesaria para la enseñanza de esta disciplina. Entre sus funciones se encuentran:

- Diseñar y asesorar metodológicamente el trabajo en las disciplinas de idiomas que forman parte de los programas que se imparten de la universidad.
- Promover el diseño y elaboración de recursos educativos, principalmente digitales, en función de la enseñanza-aprendizaje de idiomas.
- Diseñar e implementar programas de español para extranjeros.

El Centro de Idiomas ofrece cursos presenciales y semi-presenciales en varios de los niveles del MCER[1] tanto para el pregrado como para el postgrado y trabajadores en general, según la disponibilidad de profesores y las necesidades específicas de la institución [3]. A pesar de los resultados satisfactorios obtenidos, los procesos que se llevan a cabo se realizan de forma semi-informatizada presentando las siguientes dificultades:

- La recopilación de los datos de los usuarios así como el almacenamiento de sus registros a través del uso de herramientas ofimática, atrasan y dificultan el manejo de la información debido a la magnitud de los datos que se procesan. Además, propician errores humanos en el tratamiento de documentos, como errores ortográficos e información incorrecta, duplicada o ausente.
- La generación de reportes y certificados carecen de un control de estandarización posibilitando la invalidación de los mismos y el gasto innecesario de recursos tales como: materiales de oficina y esfuerzo humano.
- La descentralización de los datos almacenados enlentece la disponibilidad de información actualizada.

La Universidad de las Ciencias Informáticas apoya sus procesos sustantivos mediante el uso del Sistema de Gestión Académica: Akademos. Dicho sistema se distribuye en diferentes módulos en función de las diversas áreas y roles que comprende la universidad.

A partir de la situación problemática existente se plantea el siguiente **problema de investigación**:

¿Cómo mejorar la eficiencia de la gestión de los cursos de idiomas en la Universidad de las Ciencias Informáticas? Para este contexto se define eficiencia como capacidad para obtener un alto grado de

[1]El Marco Común Europeo de Referencia para las lenguas: aprendizaje, enseñanza, evaluación (MCER, o CEFR en inglés) es un estándar europeo, utilizado también en otros países, que sirve para medir el nivel de comprensión y expresión oral y escrita en una determinada lengua.

fiabilidad e integridad de los datos registrados y emplear moderadamente el tiempo para dar respuesta a solicitudes de cursos.

La presente investigación centra su **objeto de estudio** en: la gestión de cursos académicos delimitándose como **campo de acción**: la gestión de los cursos de idiomas en la Universidad de las Ciencias Informáticas.

En consecuencia se define como **objetivo general**: Desarrollar un módulo del Sistema de Gestión Académica que mejore la eficiencia de la gestión de los cursos de idiomas en la Universidad de las Ciencias Informáticas.

A partir del objetivo general planteado se desglosan los siguientes **objetivos específicos**:

- Elaborar los fundamentos teóricos-metodológicos de la investigación asociados a la gestión de cursos académicos.
- Diseñar la propuesta de solución a partir del proceso de desarrollo de software utilizado.
- Desarrollar el módulo Superación para el Sistema de Gestión Académica.
- Validar la solución desarrollada.

Para dar solución a los objetivos trazados se plantea la siguiente **hipótesis**: El desarrollo del módulo Superación mejora la eficiencia de la gestión de los cursos de idiomas en la Universidad de las Ciencias Informáticas.

Identificándose como **variable independiente:** módulo Superación para la gestión de cursos de idiomas en la Universidad de las Ciencias Informáticas y como **variable dependiente:** eficiencia en el proceso de gestión de los cursos de idiomas en la Universidad de las Ciencias Informáticas.

Tabla 1: Operacionalización de las variables

Variable conceptual	Dimensión	Indicador	Subindicador	Unidad de medida
Módulo Superación para la gestión de cursos de idiomas en la Universidad de las Ciencias Informáticas	Universidad de las Ciencias Informáticas	Funcionalidad	Funcionalidad	(%)
		Usabilidad	Instructibilidad	(%)
			Operatividad	(%)
Eficiencia en el proceso de gestión de los cursos de	Usuarios que hacen uso del	Fiabilidad e integridad de los datos registrados.	Alta	95-100 (%)
			Media	85-95 (%)

idiomas en la Universidad de las Ciencias Informáticas	módulo.		Baja	0-85 (%)
		Tiempo empleado para dar respuesta a solicitudes de cursos	Alta	0-10 (min)
			Media	10-20 (min)
			Baja	Más de 20 (min)

Para garantizar el cumplimiento de los objetivos específicos trazados se establecen las siguientes **tareas de investigación**:

1. Caracterización de los sistemas informáticos utilizados para la gestión de cursos académicos.
2. Caracterización de las herramientas y el proceso de desarrollo de software a utilizar en la propuesta de solución.
3. Identificación de los requisitos funcionales y no funcionales de la propuesta de solución.
4. Implementación de las funcionalidades del módulo Superación.
5. Diseño de pruebas de software para medir la calidad del módulo Superación.
6. Validación de la propuesta de solución.

Para responder al desarrollo de la investigación se emplearon los siguientes métodos investigativos:

Teóricos:

- **Método de análisis y síntesis:** posibilita descomponer al objeto de estudio en todas sus partes y cualidades, en sus múltiples relaciones, propiedades y componentes. Le acredita la síntesis, y su operación inversa establece la unión o combinación de las partes previamente analizadas, lo que permite descubrir relaciones, características generales entre los elementos de la realidad y la relación de todas sus partes.
- **Método hipotético-deductivo**: parte de una hipótesis sustentada en el desarrollo teórico de una determinada ciencia, que sigue las reglas lógicas de la deducción y permite llegar a nuevas conclusiones y predicciones empíricas, las que a su vez son sometidas a verificaciones.

Empíricos:

- **Entrevista:** para obtener información referente a las necesidades del cliente. Muy eficaz para obtener datos relevantes. La información es idónea para cuantificar y dar tratamiento estadístico.
- **Análisis de documentos**: para consultar la literatura especializada, concretar la información necesaria para definir los campos válidos para la solución, así como los métodos ideales para llevarla a cabo.

El documento está estructurado por los siguientes capítulos:

Capítulo 1. Fundamentación teórica: en este capítulo se presentan los elementos teóricos que sirven de base a la investigación. Además, se describen los conceptos fundamentales asociados al dominio del problema, se realizan la descripción y el análisis de sistemas existentes, se caracterizan las herramientas y la metodología a utilizar en la implementación de la propuesta de solución.

Capítulo 2. Descripción de la solución informática: se describen las reglas del negocio, a partir del cual se construye el modelo conceptual. Se definen las técnicas para obtener los requisitos funcionales y no funcionales. Además, se describen la arquitectura y los patrones de diseño utilizados. También, se elaboran el modelo de base de datos y el diagrama de despliegue.

Capítulo 3. Implementación y validación de la solución propuesta: se especifican los estándares de codificación utilizados. Asimismo, se realizan y describen las pruebas de software definidas para garantizar su correcto funcionamiento.

Capítulo 1: Fundamentación teórica

1. Introducción

El presente capítulo comprende el estudio de varios conceptos y características generales asociados a la gestión de cursos académicos. También, se realiza un análisis de algunas herramientas existentes para la gestión e informatización del mismo. Se caracteriza el proceso de desarrollo de software, lenguajes de programación y tecnologías que se utilizan para el desarrollo de software en la web.

1.1. Marco conceptual

Curso: periodo de tiempo establecido de forma anual para el dictado de clases en una institución educativa [4]**.**

Aprendizaje semi-presencial: proceso de enseñanza y aprendizaje que incluye la enseñanza presencial y la apoyada en las TIC (Tecnologías de la Información y las Comunicaciones). Incorpora instrucción directa, indirecta, enseñanza colaborativa y aprendizaje asistido por computadora individualizado. Este aprendizaje requiere esfuerzos rigurosos, actitud correcta, presupuesto atractivo, así como, profesores y estudiantes altamente motivados para su implementación exitosa. Como incorpora diversos modos, es complejo y organizarlo es una tarea difícil [5].

El Centro de Idiomas de la Universidad de las Ciencias Informáticas aplica dos tipos principales de exámenes centralizados:

- **Examen de diagnóstico-colocación**: es una prueba multinivel de alrededor de dos horas de duración, en la que se evalúan las cuatro habilidades (comprensión de lectura, comprensión auditiva, escritura y expresión oral).
- **Exámenes de certificación de dominio de nivel:** pueden ser de dos tipos. De un solo nivel: una prueba de cerca de tres horas en la que se evalúan las cuatro habilidades para el nivel específico de que se trate. Multinivel: una prueba de cerca de cuatro horas de duración en la que se evalúan las cuatro habilidades en un rango de diferentes niveles y certifica el que demuestre el hablante [6].

1.2. Análisis de soluciones existentes

Con el objetivo de aumentar el nivel de eficiencia de los procesos de gestión de cursos de idiomas en el Centro de Idiomas, se realizó el análisis de sistemas nacionales e internacionales, que permiten apoyar el desarrollo de la investigación para lograr adaptar el Sistema de Gestión Académica a las necesidades de los clientes.

1.2.1. Atenea

Atenea es un software de gestión de academias totalmente en línea. La disponibilidad de la información al instante y la comunicación continua con alumnos y clientes, son dos pilares importantes en la gestión de un centro de formación. Este permite la comunicación entre alumnos, familias, profesores y la administración del centro. Al mismo tiempo, dispone de todos los procesos de gestión necesarios para cualquier academia y otras entidades de formación. Además, se adapta a la legislación en términos de gestión económica y pedagógica y está preparado para ser modificado ante futuros cambios, ya que cuenta con una estructura abierta [7].

1.2.2. OfiELE

OfiELE es un software para academias y escuelas de enseñanza desarrollado por Ofimática para la gestión de academias de idiomas y escuelas en general. Desarrollado bajo la filosofía Cloud Computing (en la "nube"). El software puede ser instalado en los ordenadores locales de una escuela o en un servidor de Internet, así se puede trabajar desde cualquier lugar. Distribuido por módulos y gestión multipuesto y multiusuario por medio de contraseñas y concesión de permisos a los usuarios. Está integrado con OfiCRM, que es un software CRM (Customer Relationship Management – Gestión de la relación con el cliente) opcional para la gestión de la relación entre alumnos y grupos. Los comerciales tienen toda su actividad planificada, llevando un seguimiento de todos los contactos y actividades pendientes, así como sus comunicaciones [8].

1.2.3. aGora

aGora es un software ERP (Enterprise Resource Planning – Planificación de Recursos Empresariales) y CRM para gestión de academias, escuelas y centros de formación. Desde un mismo entorno permite gestionar alumnos y clientes, definir una planificación académica, controlar los gastos, emitir facturas o gestionar los cobros. Ofrece utilidades y herramientas de decisión e inteligencia de negocio que ayudan a responder preguntas, como por ejemplo en qué periodo del año se ha realizado la mayor matrícula y cuántos alumnos del año pasado han vuelto matricular este año. En aGora existen hasta cinco niveles de acceso, los habituales de "total", "sólo lectura" o "sin permiso" y además, para algunos apartados, tendrás los de "sólo lectura de datos propios" y "permiso total de datos propios". Con estos permisos especiales

permite que un profesor pueda entrar al programa a ver las calificaciones, pero solo aquellas que son de los alumnos que van a los cursos que éste imparte. Gestiona de forma completa y sencilla el envío de correos a alumnos, clientes, empleados y proveedores. Posibilita realizar envíos masivos con filtros personalizados y mediante campos combinados. Además, gestiona la recepción de correos de las cuentas de correo electrónico. El sistema vincula de forma automática a los alumnos, clientes y empleados que los han enviado [9].

1.2.4. SIGENU

Para apoyar el desempeño de los procesos docentes en la educación superior en Cuba existe el Sistema de Gestión de la Nueva Universidad (SIGENU) que cuenta con varios módulos desplegados en la mayor parte de las instituciones de educación superior en el país. El módulo SIGENU-WEB se encarga de gestionar la información de los estudiantes desde que realizan la matrícula hasta que se gradúan o causan baja de una Institución de Educación Superior (IES). En el nivel táctico se encuentra el módulo Sistema de Apoyo para la Toma de Decisiones (SIGENU-DSS) que permite el apoyo a la toma de decisiones en las IES acorde a los principales procesos docentes como: "matrícula", "bajas" y "graduados". En relación con el módulo Sistema de Información Docente de la Educación Superior Cubana (DMMES) se encarga del proceso de toma de decisiones pero a nivel del ministerio. El sistema SIGENU-DSS y el sistema DMMES permiten el manejo de información que favorece a los decisores tomar decisiones acertadas basándose no exclusivamente en su juicio o intuición sino en información generada a través de métodos deductivos y analíticos [10].

1.2.5. Akademos

Akademos, es un sistema desarrollado en la UCI, fue puesto en práctica en el año 2004 y permite la gestión automatizada de los elementos que intervienen en la labor académica. Mitiga los riesgos que impone el dinamismo del proceso de gestión académica, enfrentando de forma natural los cambios requeridos y adaptándose a las nuevas condiciones. Además, brinda un papel activo a todos los involucrados en el proceso, dígase profesores, estudiantes, directivos y personal de secretaría [11].

Resultado del análisis de las soluciones existentes

Las herramientas descritas, ofrecen funcionalidades que se acoplan a las tareas del negocio, sin embargo poseen las siguientes deficiencias:

- No apoyan la política y los lineamientos de informatización del país sobre el uso del software libre para garantizar la soberanía tecnológica puesto que son de código privativo [12]: aGora, Ofiele, Atenea.
- Requieren de pago de licencias: aGora, Ofiele, Atenea.
- Carecen de integración de los datos con aplicaciones externas: aGora, Ofiele, Atenea, SIGENU.
- Carecen de completitud funcional: SIGENU, Akademos.

De acuerdo con los razonamientos que se han venido realizando, estas herramientas se descartan como candidatas para su despliegue e implantación, no obstante, dada la experiencia que poseen aportan pautas en el desarrollo de soluciones similares, tales como:

- La implementación de un sistema multiusuario con permisos favorece al control, acceso y modificación de la información.
- La generación de listados y reportes permite el ahorro de tiempo y contribuye a la toma de decisiones.
- El desarrollo de la aplicación sobre plataforma web permite mayor flexibilidad a los usuarios, al solicitar matricular en determinado curso y ser notificados de su aceptación o no, desde cualquier lugar.
- El diseño de una arquitectura escalable, ofrece mejor respuesta de adaptación a cambios necesarios en el negocio.

1.3. Lenguajes, marco de trabajo y metodología de desarrollo a utilizar

1.3.1. HTML

Versión: 4.2

Descripción: HTML (HyperText Markup Language, Lenguaje de Marcado de Hipertexto) es un lenguaje descriptivo que especifica la estructura de una página web, documento de texto plano estructurado con elementos rodeados por etiquetas de apertura y cierre coincidentes. Cada etiqueta comienza y termina con corchetes angulares (<>), algunas de ellas son vacías, es decir, no pueden encerrar ningún texto, por ejemplo <img>. Las etiquetas HTML se pueden extender con atributos, los que proporcionan información

adicional que afecta la forma en que el navegador interpreta el elemento. Un archivo HTML puede ser procesado por cualquier navegador web [13].

1.3.2. CSS

Versión: 3.0

Descripción: CSS (Cascading Style Sheet, Hoja de Estilo en Cascada por sus siglas en inglés) es el lenguaje para describir la presentación de las páginas web, incluidos los colores, el diseño y las fuentes. Permite adaptar la presentación a diferentes tipos de dispositivos. Es independiente de HTML y se puede utilizar con cualquier lenguaje de marcado basado en XML[2]. La separación de HTML y el CSS hace que sea más fácil mantener los sitios, compartir hojas de estilo en las páginas y adaptar las páginas a diferentes entornos. Esto se conoce como la separación de estructura (o contenido) de la presentación [14].

1.3.3. JavaScript

Versión: 1.8.5

Descripción: Es un lenguaje de programación ligero interpretado o compilado en JIT (Just In Time, justo a tiempo) con funciones de primera clase. Es conocido como el lenguaje de scripting para páginas web, pero también, es utilizado en el servidor. Además, es un lenguaje dinámico basado en prototipos, multi-paradigma y soporta estilos orientados a objetos, imperativos y declarativos (por ejemplo, programación funcional) [15].

1.3.4. PHP

Versión: 7.2

Descripción: PHP: Hypertext Preprocessor (Preprocesador de Hipertexto PHP) es un lenguaje de código abierto interpretado, de alto nivel, embebido en páginas HTML y ejecutado en el servidor web. Este no se encuentra limitado a resultados en HTML, también, puede presentar otros resultados, como XHTML[3] y

[2]XML(siglas en inglés de eXtensible Markup Language) es un lenguaje de marcado sencillo similar al HTML. Su objetivo es facilitar la representación, almacenamiento y trasmisión de información.

[3]XHTML, acrónimo en inglés de eXtensible Hypertext Markup Language (lenguaje extensible de marcado de hipertexto), es el lenguaje de marcado pensado para sustituir a HTML como estándar para las páginas web.

archivos XML. Asimismo, puede auto-generar estos archivos y almacenarlos en el sistema en lugar de presentarlos en la pantalla. La característica más potente y destacable es su soporte para una gran cantidad de base de datos, por lo que escribir una interfaz vía web para acceder a diferentes sistemas gestores de bases de datos, es una tarea viable, efectiva y rápida con dicho lenguaje [16].

1.3.5. SQL

Descripción: Structured Query Language (Lenguaje de consultas estructurado), es un lenguaje de programación diseñado para administrar datos almacenados en bases de datos relacionales. Opera a través de declaraciones simples declarativas. Esto mantiene los datos precisos y seguros, y ayuda a mantener la integridad de las bases de datos, independientemente del tamaño [17].

1.3.6. UML

Versión: 2.5

Descripción: Lenguaje Unificado de Modelado (UML por sus siglas en inglés) es un lenguaje informático gráfico o textual que proporciona el diseño y la construcción de estructuras y modelos siguiendo un conjunto sistemático de reglas y marcos. El lenguaje de modelado es similar al lenguaje artificial [18]. Por otra parte, es un lenguaje de modelado estandarizado que permite a los desarrolladores especificar, visualizar, construir y documentar artefactos de un sistema de software. Por lo tanto, UML hace que estos artefactos sean escalables, seguros y robustos en ejecución. UML es un aspecto importante involucrado en el desarrollo de software orientado a objetos. Utiliza la notación gráfica para crear modelos visuales de sistemas de software [19].

1.3.7. Visual Paradigm

Versión: 15.0

Descripción: Herramienta CASE(Compute-Aided Software Engineering, Ingeniería de Software Asistida por Computación) que propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación. Además, proporciona a los desarrolladores de software la plataforma de desarrollo de vanguardia para crear aplicaciones de calidad. Facilita una excelente interoperabilidad con otras herramientas CASE [20].

1.3.8. GUUD

Versión: 2.0

Descripción: El Departamento de Desarrollo de la Dirección de Informatización (DIN) define para el desarrollo de la propuesta de solución como marco de trabajo GUUD, este permite la creación de aplicaciones web escritas en PHP. Constituye un híbrido entre el marco de trabajo de PHP, CodeIgniter en su versión 1.7.3 y la librería de JavaScript, jQuery en su versión 1.9.2.

1.3.9. CodeIgniter

Versión: 1.7.3

Descripción: CodeIgniter es un marco de trabajo que incluye un conjunto de herramientas para crear sitios web utilizando PHP. Su objetivo es permitir desarrollar proyectos mucho más rápido que escribiendo código desde cero, al proporcionar un amplio conjunto de bibliotecas para las tareas más comunes, así como una interfaz simple y una estructura lógica para acceder a estas bibliotecas. Por otro lado, le permite al desarrollador enfocarse en su proyecto al minimizar la cantidad de código necesario para una tarea determinada. Además, implementa el proceso de desarrollo el patrón Modelo-Vista-Controlador (Model-View-Controller, MVC), que es un estándar de programación de aplicaciones utilizadas, tanto para hacer sitios web, como programas tradicionales [21].

1.3.10. jQuery

Versión: 1.9.2

Descripción: jQuery es una biblioteca de JavaScript rápida, pequeña y con muchas funciones. Permite la manipulación de documentos HTML, el manejo de eventos, la animación y AJAX[4] mucho más simples con una API[5] fácil de usar que funciona en una gran cantidad de navegadores. Con una combinación de versatilidad y extensibilidad jQuery cambia la forma en que los desarrolladores utilizan JavaScript [22].

[4]AJAX, acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications).

[5]API: La interfaz de programación de aplicaciones, conocida también por la sigla API, en inglés, application programming interface.

1.3.11. Git

Versión: 2.17

Descripción: Es un sistema de control de versiones distribuido de código abierto y gratuito, diseñado para manejar tanto proyectos pequeños como proyectos muy grandes, con rapidez y eficiencia. Es fácil de aprender y con un rendimiento rápido. Supera a las herramientas de SCM (Source Code Management, Gestión de Código Fuente) como Subversion, CVS, Perforce y ClearCase con características como bifurcaciones locales, áreas de preparación convenientes y múltiples flujos de trabajo [23].

1.3.12. Pencil

Versión: 3.0.4

Descripción: Es una herramienta de creación de prototipos GUI (Graphical User Interface, Interfaz Gráfica de Usuario) de código abierto que está disponible para todas las plataformas. Está diseñado con el propósito de proporcionar una herramienta de creación de prototipos para que las personas puedan instalar y usar fácilmente para crear maquetas en las plataformas de escritorio populares. Además, ofrece varias colecciones de formas integradas para dibujar diferentes tipos de interfaz de usuario, desde plataformas de escritorio a móviles [24].

1.3.13. Nginx

Versión: 1.14

Descripción: Es un software de código abierto para servidores web, proxy inverso, almacenamiento en caché, balanceo de carga, transmisión de medios y más. Comenzó como servidor web diseñado para el máximo rendimiento y estabilidad. Además, de sus capacidades de servidor HTTP[6], el Nginx también, puede funcionar como un servidor proxy para correo electrónico (IMAP[7], POP3[8] y SMTP[9]) y como equilibrador de carga para servidores HTTP, TCP[10] y UDP[11] [25].

[6](Hypertext Transfer Protocol, Protocolo de Transferencia de Hipertexto)

[7](Internet Message Access Protocol, Protocolo de Acceso a Mensajes de Internet)

[8](Post Office Protocol, Protocolo de Oficina de Correos)

[9](Simple Mail Transfer Protocol, Protocolo Sencillo de Transferencia de Correo)

[10](Transfer Control Protocol, Protoco de Control de Transferencia)

[11](User Datagram Protocol, Protocolo Datagrama de Usuario)

1.3.14. Netbeans IDE

Versión: 8.2

Descripción: NetBeans IDE es el IDE (Integrated Development Environment, Entorno Integrado de Desarrollo) oficial para Java en su versión 8. Es gratuito, de código abierto y tiene una gran comunidad de usuarios y desarrolladores en todo el mundo. Permite desarrollar aplicaciones de escritorio, móviles y web de Java, así como aplicaciones HTML5 con HTML, JavaScript y CSS. El IDE también, proporciona un gran conjunto de herramientas para desarrolladores de PHP y C / C ++ [26].

1.3.15. PostgreSQL

Versión: 9.5

Descripción: PostgreSQL es un potente sistema gestor de base de datos relacional de objetos de código abierto que utiliza y amplía el lenguaje SQL combinado con muchas características que almacenan y escalan de forma segura las cargas de trabajo de datos más complicadas. Tiene una sólida reputación por su arquitectura probada, confiabilidad, integridad de datos, conjunto de características sólidas, extensibilidad y la dedicación de la comunidad de código abierto detrás del software para ofrecer constantemente soluciones innovadoras y de alto rendimiento. Este se ejecuta en todos los sistemas operativos principales, ha sido compatible con ACID[12] desde 2001. También, posee muchas características destinadas a ayudar a los desarrolladores a crear aplicaciones, a los administradores a proteger la integridad de los datos y a crear entornos tolerantes a fallas, y a administrar sus datos sin importar cuán grande o pequeño sea el conjunto de datos. Además de ser gratuito y de código abierto, PostgreSQL es altamente extensible [27].

1.3.16. pgAdmin III

Versión: 1.22.2

Descripción: pgAdmin III es un sistema completo de gestión y diseño de bases de datos PostgreSQL para sistemas Unix y Windows. Está disponible de forma gratuita según los términos de la licencia de PostgreSQL y puede redistribuirse siempre que se cumplan los términos de la licencia. El proyecto es administrado por el equipo de desarrollo pgAdmin. Su última versión es la III la cual fue desarrollada como

[12]ACID (Atomicity Consistency Integrity and Durability, Atomicidad Consistencia Integridad y Durabilidad)

sucesor de los productos pgAdmin y pgAdmin II originales, que, aunque populares, tenían limitaciones en el diseño que impedían que se los llevara a un siguiente nivel [28].

1.3.17. Firefox Developer Edition

Versión: 65.0b12

Descripción: Firefox Developer Edition es una versión de Firefox adaptada para desarrolladores, ofreciendo características y herramientas de desarrollo experimental. Permite examinar, explorar y depurar sitios y aplicaciones web, así como diagnosticar problemas de rendimiento. A continucación se presentan algunos de sus componentes:

- **Consola web**: posibilita visualizar mensajes de registro (logs) en una página web e interactuar con la página usando JavaScript.
- **Inspector de página:** permite revisar y modificar el HTML y CSS de la página.
- **Depurador JavaScript:** permite paso a paso, examinar y modificar el JavaScript que está ejecuntándose en una página.
- **Monitor de red:** muestra las solicitudes de red hechas cuando una página está cargada.
- **Herramienta de rendimiento:** analiza la capacidad de respuesta del sitio web, del JavaScript y el rendimiento del diseño.
- **Desempeño de red:** muestra cuánto tardan en cargar las partes del sitio web.

1.3.18. Scrum

Scrum es una metodología de desarrollo ágil para gestionar proyectos de software. Es ligera, simple de entender y difícil de dominar. Dentro del cual se pueden emplear varios procesos y técnicas, que aclaran la eficacia relativa de la gestión de su producto y las técnicas de trabajo para mejorar continuamente el producto, el equipo y el entorno de trabajo. Scrum está formado por los equipos de Scrum y sus roles, eventos, artefactos y reglas asociadas. Cada componente dentro del marco cumple con un propósito y es esencial para el éxito y uso del mismo. Las reglas unen los roles, eventos y artefactos, gobernando las relaciones y la interacción entre ellos [29].

Características:

- **Es un modo de desarrollo de carácter adaptable**. Lo que implica que el diseño y la estructura del resultado se construyen de forma evolutiva, esto significa que no se considera que la

descripción detallada del producto, del servicio, de la estrategia o de la arquitectura del software (según el caso) deban realizarse en la primera "fase" del proyecto.

- **Es orientado a las personas antes que a los procesos**. En Scrum los equipos son auto-organizados, con margen de decisión suficiente para tomar las decisiones que consideren oportunas. Las prácticas y el entorno de trabajo ágiles facilitan la colaboración del equipo, que es necesaria y debe basarse en la colaboración abierta entre todos según los conocimientos y capacidades de cada persona, y no según su rol o puesto.
- **Emplea un desarrollo ágil**: iterativo e incremental. El desarrollo incremental implica que al final de cada iteración se dispone de una parte del producto operativa(incremento) que se puede inspeccionar y evaluar.

Actividades:

1. **Planificación de la iteración:**

   En Scrum el desarrollo se inicia desde la visión general de producto, detallando solo a las funcionalidades que, por ser las de mayor prioridad para el negocio, se van a desarrollar en primer lugar, y pueden llevarse a cabo en un período de tiempo breve (entre 15 y 60 días).

   - Selección de requisitos (2 horas): el cliente presenta al equipo la lista de requisitos priorizada del producto o proyecto. El equipo pregunta al cliente las dudas que surgen y selecciona los requisitos más prioritarios que prevé que podrá completar en la iteración, de manera que puedan ser entregados si el cliente lo solicita.
   - Planificación de la iteración (2 horas): el equipo elabora la lista de tareas de la iteración necesarias para desarrollar los requisitos seleccionados. La estimación de esfuerzo se hace de manera conjunta y los miembros del equipo se auto-asignan las tareas, se auto-organizan para trabajar incluso en parejas (o grupos mayores) con el fin de compartir conocimiento (creando un equipo más resiliente) o para resolver juntos objetivos especialmente complejos.

2. **Ejecución de la iteración:**

   Cada uno de los ciclos de desarrollo es una iteración (sprint) que produce un incremento terminado del producto. Al final de cada sprint o iteración, se realiza una revisión con todas las personas implicadas en el proyecto. Este es el período máximo que se puede tardar en reconducir una desviación del proyecto o de las circunstancias del producto.

- Reunión de sincronización (15 minutos): se realiza cada día por el equipo, normalmente delante de un tablero físico o pizarra (Scrum Taskboard). El equipo inspecciona el trabajo que el resto está realizando (dependencias entre tareas, progreso hacia el objetivo de la iteración, obstáculos que pueden impedir este objetivo) para poder hacer las adaptaciones necesarias que permitan cumplir con la previsión de objetivos a mostrar al final de la iteración.

3. **Inspección y adaptación:**
   - Revisión (demostración de 1.5 horas): el equipo presenta al cliente los requisitos completados en la iteración, en forma de incremento de producto preparado para ser entregado. En función de los resultados mostrados y de los cambios del proyecto, el cliente realiza las adaptaciones necesarias de manera objetiva, desde la primera iteración, replanificando el proyecto.
   - Retrospectiva (1.5 horas): el equipo analiza cómo ha sido su manera de trabajar y cuáles son los problemas que podrían impedirle progresar adecuadamente, mejorando de manera continua su productividad. El facilitador se encargará de eliminar o escalar los obstáculos identificados que estén más allá del ámbito de acción del equipo.

Scrum propone prácticas de refactorización en las tareas de diseño y codificación para evitar los problemas de degradación del sistema o de la arquitectura por la evolución continua del producto.

Los principales artefactos definidos por Scrum son:

- Product backlog o Requisitos del sistema: es el inventario de características que el propietario del producto requiere, ordenado por orden de prioridad. El mismo parte de la visión del resultado que se desea obtener; y evoluciona durante el desarrollo.
- Sprint Backlog: lista de los trabajos que realizará el equipo durante el sprint para generar el incremento previsto.
- Incremento: se trata de un resultado completamente terminado y en condiciones de ser usado obtenido luego de cada sprint.

Los roles definidos por Scrum son:

- Product Owner (Propietario del producto): es una persona conocedora del entorno de negocio del cliente y de la visión del producto. Representa a todos los interesados en el producto final y es el responsable del Product Backlog.
- Equipo de desarrollo: es un conjunto multidisciplinar de personas que cubre todas las habilidades necesarias para generar el resultado. Es capaz de auto-gestionarse y auto-organizarse, y dispone de atribuciones suficientes en la organización para tomar decisiones sobre cómo realizar su trabajo.
- Scrum manager: debe garantizar el funcionamiento de los procesos y la metodología.

1.4. Conclusiones parciales

A partir de todo lo anteriormente planteado se pueden arribar a las siguientes conclusiones:

- Se adquirieron los conocimientos sobre las características principales de sistemas de gestión de cursos académicos mediante el estudio de sistemas homólogos en el panorama nacional e internacional.
- Se caracterizaron las herramientas, marco de trabajo, metodología y lenguajes de programación que permitieron formar las bases propicias para la implementación del módulo Superación.

Capítulo 2. Descripción de la propuesta de solución

2. Introducción

En el presente capítulo se presentan las características de la propuesta de solución, se define un modelo conceptual del negocio y se reflejan las reglas del mismo. Además, se definen las técnicas de obtención de requisitos, permitiendo precisar los requisitos funcionales y no funcionales. Se describe la arquitectura con el patrón arquitectónico utilizado, así como los patrones de diseño. Se representa el modelo de base de datos y el diagrama de despliegue de la propuesta de solución.

2.1. Modelo conceptual

El modelado conceptual es la actividad de decidir qué modelar. Un modelo conceptual es "una descripción no específica del software del modelo de simulación por computadora (que será, está o ha sido desarrollado), que describe los objetivos, entradas, salidas, contenido, supuestos y simplificaciones del modelo" [30].

A partir de un análisis a los procesos que requieren ser informatizados, se realizó un modelo conceptual donde se aprecian los diferentes conceptos asociados a la gestión de los cursos de idiomas por parte de los profesionales del centro. A continuación se presenta el modelo conceptual obtenido (ver figura 1).

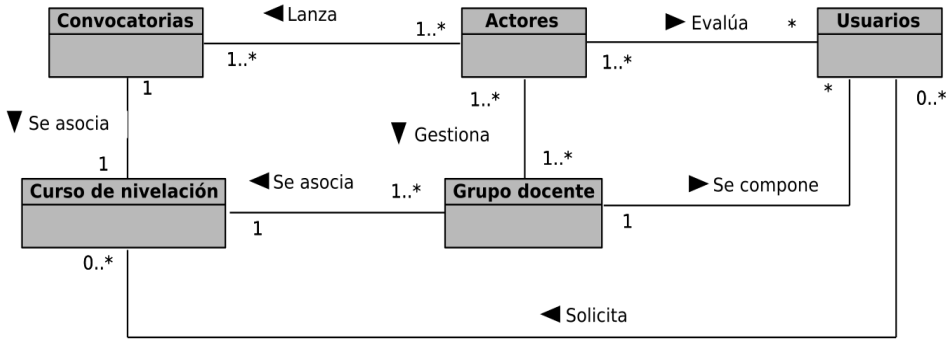


Figura 1: Modelo conceptual del negocio

A continuación se describen los conceptos asociados representados en el modelo conceptual:

Actores: trabajadores del Centro de Idiomas.

Usuarios: estudiantes, trabajadores e invitados de la universidad que acceden a los servicios que ofrece el Centro de Idiomas.

Curso de nivelación: curso impartido por el CENID de un idioma con determinado nivel.

Grupo docente: conjunto de usuarios asociados a un curso de nivelación.
Convocatoria: anuncio para un determinado curso de nivelación.

2.2. Descripción de las reglas del negocio

Las reglas del negocio son declaraciones precisas que describen, restringen y controlan la estructura, las operaciones y la estrategia de un negocio. Estas, se pueden encontrar en todas partes en forma no estructurada. Podemos describir una regla de negocios como "una declaración que define o restringe algún aspecto del negocio. Tiene la intención de afirmar la estructura del negocio, controlar o influir en el comportamiento del negocio" [31]. Constituyen la parte más cambiante del negocio. En el Centro de Idiomas, respecto a los cursos, se definen las siguientes reglas:

Generales:

- Los cursos son impartidos durante el transcurso de un semestre.
- Los cursos pueden ser cerrados (destinado a un grupo exclusivo de usuarios) o abiertos (cualquier usuario puede hacer solicitud del mismo).

Matrícula:

- La matrícula de los estudiantes a los cursos es administrada por las respectivas facultades a las que pertenecen.
- Si el usuario no ha realizado anteriormente un examen de colocación solo puede matricular en el curso del nivel más bajo.
- El usuario no puede matricular en un curso de mayor nivel del que este haya demostrado dominio en los exámenes de colocación y certificación.
- El usuario no puede matricular en un curso de nivel inferior al que haya demostrado dominio anteriormente en un examen de certificación o en un curso de nivelación.

Evaluaciones:

- Los usuarios serán evaluados según la cantidad de evaluaciones y frecuencias que definan los actores para cada curso.
- Los actores determinarán la aprobación o no de los usuarios durante cualquier momento del transcurso del curso.

2.3. Requisitos del sistema

Para saber lo que el cliente necesita resulta de vital importancia capturar los requisitos así como las cualidades o propiedades que estos deben tener. Para el desarrollo de la propuesta de solución se identificaron requisitos funcionales y no funcionales. Los requisitos funcionales se obtuvieron mediante técnicas que favorecen una comunicación más estrecha entre el cliente y el equipo de desarrollo. A continuación se caracterizan estas técnicas.

2.4. Definición de las técnicas de obtención de requisitos

El proceso de obtención de requisitos, cuya finalidad es llevar a la luz los requisitos, no solo es un proceso técnico, sino un proceso social que envuelve a diferentes personas, lo que conlleva dificultades añadidas a su realización. En la identificación de los requisitos de la propuesta de solución se utilizaron las siguientes técnicas:

- **Entrevista**: permitió analizar en detalle las necesidades del cliente y posibilitó el estudio de las expectativas del mismo respecto a la propuesta de solución, descubriendo sus motivaciones y actitudes.
- **Prototipado**: la elaboración de los prototipos permitió mostrar las funcionalidades de la propuesta de solución para su validación por parte de los interesados. Este tipo de técnica sirve para eliminar dudas sobre lo que desea el cliente y además para desarrollar la interfaz más amigable.
- **Observación:** por medio de esta técnica se obtuvo información de primera mano sobre la forma en que se efectúan las actividades. Este método permite observar la forma en que se llevan a cabo los procesos verificando la coherencia entre la guía documentada y su puesta en práctica. Los observadores experimentados saben con mayor frecuencia qué buscar y cómo evaluar la relevancia de lo que observan.

2.5. Requisitos funcionales

Los requisitos funcionales son declaraciones de servicios que el sistema debe proporcionar, cómo debe reaccionar y comportarse el mismo ante insumos y situaciones particulares. En algunos casos, los requisitos funcionales también, pueden indicar explícitamente lo que el sistema no debe hacer [32]. En la siguiente tabla se muestran los requisitos funcionales, con un total de 46, de los cuales se identificaron 6 de complejidad alta, 9 media y 31 baja, también, se estableció la prioridad para el cliente de cada uno de ellos, clasificando 22 de prioridad alta, 13 media y 11 baja.

Tabla 1: Requisitos funcionales (Elaboración propia)

No	Nombre	Prioridad	Complejidad
RFIDM1	Mostrar forma de evaluación	Baja	Baja
RFIDM2	Crear forma de evaluación	Baja	Baja
RFIDM3	Modificar forma de evaluación	Baja	Baja
RFIDM4	Detalles de la forma de evaluación	Baja	Baja
RFIDM5	Mostrar tipo de evaluación	Baja	Baja
RFIDM6	Crear tipo de evaluación	Baja	Baja
RFIDM7	Modificar tipo de evaluación	Baja	Baja
RFIDM8	Detalles del tipo de evaluación	Baja	Baja
RFIDM9	Mostrar curso	Alta	Baja
RFIDM10	Crear curso	Alta	Baja
RFIDM11	Modificar curso	Alta	Baja
RFIDM12	Detalles del curso	Alta	Baja
RFIDM13	Mostrar versiones de cursos	Alta	Baja
RFIDM14	Crear versión del curso	Alta	Baja
RFIDM15	Modificar versión del curso	Alta	Baja
RFIDM16	Detalles de la versión del curso	Alta	Baja
RFIDM17	Mostrar tipo de curso	Media	Baja
RFIDM18	Crear tipo de curso	Media	Baja
RFIDM19	Modificar tipo de curso	Media	Baja
RFIDM20	Detalles del tipo de curso	Media	Baja
RFIDM21	Mostrar formas organizativas	Media	Baja
RFIDM22	Crear forma organizativa	Media	Baja
RFIDM23	Modificar forma organizativa	Media	Baja
RFIDM24	Detalles de la forma organizativa	Media	Baja
RFIDM25	Mostrar tipologías	Media	Baja
RFIDM26	Crear tipologías	Media	Baja
RFIDM27	Modificar tipologías	Media	Baja
RFIDM28	Detalles de tipología	Media	Baja
RFIDM29	Mostrar grupos docentes	Alta	Media
RFIDM30	Crear grupo docente	Alta	Media
RFIDM31	Modificar grupo docente	Alta	Media
RFIDM32	Detalles de grupo docente	Alta	Media
RFIDM33	Asociar profesores	Alta	Media
RFIDM34	Registro de evaluaciones	Alta	Alta
RFIDM35	Registro de asistencia	Alta	Alta
RFIDM36	Mostrar documentos	Alta	Media
RFIDM37	Cerrar grupo docente	Alta	Media
RFIDM38	Ver detalles de cerrar grupo docente	Media	Baja
RFIDM39	Mostrar solicitudes de curso	Alta	Media
RFIDM40	Mostrar convocatorias de curso	Alta	Alta
RFIDM41	Crear convocatoria de curso	Alta	Alta

RFIDM42	Modificar convocatoria de curso	Alta	Alta
RFIDM43	Aprobar solicitudes de convocatoria de curso	Alta	Alta
RFIDM44	Detalles de la convocatoria del curso	Baja	Media
RFIDM45	Exportar registro de evaluaciones	Baja	Baja
RFIDM46	Exportar registro de asistencia	Baja	Baja

2.6. Requisitos no funcionales

Los requisitos no funcionales son restricciones en los servicios o funciones ofrecidos por el sistema. Incluyen restricciones de tiempo, en el proceso de desarrollo e impuestas por las normas. A menudo estos requisitos se aplican al sistema en su totalidad, en lugar de a las funciones o servicios individuales [32]. El módulo propuesto cuenta con los siguientes requisitos no funcionales:

Usabilidad:

- Solo se mostrarán a los usuarios aquellas acciones o informaciones a las que por su responsabilidad o rol dentro del negocio necesitan acceder mostrando en la vista mediante íconos y menús el acceso a la misma.
- La información que se registra estará protegida de modificaciones no deseadas de acuerdo a la estrategia de seguridad definida.
- Los términos visibles para el usuario en la aplicación serán los utilizados por los usuarios en la rama abordada con vista a una mayor comprensión por parte de los mismos de la herramienta de trabajo.
- Las vistas del sistema deben indicar en cada momento la acción que se está realizando así como los íconos deben estar representados por una imagen acorde a la acción que se realiza mediante el mismo.

Seguridad:

- El sistema puede permanecer inactivo durante 10 minutos. Al cumplirse este término se cerrará la sesión teniendo que autenticarse el usuario nuevamente.
- El tratamiento de las excepciones permitirá un seguimiento hasta guardar información acerca del lugar dónde se produjo el error y de los parámetros de configuración del sistema que lo provocaron. Cuando ocurre una excepción el sistema mostrará un mensaje explicativo del error ocurrido y permanecerá en el mismo estado sin realizar ninguna otra operación.
- El sistema de autenticación recogerá los datos necesarios de los usuarios y tendrá control de identidades.

- El sistema debe proveer algún mecanismo seguro de encriptación y trasferencia de datos.
- El sistema de autenticación permite auditoría de trazas de las acciones de los usuarios para lo que se proveerá algún mecanismo para la configuración dinámica y centralizada de la gestión de historiales.
- Se debe mantener una seguridad a nivel de usuarios y contraseñas encriptadas para el acceso a la base de datos.
- El servidor de aplicaciones y de base de datos deberá mantener una seguridad mediante un cortafuegos para proteger el código y la información.

Eficiencia:

- El sistema soportará la conexión simultánea de la mayor cantidad de usuarios posibles, con un promedio de 1000 y un máximo de 3000.
- El sistema deberá tener por cada transacción un tiempo de respuesta promedio de 1,5 segundos y un máximo de 3 segundos.

Rendimiento:

- Para el despliegue del sistema se contará en el servidor de base de datos con Postgres 8.4 o superior bajo el sistema operativo CentOS 7.0 o superior.
- Para el despliegue del sistema estará instalado en el servidor de aplicaciones web con el sistema operativo CentOS 6.2 donde se encuentre instalado PHP v7.2 con las librerías php7-ldap, php7-gd, php7-mcrypt, php7-pgsql, php7-xsl, php7-openssl, Nginx y JDK 6 o superior.

Hardware:

- El servidor de aplicación y el de base de datos tendrán los siguientes componentes de hardware: Microprocesador de 8 núcleos, 4GB de memoria RAM, 250 GB disco duro y una fuente de 800 W como mínimo.
- Para la ejecución del sistema se requiere que la PC cliente tenga los siguientes componentes de hardware: Pentium 4 o superior, 512 MB RAM y 1 GB disco duro disponible como mínimo.
- Para el uso del sistema se requiere una PC cliente con los siguientes sistemas operativos: Windows o GNU/Linux.

Soporte:

- El sistema se regirá por un estándar de código debidamente documentado en el expediente de proyecto.

- Todos los productos de trabajo generados en el desarrollo del software se regirán por unas pautas de configuración debidamente documentadas en el expediente de proyecto.

Requisitos de interfaz:

- La comunicación entre el cliente y el servidor de aplicaciones se establece a través del protocolo HTTPS[13].
- El sistema se integrará con el directorio activo de la institución cliente mediante el protocolo LDAP[14].
- Existirá un componente interno para la validación correcta de las entrada/salida de datos de cada una de las interfaces de la aplicación.
- El sistema deberá permitir la integración hacia y desde sistemas externos existentes en el ambiente de despliegue mediante servicios web por el protocolo SOAP[15].
- El sistema deberá implementar una solución arquitectónica para permitir la integración segura entre componentes internos.
- La interfaz de usuario se guiará por la vista de arquitectura de presentación.
- La comunicación entre el servidor de aplicaciones y la base de datos se lleva a través del protocolo TCP/IP.
- Se tendrán centralizados los archivos de configuración. Para ellos se utilizarán archivos XML que permitirán el fácil mantenimiento del sistema si cambia alguna de las configuraciones que posee.

Requisitos de licencia:

- El sistema se caracterizará por un licenciamento cerrado. El producto está basado en código abierto pero no está disponible bajo licenciamiento de código abierto.

Requisitos legales, de derecho de autor y otros:

- El sistema será sometido a un análisis legal por parte de los abogados y personal autorizado con vistas a declarar su autenticidad y evitar restricciones legales para su uso y comercialización documentando el resultado en el expediente del proyecto.

[13]HTTPS (del inglés Hypertext Transfer Protocol Secure) Protocolo de transferencia de hipertexto seguro.

[14]LDAP. Protocolo Ligero de Acceso a Directorios por sus siglas en inglés, es un protocolo a nivel de aplicación que permite el acceso a un servicio de directorio ordenado y distribuido para buscar diversa información en un entorno de red.

[15]SOAP es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML.

Estándares aplicables:

- Para exportar datos del sistema se usa el Formato de Documento Portátil (.pdf): norma ISO 32000
- Norma SO/IEC 8859 para codificación de caracteres.

2.7. Descripción de Historias de usuarios.

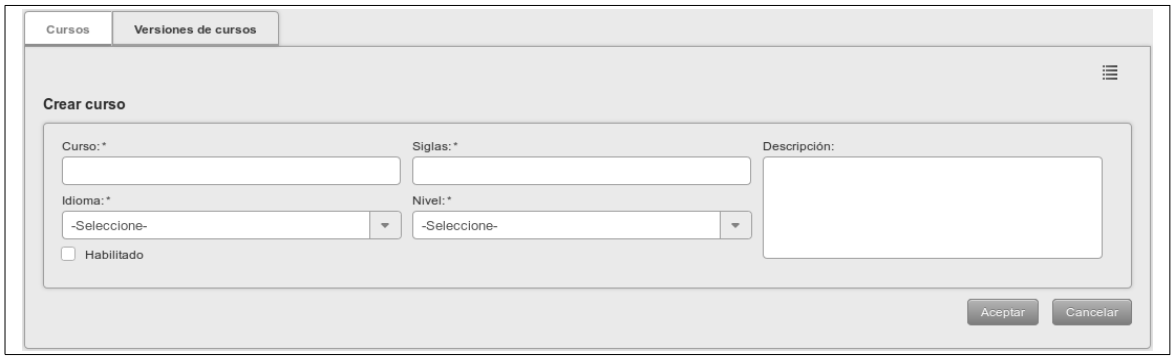
Una historia de usuario (HU) describe una funcionalidad que realizará el sistema y que aportará valor al cliente. Deben ser escritas en un lenguaje no técnico de manera que puedan ser fácilmente comprendidas. Seguidamente se exponen dos historias de usuario como ejemplos. Para ver las historias de usuario en su totalidad, consultar el expediente del proyecto Idiomas.

Tabla 2: HU_Mostrar_cursos (Elaboración propia)

Número: HU_8	**Requisito:** Mostrar cursos
Programador: Pedro Iván Fernández González	**Iteración Asignada:** 1
Prioridad: Alta	
Descripción: El requisito permite mostrar un listado con los cursos existentes. El administrador selecciona el módulo **Superación** del Sistema de Gestión de Idioma, luego en el menú lateral la agrupación funcional **Diseño**, funcionalidad **Cursos**, la pestaña **Cursos**. Se muestra un listado de los cursos con el dato: **Nombre**. Se mostrarán las opciones en el área de iconos internos: Ver detalles y Modificar. Además, la opción: Crear en el área de iconos flotantes.	
Observaciones: El usuario debe estar autenticado en el sistema con los permisos necesarios para acceder a la funcionalidad.	
Prototipo elemental de interfaz gráfica de usuario:	

Tabla 3: HU_Crear_curso (Elaboración propia)

Número: HU_9	**Requisito**: Crear curso
Programador: Pedro Iván Fernández González	**Iteración Asignada:** 1
Prioridad: Baja	
Descripción: El requisito permite crear un curso. El administrador selecciona del módulo **Superación** del Sistema de Gestión de Idioma, luego en el menú lateral la agrupación funcional **Diseño**, funcionalidad **Cursos**, la pestaña **Cursos**. En el área de iconos flotantes selecciona la opción **Crear**. Se muestran los datos a llenar: Nombre, Descripción, Idioma, Nivel, Siglas y Habilitado (Activo o no). Además, la opción: **Listar** en el área de iconos flotantes.	
Observaciones: El usuario debe estar autenticado en el sistema con los permisos necesarios para acceder a la funcionalidad. No debe existir un curso con el mismo nombre ni siglas. Los campos Nombre, Siglas, Idioma y Nivel son obligatorios.	
Prototipo elemental de interfaz gráfica de usuario:	



2.8. Descripción de la propuesta de solución

La solución propuesta tiene como principal objetivo mejorar la eficiencia de la gestión de cursos de idiomas impartidos por el Centro de Idiomas en la Universidad de las Ciencias Informáticas asistiendo a sus trabajadores.

El módulo Superación permite de manera rápida e intuitiva diseñar cursos, seleccionando el idioma, el nivel, entre otras anotaciones como una breve descripción y siglas. Los cursos pueden ser reutilizados mediante el versionado de los mismos. Estas versiones especifican a quienes está destinado el curso, las fecha inicio y fin del mismo, las formas organizativas que serán empleadas así como la cantidad y tipo de evaluaciones. Con el objetivo de informar la disponibilidad de los cursos, se pueden confeccionar convocatorias que serán visualizadas exclusivamente por las personas que pueden realizar una solicitud dentro de las fechas límites de la convocatoria. Por consiguiente, el módulo posibilita la visualización de las solicitudes efectuadas a dicho curso y la aprobación o no de las mismas. Para administrar y agrupar mejor la información de los usuarios, se facilita la creación y organización de grupos docentes ofreciendo el uso de filtros para reducir el espacio de búsqueda por año académico, grupo administrativo, área, entre otros. De manera análoga, se viabiliza la asociación de profesores a estos grupos y la especificación de uno de ellos como el principal. Los profesores pueden registrar las evaluaciones y la asistencia de sus estudiantes, así como exportar el registro de las mismas durante y/o la finalización del curso.

En la siguiente figura se presenta el mapa de navegación del módulo Superación:

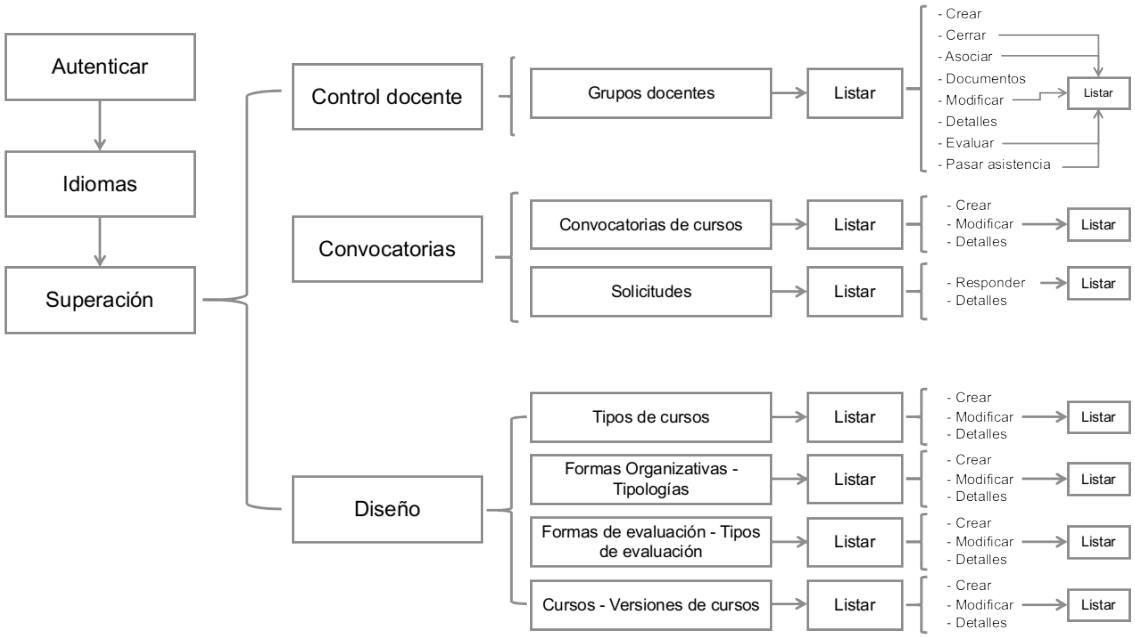


Figura 2: Mapa de navegación para hacer uso del módulo Superación.

2.8.1. Descripción de la arquitectura y diseño

La propuesta de solución está concebida como un módulo dentro del Sistema de Gestión Académica, por esta razón se ajustará a la arquitectura definida para dicho sistema, la cual propone la arquitectura Cliente-Servidor, haciendo uso del patrón Modelo-Vista-Controlador (MVC), ya que el marco de trabajo empleado está basado en dicho patrón, por su flexibilidad y ventajas.

2.8.2. Arquitectura Cliente-Servidor

El estándar IEEE 1471[16] define la arquitectura de software como “la organización fundamental de un sistema incorporados en sus componentes, sus relaciones entre sí y con el medio ambiente, y los principios que guían su diseño y evolución”. La arquitectura del software es importante porque afecta el rendimiento, la solidez, las capacidades de distribución y de mantenimiento de un sistema.

En una arquitectura Cliente-Servidor, la funcionalidad del sistema se organiza en servicios, con cada servicio entregado desde un servidor separado. Un cliente realiza una solicitud a un servidor y espera hasta que recibe una respuesta. Los clientes acceden a los servicios proporcionados por un servidor a

[16]IEEE Standard 1471: Práctica recomendada para la descripción arquitectónica de sistemas de software intensivo. Desarrollado por la IEEE Computer Society.

través de llamadas a procedimientos remotos usando un protocolo de solicitud-respuesta como el protocolo HTTP usado en la Internet.

- **Clientes**: los clientes para conectarse a los servidores disponibles y utilizar los servicios que estos brindan deben conocer el nombre del servidor. Los clientes son usuarios de estos servicios y acceden a los servidores para hacer uso de ellos.
- **Servidores**: los servicios y los servidores se pueden cambiar sin afectar a otras partes del sistema. Estos, no necesitan conocer la identidad de los clientes o cuántos clientes acceden a sus servicios.

En la siguiente figura se muestra la representación de la arquitectura:

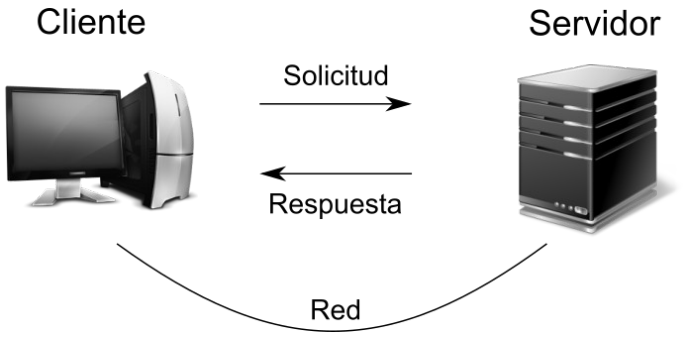


Figura 3: Arquitectura Cliente - Servidor

2.8.3. Patrón de arquitectura

Un patrón arquitectónico es una descripción estilizada y abstracta de buenas prácticas, que se ha probado en diferentes sistemas y entornos. Por lo tanto, un patrón arquitectónico debe describir una organización de sistema que haya tenido éxito en sistemas anteriores.

El patrón Modelo-Vista-Controlador es la base de la gestión de la interacción en muchos sistemas basados en web. Este patrón separa la presentación y la interacción de los datos del sistema. El patrón está estructurado en tres componentes lógicos que interactúan entre sí:

- **Modelo**: gestiona los datos del sistema y las operaciones asociadas a esos datos.
- **Vista**: define y administra cómo se presentan los datos al usuario.
- **Controlador**: gestiona la interacción del usuario (por ejemplo, pulsaciones de teclas, clics del mouse, etc.) y pasa estas interacciones a la vista y al modelo.

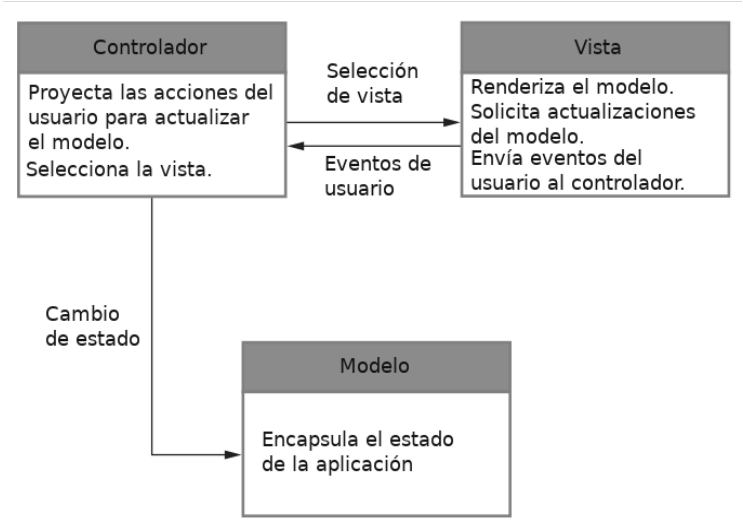


Figura 4: Patrón Modelo-Vista-Controlador

2.8.4. Patrones de diseño

Los patrones de diseño se derivaron de las ideas al sugerir que había ciertos patrones comunes de diseño de edificios que eran inherentemente agradables y efectivos. El patrón no es una especificación detallada, es una descripción del problema y la esencia de su solución, por lo que la solución puede reutilizarse en diferentes configuraciones.

Los patrones han tenido un gran impacto en el diseño de software orientado a objetos. Además, de ser probadas soluciones a problemas comunes, se han convertido en un vocabulario para hablar sobre un diseño. Un patrón de diseño bien conocido es el "Gang of Four" que originalmente describieron los autores en su libro de patrones.

2.8.5. Patrones GRASP

Los Patrones Generales de Asignación de Responsabilidades de software (GRASP, por sus siglas en inglés) describen los principios fundamentales de la asignación de responsabilidades a objetos, de forma tal que se pueda diseñar software orientado a objetos. Algunos ejemplos de estos patrones son: experto, creador, bajo acoplamiento, alta cohesión y controlador. A continuación se muestran los patrones utilizados en el desarrollo de la propuesta de solución:

- **Creador**: este patrón guía la asignación de responsabilidades relacionadas con la creación de objetos. Se evidencia en la clase loader que es el objeto load de las clases controladoras. Esta clase se encarga de cargar los elementos del marco de trabajo como librerías y modelos que se utilizaran para completar la petición. A continuación se muestra un ejemplo de su uso en la clase controladora **curso** donde se utiliza en el constructor para cargar la librería **curso_lib**.

```
public function __construct() {
    parent::__construct();
    $this->load->library("curso_lib");
}
```

Figura 5: Patrón Creador.

- **Experto:** se encarga de asignar una responsabilidad al experto en información, es decir, aquella clase que cuenta con la información necesaria para cumplir la responsabilidad. Este patrón se evidencia en las clases librerías del sistema, las cuales cuentan con la información necesaria para cumplir la responsabilidad sobre los elementos del negocio. En la siguiente figura se muestra un ejemplo de la utilización del patrón en la librería **curso_lib** donde se maneja la información referente a la tabla **tb_ncurso**.

```
class curso_lib {

    public function __construct() {
        $this->_ci = & get_instance();
        $this->_ci->load->model('tb_ncurso_mdl');
    }

    public function obtenerCursos($inicio = NULL, $limite = NULL, $elementoOrdenar = NULL,

    public function obtenerCantidadCursos($inicio = NULL, $limite = NULL, $elementoOrdenar

    public function registrarCurso($datos) {...10 lines }

    public function modificarCurso($datos) {...11 lines }
```

Figura 6: Patrón Experto.

- **Alta cohesión y Bajo acoplamiento:** la propia implementación del marco de trabajo CodeIgniter contiene estos patrones nivelados, permitiendo el uso de los componentes de forma individual en

la implementación de la solución de software que se desarrolla, evidenciando de esta forma el bajo acoplamiento, así como la dependencia entre los componentes o la alta cohesión.

- **Controlador:** asigna la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase. El patrón controlador se refleja en las clases controladoras pertenecientes al sistema, que son las clases que se encargan de obtener datos y enviarlos a las librerías y las vistas.

```
class curso extends MY_Controller {

    public function __construct() {
        parent::__construct();
        $this->load->library("curso_lib");
    }

    function index() {...3 lines }

    function listar() {...3 lines }

    function listar_cursos() {...3 lines }

    public function obtenerCursos() {...6 lines }

    public function registrarCurso() {...9 lines }
```

Figura 7: Patrón Controlador.

2.8.6. Patrones GoF

Los patrones GoF (Gang of Four por sus siglas en inglés) se clasifican en tres categorías basadas en su propósito: creacionales, estructurales y de comportamiento. Los patrones creacionales abstraen el proceso de creación de instancias y ocultan los detalles de cómo los objetos son creados o inicializados, mientras que los patrones estructurales se ocupan de cómo las clases y objetos se combinan para formar grandes estructuras y proporcionar nuevas funcionalidades. Por otra parte, los patrones de comportamiento ayudan a definir la comunicación e interacción de los objetos, así como reducir el acoplamiento.

Los patrones GoF que se utilizaron en el desarrollo del componente son:

- **Instancia única (Singleton)**: patrón de creación que garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. En la figura 8 se muestra la clase **version_curso_lib.php** la cual tiene en su constructor la instancia de la modelo

version_curso_mdl.php, de esta manera no hay necesidad de instanciar esta modelo en cualquier otra parte de la clase.

```php
<?php

class Version_curso_lib {

    public function __construct() {
        $this->_ci = & get_instance();
        $this->_ci->load->model('tb_nversion_curso_mdl');
        $this->_ci->load->model('tb_rversion_curso_forma_organizativa_mdl');
        $this->_ci->load->model('tb_rversion_curso_evaluacion_mdl');
    }
```

Figura 8: Patrón Instancia única

- **Mediador (Mediator)**: patrón de comportamiento que define un objeto que coordine la comunicación entre objetos de distintas clases, pero que funcionan como un conjunto. Se refleja en las librerías, que funcionan como mediadoras entre las clases controladoras y las modelos o acceso a datos (ver figura 9).

```php
<?php

class Version_curso_lib {

    public function __construct() {
        $this->_ci = & get_instance();
        $this->_ci->load->model('tb_nversion_curso_mdl');
        $this->_ci->load->model('tb_rversion_curso_forma_organizativa_mdl');
        $this->_ci->load->model('tb_rversion_curso_evaluacion_mdl');
    }

    public function obtenerVersionesCurso($inicio = NULL, $limite = NULL, $eleme

    public function obtenerCursoDadoId($idVersionCurso) {...3 lines }

    public function obtenerCantidadVersionesCurso($inicio = NULL, $limite = NULL

    public function modificarVersionCurso($datos) {...32 lines }
```

Figura 9: Patrón Mediador

- **Observador (Observer)**: define una dependencia de uno-a-muchos entre objetos, de forma que cuando un objeto cambie de estado se notifique y actualicen automáticamente todos los objetos que dependen de él. Se refleja en la clase **loader** que es el objeto load de las clases controladoras

(ver figura 10), cuya función es cargar los elementos del marco de trabajo dígase librerías, modelos y se encarga de actualizar la controladora instanciada.

```php
<?php
/**
 * @author Pedro Iván Fernández González <pifernandez@estudiantes.uci.cu>
 * @property curso_lib $curso_lib
 */

class Curso extends MY_Controller {

    public function __construct() {
        parent::__construct();
        $this->load->library("curso_lib");
    }

    function index() {...3 lines }

    function listar_cursos() {...3 lines }

    public function obtenerCursos() {
        $datos = $this->input->all_post(TRUE);
        if (isset($datos['cadena'])) {
            $criterio = $datos['cadena'];
        }
        $this->load->helper('grid');
        echo(grid_json($this->curso_lib, 'obtenerCantidadCursos', 'obtenerCursos',
    }
```

Figura 10: Patrón Observador

2.8.7. Modelos de datos

Un modelo de datos es una colección de conceptos y reglas que permite comunicar la estructura y comportamiento del sistema, visualizar y controlar su arquitectura a través de entidades, atributos y las relaciones entre ellos. El modelado de datos responde a una serie de preguntas específicas importantes para cualquier aplicación de procesamiento de datos. ¿Cuáles son los objetos de datos primarios que va a procesar el sistema? ¿Cuál es la composición de cada objeto de datos y qué atributos describe el objeto? ¿Dónde residen actualmente los objetos? ¿Cuál es la relación entre los objetos y los procesos que los transforman? Para responder estas preguntas, los métodos de modelado de datos hacen uso del diagrama de entidad-relación (DER). El DER, permite identificar los objetos de datos y sus relaciones mediante una notación gráfica. A continuación se presenta el modelo de datos físico principal de los utilizados en la implementación del módulo:

Capítulo 2: Descripción de la propuesta de solución

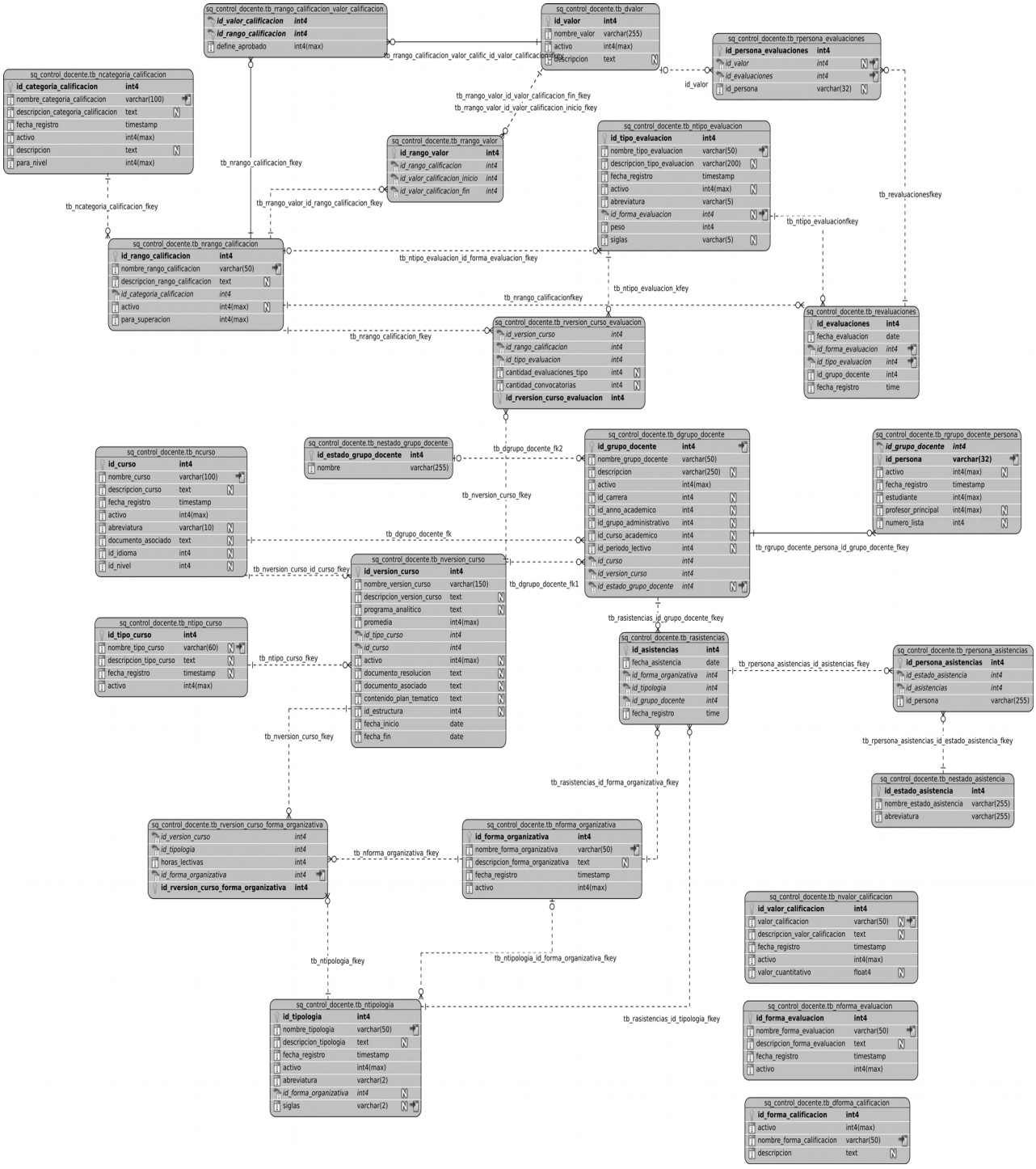


Figura 11:Modelo de datos físicos del esquema sq_control_docente.

2.8.8. Diagrama de despliegue

El diagrama de despliegue muestra cómo los componentes de software se implementan físicamente en los procesadores; es decir, el hardware, el software y el middleware utilizados para conectar los diferentes componentes en el sistema. Constituye una forma de definir y documentar el entorno objetivo. En la siguiente figura puede visualizarse el diagrama de despliegue definido para la solución propuesta**:**

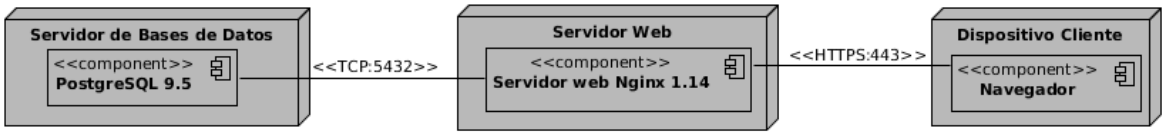


Figura 12: Diagrama de despliegue

Servidor de Bases de Datos: dispositivo físico que ejecuta al sistema gestor de base de datos PostgreSQL que almacena la información persistente del Sistema de Gestión Académica.

Servidor Web: hardware donde se ejecuta el servidor encargado de dar soporte a la aplicación web.

Dispositivo cliente: representa cualquier dispositivo que permita al usuario acceder a la aplicación web a través de un navegador.

HTTPS: protocolo empleado para establecer a través del puerto 443 la conexión segura entre el dispositivo de acceso cliente y el servidor web.

TCP: protocolo empleado para establecer la conexión entre el servidor web y el sistema gestor de base de datos usando el puerto 5432.

2.9. Conclusiones parciales

La confección del capítulo anterior propone las siguientes conclusiones:

- A partir de un modelo conceptual se reflejaron las características del negocio, los principales conceptos y las relaciones entre ellos. Se definieron las reglas relacionadas a la gestión de cursos por parte del Centro de Idiomas para su correcto funcionamiento.
- La definición de los requisitos funcionales y no funcionales de la solución propuesta permitió una mejor comprensión de los resultados que se pretenden obtener de una manera más coherente y sirvieron de guía para la implementación del módulo.

- Con la caracterización de la arquitectura Cliente-Servidor, el patrón MVC, los patrones de diseño, así como la obtención del modelo de datos y la distribución física de la propuesta de solución, se logró comprender toda la arquitectura para el desarrollo de la misma.

Capítulo 3: Implementación y evaluación de la solución

3. Introducción

Entre las fases que conforman el proceso de desarrollo de software se encuentran implementación y pruebas. En el presente capítulo se definen los estándares de codificación que debe cumplir el equipo de desarrollo con el objetivo de facilitar la legibilidad y el mantenimiento del código en lo adelante, así como los métodos y técnicas para la realización de las pruebas, con el propósito de validar la solución.

3.1. Estándares de codificación

Para el desarrollo del módulo se aplican los estándares de codificación establecidos por el Departamento de Informatización, con el propósito de estandarizar las nomenclaturas en la implementación de la aplicación web y obtener un producto estable.

3.1.1. Indentación, llaves de apertura y cierre y tamaño de las líneas

Se debe usar una indentación sin tabulaciones, con un equivalente a 4 espacios. El uso de las llaves “{}” será en una nueva línea. La longitud de las líneas de código es aproximadamente de 75 a 80 caracteres para mantener la legibilidad del código. En la figura 13 se muestra un ejemplo de cómo debe quedar el formato del código de acuerdo con el estándar utilizado.

```
function listar_cursos()
{
    echo $this->template->render('cursos/curso/cursos_view');
}
```

Figura 13: Indentación, llaves de apertura y cierre y tamaño de las líneas.

3.1.2. Convención de nomenclatura

Variables: se rigen por la nomenclatura camelCase[17]. Siempre comienzan con minúscula y en caso de nombres compuestos la primera letra de cada palabra comienza con mayúscula. En la figura 14 se muestra un ejemplo del empleo de la nomenclatura en las variables:

[17]camelCase: Es un estilo de escritura que se aplica a frases o palabras compuestas. El nombre se debe a que las mayúsculas a lo largo de una palabra en camelCase se asemejan a las jorobas de un camello. El término case se traduce como "caja tipográfica", que a su vez implica si una letra es mayúscula o minúscula.

```
$idTipo = $datos['id'];
$tipologia = $this->tipologia_lib->obtenerTipologiaDadoId($idTipo);
```

Figura 14: Variables.

Clases: siempre comienzan con mayúscula, en caso de nombre compuesto las palabras se separan con guión bajo "_" y el resto en minúscula. Se muestra un ejemplo de la nomenclatura de una clase simple y una compuesta:

```
class Grupo_docente extends MY_Controller
    class Tipologia extends MY_Controller {
```

Figura 15: Clases

Funciones: se rigen por la nomenclatura camelCase. Siempre comienzan con minúscula y en caso de nombres compuestos la primera letra de cada palabra comienza con mayúscula. Los parámetros son separados por espacio luego de la coma que los separa. Se muestra un ejemplo del uso del estándar en la función obtenerCursoDadoId:

```
public function obtenerCursoDadoId($id)
{...4 lines }
```

Figura 16: Funciones

Ficheros: siempre en minúscula y en caso de nombres compuestos se usa el guión bajo, seguido del sufijo definido para cada tipo de fichero.

- **Vistas**: intuitivo y relacionado con el formulario y/o vista que representa. Sufijo "_view".

  Ejemplo: detalles_atributo_view.php

- **Modelos**: mismo nombre de la tabla de la base de datos para la cual se crea. Sufijo "_mdl".

  Ejemplo: tb_ddato_almacenado_mdl.php

- **Librerías**: mismo nombre de la clase que representa. Sufijo "_lib".

  Ejemplo: entidad_atributo_valor_lib

- **Controladoras**: mismo nombre de la clase que representa.

  Ejemplo: Categoria_entidad.

3.1.3. Estructuras de control

Las estructuras de control incluyen if, for, foreach, while, switch. Entre las estructuras de control y los paréntesis debe de existir un espacio. Se recomienda utilizar siempre llaves de apertura y cierre, incluso en situaciones en las que técnicamente son opcionales. Con esto se aumenta la legibilidad del código y se disminuye la probabilidad de errores lógicos.

3.1.4. Documentación

Todos los archivos deben de tener la documentación asociada al mismo. Se debe de cumplir con el siguiente bloque al principio de cada clase, como se muestra en la clase **Curso_lib:**

```
<?php
/**
*@category Libreria
*@package curso_lib.php
*@author Pedro Ivan Fernandez Gonzalez <pifernandez@estudiantes.uci.cu>
*/
class Curso_lib {
```

Figura 17: Documentación

3.1.5. Buenas prácticas

Valores booleanos y nulos: los valores booleanos y nulos siempre se escriben con mayúscula, para facilitar la legibilidad del código. Se debe usar un salto de línea antes de las estructuras de control y definición de las funciones.

Etiquetas de apertura y cierre de PHP: el código PHP se debe escribir siempre utilizando las etiquetas **<?php** y **?>** y en ningún caso la versión corta **<?** y **?>**.

Usar nombres que revelen intenciones: el nombre de una variable, función o clase debe responder a una serie de cuestiones básicas. Debe indicar por qué existe, qué hace y cómo se usa. Si un nombre requiere un comentario significa que no revela su contenido.

Capítulo 3: Implementación y evaluación de la solución

3.2. Estrategia de pruebas

La calidad del software incide en gran medida en el éxito de este y en la satisfacción de sus usuarios, siendo las pruebas un elemento decisivo en la obtención de la misma. Las pruebas son un conjunto de actividades que pueden ser planeadas con anticipación y ser conducidas sistemáticamente. Por esta razón una guía para las pruebas de software, conjunto de pasos en los que podemos introducir técnicas de diseño de casos de pruebas y métodos de pruebas, debe ser definida para el desarrollo de software [33]. A continuación se muestra la estrategia de pruebas aplicada para el desarrollo del módulo:

Tabla 4: Estrategia de pruebas (Elaboración propia)

Pruebas	Métodos	Técnicas
Pruebas de unidad	Caja blanca	Prueba de camino básico
Pruebas de integración	Caja negra	Incremental
Pruebas de sistema: pruebas de rendimiento.	Caja negra	Manual
Pruebas de validación	Caja negra	Partición equivalente
Pruebas de aceptación	Caja negra	Alfa y Beta

3.2.1. Pruebas unitarias

Las pruebas unitarias verifican la funcionalidad de un pequeño subconjunto del sistema, como un objeto o un método. Estas se conocen como pruebas de programador, pruebas orientadas al desarrollador o pruebas tecnológicas. Permiten a los programadores medir lo que Kent Beck[18] ha llamado la calidad interna de su código. Las pruebas unitarias enfocan el esfuerzo de verificación en la unidad más pequeña de software que diseñe el componente o módulo de software [33].

La clase de prueba unitaria de CodeIgniter (unit_test) es bastante simple, consta de una función de evaluación y dos funciones de resultados. No se pretende que sea un conjunto completo de pruebas, sino un simple mecanismo para evaluar su código para determinar si está produciendo el tipo de datos y el resultado correctos. Como la mayoría de las otras clases en CodeIgniter, la clase de prueba unitaria se inicializa en su controlador usando la función de biblioteca load**:** $this->load->library('unit_test'). La ejecución de una prueba implica proporcionar una prueba y un resultado esperado de la siguiente manera:

[18]Kent Beck es uno de los creadores de las metodologías de desarrollo de software de programación extrema y el desarrollo guiado por pruebas, también llamados metodología ágil. Beck fue uno de los 17 firmantes originales del Manifiesto Ágil en 2001.

$this->unit->run('resultado_obtenido', 'resultado_esperado', 'nombre_prueba', 'notas'). Donde el 'resultado_obtenido' es el resultado del código que desea probar, el 'resultado_esperado' es el tipo de datos que espera, el 'nombre_prueba' es un nombre opcional que puede dar a su prueba y las 'notas' son anotaciones opcionales.

Tabla 5: Prueba estructural de caja blanca para la funcionalidad **chequearNombreCurso** (Elaboración propia)

Prueba estructural de caja blanca	Código de caso de prueba
	registrarCurso()
Probador	Pedro Iván Fernández González
Tipo de dato esperado	Booleano
Código al que se le aplica	

```
public function chequearNombreCurso($datos) {
    $nombreCurso = $datos['nombre'];
    $bandera = false;
    $cursos = $this->_ci->tb_ncurso_mdl->obtenerCursos(NULL, NULL, NULL, '', '');

    foreach ($cursos as $curso) {
        if ($curso->nombre_curso == $nombreCurso) {
            $bandera = TRUE;
            break;
        }
    }
    return $bandera;
}
```

Capítulo 3: Implementación y evaluación de la solución

Complejidad ciclomática:

Sean A: número de aristas, N: número de nodos, NP: número de nodos con predicado simple[19] y R: número de regiones rodeadas completamente por arcos del grafo.

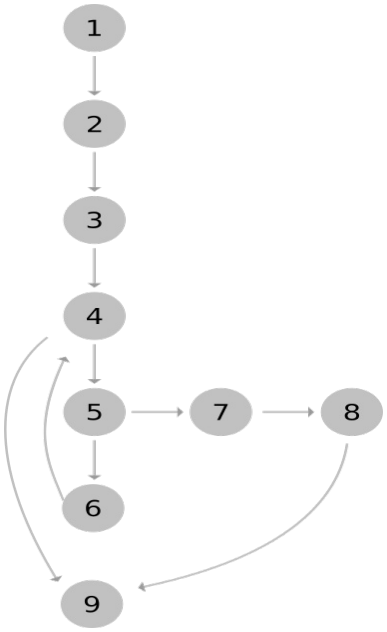
A) v(G) = A – N + 2 = 10 – 9 +2 = 3

B) v(G) = NP + 1 = 2 + 1 = 3

C) v(G) = R + 1 = 3

Caminos independientes:

1. 1→ 2→ 3→ 4→ 9
2. 1→ 2→ 3→ 4→5→ 7→ 8→ 9
3. 1→ 2→ 3→ 4→5→ 6→ 4→ 9
4. 1→ 2→ 3→ 4→5→ 6→ 4→ 5→ 7→ 8→ 9



Caso de prueba para los caminos básicos

Tipo de dato esperado:	Booleano

Función de evaluación:

```
function index() {
    $datos = array();
    $datos['nombre'] = "Curso Inglés B1";
    $resultadoEsperado = TRUE;
    $nombrePrueba = "Chequear nombre de curso";
    echo $this->unit->run($this->curso_lib->chequearNombreCurso($datos), $resultadoEsperado, $nombrePrueba);
}
```

Resultados de la prueba:

Test Name	Chequear nombre de curso
Test Datatype	Boolean
Expected Datatype	Boolean
Result	Passed
File Name	/var/www/akademos/base/application/idioma/superacion/controllers/curso.php
Line Number	20

Evaluación del caso de prueba:	Satisfactoria

[19]Nodos de los que parten dos caminos.

Se llevaron a cabo tres iteraciones para detectar no conformidades, las cuales arrojaron los siguientes resultados como se muestra en la tabla 6. Todas las no conformidades encontradas fueron resueltas.

Tabla 6: Iteración de las pruebas de caja blanca (Elaboración propia)

Iteración	Cantidad de no conformidades	Asociadas a
Primera	9	Errores de validación, entradas y salidas incorrectas.
Segunda	3	Errores de validación.
Tercera		-

3.2.2. Pruebas de integración

Las pruebas de integración son una técnica sistemática para construir la arquitectura del software y realizar pruebas para descubrir errores asociados con la interconexión . Los programadores escriben pruebas de integración de código para asegurarse de que las pequeñas unidades de código trabajen juntas según lo previsto. Esta práctica ha sido adoptada por muchos equipos, porque es una forma inteligente de analizar el diseño de software y prevenir defectos.

Existen dos tipos de integración, la incremental y la no incremental. La integración no incremental intenta construir el programa usando un enfoque donde todos los componentes se combinan de antemano. La corrección es difícil porque el aislamiento de las causas se complica por la vasta extensión de todo el programa. Una vez que se corrigen errores, aparecen otros nuevos y el proceso continúa en un bucle aparentemente interminable. Sin embargo, la integración incremental construye y prueba el programa en pequeños incrementos, donde los errores son más fáciles de aislar y corregir. Es más probable que las interfaces se prueben completamente; y se puede aplicar un enfoque de prueba sistemático [33].

Para realizar las pruebas de integración de la solución propuesta se decidió emplear la integración incremental por las facilidades que brinda al desarrollador realizando las pruebas de manera menos compleja y obteniendo mejores resultados. A continuación se presenta las diferentes estrategias empleadas.

Estrategias de integración

Top-down (descendente): La prueba de integración descendente es un enfoque incremental para la construcción de la arquitectura del software. Los módulos se integran moviéndose hacia abajo a través de

la jerarquía de control, comenzando con el módulo de control principal (programa principal). Los módulos subordinados al módulo de control principal se incorporan a la estructura ya sea en profundidad o primero en amplitud [33]. El proceso de integración se realiza en una serie de cinco pasos:

1. El módulo de control principal se utiliza como controlador de prueba y los apéndices se sustituyen por todos los componentes directamente subordinados al módulo de control principal.
2. Dependiendo del enfoque de integración seleccionado (es decir, primero en profundidad o primero en amplitud), los apéndices subordinados se reemplazan uno a la vez con componentes reales.
3. Las pruebas se realizan a medida que se integra cada componente.
4. Al completar cada conjunto de pruebas, se reemplaza otro apéndice con el componente real.
5. Se pueden realizar pruebas de regresión para garantizar que no se hayan introducido nuevos errores.
6. El proceso continúa desde el paso 2 hasta que se construye toda la estructura del programa.

Bottom-up (ascendente): Las pruebas de integración ascendente, como su nombre lo indica, comienzan la construcción y las pruebas con módulos atómicos (es decir, componentes en los niveles más bajos en la estructura del programa). Debido a que los componentes se integran de abajo hacia arriba, la funcionalidad proporcionada por los componentes subordinados a un nivel dado siempre está disponible y se elimina la necesidad de apéndices [33]. Una estrategia de integración de abajo hacia arriba se puede implementar con los siguientes pasos:

1. Los componentes de bajo nivel se combinan en grupos (a veces llamados construcciones) que realizan una sub-función de software específica.
2. Un controlador (un programa de control para pruebas) está escrito para coordinar el ingreso y salida de prueba.
3. Se prueba el conjunto o construcción.
4. Los controladores se eliminan y los grupos se combinan moviéndose hacia arriba en la estructura del programa.

Se empleó un acercamiento práctico de manera descendente para los niveles superiores de la estructura del programa y ascendente para los niveles inferiores combinando lo mejor de ambas técnicas.

Resultados

Se aplicó la prueba a la integración al módulo Exámenes, obteniéndose los resultados que a continuación se muestran a continuación.

Tabla 7: Caso de prueba de integración del componente: Crear curso (Elaboración propia)

Caso de prueba de integración del componente: Crear curso	
Sistema/componente al que se integra	Exámenes
Condiciones de ejecución	Deben existir previamente un nivel y un idioma.
Descripción de la prueba	Comprobar que el componente Crear curso, cargue los datos provenientes (Idioma y nivel). Asimismo, valide la correcta introducción de los campos. Además, que permita insertar los datos y notifique al usuario de cada acción ocurrida.
Pasos de ejecución	El probador interactúa con el componente. Se insertan los campos con valores válidos y no válidos. Se emiten o no de alertas del componente según corresponda en cada instante. Comprobar en base de datos que los valores han sido insertados satisfactoriamente.
Resultados esperados	El componente “Crear curso” brinda toda la información solicitada. Posee las validaciones pertinentes a los campos e inserta satisfactoriamente los datos.
Evaluación	Prueba satisfactoria.

3.2.3. Pruebas de sistema

El software es solo un elemento de un sistema más grande, a este lo acompañan otros elementos como por ejemplo, el hardware que lo soporta, la información que procesa y las personas que lo utilizan. Las pruebas de sistemas verifican que todos los elementos se acoplan correctamente y que se logra la función o rendimiento general del sistema. Los pasos tomados durante el diseño y las pruebas del software pueden mejorar en gran medida la probabilidad de una integración exitosa del software en el sistema más grande. Se debe anticipar posibles problemas de interfaz y diseñar rutas de manejo de errores que prueben toda la información proveniente de otros elementos [33].

3.2.3.1. Pruebas de rendimiento

Para sistemas en tiempo real e integrados, el software que proporciona la función requerida pero no cumple con los requisitos de rendimiento es inaceptable. Las pruebas de rendimiento están diseñadas para probar el rendimiento en tiempo de ejecución del software en el contexto de un sistema integrado. Estas pruebas se realizan en todos los pasos del proceso de prueba. Incluso a nivel de unidad, el rendimiento de un módulo individual puede evaluarse a medida que se realizan las pruebas. Sin embargo, no es hasta que todos los elementos del sistema están completamente integrados que se puede determinar el verdadero rendimiento de un sistema. Las pruebas de rendimiento a menudo se combinan con las pruebas de estrés y generalmente requieren instrumentación tanto de hardware como de software. Es decir, a menudo es necesario medir la utilización de los recursos (por ejemplo, los ciclos del procesador) de una manera precisa. La instrumentación externa puede monitorear los intervalos de ejecución, registrar eventos (por ejemplo, interrupciones) a medida que ocurren, y muestrea los estados de las máquinas regularmente. Al instrumentar un sistema, el probador puede descubrir situaciones que conducen a la degradación y la posible falla del sistema [33].

Resultados obtenidos

Se realizaron pruebas para comprobar que el tiempo de respuesta del módulo no excediera a los 5 segundos de forma general, así como el tiempo de respuesta por cada transacción promedio sea un máximo de 3 segundos. A continuación se muestran una prueba de rendimiento de las realizadas en el navegador web Mozilla Firefox Developer Edition.

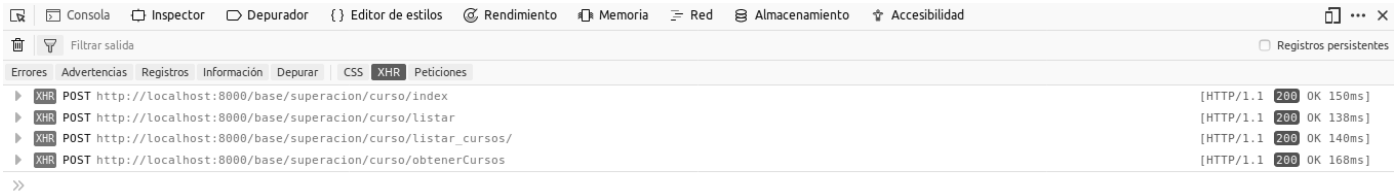


Figura 18: Prueba de rendimiento de la funcionalidad Cursos

3.2.4. Pruebas de validación

Las pruebas de validación comienzan en la culminación de la prueba de integración, cuando se han ejercitado los componentes individuales, el software se ensambla completamente como un paquete y los errores de interfaz se han descubierto y se han corregido. En la validación o nivel del sistema, la distinción

entre diferentes categorías de software desaparece. Las pruebas se centran en las acciones visibles por el usuario y en la salida del sistema reconocible por el usuario. Al igual que todos los otros pasos de prueba, la validación intenta descubrir errores, pero el enfoque está en el nivel de requisitos, en cosas que serán inmediatamente evidentes para el usuario final [33].

Para el desarrollo de las pruebas de validación se aplicó la técnica de partición de equivalencia, donde se verifica la conformidad de los requisitos.

La partición de equivalencia es un método de prueba de caja negra que divide el dominio de entrada de un programa en clases de datos a partir de los cuales se pueden derivar los casos de prueba. Un caso de prueba de acuerdo por sí solo descubre una clase de errores (por ejemplo, el procesamiento incorrecto de todos los datos de caracteres) que de otro modo podrían requerir que se ejecuten muchos casos de prueba antes de que se observe el error general. El diseño del caso de prueba para la partición de equivalencia se basa en una evaluación de clases de equivalencia para una condición de entrada [33].

En el diseño de caso de prueba hay que tener en cuenta que V indica válido, I inválido y NA que no aplica. Las variables Curso, Idioma, Nivel, Descripción, Siglas y Habilitado significan los valores de entrada de datos para los casos de pruebas; Respuesta del sistema es la variable de salida y en el Flujo central se especifican los pasos para lograr realizar las operaciones.

Resultados

Seguidamente se describe el caso de prueba Crear curso, para ver los diseños de casos de pruebas en su totalidad, consultar el expediente del proyecto Idiomas.

Tabla 8: Diseño de casos de prueba. Crear curso. Parte I (Elaboración propia)

Escenario	Descripción	Curso	Idioma	Descripción	Nivel
EC 1.1 Insertar datos correctos	Mediante este escenario se inserta un curso en el sistema.	V	V	V	V
		Curso Inglés A1	Inglés	Breve descripción	A1
EC 1.2 Insertar elementos repetidos.	Mediante este escenario se introducen datos para insertar un curso que ya	I	NA	NA	NA
		Curso Inglés A1			

	existe en el sistema.				
EC 1.3 Insertar datos incompletos.	Mediante este escenario no se introducen todos los datos para crear un curso.	I	I	NA	I
EC 1.4 Insertar datos incorrectos	Mediante este escenario se introducen datos incorrectos.	I	NA	NA	NA
		"Curso %InglesA1^"			
		NA	NA	NA	NA
		Escribir un nombre con un solo caracter.			
		I	NA	I	NA
		Escribir un nombre que termine con espacio.		Escribir una descripción que termine con espacio.	
		I	NA	I	NA
		El usuario entra una palabra con más de 30 caracteres		El usuario entra una palabra con más de 30 caracteres	
EC 1.5 Cancelar operación.	Se cancela la creación del curso.	NA	NA	NA	NA

Tabla 9: Diseño de casos de prueba. Crear curso. Parte II (Elaboración propia)

Siglas	Habilitado	Respuesta del sistema	Flujo central
V	V	El sistema crea el nuevo curso y muestra el mensaje: “El elemento ha sido creado satisfactoriamente.”	El usuario una vez autenticado en el Sistema de Gestión Académica selecciona el Componente Idioma y luego el módulo Superación. - El sistema muestra las opciones de menú. - El usuario selecciona en la agrupación funcional "Diseño" la funcionalidad "Cursos". - El sistema muestra los cursos registrados. - El usuario selecciona en el área de iconos flotantes la opción: Crear, llena los datos correctamente y presiona el botón Aceptar.
IA1	Marcado		
NA	NA	El sistema muestra un mensaje de error indicando: “El elemento ya existe.”	El usuario una vez autenticado en el Sistema de Gestión Académica selecciona el Componente Idioma y luego el módulo Superación. - El sistema muestra las opciones de menú. - El usuario selecciona en la agrupación funcional "Diseño" la funcionalidad "Cursos". - El sistema muestra los cursos registrados. - El usuario selecciona en el área de iconos flotantes la opción: Crear, llena los datos correctamente de un curso que ya está en el sistema y presiona el botón Aceptar.
I	NA	El sistema muestra encima del componente un mensaje en color rojo indicando: “Campo requerido.”	El usuario una vez autenticado en el Sistema de Gestión Académica selecciona el Componente Idioma y luego el módulo Superación. - El sistema muestra las opciones de menú. - El usuario selecciona en la agrupación funcional "Diseño" la funcionalidad "Cursos". - El sistema muestra los cursos registrados.

			- El usuario selecciona en el área de iconos flotantes la opción: Crear, llena los datos de forma incompleta y presiona el botón Aceptar.
NA	NA	El sistema muestra en rojo el mensaje encima del componente: “Entre solo letras, números y espacio o guión bajo entre palabras”	El usuario una vez autenticado en el Sistema de Gestión Académica selecciona el Componente Idioma y luego el módulo Superación. - El sistema muestra las opciones de menú. - El usuario selecciona en la agrupación funcional "Diseño" la funcionalidad "Cursos". - El sistema muestra los cursos registrados. - El usuario selecciona en el área de iconos flotantes la opción: Crear, llena los datos de forma incorrecta y presiona el botón Aceptar.
NA	NA	El sistema muestra en color rojo encima del componente el mensaje: “Entre al menos 2 caracteres.”	
NA	I	El sistema muestra en rojo encima del componente: “Solo admite espacio entre palabras.”	
	Escribir una sigla que termine con espacio.		
NA	NA	El sistema muestra en rojo encima del componente: Ha excedido el número de caracteres permitidos para una palabra.	
NA	NA	El sistema muestra el mensaje de confirmación: ¿Está seguro de realizar la acción? Si el usuario presiona el botón Aceptar el sistema retorna a la página que le dio inicio sin guardar las	El usuario una vez autenticado en el Sistema de Gestión Académica selecciona el Componente Idioma y luego el módulo Superación. - El sistema muestra las opciones de menú. - El usuario selecciona en la agrupación funcional "Diseño" la funcionalidad "Cursos". - El sistema muestra los cursos registrados. - El usuario selecciona en el área de iconos flotantes

		últimas operaciones. Si el usuario presiona el botón Cancelar se mantiene en la misma vista, mostrando las últimas operaciones realizadas.	la opción: Crear, llena o no los datos y presiona el botón Cancelar.

3.2.5. Pruebas de aceptación

Cuando se crea un software personalizado para un cliente, se realizan una serie de pruebas de aceptación para que el cliente pueda validar todos los requisitos. Conducido por el usuario final en lugar de por los ingenieros de software, una prueba de aceptación puede ir desde una “prueba de manejo” informal hasta una serie de pruebas planificadas y ejecutadas sistemáticamente. De hecho, las pruebas de aceptación pueden llevarse a cabo durante un período de semanas o meses, descubriendo así errores acumulativos que podrían degradar el sistema con el tiempo. Para descubrir errores que solo el usuario final parece poder encontrar se utiliza un proceso llamado prueba alfa y beta.

La prueba **alfa** se lleva a cabo en el sitio del desarrollador por un grupo representativo de usuarios finales. El software se utiliza en un entorno natural con el desarrollador "mirando por encima del hombro" de los usuarios y registrando errores y problemas de uso. Las pruebas alfa se realizan en un ambiente controlado.

La prueba **beta** se realiza en uno o más sitios de usuarios finales. A diferencia de las pruebas alfa, el desarrollador generalmente no está presente. Por lo tanto, la prueba beta es una aplicación "en vivo" del software en un entorno que no puede ser controlado por el desarrollador. El cliente registra todos los problemas (reales o imaginarios) que se encuentran durante las pruebas beta y los informa al desarrollador a intervalos regulares [33].

Se aplicaron ambos tipos de pruebas tomando como referencia las historias de usuarios comprobando que el sistema responde a las necesidades del cliente.

3.3. Validación de la hipótesis

Para realizar la validación de la investigación se aplicó el método **Criterio de expertos** a través del cumplimiento de los pasos siguientes:

Capítulo 3: Implementación y evaluación de la solución

1. Identificación de los posibles expertos.
2. Determinación del coeficiente de competencia de los candidatos a expertos.
3. Selección de los expertos.
4. Realización del cuestionario a los expertos.
5. Aplicación del escalamiento de Likert.

Identificación de los posibles expertos: para la identificación de los posibles expertos es necesario tomar a consideración la experiencia profesional que poseen respecto al tema, con el objetivo de que sean capaces de ofrecer valoraciones y recomendaciones pertinentes en relación con los aspectos que les serían consultados [34].

Determinación del coeficiente de competencia de los candidatos a expertos: una vez identificados los posibles expertos, se les realizó una encuesta para valorar el grado de conocimiento y el grado de influencia que cada una de las fuentes ha tenido en ese conocimiento (ver Anexo 1). El procedimiento empleado para determinar el coeficiente de competencia de los candidatos a expertos, así como los resultados obtenidos pueden ser consultados en los Anexos 2 y 3 respectivamente.

Selección de expertos: fueron seleccionados expertos con nivel alto y mayor coeficiente de competencia, dado que el coeficiente de competencia promedio del grupo de expertos sobrepasaba a 0.9. En la siguiente tabla se muestran los expertos seleccionados.

Tabla 10: Expertos seleccionados en la validación de la investigación (Elaboración propia)

Experto	Área	Cargo	Coeficiente
Ivonne de la Caridad Collada Peña	Centro de Idiomas	Director	1,00
Pedro Castro Álvarez	Centro de Idiomas	Subdirector	0,95
Nery Karen García Pando	Facultad 1. Departamento de Ciencias Sociales y Humanidades	Profesor	0,95

Realización del cuestionario a los expertos: una vez seleccionado los expertos se les realizó un cuestionario compuesto por ocho (8) afirmaciones evaluativas que permiten conocer la opinión del experto. Los parámetros evaluativos empleados fueron: MA (muy de acuerdo), DA (de acuerdo), N (Ni de acuerdo ni en desacuerdo), ED (en desacuerdo), CD (completamente en desacuerdo). El cuestionario y

las respuestas dadas por los expertos pueden ser consultadas en los Anexos 4 y 5 respectivamente. Los parámetros evaluativos se cuantificaron a través de la siguiente escala:

- Muy de acuerdo (100).
- De acuerdo (75).
- Ni de acuerdo ni en desacuerdo (50).
- En desacuerdo (25).
- Completamente en desacuerdo (0).

Aplicación del escalamiento de Likert: para el procesamiento de los resultados se empleó un método que consiste en identificar la media en cada categoría de la escala de Likert. Dicho método se aplicó por seaparado a cada una de las afirmaciones. Luego se obtuvo la media de las valoraciones individuales para obtener una valoración integral del sistema [35].

Se obtuvieron resulttados satisfactorios con valores que superan el 90 % de aprobación por parte de los expertos. El procesamiento realizado a través del escalamiento de Likert evidencia que tanto los elementos teóricos de la investigación como la propuesta de solución, tienen una alta valoración por parte de los expertos. Durante el proceso se constataron criterios favorables en el uso del módulo para mejorar la gestión de cursos de idiomas en la Universidad de las Ciencias Informáticas.

3.4. Conclusiones del capítulo

En este capítulo se han abordado los elementos de la implementación del módulo Superación, así como las pruebas realizadas al mismo y los resultados obtenidos permitiendo arribar a las siguientes conclusiones:

- El empleo de estándares de codificación contribuyó a una mejor legibilidad y comprensión del código con el objetivo de facilitar el mantenimiento del módulo Superación.
- El diseño y ejecución de la estrategia de pruebas descrita, permitió detectar y corregir deficiencias presentes durante la implementación así como el cumplimiento con las especificaciones dadas por el cliente.
- La validación de la hipótesis a través del método de criterio de expertos, ofreció una alta valoración de la propuesta de solución.

Conclusiones generales

Conclusiones generales

Una vez cumplidos los objetivos trazados en la presente investigación se concluye lo siguiente:

- Se realizó un análisis y estudio de sistemas informáticos que apoyan y dan soporte tecnológico a centros educativos en la gestión de cursos de idiomas. Dichos sistemas respaldan las características principales de las aplicaciones de su tipo sirviendo como referencia para guiar el desarrollo del módulo.
- La aplicación de la metodología de desarrollo de software Scrum, permitió la adaptación ante los cambios, así como las frecuentes reuniones con el cliente favoreció al seguimiento del estado de la aplicación y su revisión.
- La validación de la hipótesis de investigación, a través del método de criterio de expertos con escalamiento de Likert, evidenció el correcto cumplimiento de los objetivos planteados en la investigación para la mejora de la eficiencia de la gestión de cursos de idiomas en la Universidad de las Ciencias Informáticas.
- Se implementó un módulo integrado al Sistema de Gestión Académica que mejora la eficiencia de la gestión de cursos de idiomas en la Universidad de las Ciencias Informáticas. Contribuye a la superación de los trabajadores y la formación de pregrado. Agiliza los procesos que se ejecutan en el Centro de Idiomas, favorece al ahorro de recursos y apoya a la toma de decisiones.

Recomendaciones

Para garantizar el perfeccionamiento progresivo de la solución propuesta se proponen las siguientes recomendaciones:

- Desplegar el módulo desarrollado en la Universidad de las Ciencias Informáticas.
- Confeccionar los manuales de usuario del módulo Superación para apoyar la comprensión de los usuarios y que estos puedan hacer un mejor uso del mismo.

Bibliografía referenciada

[1] M. Niño-Puello, «El inglés y su importancia en la investigación científica: algunas reflexiones», *Rev. Colomb. Cienc. Anim. - RECIA*, pp. 243-254, ene. 2013.
[2] «Inglés en la universidad: replantean política de enseñanza», *Granma.cu*. [En línea]. Disponible en: http://www.granma.cu/cuba/2018-05-31/ingles-en-la-universidad-replantean-politica-de-ensenanza-31-05-2018-23-05-50. [Accedido: 14-nov-2018].
[3] «El centro de idioma | Portal del Centro de Idiomas». [En línea]. Disponible en: https://cenid.uci.cu/es/el-centro-de-idioma. [Accedido: 08-abr-2019].
[4] «Definición de curso — Definicion.de», *Definición.de*. [En línea]. Disponible en: https://definicion.de/curso/.
[5] Lalima y K. L. Dangwal, «Blended Learning: An Innovative Approach», *Univers. J. Educ. Res.*, vol. 5, n.º 1, pp. 129-136, ene. 2017.
[6] «Tipos Principales de Exámenes», *Portal del Centro de Idiomas*, 29-jun-2016. [En línea]. Disponible en: https://cenid.uci.cu/es/tipos-principales-de-examenes.
[7] «Gestión de academias de inglés: Atenea - Ender Aplicaciones», *Ender*, 29-jun-2011. .
[8] «Software Academias | Software para Escuelas de Idiomas - OfiELE». [En línea]. Disponible en: https://www.ofiele.es/. [Accedido: 14-nov-2018].
[9] «Software de gestion para centros de formacion y academias con cientos de herramientas y funciones - aGora ERP». [En línea]. Disponible en: https://www.agora-erp.com/cu/features. [Accedido: 12-dic-2018].
[10] infoadmin, «LAS TIC EN LA GESTIÓN UNIVERSITARIA CUBANA: BARRERAS, PRINCIPIOS, ACCIONES», *Informática Habana 2018*, 14-mar-2018. [En línea]. Disponible en: http://www.informaticahabana.cu/es/node/4075. [Accedido: 22-nov-2018].
[11] I. B. Núñez, «REFLEXIÓN BIBLIOGRÁFICA SOBRE SISTEMAS INFORMÁTICOS PARA LA GESTIÓN DE INFORMACIÓN DE MAESTRÍAS EN CENTROS UNIVERSITARIOS», *Universidad&Ciencia*, vol. 6, n.º 3, pp. 26-35, oct. 2017.
[12] «Lineamientos», *PCC*, 25-ene-2018. [En línea]. Disponible en: https://www.pcc.cu/es/lineamientos. [Accedido: 17-may-2019].
[13] «HTML5», *MDN Web Docs*. [En línea]. Disponible en: https://developer.mozilla.org/en-US/docs/Glossary/HTML5. [Accedido: 13-nov-2018].
[14] «HTML & CSS - W3C». [En línea]. Disponible en: https://www.w3.org/standards/webdesign/htmlcss. [Accedido: 13-nov-2018].
[15] «JavaScript», *MDN Web Docs*. [En línea]. Disponible en: https://developer.mozilla.org/en-US/docs/Web/JavaScript. [Accedido: 13-nov-2018].
[16] «PHP», *MDN Web Docs*. [En línea]. Disponible en: https://developer.mozilla.org/en-US/docs/Glossary/PHP. [Accedido: 16-nov-2018].
[17] «List of SQL Commands | Codecademy». [En línea]. Disponible en: https://www.codecademy.com/articles/sql-commands. [Accedido: 13-nov-2018].
[18] «What is Modeling Language? - Definition from Techopedia», 14-nov-2018. [En línea]. Disponible en: https://www.techopedia.com/definition/20810/modeling-language. [Accedido: 14-nov-2018].
[19] «What is Unified Modeling Language (UML)? - Definition from Techopedia», 14-nov-2018. [En línea]. Disponible en: https://www.techopedia.com/definition/3243/unified-modeling-language-uml. [Accedido: 14-nov-2018].

Bibliografía referenciada

[20] «Visual Paradigm Product Overview», 14-nov-2018. [En línea]. Disponible en: https://www.visual-paradigm.com/support/documents/vpuserguide/12/13/5963_visualparadi.html. [Accedido: 14-nov-2018].

[21] «Welcome to CodeIgniter — CodeIgniter 3.1.9 documentation». [En línea]. Disponible en: https://www.codeigniter.com/user_guide/general/welcome.html. [Accedido: 13-nov-2018].

[22] J. F.- js.foundation, «jQuery». .

[23] «Git». [En línea]. Disponible en: https://git-scm.com/. [Accedido: 14-nov-2018].

[24] «Home - Pencil Project». [En línea]. Disponible en: https://pencil.evolus.vn/. [Accedido: 06-abr-2019].

[25] «What is NGINX? - NGINX». [En línea]. Disponible en: https://www.nginx.com/resources/glossary/nginx/. [Accedido: 04-dic-2018].

[26] «NetBeans IDE - Overview». [En línea]. Disponible en: https://netbeans.org/features/index.html. [Accedido: 13-nov-2018].

[27] «PostgreSQL: About». [En línea]. Disponible en: https://www.postgresql.org/about/. [Accedido: 13-nov-2018].

[28] «Introduction — pgAdmin III 1.22.2 documentation». [En línea]. Disponible en: https://www.pgadmin.org/docs/pgadmin3/1.22/introduction.html. [Accedido: 14-nov-2018].

[29] «Scrum Guide | Scrum Guides». [En línea]. Disponible en: https://scrumguides.org/scrum-guide.html. [Accedido: 14-nov-2018].

[30] S. Robinson, «Conceptual modelling for simulation Part I: definition and requirements», *J. Oper. Res. Soc.*, vol. 59, n.º 3, pp. 278-290, mar. 2008.

[31] L. Motiejūnas y R. Butleris, «BUSINESS RULES MANIPULATION MODEL», *Inf. Technol. Control*, vol. 36, n.º 3, abr. 2015.

[32] I. Sommerville, *Software engineering 9th Edition*, vol. 137035152. 2011.

[33] Roger S. Pressman, *Software Engineering: A Practitioner'S Approach, 8th Edition*, 8th Edition. 2014.

[34] «CRITERIO DE EXPERTOS. SU PROCESAMIENTO A TRAVÉS DEL MÉTODO DELPHY». [En línea]. Disponible en: http://www.ub.edu/histodidactica/index.php?option=com_content&view=article&id=21:criterio-de-expertos-su-procesamiento-a-traves-del-metodo-delphy&catid=11:metodologia-y-epistemologia&Itemid=103.

[35] O. Llauradó, «La escala de Likert: qué es y cómo utilizarla». [En línea]. Disponible en: https://www.netquest.com/blog/es/la-escala-de-likert-que-es-y-como-utilizarla.

Anexos

Anexo 1. Encuesta para determinar el coeficiente de competencias de los expertos

Nombre y apellidos _____

Usted ha sido seleccionado como posible experto para ser consultado respecto a temas relacionados a la gestión de cursos académicos, específicamente cursos de idiomas, con vista a la investigación que se está llevando a cabo. Agradecemos sinceramente su valiosa cooperación.

1. Marque con una cruz (X) en la tabla siguiente el valor que se corresponde con el grado de conocimiento que usted posee sobre la gestión de cursos de idiomas. (Escala ascendente).

0	1	2	3	4	5	6	7	8	9	10

2. Realice una autoevaluación del grado de influencia que cada una de las fuentes que le presentamos a continuación ha tenido en su conocimiento. Marque con una cruz (X) según corresponda en A (alto), M (medio) o B (bajo).

No	Fuente de argumento	Grado de influencia		
		B	M	A
1	Análisis teórico realizado			
2	Conocimiento de investigaciones y/o publicaciones nacionales e internacionales			
3	Experiencia obtenida durante su vida laboral			
4	Intuición			
5	Conocimiento del estado actual del problema en la universidad			

Anexo 2. Procedimiento empleado para determinar el coeficiente de competencia de los candidatos a expertos

Cálculo del coeficiente de competencia de los expertos que evaluaron el módulo desarrollado. El cálculo de dicho coeficiente se realiza de la forma siguiente:

Kcomp = ½ (Kc + Ka)

Donde:

Kcomp: Coeficiente de competencia.

Kc: Coeficiente de conocimiento o información que tiene el experto acerca del problema, calculado sobre la valoración del propio experto en una escala de 0 a 10 y multiplicado por 0,1.

Ka: Coeficiente de argumentación o fundamentación de los criterios del experto, obtenido como resultado de la suma de los puntos de acuerdo a la siguiente tabla patrón:

Tabla 11: Fuentes de argumentación del conocimiento de los expertos.

No	Fuentes de argumentación	Alto(A)	Medio(M)	Bajo(B)
1	Análisis teórico realizado	0,30	0,25	0,15
2	Conocimiento de investigaciones y/o publicaciones nacionales e internacionales.	0,20	0,10	0,05
3	Experiencia obtenida durante su vida laboral.	0,30	0,20	0,10
4	Intuición.	0,10	0,05	0,05
5	Conocimiento del estado actual del problema en la universidad.	0,10	0,10	0,05
	Total	**1,00**	**0,70**	**0,40**

Se plantea entonces que:

- La Competencia del experto es de Alta (A): Si **1 > Kcomp > 0,7**
- La Competencia del experto es Media (M): Si **0,4 < Kcomp < 0,7**
- La Competencia del experto es Baja (B): Si **Kcomp =< 0,4**

Anexo 3. Resultado del cuestionario aplicado a los candidatos a expertos para determinar su nivel de competencia

Tabla 12: Resultado de la encuesta aplicada a los candidatos a expertos para determinar nivel de competencia (Elaboración propia)

Expertos	Kc	1.1	1.2	1.3	1.4	1.5	Ka	Kcomp	valor
1	0,90	0,30	0,20	0,30	0,10	0,10	1,00	0,95	alto
2	1,00	0,30	0,20	0,30	0,10	0,10	1,00	1,00	alto
3	1,00	0,30	0,20	0,30	0,10	0,10	1,00	1,00	alto
4	0,80	0,25	0,20	0,30	0,05	0,10	0,90	0,85	alto
5	0,90	0,30	0,20	0,30	0,10	0,10	1,00	0,95	alto
6	0,80	0,25	0,1	0,30	0,05	0,10	0,80	0,80	alto

Anexo 4. Cuestionario a expertos

Módulo Superación para la gestión de cursos de idiomas en el CENID				
Datos del encuestado				
Entidad:				
Área:				
Nombre y Apellidos:				
Cargo o Rol:				
Nivel Escolar:	Técnico Medio __		Universitario __	
Categoría docente:	Instructor __	Asistente __	Auxiliar __	Titular __
Categoría científica:	Especialista __	Máster __	Doctor __	
Años de experiencia:				
Afirmaciones			Respuesta	
1. El módulo posee una interfaz intuitiva, amigable y fácil de utilizar.			MA __ DA __ N __ ED __ CD __	
2. El módulo garantiza la fiabilidad e integridad de los datos registrados de las personas matriculadas a un curso.			MA __ DA __ N __ ED __ CD __	
3. El módulo favorece la obtención en poco tiempo de información confiable y actual.			MA __ DA __ N __ ED __ CD __	
4. El módulo permite visualizar las solicitudes de cursos.			MA __ DA __ N __ ED __ CD __	
5. El módulo permite dar respuesta a las solicitudes de cursos.			MA __ DA __ N __ ED __ CD __	
6. El módulo posibilita realizar reportes de las evaluaciones y la asistencia de estudiantes.			MA __ DA __ N __ ED __ CD __	
7. El módulo favorece la toma de decisiones para el centro.			MA __ DA __ N __ ED __ CD __	
8. El módulo puede ser de gran utilidad para agilizar en tiempo y el ahorro de recursos durante la gestión de cursos de idiomas en la Universidad.			MA __ DA __ N __ ED __ CD __	

Anexo 5. Respuestas dadas por los expertos para cada indicador

5: Muy de acuerdo (MA)

4: De acuerdo (DA)

3: Ni de acuerdo ni en desacuerdo (N)

2: En desacuerdo (ED)

1: Completamente en desacuerdo (CD)

Experto	Indicador							
	1	2	3	4	5	6	7	8
1	4	5	5	5	5	5	5	5
2	4	5	5	5	5	5	5	5
3	4	5	5	5	5	5	5	5