UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 1

“**SEIE.** Sistema de Evaluación Integral Estudiantil”

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

Autor: Alcides Rodríguez Rodríguez

Tutores:

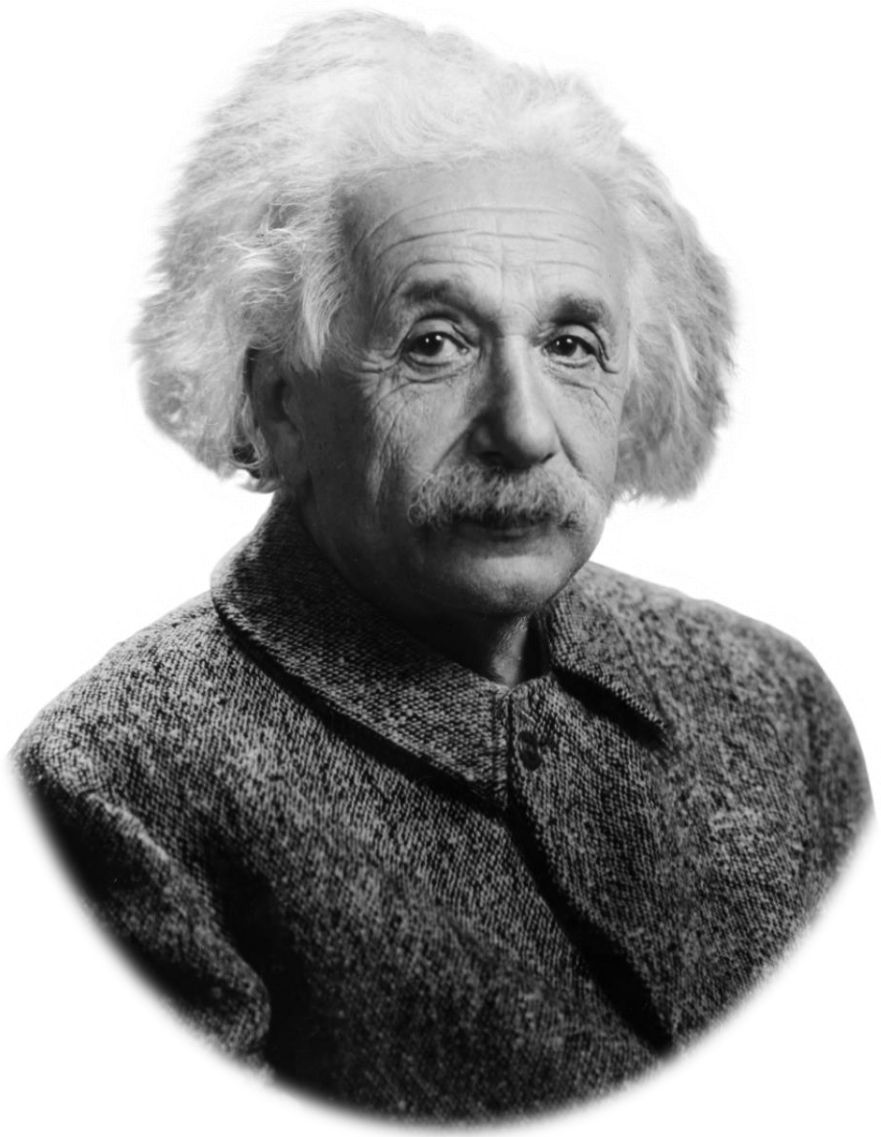
Mr. Aneity García Marín

Ing. Leiny Pons Flores

Cotutor:

Ing. Duriet Aguilera Alvarez

La Habana, 2020



"En medio de la dificultad yace la oportunidad"

Albert Einstein.

Declaración jurada de autoría

Declaro por este medio que yo **Alcides Rodríguez Rodríguez**, con carné de identidad **96090517221** soy el autor principal del trabajo titulado "**SEIE**. Sistema de Evaluación Integral Estudiantil" y autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

Para que así conste firmo la presente a los _____ días del mes de ___junio_____ de _____2020_

_____

Alcides Rodríguez Rodríguez

Autor

_____ _____

Mr. Aneity García Marín Ing. Leiny Pons Flores

Tutor Tutor

_____

Ing. Duriet Aguilera Alvarez

Cotutor

Resumen

La Federación Estudiantil Universitaria (FEU) en la Universidad de las Ciencias Informáticas lleva a cabo un importante número de procesos que requieren reducir el tiempo de procesamiento de la información que se genera, estandarizar la documentación que aportan las diferentes áreas en relación con la organización y elevar la persistencia de los datos.

En el presente informe se muestran los resultados de la implementación de la herramienta Sistema de Evaluación Integral Estudiantil (SEIE). Esta es una aplicación informática que asegura la calidad del proceso de evaluación estudiantil. Se centra principalmente en el proceso de integralidad y caracterización de los estudiantes registrando mediante evidencias la trayectoria de cada uno de ellos en un expediente digital.

Cuenta con 10 módulos que se integran entre sí y permiten el correcto funcionamiento, integridad y persistencia de los datos que se introducen y procesan.

Como parte de la investigación se valida la herramienta a partir de diferentes pruebas al sistema y se realizó una encuesta para medir la satisfacción de los usuarios que interactúan con el software.

Palabras Clave: evaluación integral, sistema informático, gestión de información, Federación Estudiantil Universitaria.

Índice de Contenido

Índice de tablas

Índice de Figuras

INTRODUCCIÓN

Introducción

Las aceleradas transformaciones científicas, tecnológicas, en los sistemas de producción, económico, político y cultural, que caracteriza a la sociedad del siglo XXI, demandan de la educación cambios en el quehacer formativo, y particularmente del nivel superior la formación de la persona, del ciudadano y profesional capaz de adaptarse a los nuevos escenarios sociales y hacer frente a las problemáticas que estos traen consigo. Es por ello que las Instituciones de Educación Superior (IES) del presente siglo, orientan sus esfuerzos en promover la formación integral del estudiante desarrollando un pensamiento analítico, crítico, creativo y propositivo (Pérez, 2016).

El Ministerio de Educación Superior (MES) en Cuba tiene entre sus mayores desafíos, el logro de la formación profesional del estudiante en el que se combine el desarrollo intelectual con el sentido de la responsabilidad ética, social, ambiental, disposición y capacidad de gestión para participar de manera activa, crítica y creativa en la etapa histórica concreta en que desarrolla su vida.

Una vía que contribuye a alcanzar ese reto es el proceso de evaluación integral del estudiante al ofrecerle elementos cualitativos y cuantitativos que revelan datos sobre la forma multifacética y armónica de cómo transcurre su desarrollo y los resultados obtenidos; en la misma medida, le aporta mecanismos para autorreconocer (o que otros reconozcan) tanto las fallas como las potencialidades y oportunidades que lo ayudan a superarlas, al proponerse metas, compromisos, tareas, acciones en aras de incrementar su implicación en el proceso de formación (Acevedo, et al., 2017).

La Universidad de las Ciencias Informáticas (UCI), como centro docente-productor que forma profesionales altamente calificados y comprometidos en la rama de la informática lleva a cabo el proceso de evaluación estudiantil, denominado proceso de Integralidad para los estudiantes de primero a cuarto año y proceso de Caracterización para los de quinto, desarrollando las capacidades, actitudes, valores y potencialidades del futuro profesional. Esto se logra durante todo el periodo de formación en el pregrado a partir del control y evaluación de los estudiantes en el ejercicio de sus actividades.

En cada curso escolar el estudiante debe entregar una autoevaluación del período señalado que es tomada como base por la brigada para otorgarle una evaluación. La forma de realizar este proceso trae consigo que

INTRODUCCIÓN

la veracidad de esa información se vea comprometida, reflejando el poco control de la trayectoria estudiantil ya que las actividades no son registradas en el momento en que ocurren.

Según los registros de estos procesos en años anteriores, obtenidos en las actas de los Consejos de la FEU, una cifra cercana al 60% de los estudiantes no entrega su caracterización o lo hace con mala calidad. Lo anterior obliga a los jefes de brigada a recopilar todas las evidencias necesarias para llevar a cabo esta evaluación, ralentizando el envío de la información a organismos superiores y el cierre del proceso.

Al no existir una estandarización en la documentación a almacenar, cada estudiante decide en que formato enviar sus datos: PDF, Microsoft Office u Open Office/Libre Office, lo que torna engorroso los pasos de unificar y procesar la información. Tampoco se cuenta con un archivo histórico para la consulta de la información de años anteriores.

En estos momentos todo este proceso se realiza de forma manual lo que hace más engorroso el control de esta información, la que se almacena en el mejor de los casos en documento Word. Luego es enviada a niveles superiores a través de correo electrónico, dispositivos de almacenamiento de datos, lo que resulta compleja su manipulación y almacenamiento, afectándose así la agilidad y calidad del proceso. Esto trae consigo la existencia de duplicados o ausencia de la información, dificultades debido a la gestión manual de grandes volúmenes de información y en la confección de la caracterización de los estudiantes.

Lo anterior evidencia la poca fiabilidad en la información que se recopila, comprometiéndose así la integridad, la calidad y la autenticidad de la misma.

Por tal situación se plantea como **Problema de investigación:** ¿Cómo contribuir a la gestión y control de la información del proceso de Evaluación Integral Estudiantil en la Universidad de las Ciencias Informáticas?

Se define como **objeto de estudio**: sistemas de gestión de información en centros educacionales.

Objetivo general: Desarrollar un sistema web que contribuya a la gestión y control de la información que se genera en el proceso de Evaluación Integral Estudiantil en la Universidad de las Ciencias Informáticas.

Dentro del objeto de estudio se encuentra como **campo de acción**: sistema informático para la gestión y control del proceso de Evaluación Integral Estudiantil.

INTRODUCCIÓN

Objetivos específicos:

- Elaborar el marco teórico-metodológico de la investigación referente al desarrollo de sistemas de gestión de información orientados al proceso de Evaluación Integral Estudiantil.
- Diseñar el sistema de gestión de la información para el proceso de Evaluación Integral Estudiantil en la Universidad de las Ciencias Informáticas (UCI).
- Implementar el sistema de gestión de la información para el proceso de Evaluación Integral Estudiantil en la Universidad de las Ciencias Informáticas (UCI).
- Validar el sistema de gestión de la información para el proceso de Evaluación Integral Estudiantil en la Universidad de las Ciencias Informáticas (UCI).

Para darle cumplimiento al objetivo planteado se responderán las siguientes **preguntas científicas**:

- ¿Cuáles con los supuestos teóricos que sustentan la investigación referente al desarrollo de sistemas de gestión de información orientados al proceso de Evaluación Integral Estudiantil?
- ¿Qué aspectos deben tenerse en cuenta para realizar el análisis y diseño del sistema para el proceso de Evaluación Integral Estudiantil en la Universidad de las Ciencias Informáticas (UCI)?
- ¿Cómo implementar a partir del análisis y diseño realizado el sistema para el proceso de Evaluación Integral Estudiantil en la Universidad de las Ciencias Informáticas (UCI)?
- ¿Qué resultados se obtendrán al validar el sistema para el proceso de Evaluación Integral Estudiantil en la Universidad de las Ciencias Informáticas (UCI)?

Las **tareas de investigación** a desarrollar para dar respuesta a los objetivos específicos:

- Sistematización de los principales referentes teóricos que permiten la elaboración del marco teórico de la investigación.
- Análisis valorativo de sistemas nacionales e internacionales existentes que gestionan la información referente al proceso de evaluación integral estudiantil.
- Selección de las herramientas y tecnologías utilizadas en la implementación del sistema de gestión de la información para el proceso de Evaluación Integral Estudiantil en la Universidad de las Ciencias Informáticas (UCI).

INTRODUCCIÓN

- Identificación de los requerimientos funcionales y no funcionales del sistema de gestión de la información para el proceso de Evaluación Integral Estudiantil en la Universidad de las Ciencias Informáticas (UCI).
- Diseño del sistema de gestión de la información para el proceso de Evaluación Integral Estudiantil en la Universidad de las Ciencias Informáticas (UCI).
- Implementación del sistema de gestión de la información para el proceso de Evaluación Integral Estudiantil en la Universidad de las Ciencias Informáticas (UCI).
- Validación del sistema de gestión de la información para el proceso de Evaluación Integral Estudiantil en la Universidad de las Ciencias Informáticas (UCI).

Para el desarrollo de la investigación se emplearon los siguientes **métodos de la investigación científica**:

Métodos Teóricos

Histórico-lógico: se utiliza para conocer el uso de los sistemas de gestión existentes a nivel internacional y nacional que tengan similitud con el que se va a implementar, así como las investigaciones que se realizaron anteriormente referentes al tema.

Analítico-sintético: se utiliza para el análisis bibliográfico de la investigación y para sintetizar las características de los procesos y sistemas estudiados.

Modelación: se utiliza para crear abstracciones con el objetivo de explicar la realidad. Se modelan todos los procesos de la trayectoria estudiantil relacionados con las esferas de cultura, deporte, investigación, formación docente y residencia de la UCI.

Métodos empíricos:

Observación: se utiliza para percibir cómo fluyen los procesos involucrados en la trayectoria estudiantil relacionados con las esferas de cultura, deporte, investigación, formación docente y residencia de la UCI.

Entrevista: la entrevista se utiliza como técnica de recopilación de información para entender los procesos involucrados en la trayectoria estudiantil relacionados con las esferas de cultura, deporte, investigación, formación docente y residencia de la UCI.

INTRODUCCIÓN

Análisis documental: se evidencia en el estudio de los documentos que establecen procedimientos para la gestión de los procesos involucrados en la trayectoria estudiantil relacionados con las esferas de cultura, deporte, investigación, formación docente y residencia de la UCI.

Estructura del documento

La estructura del trabajo de diploma, siguiendo un orden lógico ha sido dividida en tres capítulos:

El **primer capítulo: "Fundamentación teórica"**, consiste en una fundamentación teórica del resto de la investigación y un estudio del estado del arte de los sistemas informáticos para la gestión de los procesos de la Federación Estudiantil Universitaria. Se identifican los procesos de llevados a cabo por la organización. Se selecciona la metodología, las herramientas y las tecnologías a utilizar a partir de una revisión de las existentes en el mundo.

El **segundo capítulo: "Descripción de la solución propuesta"**, abarca la descripción de los procesos de negocios, la definición de los requisitos funcionales y no funcionales, el diseño y arquitectura de la aplicación a desarrollar. Se describen los patrones de diseño y el estilo arquitectónico empleado.

En el **tercer capítulo: "Implementación y validación del sistema"**, se analizan los resultados de la implementación del sistema propuesto, además de verificarse requisitos funcionales a través de pruebas y donde se evalúa la calidad del producto desarrollado. Se efectúa además un estudio de satisfacción a los clientes finales que consumirán la aplicación web.

CAPÍTULO 1: Fundamentación teórica del Sistema de Evaluación Integral Estudiantil (SEIE)

1.1 Introducción

En el presente capítulo se identifican los procesos que lleva la FEU, además de una visión de los principales conceptos y elementos teóricos para el desarrollo de la aplicación. Se analizan las tecnologías, métodos, técnicas y herramientas necesarias para la implementación del software. Se realiza un estudio del estado del arte de los sistemas informáticos para la gestión de los procesos de la Evaluación Integral Estudiantil, identificando tendencias y requisitos funcionales.

1.2 Conceptos asociados a la investigación.

La gestión de la información en las organizaciones

1.2.1 Gestión de la información

La Gestión de Información es un proceso estratégico que tiene lugar en una organización de cualquier tipo (incluidas las comunidades y otras entidades de carácter social). Es un proceso que abarca todos los procesos y actividades de esa organización y sus componentes por lo que tiene una estrecha relación con el sistema que lo rige y participan en él diferentes componentes (Duarte, 2016).

La gestión de información se define como el proceso mediante el cual se obtienen, despliegan o utilizan recursos básicos (económicos, físicos, humanos, materiales) para manejar información dentro y para la sociedad a la que sirve. Tiene como elemento básico, la gestión del ciclo de vida de este recurso y se desarrolla en cualquier organización (Rodríguez, 2015).

La Gestión de Información se concibe como un proceso estratégico de planificación, organización, dirección y control, de forma eficaz y eficiente, de las estrategias, recursos, procesos, sistemas, productos, servicios, y demás capacidades informacionales existentes en una organización o en la sociedad, con el objetivo de mejorar el desempeño, la toma de decisiones, la adaptación al cambio, y la creación de fortalezas y ventajas competitivas por parte de los individuos, las organizaciones y la administración pública (Cruz, 2015).

A partir del análisis de los conceptos referentes a la Gestión de la Información expuestos anteriormente se decide utilizar el brindado por el Dr. Yunier Rodríguez Cruz , pues define todos los procesos que forman parte de la gestión de la información ya que se puede vincular de forma eficiente a la gestión de la

información que se trata en el proceso de Evaluación Integral Estudiantil en la Universidad de las Ciencias Informáticas y expone los objetivos que se pretenden lograr con la solución del problema planteado, como la optimización de tiempo y contribuir a mejorar el control que se realiza en el momento de realizar el proceso.

1.2.2 Software de Gestión

Las aplicaciones o software de gestión son aquellas diseñadas para sustituir uno o varios procedimientos, tanto comerciales como administrativos, que habitualmente realiza una persona en una empresa o institución de forma presencial o por un software que permite realizar al cliente los mismos procedimientos de forma no presencial, disminuyendo el esfuerzo empleado para la realización de los mismos (Martelo, y otros, 2015).

1.2.3 Evaluación integral estudiantil

La Federación Estudiantil Universitaria (FEU), como organización representativa de los jóvenes universitarios define el concepto de integralidad como un proceso de seguimiento a las potencialidades y actitudes de los universitarios en el ejercicio de sus actividades como estudiantes, tanto académicas, investigativas, políticas, extensionistas, que tiene como objetivo contribuir a la formación de profesionales cada vez más preparados y comprometidos con la sociedad, incondicionales a la Revolución (Laguna, 2018).

Las consideraciones anteriores han propiciado al autor de la investigación referirse a la evaluación integral como un conjunto de actividades sistemáticas de recopilación, análisis e interpretación de la información que se genera producto del desempeño del estudiante universitario como parte de su formación integral en las universidades, que, apoyada en la comparación con criterios referentes, permite emitir una cualificación del objeto evaluado.

1.3 Estado del arte

Principales sistemas de gestión de la información en el proceso de Evaluación Integral Estudiantil:

1.3.1 Sistemas homólogos de gestión de la información relacionados al proceso de evaluación integral estudiantil en el mundo

El estudio de los sistemas homólogos se realiza para tener una referencia acerca de la forma de informatizar los procesos de una organización. Actualmente existen diversas soluciones informáticas encargadas de gestionar la información de los procesos estudiantiles que se realizan dentro de unidades educacionales que se extienden por Cuba y el mundo.

ALEXIA

Desarrollada por Educaria, empresa dedicada a prestar asistencia y servicios informáticos ubicada en Madrid, España, es totalmente web y tiene más de 1.200 centros educativos usuarios. Aporta flexibilidad en todas las áreas de gestión: facturación, impagados, ventas, contabilidad, evaluación, programación de aula, seguimiento del alumno, comunicación (web y app), etcétera. Dispone de cuadro de mando para extraer indicadores a nivel de centro y agrupación. Se integra con G Suite, Office 365, Moodle, y otros aplicativos, y permite incorporar soluciones específicas de horarios, calidad y aprendizaje, con herramientas para que el centro pueda desarrollar su propia estrategia de contenidos. La usan docentes, a quienes se les facilita el acompañamiento de sus estudiantes al poder observar calificaciones con el fin de evaluar rendimientos y elaborar hipótesis sobre la ruta de sus aprendizajes. Datos que pueden cruzar, a su vez, con la asistencia y con las herramientas que le ofrece plataforma de enseñanza-aprendizaje Alexia. A través de la misma los centros escolares pueden gestionar con eficacia y rapidez las tareas administrativas de su Secretaría, volcar calificaciones, controlar asistencia y comunicarse con las familias según diversos y numerosos objetivos.

Desventajas: este sistema, aunque cuenta con un gran número de funcionalidades relacionadas con la gestión académica estudiantil, así como otras soluciones, no posee un registro de información referente a la participación de los estudiantes en las diversas dimensiones educativas ya sean curricular, sociopolítico y extensionista siendo estas las principales características demandadas por la institución con el objetivo de llevar una evaluación eficiente de los estudiantes.

ÁGORA

Ágora es un producto de software estándar de gran calidad para la Gestión de Centros Docentes y Academias de todo tipo, desarrollado por la empresa española Kherian Soft. Se adapta a cualquier tipo de centro o formación, ya sea de tipo oficial o de carácter libre (academias de enseñanza general, de idiomas, informática, música u oposiciones entre otras), ya se trate de un pequeño centro o una gran empresa de formación con una gestión centralizada de varias sucursales distantes geográficamente. Para ello, la aplicación cuenta con varias versiones y múltiples opciones de organización docente para utilizar en cada caso.

Ventajas: aporta soluciones en diversos sentidos, como son la gestión centralizada de datos a través de la cual se gestionan de forma centralizada todas las bases de datos de alumnos, profesores, aulas y clases; la generación automática de la documentación. Gestiona con gran facilidad excepciones que dificultan la planificación tales como: clases que cambian de horario, profesores que están de baja y han de ser sustituidos, entre otras. Con el control automatizado de docencia real, el programa calculará cada día la docencia que, en función de los horarios y las matrículas, teóricamente se habrá impartido, generando los registros correspondientes, permite al usuario verificarlos periódicamente para ajustar - con un simple clic de ratón - las incidencias (clase aplazada, no impartida, impartida, aunque no esté planificada). Es especialmente eficaz para calcular los pagos a profesores. Así mismo, calculará de forma automática la disponibilidad horaria de profesores y aulas, y de plazas en grupos y clases. El control automatizado de asistencia real permite un registro automatizado de la asistencia.

Desventajas: la principal desventaja de este software radica en su carácter propietario, por lo que habría que pagar para su utilización e instalación en la UCI y por otro lado no cuenta con la capacidad de gestionar los datos relacionados a la participación de los estudiantes en las diversas dimensiones educativas ya sean curricular, sociopolítico y extensionista siendo estas las principales características demandadas por la institución con el objetivo de llevar una evaluación eficiente de los estudiantes.

1.3.2 Sistemas homólogos de gestión de la información relacionados al proceso de evaluación integral estudiantil en Cuba

AKADEMOS

Sistema de Gestión Universitaria de la Universidad de las Ciencias Informáticas (UCI), conocido como Akademos, es un sistema de planificación empresarial de recursos universitarios que incluye la concepción

de varias líneas de desarrollo agrupadas en las áreas de procesos. (Flores, y otros, 2016). Fue desarrollado para mejorar la planeación, organización y dirección de los procesos universitarios facilitando su ejecución y control tanto para trabajadores como para estudiantes.

Desventajas: no cuenta con la capacidad de gestionar sistemáticamente los datos relacionados a la trayectoria de los estudiantes en las esferas de cultura, deporte y residencia siendo estas tres de las principales características demandadas por la institución con el objetivo de llevar una evaluación integral eficiente de los estudiantes.

SIGENU

El Sistema de Gestión de la Nueva Universidad (SIGENU) es un software desarrollado en la Universidad Tecnológica de la Habana "José Antonio Echeverría" (CUJAE) en el 2007 que se perfilaba, en su diseño original, como el resultado de la acción mancomunada de varios IES, en el desarrollo de módulos que automatizarían la gestión de los procesos fundamentales y de apoyo de las universidades: matrícula, proceso docente, postgrado, relaciones internacionales, investigación, etc. No obstante, los esfuerzos realizados por los desarrolladores y la dirección del Ministerio de Educación Superior (MES), hasta la actualidad, sólo se ha extendido a todos las IES el módulo de matrícula y promoción (Ruiz, et al., 2018).

Desventajas: el sistema no brinda la posibilidad de llevar un seguimiento de la participación de los estudiantes en eventos de investigación, cultura y deporte, formación docente y residencia los cuales forman parte del proceso de caracterización de los estudiantes**.** Además, su interfaz principal necesita ser instalada en cada computadora que se va a utilizar por ser esta una aplicación de escritorio, lo que ocasiona pérdidas de tiempo y limitaciones en su uso.

DATAFEU

Esta es una aplicación informática que asegura la calidad del proceso de evaluación estudiantil y apoya a sus principales dirigentes en el proceso de toma de decisiones. Se centra principalmente en el proceso de integralidad y caracterización de los estudiantes registrando mediante evidencias la trayectoria de cada uno de ellos en un expediente digital.

DataFEU fue desarrollado sobre los marcos de trabajo *Symfony y Ext JS* empleando MySQL como servidor de base de datos, ajustándose a la soberanía tecnológica por la cual aboga Cuba. Cuenta con cinco subsistemas que se integran entre sí y permiten el correcto funcionamiento, integridad y persistencia de los

datos que se introducen y procesan. Brinda información a sistemas externos mediante servicios web que garantizan la interoperabilidad del mismo. Como parte de la investigación se valida la herramienta a partir de diferentes pruebas al sistema y se muestra el resultado de una encuesta para medir la satisfacción de los usuarios que interactúan con el software.

Desventajas: El sistema en estudio realiza una gestión de roles muy extensa lo que implica que el estudiante para acceder a la información de la evaluación tenga que esperar a que el presidente de brigada le brinde los permisos necesarios de acceso al sistema, luego a su vez obtenga dichas autorizaciones del presidente de la FEU de la facultad y del presidente de la FEU de la universidad. Esta escalabilidad en roles provoca que el estudiante se sienta desmotivado al acceder al sistema o pase demasiado tiempo para subir una evidencia o actualizar determinada información de la evaluación. Por otra parte, el sistema se encarga de la gestión de todos los procesos estudiantiles demandando una gran capacidad de almacenamiento en los servidores institucionales. Teniendo en cuenta a lo anterior planteado DataFEU no se encuentra en explotación debido a que la capacidad de almacenamiento de los servidores UCI esté el máximo o en fecha de caducidad y se prioricen otros recursos y nuevas plataformas que han dado al traste con los nuevos cambios en la universidad. Además, el sistema no estaba diseñado para la situación actual de la universidad con la inserción de nuevas carreras y la fusión de facultades.

Tabla 1 Comparación sistemas homólogos (Elaboración propia)

Sistemas homólogos	Aspectos comunes	Particularidades
Alexia	✓ Gestiona con eficacia y rapidez las tareas administrativas.	✓ No posee un registro de información referente a la participación de los estudiantes en las diversas dimensiones educativas.
Agora	✓ Centralización de la base de datos. ✓ Generación automática de la documentación.	✓ Carácter propietario, por lo que hay que pagar para su utilización e instalación en la UCI. ✓ No gestiona los datos relacionados a la participación de los estudiantes en las diversas dimensiones educativas.
Sigenus	✓ Desarrollo de módulos para la gestión de procesos fundamentales de un centro universitario.	✓ Es una aplicación de escritorio. ✓ Solo implementa los módulos de matrícula y promoción sin tener en cuenta la participación de los estudiantes en las diversas dimensiones educativas.
Akademos	✓ Mejora la planeación, organización y dirección de los procesos universitarios.	✓ No gestiona sistemáticamente los datos relacionados a la trayectoria de los estudiantes en las esferas de cultura, deporte y residencia a tener en cuenta en su evaluación integral.
DataFEU	✓ Se centra en el proceso de integralidad y caracterización de los estudiantes. ✓ Registra la trayectoria de los estudiantes en un expediente digital.	✓ El sistema no estaba diseñado para la situación actual de la universidad con la inserción de nuevas carreras y la fusión de facultades. ✓ Presenta cierta escalabilidad en roles provocando que el estudiante se sienta desmotivado al acceder al sistema.

1.3.3 Conclusiones de las soluciones estudiadas

Se puede concluir que las propuestas anteriores no solventan las nuevas situaciones. No cuentan con la capacidad de llevar el control, mantenimiento y gestión de la información relacionada con la evaluación integral y el desempeño de los estudiantes en las diversas dimensiones educativas o presentan una cierta escalabilidad de roles dependiendo de permisos otorgados por niveles estudiantiles lo que resulta complicado y extenso la realización de este proceso para el estudiante. En algunas no se puede acceder al código fuente ya que es privativo o su interfaz principal necesita ser instalada en cada computadora que se va a utilizar por ser esta una aplicación de escritorio, lo que ocasiona pérdidas de tiempo y limitaciones en su uso. Por tanto, se ratifica la necesidad de implementar un nuevo Sistema para la Evaluación Integral Estudiantil en la Universidad de las Ciencias Informáticas (UCI).

1.4 Metodología de desarrollo, tecnologías y lenguajes

Debido a que la universidad no cuenta con un sistema que le permita gestionar la evaluación integral de los estudiantes vinculados a las diferentes dimensiones educativas, se ha decidido implementar una aplicación que disponga de las funcionalidades fundamentales para garantizar una respuesta adecuada de los procesos que se llevan a cabo. Para conseguir desarrollar un sistema que tenga la calidad y la eficiencia requerida se utilizaron las siguientes tecnologías, metodología y leguajes para su implementación.

1.4.1 Metodología de desarrollo

Una metodología de desarrollo de software es un proceso o conjunto de procedimientos, técnicas y documentación que permiten a los desarrolladores guiar y ejecutar el proyecto con el objetivo de crear nuevas aplicaciones de calidad que satisfagan las expectativas del cliente (Enríquez, et al., 2017). Pressman define la metodología como "un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo de sistemas computaciones".

En el campo de desarrollo de software existen dos tipos de metodologías, las denominadas tradicionales (formales) y las ágiles.

Las primeras son un tanto rígidas, exigen una documentación exhaustiva y se centran en cumplir con el plan del proyecto definido totalmente en la fase inicial del desarrollo del mismo; mientras que la segunda enfatiza el esfuerzo en la capacidad de respuesta a los cambios, las habilidades del equipo y mantener una buena relación con el usuario (Enríquez, et al., 2017).

El Proceso Unificado Ágil de Scott Ambler o *Agile Unified Process* (**AUP**) en inglés es una versión simplificada del Proceso Unificado de Rational (RUP). Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP. El AUP aplica técnicas ágiles incluyendo:

- Desarrollo Dirigido por Pruebas (*test driven development* - TDD en inglés)
- Modelado ágil
- Gestión de Cambios ágil
- Refactorización de Base de Datos para mejorar la productividad.

Al no existir una metodología de software universal, ya que toda metodología debe ser adaptada a las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigiéndose así que el proceso sea

configurable. Se decide hacer una variación de la metodología AUP, de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la UCI.

AUP-UCI es una variación de la metodología AUP. En aras de aumentar la calidad del software que se produce, esta metodología se apoya en el Modelo CMMI-DEV v1.3. (por sus siglas en inglés *Capability Maturity Model Integration Development*, Integración de Modelos de Madurez de Capacidades para Desarrollo). El mismo constituye una guía para aplicar las mejores prácticas y obtener un producto o servicio con calidad en una entidad desarrolladora (Sánchez, 2014).

Fases de la metodología:

1. **Inicio:** El objetivo de esta fase es llevar a cabo las actividades relacionadas con la planeación del proyecto. Se realiza un estudio inicial de la organización que actúa como cliente y se obtiene información acerca del alcance del proyecto, se realizan estimaciones de tiempo y esfuerzo, y finalmente se decide si se ejecuta o no.
2. **Ejecución:** En esta etapa se ejecutan las actividades requeridas para desarrollar el software. Durante el desarrollo se modela el negocio, se obtienen los requisitos, se elabora la arquitectura y el diseño, se implementa y se libera el producto.
3. **Cierre:** En esta fase se analizan tanto los resultados del proyecto como su ejecución y se llevan a cabo las actividades formales de cierre de proyecto.

La metodología **AUP-UCI** se ajusta al ambiente que presenta el negocio que se desea informatizar y da la posibilidad al cliente de siempre acompañar al equipo de desarrollo para convenir los requisitos y así poder implementarlos. También, es una de las que mayor prestigio presenta hoy día entre los desarrolladores de la UCI, pues fue aprobada cuando la institución se sometió al proceso de certificación CMMI nivel 2 para el desarrollo de software.

La metodología **AUP-UCI** propone 7 disciplinas:

1. **Modelado de negocio:** es la disciplina destinada a comprender los procesos de negocio de una organización. Se comprende cómo funciona el negocio que se desea informatizar para tener garantías de que el software desarrollado va a cumplir su propósito. Para modelar el negocio se proponen las siguientes variantes

- Casos de Uso del Negocio (CUN).
- Descripción de Proceso de Negocio (DPN).
- Modelo Conceptual (MC).

A partir de las variantes anteriores se condicionan cuatro escenarios para modelar el sistema en la disciplina Requisitos

2. **Requisitos:** El esfuerzo principal en la disciplina Requisitos es desarrollar un modelo del sistema que se va a construir. Esta disciplina comprende la administración y gestión de los requisitos funcionales y no funcionales del producto. Existen tres formas de encapsular los requisitos [Casos de Uso del Sistema (CUS), Historias de usuario (HU) y Descripción de requisitos por proceso (DRP)], agrupados en cuatro escenarios condicionados por el Modelado de negocio.

Para la realización del proyecto se definió en la disciplina Requisitos utilizar el escenario No 4, que expone modelar HU (Historias de usuario) ya que el negocio fue modelado a través de modelo conceptual.

3. **Análisis y diseño:** En esta disciplina, si se considera necesario, los requisitos pueden ser refinados y estructurados para conseguir una comprensión más precisa de estos, y una descripción que sea fácil de mantener y ayude a la estructuración del sistema (incluyendo su arquitectura). Además, en esta disciplina se modela el sistema y su forma (incluida su arquitectura) para que soporte todos los requisitos, incluyendo los requisitos no funcionales. Los modelos desarrollados son más formales y específicos que el de análisis.
4. **Implementación:** En la implementación, a partir de los resultados del análisis y diseño se construye el sistema.
5. **Pruebas internas**: En esta disciplina se verifica el resultado de la implementación probando cada construcción, incluyendo tanto las construcciones internas como intermedias, así como las versiones finales a ser liberadas. Se deben desarrollar artefactos de prueba como: diseños de casos de prueba, listas de chequeo y de ser posible componentes de prueba ejecutables para automatizar las pruebas.
6. **Pruebas de liberación:** Pruebas diseñadas y ejecutadas por una entidad certificadora de la calidad externa, a todos los entregables de los proyectos antes de ser entregados al cliente para su aceptación.
7. **Pruebas de Aceptación:** Es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido (Sánchez, 2014).

1.4.2 Herramientas y Lenguaje

Las herramientas son aplicaciones, programas o meramente instrucciones que ofrecen la posibilidad de realizar determinadas funcionalidades con disímiles propósitos. De otro modo, las tecnologías son el conjunto de conocimientos técnicos, ordenados científicamente que permiten diseñar y crear bienes y servicio. A continuación, se muestran todas las herramientas, tecnologías y lenguajes utilizadas en el desarrollo del sistema de gestión y control de la información para el proceso de evaluación integral estudiantil en la Universidad de las Ciencias Informáticas**.**

Lenguaje de modelado

Un lenguaje de modelado es un lenguaje artificial para expresar modelos, habitualmente estos se muestran en forma de diagramas por comodidad. Los lenguajes de modelado, al igual que los lenguajes naturales, poseen un conjunto de palabras que existen en el lenguaje (léxico); y un conjunto de reglas que nos dicen cómo se pueden combinar dichas palabras para componer "frases" que tengan sentido (sintaxis). Estas "palabras" de los lenguajes de modelado no se transmiten mediante texto o sonido como en el caso de los lenguajes naturales, sino que, habitualmente, lo hacen en forma de iconos o dibujos para facilitar la visualización formal de los conceptos, que son abstractos (Patricia, y otros, 2015).

UML 2.0

Se define como un lenguaje gráfico para visualizar, especificar, construir y documentar los artefactos de un sistema. Proporciona una forma estándar de diagramar planos de un sistema, abarcando las partes conceptuales (funciones del sistema, y en principio también procesos industriales), y los objetos concretos (clases escritas en lenguajes de programación específico, esquemas de bases de datos, componentes de software reutilizables) (Patricia, y otros, 2015).

Herramienta de Modelado

Dentro de las principales herramientas para el modelado se encuentran herramientas de Ingeniería de Software Asistida por Computadora (*CASE* por sus siglas en inglés) definidas como “Herramientas individuales para ayudar al desarrollador de software o administrador de proyecto durante una o más fases del desarrollo de software (o mantenimiento)” (Universidad Politécnica de Valencia, 2016).

Visual Paradigm

Es una herramienta CASE que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, implementación y pruebas. Ayuda a una rápida construcción de aplicaciones con mejor

calidad y a un menor costo. Permite construir diagramas de diversos tipos, código inverso, generar código desde diagramas y generar documentación (Lianet, y otros, 2012).

Para la presente investigación se utiliza esta herramienta en su versión 8.0

Lenguajes de programación

Un lenguaje de programación es un lenguaje diseñado para describir el conjunto de acciones consecutivas que un equipo debe ejecutar. Por lo tanto, es un modo práctico para que los seres humanos puedan dar instrucciones a un equipo (Rolando, y otros, 2016).

A continuación, se muestra el lenguaje de programación utilizado para el desarrollo del sistema de gestión y control de la información para el proceso de evaluación integral estudiantil en la Universidad de las Ciencias Informáticas.

Python

Python es un lenguaje de programación poderoso y fácil de aprender. Cuenta con estructuras de datos eficientes y de alto nivel y un enfoque simple pero efectivo a la programación orientada a objetos. La elegante sintaxis de Python, junto con su naturaleza interpretada, hacen de éste un lenguaje ideal para *scripting* y desarrollo rápido de aplicaciones en diversas áreas y sobre la mayoría de las plataformas (Rolando, y otros, 2016).

Las ventajas de usar Python son:

- **Facilidad de uso**. Una persona puede empezar a hacer programas sencillos en Python en muy poco tiempo. A esto contribuye la gestión automática de memoria o las operaciones sencillas de lectura y escritura, a diferencia de otros lenguajes como C.
- **Legibilidad del código**. La estructura del código es bastante natural y promueve una forma de escribir que facilita su lectura. Esta es una ventaja importante frente a lenguajes dirigidos al mismo sector.
- **Abundancia de bibliotecas**. Hay muchas bibliotecas disponibles para extender la funcionalidad básica de Python a cualquier campo.
- **Gran base de usuarios**. Esto hace que exista mucho código disponible en internet y que los foros de usuarios sean bastante activos, por lo que es fácil encontrar ayuda cuando se necesita.

Para el desarrollo de este sistema web se utilizó Python en su versión 3.5.1

1.4.3 Servidor web

Un servidor Web es un programa que utiliza el protocolo de transferencia de hipertexto, HTTP (*Hypertext Transfer Protocol*), para servir los archivos que forman páginas Web a los usuarios, en respuesta a sus solicitudes, que son reenviados por los clientes HTTP de sus computadoras. Las computadoras y los dispositivos dedicados también, pueden denominarse servidores Web (Rouse, 2015).

1.4.4 Entorno de desarrollo

Un Entorno de Desarrollo Integrado, o bien conocido como IDE (por sus siglas en inglés *Integrated Development Environment*), es una aplicación de software, que proporciona servicios integrales para facilitarle al programador de computadora el desarrollo de software. Normalmente, un IDE consiste de un editor de código fuente, herramientas de construcción automáticas y un depurador. La mayoría de los IDEs tienen autocompletado inteligente de código (Suárez, 2016).

PyCharm IDE

PyCharm es uno de los entornos de desarrollo más completos para Python. Es parte de la suite de herramientas de programación ofrecidas por *JetBrains*, que cuenta con entornos para construir código en distintos idiomas como PHP y Ruby (Python, 2019).

Ventajas de PyCharm

Trabajar con *PyCharm* tiene ventajas básicas (similares a las ofrecidas por otros IDE) pero también algunas específicas a las cuales debe su popularidad. Es así que *PyCharm* tiene un editor inteligente, que permite completar código con algunos atajos de teclado.

Una de las características notables de *PyCharm* es la posibilidad que tiene de refactorizar el código, que en términos generales, significa modificar el código sin comprometer la ejecución del mismo.

Para llevar a cabo la implementación del sistema web se hará uso de *PyCharm* en su versión 2017.2.2

1.4.5 Sistema Gestor de Bases de Datos

Es una herramienta de propósito general útil para estructurar, almacenar y controlar los datos ofreciendo interfaces de acceso a la base de datos. Tareas fundamentales que desempeñan estos sistemas hacen referencia a la seguridad de acceso a los datos, al mantenimiento de la integridad de los datos, a mecanismos de recuperación debidos a fallos físicos y lógicos, al control de concurrencia en el momento de acceder a los datos y a la eficiencia del sistema evaluada, generalmente, en términos del tiempo de respuesta a las consultas de los usuarios (Rolando, y otros, 2017).

PostgreSQL

"PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado"(PostgreSQL, 2017).

"PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando" (Rolando, y otros, 2017).

Para desarrollar el sistema de gestión de la información para el proceso de Evaluación Integral Estudiantil en la Universidad de las Ciencias Informáticas se va a utilizar como Sistema Gestor de Base de Dato PostgreSQL en su versión 9.4.1.

1.4.6 Marcos de Trabajo

Un marco de trabajo o *framework* en inglés, está diseñado para apoyar el desarrollo de aplicaciones web. Es un software o conjunto de librerías que tratan de facilitar aquellas actividades comunes que son realizadas durante el desarrollo de una aplicación web, como, por ejemplo: acceso a la base de datos, uso de plantillas, manejo de sesiones; además, promueve la reutilización de código.

Django

Django es un *framework* de código libre que permite el desarrollo web ágil y utilizando el menor número de código posible; requiere también de un nivel de conocimientos básicos del lenguaje de programación Python (Cabrera, 2016).

Para el desarrollo del sistema web se decidió usar Django en su versión 1.11.8.

Conclusiones del capítulo

- Se identificaron los procesos que lleva a cabo la FEU, los cuales generan mucha dinámica a partir de los diferentes escenarios en el funcionamiento de la organización.
- Se realizó un análisis de los conceptos necesarios para el entendimiento de la creación del sistema a desarrollar. Además de profundizar en los aspectos teóricos necesarios para el desarrollo del trabajo.
- A partir del estudio del estado del arte se concluye que las soluciones estudiadas no solventan las nuevas situaciones; por tal motivo se ratifica la necesidad de implementar un sistema de gestión y control de la información para el proceso de evaluación integral estudiantil en la Universidad de las Ciencias Informáticas.
- Luego de realizada una revisión de los principales elementos teóricos para el desarrollo de la investigación, se decide desarrollar una aplicación web de tipo interactiva. Se define como marco de trabajo: el lenguaje de programación Python, el *framework* de código libre Django, como entorno de desarrollo *PyCharm* y como sistema gestor de base de datos fue elegido PostgreSQL.

CAPÍTULO 2. Características y diseño del Sistema de Evaluación Integral Estudiantil (SEIE)

2.1. Introducción

En el presente capítulo se realiza la descripción de la propuesta de solución. Además, se exponen las principales características del sistema, y se definen los requisitos funcionales y no funcionales necesarios para lograr que el Sistema para la Evaluación Integral Estudiantil (SEIE) funcione correctamente. Se identifican las actividades que se informatizarán y se construyen los artefactos correspondientes al análisis y diseño. Se caracteriza la arquitectura y patrones de diseño implementados en el marco de trabajo.

2.2 Procesos dentro de la FEU

El secretariado de la FEU a cada nivel es el encargado de establecer y llevar a cabo todos los procesos de la organización para conservar un correcto funcionamiento. Dentro de los cuales se encuentran:

1. Proceso de integralidad: se realiza al concluir cada curso académico entre el primer y cuarto año de la carrera. Procura evaluar a todos los estudiantes según su desempeño durante el período señalado. El resultado final es una evaluación cualitativa de bien, regular o mal, a partir del cumplimiento de un grupo de indicadores que se discuten y validan en cada brigada a inicios del semestre.

2. Proceso de caracterización: se realiza al concluir el último curso de la carrera y establece una evaluación final del desempeño del estudiante durante los años de estudio, a partir del resumen de evidencias y las evaluaciones del proceso de integralidad. Las evaluaciones generadas tributan directamente al proceso de ubicación laboral.

3. Gestión de eventos y actividades: se encarga de controlar, atender y avisar sobre las actividades a desarrollarse en cualquier esfera de la vida universitaria. Cada actividad o evento se relaciona con una esfera de trabajo. Los eventos se clasifican de acuerdo a su magnitud.

2.2.1 Pautas para el proceso de integralidad de la FEU UCI

Para el desarrollo de la propuesta de solución se tuvieron en cuenta las pautas que define la FEU UCI para el proceso de integralidad estudiantil. Para más información ver ANEXO 5 (Indicadores a medir para evaluaciones de integralidad)

Los estudiantes serán autoevaluados por ellos y además teniendo en cuenta la opinión de sus profesores guías en el cumplimiento de los deberes estudiantiles que recogen los estatutos de la organización. Además, si es dirigente estudiantil debe ser evaluado por su desempeño en las funciones que desempeña.

Según los analizados por la brigada en cada uno de los aspectos se otorgarán las evaluaciones siguientes:

- Bien (B)
- Regular (R)
- Mal (M)
- Vanguardia integral por esfera

2.3 Descripción de la propuesta de solución

El sistema web para la gestión y control de la evaluación integral estudiantil está compuesto por 10 módulos a petición del cliente: Perfil, Deporte, Residencia, Cultura, Investigación, Trabajo Socialmente Útil (TSU), Guardia estudiantil, Producción, FEU-UJC y Repositorio. En la presente investigación se implementarán cada uno de ellos.

El módulo **Perfil**, se encarga de archivar todas actividades en las que el estudiante haya participado y la evaluación asignada teniendo en cuenta las pautas de evaluación que lleva cabo la FEU para emitir una valoración (Bien, Regular, Mal y vanguardia integral por esfera), así como la constancia de los méritos obtenidos en cualquier dimensión educativa, y los cargos a nivel de facultad o universidad que haya ejercido. Posteriormente se realizan los cálculos necesarios atendiendo al cumplimiento de los deberes estudiantiles que recogen los estatutos de la organización. Nunca será evaluado como integral o vanguardia por esfera un estudiante cuyo desempeño docente no sea satisfactorio. Además, el sistema debe permitir al estudiante exportar su perfil en formato PDF.

El módulo **Deporte**, se encarga de gestionar todas las actividades deportivas que existen en la universidad y a las cuales cada estudiante puede participar. Además, permite archivar los resultados obtenidos en cada actividad.

El módulo **Residencia**, se encarga de gestionar todas las actividades que se realizan en la residencia estudiantil, así como, los resultados y evaluaciones obtenidas.

El módulo **Cultura**, se encarga de gestionar todas las actividades culturales que se realizan en la Universidad, así como archivar los resultados obtenidos en cada actividad.

El módulo **Investigación**, se encarga de gestionar todas las actividades investigativas que se realizan en la universidad, así como los resultados obtenidos en cada una de ellas.

El módulo **FEU-UJC**, se encarga de archivar los cargos que ha ocupado el estudiante en el Consejo de la FEU o Comité Primario de la UJC tanto en la facultad como en la Universidad, así como distinciones que haya alcanzado y las evaluaciones adquiridas por cada actividad realizada.

El módulo **Producción**, se encarga de archivar los eventos vinculados a la producción en los que haya participado el estudiante, así como los resultados obtenidos. Además, se tiene en cuenta las cantidades de ausencias al centro al cual pertenece.

El módulo **Trabajo Socialmente Útil (TSU)**, se encarga de archivar las evaluaciones obtenidas por el estudiante durante la etapa de TSU.

El módulo **Guardia estudiantil**, se encarga de archivar la cantidad de ausencias que ha presentado el estudiante, así como el tipo de ausencia: justificada e injustificada.

El módulo **Repositorio,** se encarga de archivar todos los perfiles de cada estudiante ordenados por año académico con la opción de exportar cada perfil en formato PDF.

2.4 Modelo de dominio

Un modelo de dominio es una representación visual de las clases conceptuales u objetos del mundo real en un dominio de interés. Este modelo se describe mediante diagramas de clase, que se van refinando con los detalles y sirven de base para el diseño de la capa lógica o de negocio del sistema y, posteriormente, para la construcción de los programas orientados a objetos. Estos diagramas muestran las clases del dominio y las relaciones entre ellas a través de asociaciones (Casadiego, 2017).

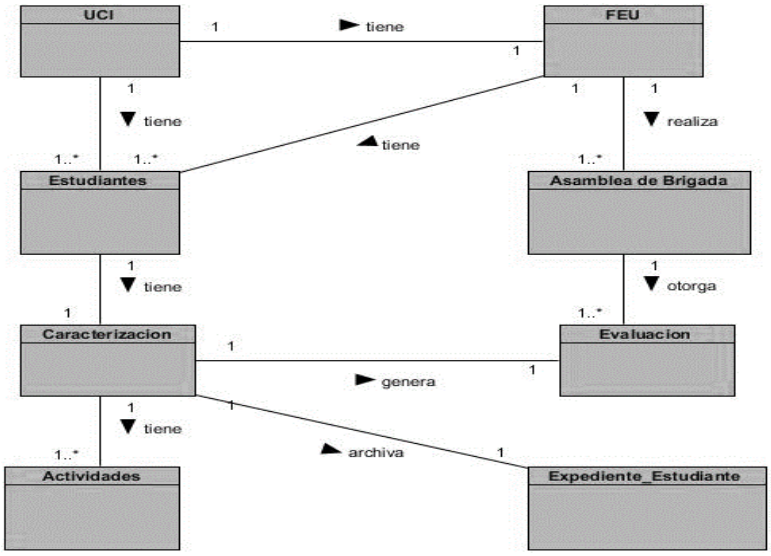


ILUSTRACIÓN 1 MODELO DE DOMINIO (ELABORACIÓN PROPIA)

2.4.1 Descripción de los objetos representados en el Modelo de dominio

UCI: Universidad de las Ciencias Informáticas, centro de altos estudios donde se pondrá en práctica el Sistema para la Evaluación Integral Estudiantil (SEIE).

FEU: entidad que representa la federación estudiantil universitaria la cual agrupa todos los estudiantes universitarios del país.

Estudiantes: usuario que dispondrá del SEIE, para realizar el proceso de caracterización y evaluación integral estudiantil.

Evaluación: unidad de evaluación que contiene todas las evaluaciones definidas a partir de las actividades en las que participa cada estudiante.

Actividades: entidad que representa las diferentes dimensiones educativas que intervienen en el proceso de caracterización e integralidad estudiantil.

Caracterización: entidad que representa el proceso de definición de las particularidades individuales de los estudiantes en valoración a su entorno escolar.

Asamblea de brigada: entidad que establece la reunión general de miembros de un colectivo de estudiantes para decidir sobre la caracterización y evaluación integral estudiantil según las diferentes

dimensiones educativas.

Expediente_estudiante: entidad que representa el documento oficial expedido por las administraciones educativas de la universidad, que refleja los resultados de la evaluación y el progreso académico del estudiante y tiene valor acreditativo de sus estudios realizados.

2.5 Especificación de Requisitos

Un requisito es simplemente una declaración abstracta de alto nivel de un servicio que debe proporcionar el sistema o una restricción de éste. En el otro extremo, es una definición detallada y formal de una función del sistema (Martinez, et al., 2015).

2.5.1 Requisitos Funcionales

Los requisitos funcionales son aquellos directamente relacionados con las funciones o las reacciones que el sistema debe proporcionar y de igual forma son prioritarios en caso de no existir la posibilidad de desarrollar todos los requisitos (funcionales o no funcionales) en escases de tiempo y recursos (Cardozzo, 2016).

TABLA 2 REQUISITOS FUNCIONALES (ELABORACIÓN PROPIA)

Requisitos funcionales	Descripción de requisitos funcionales	Prioridad	Complejidad
RF1	Autenticar usuario	alta	baja
RF2	Búsqueda avanzada	baja	alta
RF3	Insertar actividad en la residencia	alta	alta
RF4	Modificar actividad en la residencia	alta	alta
RF5	Eliminar actividad en la residencia	alta	alta
RF6	Listar actividad en la residencia	alta	alta
RF7	Insertar actividad en cultura	alta	alta
RF8	Modificar actividad en cultura	alta	alta
RF9	Eliminar actividad en cultura	alta	alta
RF10	Listar actividad en cultura	alta	alta
RF11	Insertar actividad en deporte	alta	alta
RF12	Modificar actividad en deporte	alta	alta

RF13	Eliminar actividad en deporte	alta	alta
RF14	Listar actividad en deporte	alta	alta
RF15	Seleccionar cantidad de ausencias	alta	alta
RF16	Seleccionar tipo de ausencia	alta	alta
RF17	Seleccionar evaluación en TSU	alta	alta
RF18	Seleccionar evaluación en Producción	alta	alta
RF26	Seleccionar evento en Producción	media	baja
RF19	Insertar actividad en investigación	alta	alta
RF20	Modificar actividad en investigación	alta	alta
RF21	Eliminar actividad en investigación	alta	alta
RF22	Listar actividad en investigación	alta	alta
RF23	Seleccionar evaluación en FEU-UJC	media	baja
RF24	Seleccionar cargo en FEU-UJC	media	baja
RF25	Insertar distinción en FEU-UJC	media	baja
RF27	Mostrar evaluación	alta	alta
RF28	Actualizar perfil	alta	alta
RF29	Exportar perfil	alta	alta
RF30	Seleccionar perfil en Repositorio	alta	media
RF31	Exportar perfil en Repositorio	alta	media
RF32	Mostrar perfiles en Repositorio	alta	baja

2.5.2 Requisitos No Funcionales

Los requisitos no funcionales son aquellos requisitos que no describen información a guardar, ni funciones a realizar, sino que son propiedades que hacen al producto atractivo, usable, rápido o confiable. Además, se conocen como un conjunto de características de calidad, que es necesario tener en cuenta al diseñar e implementar el software. (Sommerville, 2015)

- ✓ **Usabilidad:**

RnF_1: El sistema debe ser una aplicación web.

RnF 2: La aplicación debe mostrar una interfaz con elementos identificativos al contexto donde se aplica.

RnF_3: La aplicación debe estar dirigida a registrar la información referente al proceso de evaluación estudiantil de la FEU.

- ✓ **Confiabilidad:**

RnF_4: El sistema de autenticación recogerá datos necesarios de los usuarios y tendrá control de identidades de los mismos.

- ✓ **Seguridad:**

RnF_5: El sistema debe ser tolerante a fallos, y mostrar solo la información necesaria para orientar al usuario.

- ✓ **Rendimiento:**

RnF_6: Los tiempos de respuesta del sistema deben ser rápidos, no mayores de 2 segundos para cualquier operación realizada por el usuario.

RnF_7: Solo tendrán acceso al sistema aquellos usuarios que puedan autenticarse por el dominio de la Universidad y el Administrador del sistema.

RnF_8: Ante los errores que puedan ocasionarse en el sistema no se deben mostrar detalles de información que puedan comprometer su seguridad e integridad de los datos.

2.6 Historias de Usuario

En la metodología de desarrollo de software *AUP* variación UCI los requisitos que debe cumplir el software son especificados por los clientes en las denominadas historias de usuario. Estas historias son descompuestas en tareas de programación y asignadas a los programadores. A continuación, se describen las historias de usuario del sistema de acuerdo a la siguiente plantilla:

Historia de Usuario	
Número: *Número de la Historia de Usuario (HU), incremental en el tiempo.*	**Usuario:** *El usuario del sistema que utiliza o protagoniza la historia.*
Nombre de historia: *El nombre de la HU, sirve para identificarla fácilmente entre los desarrolladores y los clientes.*	
Prioridad en negocio: *Qué tan importante es para el cliente (Alta, Media o Baja).*	**Riesgo en desarrollo:** *Qué tan difícil es para el desarrollador (Alto, Medio o Bajo)*

Programador Responsable: *Persona que se encarga de la implementación de la funcionalidad.*
Descripción: *La descripción de la historia, detallando las operaciones del usuario y opcionalmente las respuestas del sistema.*

Para el diseño de la propuesta de solución fueron generadas un total de 34 Historias de Usuarios (HU), a continuación, solo se muestran tres (3) de ellas correspondientes a los requisitos de mayor prioridad, las restantes Historias de Usuario (HU) se encuentran en los anexos:

TABLA 3 HU AUTENTICAR USUARIO (ELABORACIÓN PROPIA)

Historia de Usuario	
Número: HU_1	**Usuario:** Estudiante
Nombre de historia: Autenticar usuario.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Programador Responsable: Alcides Rodríguez Rodríguez.	
Descripción: el usuario es el encargado de acceder al sitio con sus credenciales (usuario y contraseña) de su cuenta UCI.	

TABLA 4 HU INSERTAR ACTIVIDAD EN DEPORTE (ELABORACIÓN PROPIA)

Historia de Usuario	
Número: HU_5	**Usuario:** Administrador y Estudiante
Nombre de historia: Gestionar actividad en deporte	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Programador Responsable: Alcides Rodríguez Rodríguez.	
Descripción: la funcionalidad Gestionar actividad en deporte debe permitir adicionar , mostrar, modificar y eliminar una actividad deportiva en el sistema que será seleccionada por el estudiante antes de emitir su evaluación.	

TABLA 5 HU MOSTAR EVALUACIÓN (ELABORACIÓN PROPIA)

Historia de Usuario	
Número: HU_32	**Usuario:** Administrador
Nombre de historia: Mostrar evaluación	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Programador Responsable: Alcides Rodríguez Rodríguez.	
Descripción: El usuario se encarga de mostrar en el sistema la evaluación cualitativa (Bien, Mal o Regular) del estudiante a partir del cálculo de parámetros en su caracterización o el análisis de su integralidad.	

2.7 Estilo Arquitectónico

La Arquitectura de Software es, a grandes rasgos, una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes según se la percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema. La vista arquitectónica es una vista abstracta, aportando el más alto nivel de comprensión y la supresión o diferimiento del detalle inherente a la mayor parte de las abstracciones (Rivera, 2010).

Para la solución propuesta se decidió usar el *framework* de desarrollo Django, lo que implica una serie de decisiones de diseño. Django sigue una arquitectura Modelo-Vista-Controlador solo que hace una adaptación de esta a Modelo-Vista-Plantilla (a partir de ahora MTV por sus siglas en inglés*, Model Template View*). Por tanto, el sistema propuesto hereda una arquitectura MTV.

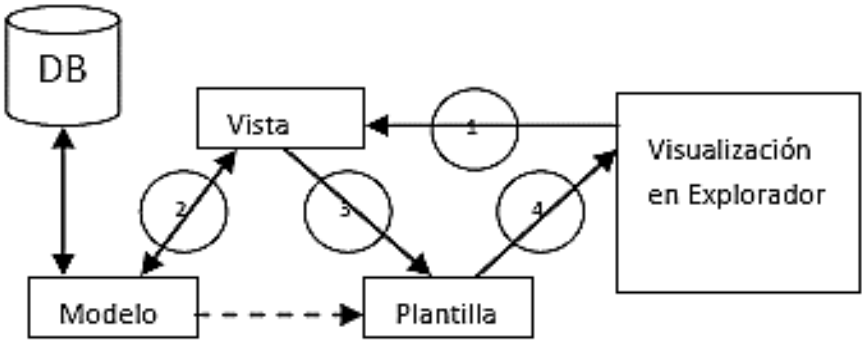


ILUSTRACIÓN 2 FUNCIONAMIENTO DE MTV DJANGO **(RIVERA, 2010)**

1. El Navegador manda una solicitud
2. La vista interactúa con el modelo para obtener datos
3. La vista llama a la plantilla
4. La plantilla renderiza la respuesta a la solicitud del navegador

Modelo: Contiene toda la información sobre los datos. Cada una de las entidades de la base de datos se encuentra en el modelo en forma de clases de Python, y sus atributos se almacenan en variables con ciertos parámetros. También estos archivos poseen métodos, lo que permite indicar y controlar el comportamiento de los datos.

Vista: Es la capa de la lógica de negocios, contiene la lógica que accede al modelo y la delega a la plantilla apropiada. Esta capa sirve de "puente" entre el modelo y la plantilla, se presenta en forma de funciones de Python y su función principal es determinar qué datos serán visualizados en las plantillas.

Plantilla: Define la interfaz de las páginas web, o sea, cómo se van a mostrar los datos al usuario.

Para cumplir con la arquitectura seleccionada, la implementación del sistema cuenta con la siguiente estructura:

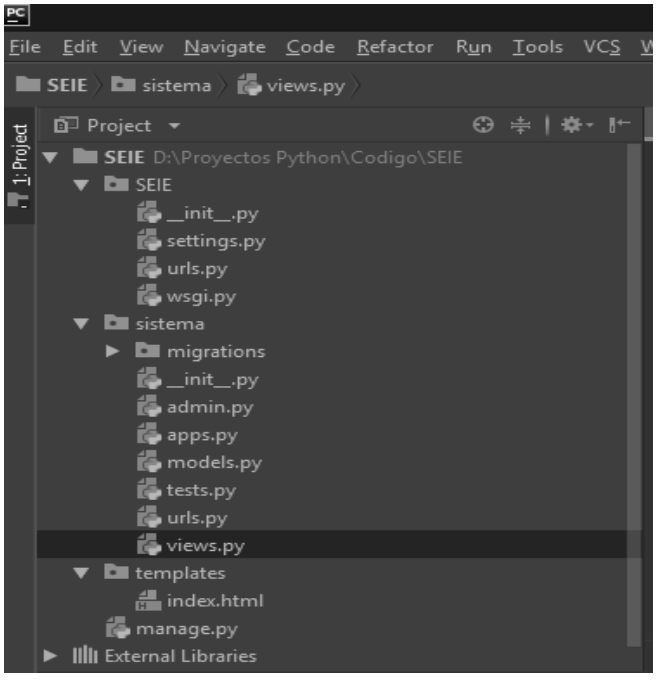


ILUSTRACIÓN 3 ESTRUCTURA DE DIRECTORIOS DEL SISTEMA (ELABORACIÓN PROPIA)

2.8 Diagrama de Clases del Diseño

Un Diagrama de Clases de Diseño muestra la especificación para las clases de software de una aplicación. Incluye la siguiente información:

- Clases, asociaciones y atributos
- Interfaces, con sus operaciones y constantes
- Métodos
- Navegabilidad
- Dependencias

Se utilizan los siguientes estereotipos:

TABLA 6 TABLA DE ESTEREOTIPOS PARA LAS CLASES

Estereotipos para las clases	
Estereotipo	**Descripción**
 *Client Page*	Representan páginas que son dibujadas por el navegador web y pueden ser una combinación de algún o algunos lenguajes de marcado, *scripts* del lado del cliente, islas de datos, etc.
 *Server Page*	Representa una página web que tiene *scripts* ejecutados por el servidor. Estos *scripts* interactúan con los recursos que se encuentran al alcance del servidor. Sólo puede mantener relaciones con objetos que se encuentren en el servidor.
 *Form*	Representa una colección de campos de entrada que forman parte con una página del lado cliente (*Client Page*). Tiene una correspondencia directa con la etiqueta <*FORM*> de *HTML*.
Estereotipos para las Relaciones entre las Clases	
Link	Representa un apuntador desde una "*client page*" hacia una "*client*

	page" o "*server page*". Corresponde directamente con una etiqueta <a> de *HTML*
Submit	Esta relación siempre se da entre una "*form*" y una "*server page*", por supuesto, la "*server page*" procesa los datos que la "*form*" le envía (*submits*)
Build	Sirve para identificar cuáles "*server page*" son responsables de la creación de una "*client page*". Una "*server page*" puede crear varias "*client page*", pero una "*client page*" sólo puede ser creada por una sola "*server page*". Esta relación siempre es unidireccional
Redirect	Esta es también una relación unidireccional que indica que una página web redirige hacia otra. En caso de que la página origen sea una "*client page*" esta asociación corresponderá con la etiqueta "*META*" y valor *HTTP-EQUIV* de "*Refresh*".

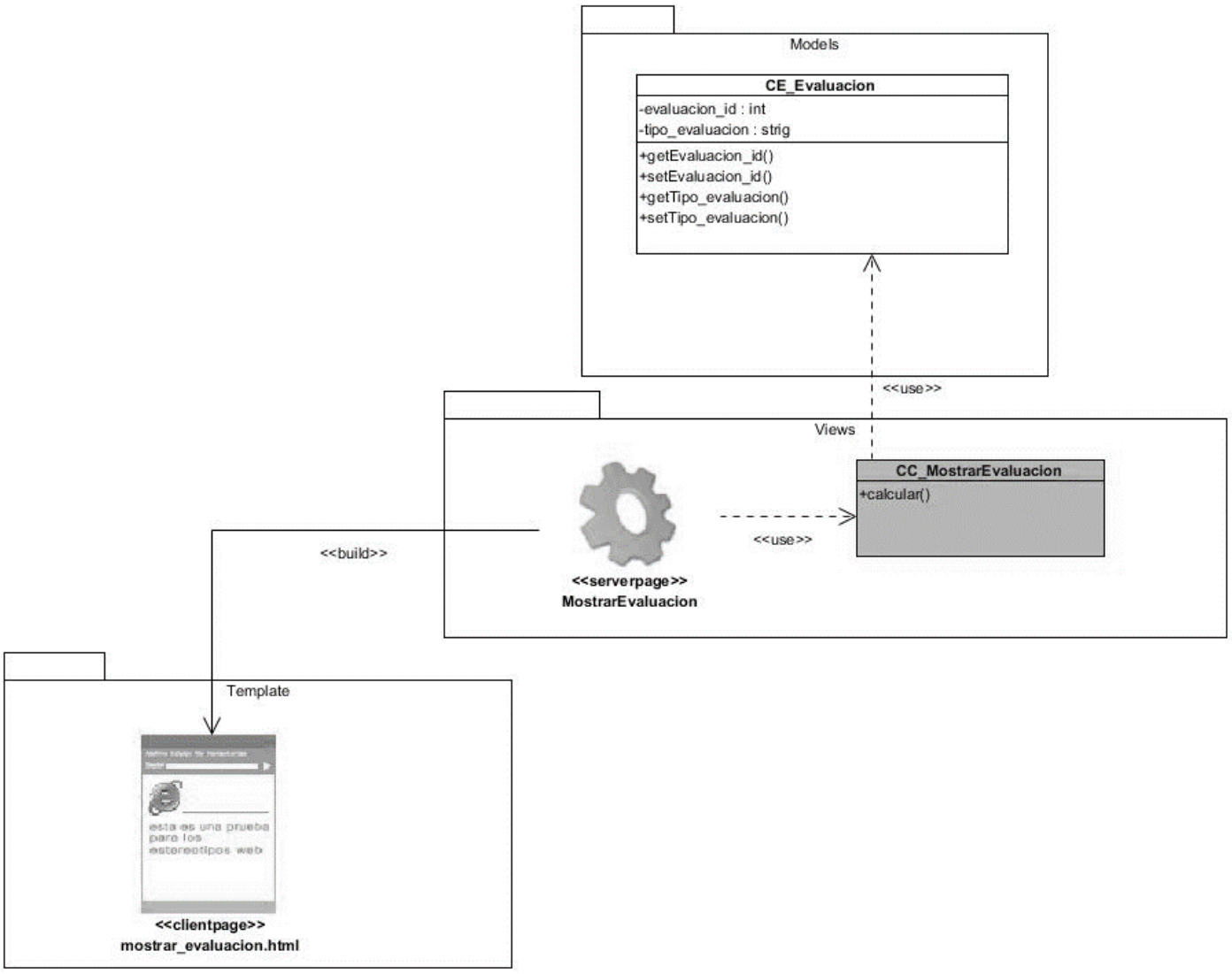


ILUSTRACIÓN 4 DIAGRAMA DE CLASE DEL DISEÑO HU MOSTRAR EVALUACIÓN (ELABORACIÓN PROPIA)

2.9 Patrones del Diseño

Los patrones de diseño son descripciones de clases y objetos relacionados que están particularizados para resolver un problema de diseño general en un determinado contexto. Un patrón de diseño nomina, abstrae e identifica los aspectos clave de una estructura de diseño común, lo que los hace útiles para crear un diseño orientado a objetos reutilizable. El patrón de diseño identifica las clases e instancias participantes, sus roles y colaboraciones, y la distribución de responsabilidades (Escalante, 2015).

Patrones Gof

Representan patrones que dan soluciones técnicas basadas en programación orientada a objeto (POO), que favorecen la reutilización del código (Guerrero, 2013).

El catálogo de patrones más famoso es el contenido en el libro "*Design Patterns: Elements of Reusable Object-Oriented Software*", también conocido como: El libro *GOF* (*Gang-Of-Four Book*). Según este documento, estos patrones se clasifican según su propósito en creacionales, estructurales y de composición, mientras que respecto a su ámbito se clasifican en clases y objetos:

Decorator **(Decorador):** Permite agregar responsabilidades adicionales a un objeto dinámicamente, proporcionando una alternativa flexible a la especialización mediante herencia, cuando se trata de añadir funcionalidades.

Como ejemplo de este patrón se evidencia la utilización de los decoradores de *Django* o definidos por el programador, cómo es el caso de *login_required*:

```
@login_required
def index(request):
    return render(request, 'base/base.html')
```

ILUSTRACIÓN 5 DECORADORES PARA EL CONTROL DE SESIONES Y PERMISOS (ELABORACIÓN PROPIA)

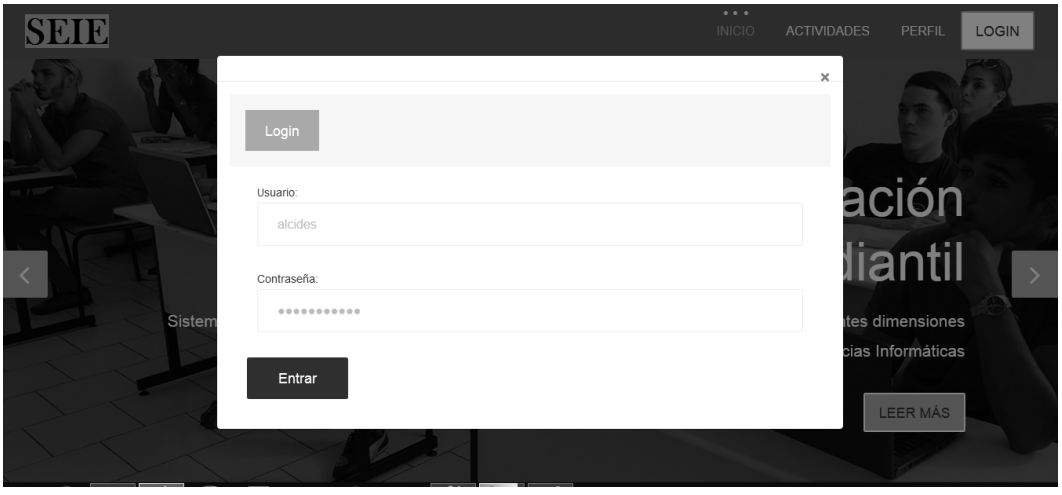


ILUSTRACIÓN 6 EJEMPLO PATRÓN DECORATOR

Patrones GRASP

Los Patrones de Principios Generales para Asignar Responsabilidades (*GRASP*) por sus siglas en inglés, describen los principios fundamentales del diseño de objetos y la asignación de responsabilidades, expresados como patrones.

Según Visconti (2012), los patrones *GRASP* son:

Experto: garantiza que la responsabilidad de la creación de un objeto o la implementación de un método, recaiga sobre la clase que conoce toda la información necesaria para crearlo, lo que contribuye a un adecuado encapsulamiento, favoreciendo la robustez y fácil mantenimiento del sistema.
Creador: se asigna la responsabilidad a una clase de crear cuando contiene, agrega, compone, almacena o usa otra clase, lo que brinda una alta posibilidad de reutilizar la clase creadora.
Este patrón se pone de manifiesto en las vistas, donde se implementan las clases controladoras para crear objetos del modelo de datos, permitiendo acceder a estos de forma directa en cada una de las vistas para enviar la información necesaria hacia las plantillas, respondiendo a las peticiones del usuario a través de un navegador web.

```
from .models import Deporte
from django.views.generic import CreateView


class ACTDeporteCreate(CreateView):
    model = Deporte
    template_name = 'deporte/deporte.html'
```

ILUSTRACIÓN 7 CÓDIGO FUENTE PARA CREAR ACTIVIDADES DEPORTIVAS (ELABORACIÓN PROPIA)

Bajo acoplamiento: es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Una clase con bajo (o débil) acoplamiento no depende de muchas otras clases.
Este patrón ya viene incluido con *Django,* el cual permite un bajo acoplamiento entre las piezas, lo que evita las dependencias, por ejemplo, a la hora de realizar cambios en las configuraciones de las *URL*, en la BD y las plantillas *HTML*, basta solo con realizarlo una sola vez.

Alta Cohesión: asigna responsabilidades de manera que una clase no tenga muchas funcionalidades no relacionadas o no realice un trabajo excesivo. Este patrón incrementa la claridad y facilita la comprensión del diseño. Al usar este patrón se garantiza una mejor eficiencia en cuanto al tiempo de respuesta de las peticiones del usuario.

Una de las características de *Django* es la organización del trabajo en cuanto a la estructura del proyecto, lo cual permite crear y trabajar con clases con una alta cohesión. Por ejemplo, se puede observar en el sistema que cada clase controladora se ajusta a manejar solo las responsabilidades correspondientes a las entidades con las que se relaciona. Esto hace posible que el sistema sea flexible a cambios sustanciales con efecto mínimo.

Controlador: Asignar la responsabilidad de administrar un mensaje de evento del sistema a una clase que representa el sistema global, dispositivo, subsistema o representa un escenario de caso de uso en el que tiene lugar el evento del sistema.

Este patrón es empleado en todo el sistema debido a que cada uno de los eventos generados por el usuario es redirigido a una clase o función controladora que realiza las operaciones solicitadas, manteniendo siempre la alta cohesión (Larman., 2003).

2.10 Modelo de datos

El modelo de datos describe de forma abstracta las entidades y sus características, además de las relaciones entre estas dentro de la base de datos. Las entidades son objetos que guardan información necesaria para el sistema. Su símbolo es un rectángulo. Los atributos son características de una entidad, se representan colocando su nombre dentro del rectángulo de la entidad. Los atributos se clasifican en: obligatorios, opcionales, claves foráneas y claves primarias (estas se dividen en simples y compuestas). Gráficamente la clave primaria se representa con una llave y la foránea con una flecha en dirección izquierda. Las relaciones muestran la asociación entre dos entidades, representadas por una línea discontinua que une a las entidades involucradas.

Conclusiones del capítulo

- El desarrollo del presente capítulo permitió sentar las bases para la construcción del sistema propuesto a partir del modelado de la propuesta de solución, propició la obtención de las principales funcionalidades que debe cumplir la aplicación y quedaron manifestadas en la descripción de requisitos de software utilizando el escenario 4.

- La utilización de los patrones de diseño permitió identificar aspectos importantes de la estructura del diseño del sistema web propuesto, garantizando una mayor organización e hizo el código más eficiente.

- Adoptar la arquitectura de software Modelo-Vista-Plantilla propuesta por el marco de trabajo utilizado, abonó una propicia organización del sistema a implementar.

CAPÍTULO 3. Implementación y prueba del Sistema de Evaluación Integral Estudiantil (SEIE)

3.1 Introducción

En el presente capítulo se muestran los artefactos correspondientes a las etapas de implementación y pruebas del sistema. El análisis ofrece un refinamiento mayor de los requisitos dándoles una estructura de mayor comprensión, además de servir como una primera aproximación al diseño. El diseño por su parte es el responsable de dar una forma más consistente al sistema, utilizando el lenguaje de programación final, por lo que el resultado del modelo del negocio es un paso previo al flujo de trabajo de implementación.

En este capítulo realizaremos, pues, el análisis y el diseño del sistema que proponemos, dejándolo listo para otros trabajos posteriores que puedan realizar su implementación.

3.2 Diagrama de Componentes

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones. Muestran las opciones de realización incluyendo código fuente, binario y ejecutable. Los componentes representan todos los tipos de elementos software que entran en la fabricación de aplicaciones informáticas. Pueden ser simples archivos, bibliotecas cargadas dinámicamente, etc. (Torres, 2002).

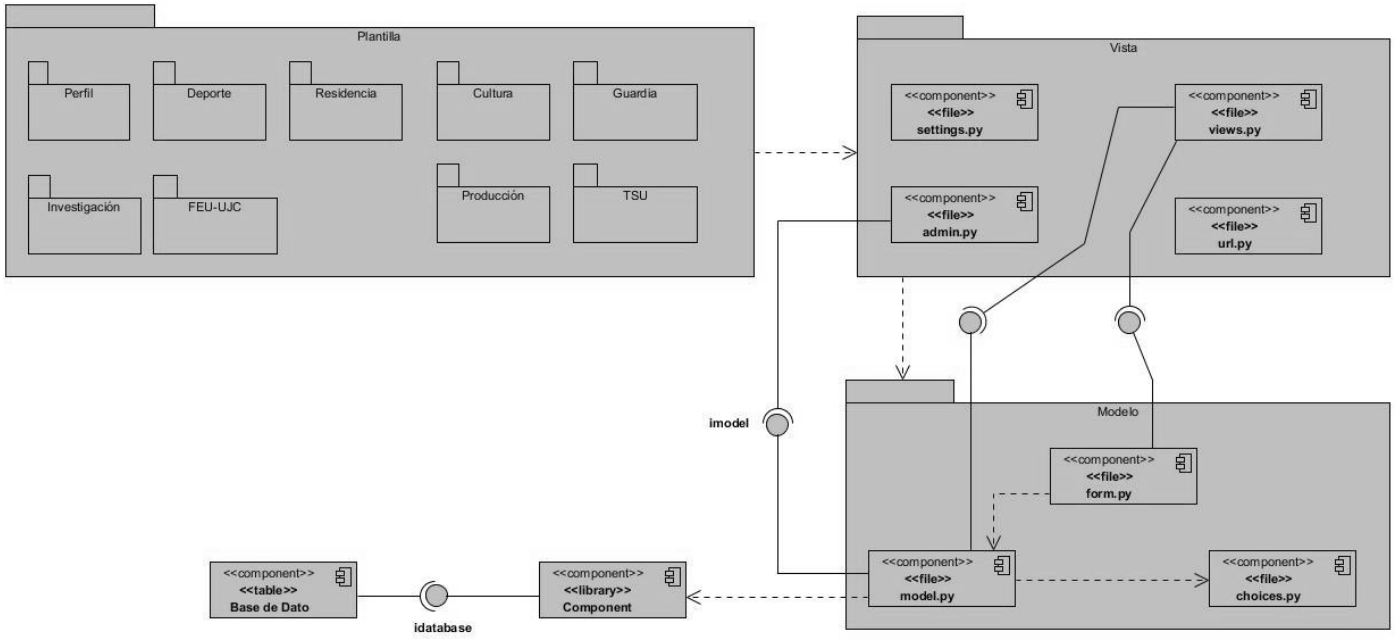


ILUSTRACIÓN 8 DIAGRAMA DE COMPONENTES (ELABORACIÓN PROPIA)

3.3 Diagrama de Despliegue

Los diagramas de despliegue muestran la disposición física de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos modos. Modela la distribución en tiempo de ejecución de los elementos de procesamiento y componentes de software, junto a los procesos y objetos asociados (Torres, 2002).

A continuación, se muestra el diagrama de despliegue definido para la propuesta de solución, que cuenta con 3 nodos principales. El nodo PC Cliente que requiere de un navegador web, el nodo Servidor Web donde debe estar instalado el sistema web para la Evaluación Integral Estudiantil (SEIE) y el nodo SGBD en el cual debe estar instalado el Sistema Gestor de Bases de Datos PostgreSQL con la base de datos del sistema. El Nodo PC Cliente representa las estaciones de trabajo de los usuarios que se conectan al sistema, como: estudiante, profesor guía y administrador, las mismas realizan peticiones al Servidor Web mediante protocolo HTTP (*Hypertext Transfer Protocol*) a través del puerto 443. El nodo Servidor Web, a su vez, estará conectado al nodo SGBD mediante el protocolo TCP\IP a través del puerto 5432.

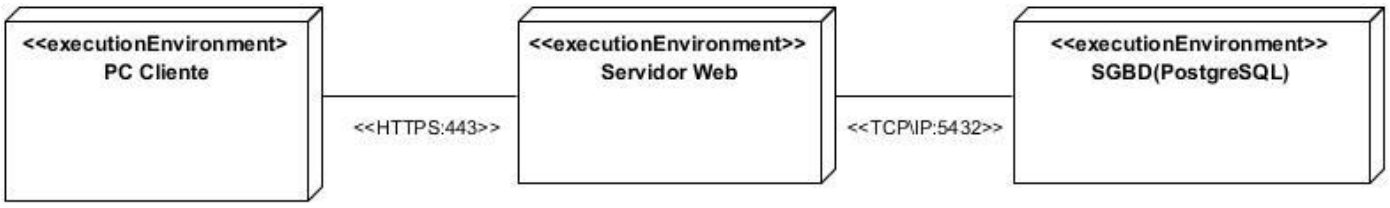


ILUSTRACIÓN 9 DIAGRAMA DE DESPLIEGUE (ELABORACIÓN PROPIA)

3.4 Interfaz gráfica del usuario

Una vez finalizado el desarrollo del software es posible visualizar las pantallas principales de SEIE, donde se observa el resultado obtenido durante la implementación de las historias de usuarios descritas en el capítulo anterior.

ILUSTRACIÓN 10 INTERFAZ PRINCIPAL DEL SISTEMA (ELABORACIÓN PROPIA)

ILUSTRACIÓN 11 INTERFAZ DE INFORMACIÓN SOBRE EL SISTEMA SEIE (ELABORACIÓN PROPIA)

ILUSTRACIÓN 12 INTERFAZ DE INFORMACIÓN SOBRE LAS PAUTAS DE INTEGRALIDAD DE LA FEU-UCI (ELABORACION PROPIA)

3.5 Estándares de Codificación

Un estándar de codificación comprende todos los aspectos de la generación del código fuente de un software. Es preciso definir una serie de pautas que lleven por objetivo uniformar la estructura del código de manera tal que este sea lo más legible posible. Cuando el proyecto de software incorpore código fuente previo, o bien cuando realice el mantenimiento de un sistema de software creado anteriormente, el estándar de codificación debería establecer cómo operar con la base de código existente (Calleja, 2015).

Para la definición de los estándares de codificación que se utilizaron en el presente trabajo de diploma se tuvieron en cuenta las características que rigen a Python. A continuación, se listan los estándares por los que se rige el autor:

1. Los nombres de los atributos, variables y parámetros tendrán todas las letras en minúsculas y usarán el guion bajo como delimitador entre palabras.
2. Declarar las variables en líneas separadas.
3. Los métodos deben comenzar con letra mayúsculas.

4. La mayoría de los elementos se deben nombrar usando sustantivos.
5. Concatenación de cadenas: Mantener un espacio a ambos lados de los operadores de concatenación (.) y (. =).
6. Comentarios: Pueden utilizarse los símbolos (//) y (/* */) indistintamente. Los comentarios deben terminar con la puntuación adecuada.
7. Nombres de funciones y miembros de clases: Las funciones deben nombrarse con letras minúsculas y las palabras separadas por un guión bajo (_). Los nombres de los miembros privados de las clases deben comenzar con un guión bajo.
8. Constantes: Las constantes serán nombradas completamente con letras mayúsculas y las partes del nombre serán separadas por un guión bajo.

3.6 Pruebas del Software

Las pruebas del software es la verificación dinámica del comportamiento de un programa contra el comportamiento esperado, usando un conjunto finito de casos de prueba, seleccionados de manera adecuada (Paz, 2015).

El objetivo de las pruebas del software es descubrir errores. Este objetivo se logra mediante una serie de pasos. Las de unidad e integración se concentran en la verificación funcional de un componente y en la incorporación de componentes en una arquitectura de software. Las pruebas de validación demuestran la conformidad con los requerimientos del software y las del sistema validan el software una vez que se incorporó en un sistema más grande. Cada paso de la prueba se logra a través de una serie de técnicas de prueba sistemáticas que auxilian en el diseño de casos de prueba. Con cada paso de prueba, se amplía el nivel de abstracción con la que se considera el software (Pressman, 2010).

Para garantizar la calidad requerida de la solución propuesta, así como el cumplimiento de los requisitos especificados, se desarrollaron pruebas que garantizan el cumplimiento de dichas acciones.

3.6.1 Estrategias de Pruebas

Una estrategia de prueba de software proporciona una guía que describe los pasos que deben realizarse como parte de la prueba, cuándo se planean y se llevan a cabo dichos pasos, y cuánto esfuerzo, tiempo y recursos se requerirán. Por tanto, cualquier estrategia de prueba debe incorporar la planificación, el diseño de casos de prueba, la ejecución y la recolección y evaluación de los resultados. Una estrategia de prueba de software debe ser suficientemente flexible para promover un uso personalizado de la prueba. Al mismo

tiempo, debe ser suficientemente rígida para alentar la planificación razonable y el seguimiento de la gestión conforme avanza el proyecto (Pressman, 2010).

3.6.2 Pruebas de Aceptación

Cuando se construye software para un cliente, se llevan a cabo las pruebas de aceptación para permitir que el cliente valide todos los requisitos. Son ejecutadas antes de que la aplicación sea instalada dentro de un ambiente de producción. La prueba de aceptación es conducida a determinar como el sistema satisface sus criterios de aceptación validando los requisitos que han sido levantados para el desarrollo, incluyendo la documentación y procesos de negocio (Pressman, 2010).

Prueba alfa

La prueba alfa se lleva a cabo, por un cliente, en el lugar de desarrollo. Se usa el software de forma natural con el desarrollador como observador del usuario y registrando los errores y los problemas de uso. Las pruebas alfa se llevan a cabo en un entorno controlado (Pressman, 2010).

Prueba beta

La prueba beta se lleva a cabo por los usuarios finales del software en los lugares de trabajo de los clientes. El cliente registra todos los problemas que encuentra durante la prueba beta e informa a intervalos regulares al desarrollador. Como resultado de los problemas informados durante la prueba beta, el desarrollador del software lleva a cabo modificaciones y así prepara una versión del producto de software para toda la clase de clientes (Pressman, 2010).

Resultados de las pruebas de aceptación

Para la ejecución de las pruebas de aceptación se realizaron varias reuniones con el cliente, donde se le explicó el funcionamiento del sistema y se observó su interacción con los componentes del mismo identificando un conjunto de no conformidades agrupadas en errores de validación e interfaz. Luego se realizaron pruebas por el cliente en su entorno de trabajo y sin observadores, permitiéndole iniciar el trabajo con el sistema en una situación real donde identificó un conjunto de no conformidades relacionadas con funciones incorrectas, las mismas fueron erradicándose a medida que eran informadas. La realización de las pruebas de aceptación a la solución utilizando las técnicas alfa y beta, permitió demostrar que el sistema cumple satisfactoriamente los requisitos del cliente, emitiéndose el acta de aceptación, garantizando evaluar el grado de calidad del sistema (**Ver acta de aceptación Anexo 3).**

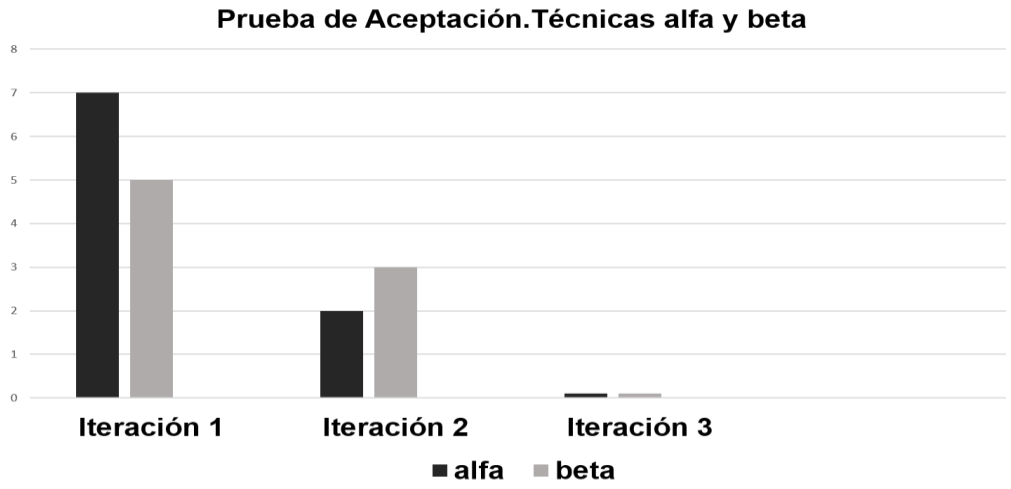


ILUSTRACIÓN 13 RESULTADOS PRUEBA DE ACEPTACIÓN (ELABORACIÓN PROPIA)

3.6.3 Tipos de prueba

Caja negra:

Las pruebas de caja negra, también denominada prueba de comportamiento, se centran en los requisitos funcionales del software. La misma permite al ingeniero del software obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. La prueba de caja negra no es una alternativa a las técnicas de prueba de caja blanca. Más bien se trata de un enfoque complementario que intenta descubrir diferentes tipos de errores. La prueba de caja negra intenta encontrar errores de las siguientes categorías: funciones incorrectas o ausentes, errores de interfaz, errores en estructuras de datos o en accesos a bases de datos externas, errores de rendimiento y errores de inicialización y de terminación (Pressman, 2010).

La prueba de caja negra aplicada al sistema se denomina Partición de Equivalencia que consiste en los valores de entrada del programa o del sistema, se dividen en grupos que vayan a tener comportamiento similar, de manera que puedan ser procesados de la misma forma. Las particiones de equivalencia o clases son aplicables a datos válidos y datos no válidos. También, pueden aplicarse a los valores de salida, valores internos, valores relativos al tiempo o a los parámetros de interfaz. (Pressman, 2010)

Como resultado de las pruebas de caja negra se realizó un caso de prueba por cada requisito (tomando los diseños basados en requisitos), en los cuales se detectaron varias no conformidades en revisiones internas

realizadas por el equipo de desarrollo, las cuales fueron resueltas. Los restantes casos de pruebas se encuentran en los **Anexos #2**

TABLA 7 CASO DE PRUEBA INSERTAR ACTIVIDAD DEPORTIVA (ELABORACIÓN PROPIA)

Nombre de la sección	**Escenario**	**Acción realizada**	**Respuesta del sistema**	**Resultado de la prueba**
SC 1: Insertar Actividad deportiva	EC 1.1: Introducir los datos actividad deportiva	El administrador introduce los datos correspondientes a actividad deportiva tales como: evento, resultados	El sistema inserta la nueva actividad deportiva	Resultado Satisfactorio.
	EC 1.2: La actividad deportiva insertado está incompleto.	El administrador selecciona el botón aceptar para insertar actividad deportiva	El sistema muestra un mensaje de alerta que existen campos vacíos.	Resultado Satisfactorio.
SC 2: Editar actividad deportiva	EC 2.1: Editar actividad deportiva	El administrador introduce los datos que desea editar.	El sistema edita actividad deportiva.	Resultado Satisfactorio.
	EC 2.1: la actividad deportiva a editar está incompleta.	El administrador selecciona el botón aceptar para editar la actividad deportiva	El sistema muestra un mensaje de alerta que existen campos vacíos.	Resultado Satisfactorio.

SC 3: Eliminar la actividad deportiva.	EC 2.1: Eliminar la actividad deportiva.	El administrador selecciona la actividad deportiva que desea eliminar	El sistema elimina actividad deportiva	Resultado Satisfactorio.
SC 4: Listar actividad deportiva	EC 3.1: Listar actividad deportiva	El administrador municipal selecciona la opción listar actividad deportiva	El sistema lista las actividades deportivas.	Resultado Satisfactorio.

Caja blanca:

Las pruebas de caja blanca, denominadas a veces prueba de caja de cristal es el método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de prueba. Mediante este tipo, el ingeniero del software puede obtener los casos de prueba que garanticen que se ejercita por lo menos una vez todos los caminos independientes de cada módulo; se ejerciten todas las decisiones lógicas en sus vertientes verdadera y falsa; se ejecuten todos los ciclos en sus límites y con sus límites operacionales, y se ejerciten las estructuras internas de datos para asegurar su validez.

La prueba de caja blanca empleada en la solución fue la prueba del camino básico, a partir del cálculo de la complejidad ciclomática del algoritmo a ser analizado. Para realizarla se deben enumerar las sentencias de código y a partir de ahí elaborar el grafo de flujo de esta funcionalidad.

Se definieron una serie de pasos a seguir:

1. Notación del grafo de flujo: usando el código como base se realiza la representación del grafo de flujo, mediante una sencilla notación. Cada construcción estructurada tiene su correspondiente símbolo.
    - Nodo: cada círculo denominado nodo, representa una o más sentencias procedimentales.

- Arista: las flechas del grafo de flujo, denominadas aristas, representan el flujo de control y son análogas a las flechas del diagrama de flujo.
- Región: las áreas delimitadas por aristas y nodos se denominan regiones.

2. Complejidad ciclomática: es una métrica que proporciona una medición cuantitativa de la complejidad lógica de un programa. El valor calculado define el número de caminos independientes del conjunto básico de un programa. Esto indica el límite superior para el número de pruebas que se deben realizar, para asegurar que se ejecuta cada sentencia al menos una vez. Se utilizó la siguiente forma: V (G), de un grafo de flujo G se define como: V (G)=A-N+2, donde A es el número de aristas del grafo de flujo y N es el número de nodos.
3. Determinar un conjunto básico de caminos linealmente independientes: el valor de V (G) es el número de caminos linealmente independientes de la estructura de control del programa.
4. Obtención de casos de prueba: se realizan los casos de pruebas que forzarán la ejecución de cada camino del conjunto básico.

3.6.5 Pruebas de rendimiento

Pruebas de carga.

La carga de trabajo se refiere a la capacidad máxima que tiene un servidor web, para atender a un conjunto de usuarios de manera simultánea. Por ello, las actividades de esta etapa tienen relación con comprobar, de manera anticipada, el funcionamiento que tendrá el servidor del Sitio Web cuando esté en plena operación (Díaz y Dosting, 2014).

Las pruebas en este caso consisten en simular una carga de trabajo similar y superior a la que tendrá cuando el sitio esté funcionando, con el fin de detectar si el software instalado cumple con los requerimientos de muchos usuarios simultáneos y también si el hardware es capaz de soportar la cantidad de visitas esperadas (Serna, Bedoya, y López, 2015).

Pruebas de estrés.

Enfocada a evaluar cómo el sistema responde bajo condiciones anormales. (Extrema sobrecarga, insuficiente memoria, servicios y hardware no disponible, recursos compartidos no disponible (Pérez y Paumier, 2015).

Resultados de las pruebas de carga y estrés.

Con el objetivo de saber con un alto nivel de proximidad el soporte del Sistema de Evaluación Integral Estudiantil (SEIE), teniendo en cuenta que la matrícula de los estudiantes en la Facultad 1 oscila 1000 estudiantes, se desarrollaron las pruebas para un total de 700,1000 y 3000 usuarios concurrentes.

Las pruebas se desarrollaron con el apoyo de la herramienta Apache Meter 2.12, en la que se simuló el entorno donde debe interactuar el sistema para obtener la información más correcta acerca del comportamiento y resultados en general. Por lo que fue elegido un ambiente de gama con las siguientes características:

Software:

- Sistema Operativo: Windows 10 Arquitectura de 64bits
- Sistema Gestor de Bases de Datos: PostgreSQL 9.4
- Servidor Web: Apache v2.4
- Máximo de hilos concurrentes: entre 700 y 1000

Hardware:

- Microprocessor: Intel Core(TM) i5 – 2120M CPU @ 3.30 GHz
- Memoria RAM: 6GB
- Tarjeta de Red: Ethernet 10/100 Mbps

Se describen a continuación las variables que miden los resultados de las pruebas.

Cantidad de Hilos (Cant de Hilos): cantidad de usuarios que se conectan concurrentemente a la aplicación.

#Muestras: cantidad de peticiones realizadas para cada URL.

Media: tiempo promedio en milisegundos en que se obtiene los resultados.

Min: tiempo mínimo que demora un hilo en acceder a una página.

Max: tiempo máximo que demora un hilo en acceder a una página.

%Error: porciento de error de las páginas que no llegan a cargar de manera correcta.

Rendimiento: el rendimiento se mide en cantidad de solicitudes por segundo.

Kb/sec: velocidad de carga de las páginas**.**

La siguiente tabla muestra el resultado que arrojó la prueba para un total de 700, 1000 y 3000 usuarios concurrentes, donde el sistema responde en un tiempo promedio de 1.5, 1.1 y 1.0 segundos respectivamente.

TABLA 8: RESULTADOS DE LAS PRUEBAS DE CARGA Y ESTRÉS.

CANT DE HILOS	#MUESTRAS	MEDIA	MIN	MÁX	%ERROR	RENDIMIENTO	KB/S
700	**1003**	**1542**	**45**	**3122**	**0.00**	**89.14**	**1003**
1000	**2600**	**1145**	**140**	**4255**	**0.01**	**78.25**	**1500**
3000	**2800**	**1002**	**210**	**4756**	**0.27**	**67.25**	**2100**

Conclusiones del capítulo

- Con el transcurso del proceso de implementación el uso de los estándares de codificación contribuyó a tener una adecuada uniformidad y legibilidad en el código fuente.

- El diagrama de componentes permitió la posibilidad de visualizar la dependencia entre los componentes del sistema, además, reflejó de manera más detallada la ubicación física de los componentes de acuerdo a la arquitectura empleada.

- La aplicación de las pruebas de software a la solución desarrollada permitió encontrar errores que afectaban el funcionamiento del sistema, posibilitando corregirlos a tiempo para que el mismo cumpliera totalmente con los requisitos funcionales definidos en la etapa de análisis.

Conclusiones generales

Conclusiones generales:

La presente investigación tuvo como base el desarrollo de un Sistema de Evaluación Integral Estudiantil (SEIE), para ello se dio cumplimiento a una serie de objetivos específicos, los cuales fueron cumplidos satisfactoriamente, por lo que se puede arribar a las siguientes conclusiones:

- La elaboración del marco teórico-metodológico que sustenta el desarrollo y utilización de un Sistema de Evaluación Integral Estudiantil (SEIE) ratificó la necesidad de informatizar los procesos que se llevan a cabo en la Universidad de las Ciencias Informáticas (UCI).

- La obtención, descripción y validación de los requisitos definieron las características técnicas y funcionales de la propuesta de solución.

- A partir del análisis y diseño del sistema web se obtuvieron los artefactos necesarios para guiar el proceso de desarrollo del mismo.

- Las pruebas internas realizadas permitieron probar el correcto funcionamiento del sistema y los resultados satisfactorios de las mismas otorgaron validez a la presente investigación, siendo esto la base para que el sistema pueda favorecer los procesos estudiantiles que tienen lugar en la Universidad de las Ciencias Informáticas (UCI).

Recomendaciones

Para dar continuidad a la presente investigación, el autor recomienda:

- El despliegue de la aplicación en todas las universidades o sedes del país.
- La incorporación de otras funcionalidades referentes a la trayectoria estudiantil.

Referencias Bibliográficas

Referencias Bibliográficas

Universidad Politécnica de Valencia. 2016. *Introducción a Herramientas CASE y System Architect.* 2016.

Acevedo, MSc. Isis Neisy Ramos, Acosta, Dra. C. Aurelia Massip y Nazco, Dra. C. Marta Alfonso. 2017. *LA EVALUACIÓN INTEGRAL DEL ESTUDIANTE UNIVERSITARIO COMO VÍA PARA ESTIMULAR EL.* 2017.

Cabrera, Robinson Danilo Chávez. 2016. *"ANÁLISIS DEL FRAMEWORK DJANGO PARA IMPLEMENTAR APLICACIONES WEB CON BASE DE DATOS MARIADB Y METODOLOGÍA DE DESARROLLO SCRUM.* 2016.

Cabrera, Robinson Danilo Chàvez. 2016. *ANÁLISIS DEL FRAMEWORK DJANGO PARA IMPLEMENTAR APLICACIONES WEB CON BASE DE DATOS MARIADB Y METODOLOGÍA DE DESARROLLO SCRUM. .* Ecuador : s.n., 2016.

Calleja, Manuel Arias. 2015. *Estándares de codificación.* 2015.

Campoverde, Melida Rocio. 2015. *EVALUACIÓN INTEGRAL DE LOS DOCENTES UNIVERSITARIOS DESDE VARIOS PUNTOS DE VISTA.* 2015.

Cardozzo, Daniel Ramos. 2016. Desarrollo de Software: requisitos estimaciones y analisis. 2da Edicion. *Desarrollo de Software.* 2016.

Casadiego, Jorge Manuel Pacheco. 2017. *Metodologia para elaborar el Modelo Conceptual de Datos .* Colombia : s.n., 2017.

Castelao, Paula Montoto. 2008. *Introducción al Diseño con Patrones.* Universidad de Coruña : s.n., 2008.

Craig, Larman. 2003. *UML y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado.*

Cruz, Yunier Rodríguez. 2015. *Gestión de la Información y del Conocimiento para la toma de desiciones organizacionales.* 2015.

Duarte, Gloria Ponjuán. 2016. *Gestión de la Información: Presiciones Conceptuales a partir de sus orígenes.* 2016.

Referencias Bibliográficas

Enríquez, Ruíz José Luis, Flores, Flores Eder y Honores, Solano Cesar. 2017. *Metodologia de Desarrollo de Software.* 2017.

Erick Nùñez Navarrete,Darìo Rìos Navarro. 2017. *Desarrollo de un marco de trabajo (framework) para el desarrollo de aplicaciones web en la Universidad Nacionalde CostaRica.* Costa Rica : s.n., 2017.

Escalante, Lain Càrdenas. 2014. *Aplicación de Patrones de Diseño para Garantizar Alta Flexibilidad en el Software.* 2014.

Escalante, Mg.Lain Cárdenas. 2015. *Aplicación de Patrones de Diseño para Garantizar Alta Flexibilidad en el Software. .* 2015.

Flores, Isabel Gonzalez, Cruz, Cesar Manuel Cruzada de la y Miguel, Vladimir Medina. 2016. Una contribución a la gestión de la informacion de ciencia, tecnologia e innovacion. Habana, Cuba : s.n., 2016.

Francisco José García Peñalvo,Alicia García Holgado . 2017. *Fundamentos de la Vista de Casos de Uso .* Salamanca,España : s.n., 2017.

Guerrero, C. A., y Johanna M. Suárez, L. E. 2013. *Patrones de Diseño GOF (The Gang of Four) en el contexto de Procesos de Desarrollo de Aplicaciones Orientadas a la Web.* s.l. : Informacion tecnologica, 2013.

Hernández, Coromoto León. 2006. *Herramientas y Lenguajes de Programación.* 2006.

Informáticas, Universidad de la Ciencias UCI. 2015. *Metodologia de desarrollo para la actividad productiva de la UCI.* 2015.

Laguna, Adrián Leyva. 2018. *PROCEDIMIENTO CON ENFOQUE MULTICRITERIO PARA LA EVALUACIÓN INTEGRALIDAD EMULATIVAS DE LA FEU.* Holguin Cuba : 2018.

Larman., Craig. 2003. *Modelo del Dominio.* s.l. : Prentice Hall, 2003.

Lianet, Cabrera González y Roberto, Pompa Torres Enrique. 2012. *Extension de Visual Paradigm for UML para el desarrollo dirigido por modelos de aplicaciones de gestión de información.* 2012.

Martelo, Raúl J., Madera, Jhonny E. y Betín, Andrés D. 2015. Scielo. *Software para Gestión Documental, un Componente Modular del Sistema de Gestión de Seguridad de la Información (SGSI).* [En línea] 2015. https://scielo.conicyt.cl/scielo.php?pid=S0718-07642015000200015&script=sci_arttext&tlng=en.

Martinez y Imelda. 2015. *Ponderación de requisitos de software usando técnicas cognitivas y orientación por objetivos.*

Referencias Bibliográficas

Millàn, Martha Elena. 2016. *Fundamentos de Bases de Datos .* Madrid : s.n., 2016.

Patricia, López y Francisco, Ruiz. 2015. *Lenguaje Unificado de Modelado - UML.* 2015.

Paz, Juan Andres Mera. 2015. *Análisis del proceso de pruebas.* 2015.

Paz, Julián Andrés Mera. 2015. *Análisis del Proceso de Pruebas de Calidad de Software.* 2015.

Peño, Jose Manuel Sanchez. 2015. *Pruebas de Software.Fundamentos y Tecnicas .* 2015.

Pérez, Fredy Vázquez. 2017. *La formación integral del estudiante universitario.* Mexico : s.n., 2017.

Pérez, Mtro. Fredy Vázquez. 2016. *Comprehensive training college student: the case of the UNACH.* 2016.

Pressman, Roger S. 2010. *Ingenieria del Software.* 2010.

Python, Escuela de. 2019. jetbrains. *Welcome to PyCharm help.* [En línea] 2019. https://www.jetbrains.com/help/pycharm/2017.1/pycharm-2017.1-help.pdf.

Revista de Información, Tecnología y Sociedad . 2015. *Phython – DjangoFramework de desarrollo web para perfeccionistasBasado en el Modelo MTV.* España : s.n., 2015.

Rivera, Alcides Neptalí. 2010. *Estaudio de la Arquitectura de Software.* 2010.

Rodríguez, Miguel Ángel Palacios. 2015. *Gestión de Información como eje transversal para el éxito de las organizaciones.* 2015.

Rolando, Molina Ríos Jimmy, Nancy, Loja Mora y Paola, Zea Ordóñez. 2016. *Evaluación en los Frameworks en el desarrollo de Aplicaciones Web con Python.* 2016.

Rolando, Molina Ríos Jimmy, Paola, Zea Ordóñez y Fabían, Castillo Fausto. 2017. *Administración de Bases de Datos con PostgreSQL.* 2017.

Ruiz, Jhones Alina y Vidal, Larramendi Julio. 2018. Las TIC en la Gestion universitaria cubana: barreras, principios y acciones. Habana, Cuba : s.n., 2018.

Sánchez, Tamara Rodríguez. 2014. *Metodología de desarrollo para la Actividad productiva de la UCI.* 2014.

Sommerville, Ian. 2015. *Requerimientos del software.* 2015.

Suárez, Yuniel. 2016. Doofy's Blog. *Qué es un IDE – DeProgramación.* [En línea] 2016.

Referencias Bibliográficas

Torres, Patricio Letelier. 2002. *Desarrollo de Software Orientado a Objeto usando UML.* Departamento Sistemas Informáticos y Computación (DSIC) Universidad Politécnica de Valencia (UPV) - España : s.n., 2002.

Visconti, Marcello y Astudillo, Hernan. 2012. *Fundamentos de Ingeniería de Software.* 2012.

ANEXOS

Anexos

1. Historias de Usuario (HU)

Las restantes Historias de Usuario (HU) se encuentran en la carpeta del proyecto.

Historia de Usuario	
Número: HU_2	**Usuario:** Estudiante
Nombre de historia: Búsqueda avanzada.	
Prioridad en negocio: media	**Riesgo en desarrollo:** bajo
Programador Responsable: Alcides Rodríguez Rodríguez.	
Descripción: el usuario realiza la búsqueda de los criterios de las actividades en las diferentes dimensiones educativas	

Historia de Usuario	
Número: HU_3	**Usuario:** Administrador y Estudiante
Nombre de historia: Gestionar actividad en residencia	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Programador Responsable: Alcides Rodríguez Rodríguez.	

ANEXOS

Descripción: la funcionalidad gestionar actividad en residencia debe permitir adicionar , mostrar, modificar y eliminar actividades en el sistema correspondiente a la residencia y luego será seleccionada por el estudiante antes de emitir su evaluación.

2. Cuestionario para evaluar la satisfacción de SEIE

Estimado estudiante:

A fin de mejorar el servicio que brinda la herramienta SEIE ponemos a tu disposición una encuesta anónima para conocer tus opiniones y criterios. Nos será de gran ayuda y utilidad contar con tu transparencia y claridad en las respuestas

1. ¿De qué Facultad eres?

a. Facultad 1.

b. Facultad 2.

c. Facultad 3.

d. Facultad 4.

f. Facultad de Ciencias y Tecnologías Computacionales.

2. ¿Qué año cursas?

a. Primero.

b. Segundo.

c. Tercero.

d. Cuarto.

e. Quinto.

3. ¿Eres dirigente estudiantil?

a. Sí.

b. No.

ANEXOS

4. ¿Con qué frecuencia entras al sistema?

a. Todos los días.

b. Una vez por semana.

c. Regularmente.

d. Casi nunca.

e. Nunca.

5. ¿Cómo ves a SEIE?

	Si	No	Tengo mis dudas
Ayuda al control y seguimiento de la trayectoria estudiantil.			
Es un mecanismo para obligarte a ir a las actividades.			
Permite evaluar cómo marcho durante el curso, además de fijarme metas altas.			
El perfil de los estudiantes debe estar público para que todos lo vean			
Todas las actividades por mínimas que sean deben registrarse sin excepción.			

Es un sistema confiable y veraz en la información.			

6. ¿Qué otras funcionalidades crees que debería brindar SEIE?

7. De los criterios de medida que se muestran debajo, seleccione un valor de 1 (Baja/Mala) a 5 (Alta/Excelente)

	1	2	3	4	5
Interfaz gráfica					
Estructuración de las áreas de contenido. (¿Pasas trabajo para encontrar la información?)					
Velocidad en la navegación.					
Usabilidad					
Complejidad (¿Es fácil de utilizar?).					

8. ¿Crees que el sistema debe ser más interactivo, enfocado a una red social donde puedas emitir criterios sobre tus compañeros, actividades y demás procesos de la organización y su funcionamiento?

a. Sí.
b. No.
c. Perdería la esencia.
d. No sé, tengo mis dudas

9. Alguna sugerencia o criterio que quieras transmitir.

3. Indicadores a medir para evaluaciones de integralidad

Investigación

ANEXOS

(5 puntos):

- ✓ Participación en al menos 2 eventos no colaterales en el curso a nivel de universidad.
- ✓ Tener al menos 1 premio en eventos a nivel de Universidad o tener al menos 1 premio en eventos a nivel Nacional o Internacional o tener una publicación científica.

(4 puntos):

- ✓ Participación en al menos 3 eventos no colateral en el curso a nivel de facultad.
- ✓ Tener 1 resultados en eventos a nivel de facultad.
- ✓ Participación en al menos 1 evento a nivel de universidad o tener una publicación científica

(3 puntos):

- ✓ Participación en al menos 3 evento no colateral en el curso a nivel de Facultad
- ✓ Tener al menos 1 resultados en eventos a nivel de facultad

(2 puntos):

Debe cumplirse como mínimo 1 de los siguientes aspectos.

- ✓ Participación en al menos 2 eventos a nivel de facultad o tener 1 resultados evento colateral a nivel de Universidad
- ✓ Participación en 1 evento no colateral a nivel Facultad y 2 eventos colaterales a cualquier nivel

(1 punto):

- ✓ Participación en al menos 2 eventos colaterales a cualquier nivel

(0 punto):

- ✓ No cumplir ninguna de las anteriores.

Nota: La participación en evento no puede ser ni como organizador ni en actividad colateral.

Se definen como eventos investigativos:

Fórum de Historia

Seminario Juvenil Martiano

ANEXOS

Mi Web x Cuba

Jornada Científica Estudiantil

Exposición de trabajos de Ingeniería de Software

Peña Tecnológica

Copa Pascal Olimpiada de Matemática

Concurso de Base de Datos

Cultura

(5 puntos):

Debe cumplirse como mínimo 1 de los siguientes aspectos.

- ✓ Haber participado en el Festival de Artistas Aficionados y obtener al menos 1 resultado en el mismo.
- ✓ Haber participado activamente en la 1era y 2da parte del FAA.

(4 puntos):

- ✓ Haber participado en el Festival de Artistas Aficionados. (Manifestaciones)

De no haber participado en el FAA para obtener la calificación debe cumplir con al menos 2 de las siguientes condiciones:

Haber participado en las Peñas que se realizan en la Universidad (Música, Danza, Teatro, Literatura, etc.) Haber apoyado sistemáticamente a la realización del Festival (escenografía, actividades colaterales, apoyo a los artistas).

Ser un artista muy reconocido en la Facultad por su participación en las actividades de la misma (vespertinos, inauguraciones, clausuras, peñas, etc.)

(3 puntos):

Debe cumplirse como mínimo 1 de los siguientes aspectos.

- ✓ Haber participado en las Peñas en la Universidad

- ✓ Haber apoyado a la realización de los Festivales (escenografía, actividades colaterales, apoyo a los artistas).

(2 puntos):

- ✓ Haber participado en alguna actividad de la facultad como artista.

(1 punto):

- ✓ Haber participado en alguna actividad de la facultad.

(0 punto):

No cumplir ninguna de las anteriores.

Deporte

(5 puntos):

Debe cumplirse como mínimo 2 de los siguientes aspectos.

- ✓ Haber participado como atleta en los Juegos Mella.
- ✓ Haber obtenido medallas (oro, plata o bronce) en los Juegos Mella.
- ✓ Haber participado al menos uno de los eventos convocados por el proyecto Marabana, (MaraCuba, Marabana, Cacahual, Maratón Terry Fox, Maratón 10 de octubre, Maratón por el Día Olímpico, etc.).
- ✓ Haber participado en los Juegos Provinciales Giraldillos.
- ✓ Haber participado en los Juegos Nacionales (Universidades).

(4 puntos):

Debe cumplirse como mínimo 2 de los siguientes aspectos.

- ✓ Haber participado como atleta en los Juegos Mella.
- ✓ Haber participado en al menos 1 Copa UCI y en al menos 1 Copa Interna.
- ✓ Haber participado al menos uno de los eventos convocados por el proyecto Marabana, (MaraCuba, Marabana, Cacahual, Maratón Terry Fox, Maratón 10 de octubre, Maratón por el Día Olímpico, etc.).

(3 puntos):

Si se cumple como mínimo 2 de los siguientes aspectos

- ✓ Haber participado como atleta, activista u organizador en los Juegos Mella.
- ✓ Haber participado como atleta, activista u organizador en los Juegos Inter-Años.
- ✓ Haber participado en al menos 1 evento convocados por el proyecto Marabana, (MaraCuba, Marabana, Cacahual, Maratón Terry Fox, Maratón 10 de octubre, Maratón por el Día Olímpico).
- ✓ Haber participado en al menos 1 Copa UCI o Copa Interna.

(2 puntos):

- ✓ Haber participado como atleta, activista u organizador en los Juegos Mella.

(1 punto):

- ✓ Haber participado como atleta, activista u organizador en los Juegos Inter-Años.

(0 punto):

- ✓ No cumplir ninguna de las anteriores.

Residencia

Nota: Invalida la posibilidad de obtener 4 o 5 puntos el hecho de tener alguna sanción relacionada con indisciplinas cometidas en el área de le residencia.

(5 puntos):

- ✓ El estudiante tiene que haber sido evaluado de B todos los meses y tiene que tener una trayectoria marcada por ser muy activo en los distintos procesos y actividades (Mi Beca + Bonita, Apartamentos Modelos, Apartamentos especializados, Paradas de Beca, otras) que ha desarrollado la organización en la residencia estudiantil.

(4 puntos):

- ✓ El estudiante tiene que haber sido evaluado de B todos los meses.

(3 puntos):

- ✓ El estudiante puede tener hasta 4 evaluaciones de R y tiene que ser muy activo en los distintos procesos y actividades que ha desarrollado la organización en la residencia estudiantil.

(2 puntos):

- ✓ En las evaluaciones mensuales del estudiante predominan las evaluaciones de R.

(1 punto):

- ✓ En las evaluaciones mensuales el estudiante puede tener hasta 3 evaluaciones de M.

Otras combinaciones:

- ✓ Estudiante sancionado con trayectoria de R tiene un 1 punto y si es muy activo en los distintos procesos y actividades que ha desarrollado la organización en la residencia estudiantil entonces obtiene 2 Puntos.
- ✓ Estudiante sancionado con trayectoria de B o E tiene 2 puntos y si es muy activo en los distintos procesos y actividades que ha desarrollado la organización en la residencia estudiantil entonces obtiene 3 Puntos.

TSU

(5 puntos):

- ✓ Obtener B en el TSU.

(3 puntos):

- ✓ Obtener R en el TSU.

(0 puntos):

- ✓ Obtener M en el TSU.

Guardia estudiantil

ANEXOS

(5 puntos):

- ✓ No tener más de 2 ausencias justificadas a la guardia.
- ✓ No tener ausencias injustificadas. (Se considera injustificada toda aquella ausencia que no se informen las causas previamente al momento de la guardia.)

(4 puntos):

- ✓ No tener más de 3 ausencias justificadas a la guardia.
- ✓ No tener ausencias injustificadas. (Se considera injustificada toda aquella ausencia que no se informen las causas previamente al momento de la guardia.)

(3 puntos):

- ✓ No tener más de 4 ausencias justificadas a la guardia.
- ✓ No tener más de 1 ausencias injustificadas. (Se considera injustificada toda aquella ausencia que no se informen las causas previamente al momento de la guardia.)

(2 puntos):

- ✓ No tener más de 5 ausencias justificadas a la guardia.
- ✓ No tener más de 2 ausencias injustificadas. (Se considera injustificada toda aquella ausencia que no se informen las causas previamente al momento de la guardia.)

(1 punto):

- ✓ No tener más de 6 ausencias justificada a la guardia.
- ✓ No tener más de 3 ausencias injustificadas. (Se considera injustificada toda aquella ausencia que no se informen las causas previamente al momento de la guardia.)

(0 punto):

- ✓ No cumplir con ninguna de las anteriores.

FEU-UJC

(5 puntos):

Deben cumplirse al menos 1 de los siguientes elementos.

- ✓ Haber ocupado responsabilidades en el Consejo de la FEU o Comité Primario de la UJC en la Universidad, no evaluado de M.
- ✓ Haber ocupado responsabilidades en los Secretariados o el Comité Primario de las Facultades, evaluado de B.
- ✓ Haber obtenido la distinción 13 de marzo para dirigentes estudiantiles o Destacado como dirigente estudiantil en la UJC.
- ✓ Haber obtenido la distinción “Joven XX aniversario de la FEU” o “Joven XX aniversario de la UJC”.

(4 puntos).

Deben cumplirse al menos 1 de los siguientes elementos.

- ✓ Haber ocupado responsabilidades en los Secretariados o el Comité Primario de las Facultades, no evaluado de M.
- ✓ Haber ocupado responsabilidades en los Consejo de la FEU de las Facultades o como Secretario de C/B, evaluado de B.
- ✓ Tener reconocimientos que avalen el desempeño dentro de las organizaciones. (Destacado en las actividades convocadas, etc.)

(3 puntos):

Deben cumplirse al menos 1 de los siguientes elementos:

- ✓ Haber ocupado responsabilidades en los Consejo de la FEU de las Facultades o como Secretario de C/B. (Aquí se incluyen los grupos de trabajo), no evaluado de M.
- ✓ Tener al menos 1 reconocimiento que avalen el desempeño dentro de las organizaciones.
- ✓ Ser donante de sangre.
- ✓ Haber participado en al menos 3 ocasiones como organizador de eventos y/o actividades de la organización. (Paradas de BK, vespertinos o matutinos, Comedor Modelo, Eventos FEU (FAA, JICI,

JCE, Fórum de Historia, Juegos Mella etc.), Inauguraciones o Clausuras, Proyectos Socioculturales, entre otros)

(2 puntos):

Deben cumplirse al menos 1 de los siguientes elementos:

- ✓ Haber participado en al menos 2 ocasiones como organizador de eventos y/o actividades de la organización. (Paradas de BK, vespertinos o matutinos, Comedor Modelo, Eventos FEU (FAA, JICI, JCE, Fórum de Historia, etc.), Inauguraciones o Clausuras, Proyectos Socioculturales, entre otros).
- ✓ No tener ausencias a las reuniones de brigada.

(1 punto):

Deben cumplirse con al menos 1 de los elementos:

- ✓ Haber participado en al menos 1 ocasión como organizador de eventos y/o actividades de la organización. (Paradas de BK, vespertinos o matutinos, Comedor Modelo, Eventos FEU (FAA, JICI, JCE, Fórum de Historia, etc.), Inauguraciones o Clausuras, Proyectos Socioculturales, entre otros).
- ✓ No tener más de 2 ausencias a las reuniones de brigada.

(0 punto):

- ✓ No cumplir con ninguna de las pautas anteriores.

Producción

(5 puntos):

Deben cumplirse al menos 2 de los siguientes elementos:

- ✓ Participación en los Activos de Producción o Asambleas de Producción.
- ✓ Participación en el proyecto, con resultados satisfactorios (evaluado de 5 puntos)
- ✓ Participación en al menos un despliegue con resultados satisfactorios.
- ✓ Participación en un evento internacional exponiendo o como implementador de algún resultado en la producción. - Participar como expositor, organizador o desarrollador en la FESI.

- ✓ Ser reconocido como Destacado en la Producción.
- ✓ Haber validado el desempeño de un rol en cualquier especialidad de la Producción mediante el Concurso de Certificación de Roles.

(4 puntos):

Deben cumplirse al menos 2 de los siguientes elementos:

- ✓ Participación en el proyecto, con resultados satisfactorios (evaluado de 4 puntos)
- ✓ Participación en los Activos de Producción o Asambleas de Producción.
- ✓ Participar como expositor, organizador o desarrollador en la FESI.

(3 puntos):

Deben cumplirse al menos 2 de los siguientes elementos.

- ✓ Participación en el proyecto, con resultados satisfactorios (evaluado de 3 puntos)
- ✓ Participación en los Activos de Producción o Asambleas de Producción.
- ✓ Participar como expositor, organizador o desarrollador en la FESI.

(2 puntos):

Deben cumplirse al menos 1 de los siguientes elementos.

- ✓ Participación en el proyecto, con resultados satisfactorios (evaluado de 3 puntos)
- ✓ Participación en los Activos de Producción o Asambleas de Producción

(1 punto):

- ✓ Participación en el proyecto, con resultados satisfactorios (evaluado de 3 puntos)

(0 punto):

- ✓ No cumplir ninguna de las anteriores.

<u>Evaluación Final por rango de puntos (1ro)</u>

ANEXOS

1. Para obtener una evaluación de Bien (B) debe tener un rango de puntos entre 19 y 30.

2. Para obtener una evaluación de Regular (R) debe tener un rango de puntos entre 4 y 18.

3. Para obtener una evaluación de Mal (M) debe tener un rango de puntos entre 0 y 3.

Evaluación Final por rango de puntos (2do)

4. Para obtener una evaluación de Bien (B) debe tener un rango de puntos entre 24 y 35.

5. Para obtener una evaluación de Regular (R) debe tener un rango de puntos entre 9 y 23.

6. Para obtener una evaluación de Mal (M) debe tener un rango de puntos entre 0 y 8.

Evaluación Final por rango de puntos (3ro y 4to)

1. Para obtener una evaluación de Bien (B) debe tener un rango de puntos entre 29 y 40.

2. Para obtener una evaluación de Regular (R) debe tener un rango de puntos entre 14 y 28.

3. Para obtener una evaluación de Mal (M) debe tener un rango de puntos entre 0 y13.