



FACULTAD 4

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

**Título: Sistema de gestión para el apoyo de
la integralidad de los estudiantes de 5to
año**

Autor: Asael Caraballo Oquendo

Javier Mancha Cabrera

Tutor: M. Sc. Yordankis Matos López

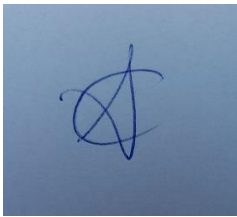
La Habana, 7 de diciembre de 2022

“Año 64 de la Revolución”

DECLARACIÓN DE AUTORÍA

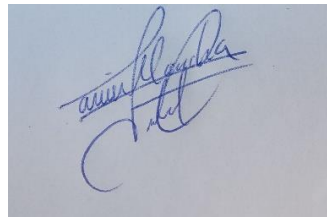
Declaramos ser autores de la presente tesis que tiene por título: Sistema de gestión para el apoyo de la integralidad de los estudiantes de 5to año y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmo la presente a los 7 días del mes de diciembre del año 2022.

Asael Caraballo Oquendo



Firma del Autor

Javier Mancha Cabrera



Firma del Autor

Yordankis Matos López



Firma del Tutor

Agradecimientos

Javier Mancha Cabrera:

Agradezco a mi familia:

- Como no podría ser de otra forma, la primera persona en mencionar tiene que ser mi madre Beatriz, sin cuyo apoyo incondicional y amor infinito yo no estaría aquí ahora. Agradecer siempre se quedará corto contigo.

- A mi padre Javier, por enseñarme que el límite de lo que puedas hacer lo pones tú mismo.

- A mis abuelos Miguel Ángel y Neyda Lidia, por su paciencia oportuna y ser siempre los mejores maestros de vida y escuela.

- A mi hermana Yisel, tu férreo carácter me enseñó a valerme por mí mismo desde temprana edad. Gracias por preocuparte tanto por mí cuando debo ser yo quien te cuide.

-A mi esposa Brenda, eres mi estrella guía, la claridad de mi mente y mi razón, la única que necesito. Gracias por tu valentía, tu honestidad y por enseñarme que puedo ser verdaderamente feliz. Te amo.

Agradezco en general a mi familia, todos en algún momento han sido una pieza importante de mi crecimiento como persona, y me han apoyado cuando lo necesité.

Agradezco profundamente, además:

- A mi mejor amiga Daniela, gracias por ser un pilar en mi vida y recordarme en mis más oscuros momentos quien soy.

- A mi amigo Lemsoy, por ser el hermano mayor tardío con el que tanto disfruto compartir.

- A mis amistades de la vocacional, El Rubio, Orta, Carlos, Kinki y muchos otros. El tiempo que pasé con ustedes lo atesoraré siempre.

- A Lía, Roberto y Asael. Por darme una familia aquí cuando no tenía ninguna. A este último además por ser capaz de entenderme tan bien, lo que hizo trabajar juntos en una tesis un reto posible.

- A mi amigo de la infancia Andy, por todas las risas, los videojuegos compartidos y en general ser unos incomprendidos juntos.

- A mis amigos de la infancia Omarito y Dayron, porque todo niño necesita cicatrices en las rodillas.

- A todos los profesores que han contribuido a mi formación en este centro de altos estudios, pero en especial a la profe Yadira, por intentar con tanta fuerza hacer de nosotros hombres y mujeres de provecho (trabajo aún en proceso) y a nuestro tutor Yordankis por su guía y paciencia.

Siempre se me quedarán personas que merecen estar aquí, mi memoria sigue siendo... lo que siempre ha sido, un desastre. Todos los que han llegado a conocerme y apreciarme lo saben, gracias por lidiar con eso. Gracias a todos por hacer de mi vida una mejor experiencia.

Asael Caraballo Oquendo:

Quisiera agradecer primeramente a toda familia si tengo que mencionar en especial pues sería a todos, gracias por su apoyo constante a lo largo de mi carrera, a su sabiduría y enseñanzas para la vida y progresar como persona. A mis profesores por su influencia en mi desarrollo como futuro ingeniero. Agradecer principalmente a mis amigos Juan José Gonzales Salas, Arturo Bolívar Labaut del Rio, Liany Sobrino Miranda, Luis Ernesto Clarke Samuels, Sergio Daniel Ortiz Jova, Rolando Pérez Domínguez, Samira de las Mercedes Enríquez González, son demasiados XD, y por supuesto, mis compañeros de cuarto Roberto Reyes Martínez y Javier Mancha Cabrera, este último con el que más he convivido a lo largo de estos 5 años de la carrera, a mis amigos de antes de la universidad Hoffman Noda Quintana, Roxana Martínez Abreu, María de Jesús Ríos García, a mis compañeros de turbeo intenso hasta altas horas de la madrugada, todos sois los mejores. Por último, y no los menos importantes, no podrían faltar por su ayuda en la recta final de mi carrera, mi tutor Yordankis Matos López y

nuestra profesora Yadira Ramírez Rodríguez por estar tan pendiente de sus estudiantes. A todos los que están y los que faltan, muchas gracias.

Resumen

Para toda universidad es de vital importancia la integralidad y evaluación de sus estudiantes. Si este proceso es asistido por un sistema informático tomando en cuenta los nuevos avances de las Tecnologías de la Información y las Comunicaciones se garantiza un impacto positivo provocando que la información generada esté más organizada y disponible. La investigación, cuyos resultados se describen en este informe, tuvo como propósito desarrollar un sistema de gestión para el apoyo de la integralidad de los estudiantes de 5to año, que permitió aumentar la disponibilidad y eficiencia de este proceso a sus usuarios. El desarrollo del mismo fue guiado por la metodología de desarrollo de software XP. Para la implementación de este se utilizó como marco de trabajo web Django con su lenguaje de programación Python. Para validar que la solución cumpliera con los requisitos definidos por el cliente, se aplicaron pruebas unitarias y pruebas de aceptación.

Palabras clave: evaluación, integralidad, estudiantes

Abstract

For every university it is of vital importance the integrality and evaluation of its students. If this process is assisted by a computer system considering the new advances in Information and Communication Technologies, a positive impact is guaranteed, causing the information generated to be more organized and available. The purpose of the research, whose results are described in this report, was to develop a management system for the support of the 5th year students' integrality, which allowed increasing the availability and efficiency of this process to its users. Its development was guided by the XP software development methodology. For its implementation, Django with its Python programming language was used as a web framework. To validate that the solution complied with the requirements defined by the client, unit tests and acceptance tests were applied.

Keywords: evaluation, integrality, students

CONTENIDO

Introducción	1
1 Capítulo I. Estudio del estado del arte	4
1.1 Conceptos asociados al tema.....	4
1.2 Análisis de sistemas homólogos	5
1.3 Metodología para el desarrollo del software	7
1.4 Herramientas y tecnologías	11
1.4.1 Lenguaje de programación	11
1.4.2 Marco de trabajo para el desarrollo de la solución informática	14
1.4.3 Herramienta CASE	19
1.4.4 Lenguaje de modelado	21
1.4.5 Entorno de desarrollo integrado	23
1.4.6 Gestor de base de datos	25
1.5 Conclusiones parciales.....	28
2 Capítulo II. Propuesta de solución	29
2.1 Requisitos funcionales.....	29
2.2 Requisitos no funcionales	36
2.3 Historias de usuario.....	37
2.4 Estimación de tiempo por historia de usuario	40
2.5 Plan de iteraciones.....	42
2.6 Plan de entrega.....	49
2.7 Tarjetas CRC.....	50
2.8 Arquitectura del sistema	51
2.9 Diseño de la base datos del sistema.....	52
2.10 Patrones de diseño.....	53
2.11 Prototipo de interfaz de usuario	55
2.12 Conclusiones parciales.....	56
3 Capítulo III. Implementación y Pruebas	57

3.1	Implementación.....	57
3.1.1	Diagrama de despliegue.....	57
3.1.2	Estándares de codificación	58
3.2	Estrategia de pruebas.....	59
3.2.1	Pruebas unitarias.....	60
3.2.2	Pruebas de aceptación	67
3.3	Resultados de las pruebas.....	72
3.4	Conclusiones parciales.....	75
4	Conclusiones.....	76
5	Recomendaciones.....	77
6	Referencias bibliográficas	78
7	Anexos.....	81

Índice de ilustraciones

ILUSTRACIÓN 1	MATRIZ DE TRAZABILIDAD	40
ILUSTRACIÓN 2	DIAGRAMA ENTIDAD-RELACIÓN	53
ILUSTRACIÓN 3	INTERFAZ DE ACCESO	55
ILUSTRACIÓN 4	INTERFAZ PROTOTIPO DE MENÚ PRINCIPAL.....	56
ILUSTRACIÓN 5	DIAGRAMA DE DESPLIEGUE	57
ILUSTRACIÓN 6	RESULTADOS PRUEBAS UNITARIAS	67

Índice de tablas

TABLA 1HU1.....	38
TABLA 2HU2.....	38
TABLA 3HU3.....	39
TABLA 4 ESTIMACIÓN DE TIEMPO.....	41
TABLA 5 PLAN DE ITERACIONES	42
TABLA 6 TAREA DE INGENIERÍA 1	45
TABLA 7 TAREA DE INGENIERÍA 2	45
TABLA 8 TAREA DE INGENIERÍA 3	46
TABLA 9 TAREA DE INGENIERÍA 8	46
TABLA 10 TAREA DE INGENIERÍA 9.....	47

TABLA 11 TAREA DE INGENIERÍA 10.....	47
TABLA 12 TAREA DE INGENIERÍA 13.....	48
TABLA 13 TAREA DE INGENIERÍA 14.....	48
TABLA 14 TAREA DE INGENIERÍA 15.....	49
TABLA 15 PLAN DE ENTREGA.....	49
TABLA 16 TARJETA CRC 1.....	50
TABLA 17 TARJETA CRC 2.....	50
TABLA 18 TARJETA CRC 3.....	51
TABLA 19 CASO DE PRUEBA DE ACEPTACIÓN 1	68
TABLA 20 CASO DE PRUEBA DE ACEPTACIÓN 2	68
TABLA 21 CASO DE PRUEBA DE ACEPTACIÓN 3	69
TABLA 22 CASO DE PRUEBA DE ACEPTACIÓN 4	69
TABLA 23 CASO DE PRUEBA DE ACEPTACIÓN 5	70
TABLA 24 CASO DE PRUEBA DE ACEPTACIÓN 6	70
TABLA 25 CASO DE PRUEBA DE ACEPTACIÓN 7	70
TABLA 26 CASO DE PRUEBA DE ACEPTACIÓN 8	71
TABLA 27 CASO DE PRUEBA DE ACEPTACIÓN 9	71
TABLA 28 RESULTADOS DE LAS PRUEBAS	72
TABLA 29 GRÁFICO DE PRUEBAS	73
TABLA 30 HU4	81
TABLA 31 HU5	81
TABLA 32 HU6	82
TABLA 33 HU7	82
TABLA 34 HU8	83
TABLA 35 HU9	83
TABLA 36 HU10	84
TABLA 37 HU11	84
TABLA 38 HU12.....	85
TABLA 39 HU13.....	86
TABLA 40 HU14.....	86
TABLA 41 HU15.....	86
TABLA 42 HU16.....	87
TABLA 43 HU17.....	88
TABLA 44 HU18.....	88
TABLA 45 HU19.....	88
TABLA 46 HU20.....	89
TABLA 47 HU21.....	89
TABLA 48 HU22.....	90
TABLA 49 HU23.....	90
TABLA 50 HU24.....	91
TABLA 51 HU25.....	91
TABLA 52 HU26.....	92
TABLA 53 HU27.....	92

TABLA 54 HU28	93
TABLA 55 HU29	93
TABLA 56 HU30	94
TABLA 57 HU31	95
TABLA 58 HU32	95
TABLA 59 HU33	96
TABLA 60 TARJETA CRC 4.....	96
TABLA 61 TARJETA CRC 5.....	96
TABLA 62 TARJETA CRC 6.....	97
TABLA 63 TARJETA CRC 7.....	97
TABLA 64 TARJETA CRC 8.....	97
TABLA 65 TARJETA CRC 9.....	98
TABLA 66 CP ACEPTACIÓN 10	98
TABLA 67 CP ACEPTACIÓN 11	98
TABLA 68 CP ACEPTACIÓN 12	99
TABLA 69 CP ACEPTACIÓN 13	99
TABLA 70 CP ACEPTACIÓN 14	99
TABLA 71 CP ACEPTACIÓN 15	100
TABLA 72 CP ACEPTACIÓN 16	100
TABLA 73 CP ACEPTACIÓN 17	101
TABLA 74 CP ACEPTACIÓN 18	101
TABLA 75 CP ACEPTACIÓN 19	101
TABLA 76 CP ACEPTACIÓN 20	102
TABLA 77 CP ACEPTACIÓN 21	102
TABLA 78 CP ACEPTACIÓN 22	103
TABLA 79 CP ACEPTACIÓN 23	103
TABLA 80 CP ACEPTACIÓN 24	103
TABLA 81 CP ACEPTACIÓN 25	104
TABLA 82 CP ACEPTACIÓN 26	104
TABLA 83 CP ACEPTACIÓN 27	104
TABLA 84 CP ACEPTACIÓN 28	105
TABLA 85 CP ACEPTACIÓN 29	105
TABLA 86 CP ACEPTACIÓN 30	106
TABLA 87 CP ACEPTACIÓN 31	106
TABLA 88 CP ACEPTACIÓN 32	106
TABLA 89 CP ACEPTACIÓN 33	107
TABLA 90 CP ACEPTACIÓN 34	107
TABLA 91 CP ACEPTACIÓN 35	108
TABLA 92 CP ACEPTACIÓN 36	108
TABLA 93 CP ACEPTACIÓN 37	108
TABLA 94 CP ACEPTACIÓN 39	109
TABLA 95 CP ACEPTACIÓN 40	109
TABLA 96 CP ACEPTACIÓN 41	110

TABLA 97 CP ACEPTACIÓN 42	110
TABLA 98 TAREA DE INGENIERÍA 4	110
TABLA 99 TAREA DE INGENIERÍA 5	111
TABLA 100 TAREA DE INGENIERÍA 6	111
TABLA 101 TAREA DE INGENIERÍA 7	111
TABLA 102 TAREA DE INGENIERÍA 11	112
TABLA 103 TAREA DE INGENIERÍA 12	112
TABLA 104 TAREA DE INGENIERÍA 16	112
TABLA 105 TAREA DE INGENIERÍA 17	113
TABLA 106 TAREA DE INGENIERÍA 18	113
TABLA 107 TAREA DE INGENIERÍA 19	113
TABLA 108 TAREA DE INGENIERÍA 20	114
TABLA 109 TAREA DE INGENIERÍA 21	114
TABLA 110 TAREA DE INGENIERÍA 22	114
TABLA 111 TAREA DE INGENIERÍA 23	115
TABLA 112 TAREA DE INGENIERÍA 24	115
TABLA 113 TAREA DE INGENIERÍA 25	115
TABLA 114 TAREA DE INGENIERÍA 26	116
TABLA 115 TABLA 115 TAREA DE INGENIERÍA 27	116
TABLA 116 TABLA 115 TAREA DE INGENIERÍA 28	116
TABLA 117 TABLA 115 TAREA DE INGENIERÍA 29	117
TABLA 118 TABLA 115 TAREA DE INGENIERÍA 30	117
TABLA 119 TABLA 115 TAREA DE INGENIERÍA 31	118
TABLA 120 TABLA 115 TAREA DE INGENIERÍA 32	118
TABLA 121 TABLA 115 TAREA DE INGENIERÍA 33	118

INTRODUCCIÓN

El término integral hace alusión a la idea de totalidad. Así, la educación integral se entendería como el desarrollo perfectivo del ser humano completo, en todas y cada una de sus dimensiones (física, intelectual, social, moral, religiosa...). En este mismo sentido, el profesor Gervilla alude al concepto educación integral, relacionada con el concepto de totalidad: "la educación del hombre completo, de todas y cada una de sus facultades y dimensiones". (Álvarez)

La formación integral del estudiante es el objetivo central del proceso docente educativo que se desarrolla en la educación superior. Por lo que la evaluación sistemática durante la educación es de vital importancia para el desarrollo de los modos de actuación con la pertinencia deseada en los futuros egresados. Con la aparición de los sistemas para gestionar la información se ha contribuido a que se gestionen mejor los procesos y que la información que se genere esté más organizada y disponible.

En la Universidad de las Ciencias Informáticas se hace necesaria la informatización de este proceso que se desarrolla para así poder mantener un mayor control sobre los documentos y poder tomar ciertas decisiones en el menor tiempo posible, debido a que los estudiantes tienen que esperar por el profesor y este a su vez esperar por la confección de la planilla y que le sea entregada por secretaria docente por vía correo, luego este se la envía a sus estudiantes los cuales tienen que llenar esta planilla y enviársela a su profesor. El profesor imprime las planillas se firman por una comisión y después son regresadas a secretaría docente donde se archivan; como consecuencia en ocasiones la información no se entrega en tiempo y con la calidad que requiere este tipo de proceso; los estudiantes tienen que mostrar evidencias en las actividades que han participado, muchas veces se pierde la información o esta información nunca llega al profesor porque el estudiante olvida mandar la planilla y se

crea un retraso cuando los estudiantes necesitan modificar la planilla porque omitieron u olvidaron algún evento.

La situación planteada anteriormente permitió identificar la siguiente problemática:

- ¿Cómo mejorar el proceso de la evaluación integral estudiantil de la Universidad de las Ciencias Informáticas?

Se identifica como **objeto de estudio**: los sistemas informáticos para el proceso de evaluación integral de estudiantes.

Campo de acción: los sistemas informáticos para el proceso de evaluación integral de estudiantes de 5to año en la Universidad de las Ciencias Informáticas.

Este problema se enmarca en el objetivo de: desarrollar una aplicación web que permita al profesorado gestionar la integralidad de los estudiantes en la Universidad de las Ciencias Informáticas.

Durante el desarrollo de este trabajo se trazan las siguientes **tareas de investigación** para darle cumplimiento al objetivo:

- ✓ Análisis de sistemas homólogos.
- ✓ Determinación de las herramientas y la metodología adecuada para el desarrollo de una aplicación que mejore el proceso de la integralidad estudiantil.
- ✓ Definición de los requisitos funcionales dados por el cliente.
- ✓ Realización de los artefactos ingenieriles acorde a la metodología seleccionada.
- ✓ Definición de la arquitectura y patrones de diseño.
- ✓ Realización de las pruebas al software.

Para apoyar el desarrollo de la investigación se emplean los siguientes **métodos científicos**:

Métodos Teóricos:

- Analítico Sintético: Se realizó un análisis de los documentos y la información referente a la forma en que se realiza la evaluación integrada en la UCI, lo que permitió la identificación de las dificultades con que cuenta el proceso.
- Histórico-lógico: como resultado de la utilización de este método, se lograron conocer los antecedentes y tendencias actuales de los Sistemas de Evaluación Integrada, tomando dichas observaciones como guía para el desarrollo de la presente investigación.

Métodos Empírico:

Observación: Permitted obtener un conocimiento de cómo se realiza el proceso de evaluación integral, para lograr su mejora con el nuevo sistema.

Con la implementación de este sistema se espera informatizar esta área de la formación estudiantil para la institución que aún presenta dificultad y demora a la hora de la elaboración de esta tarea, posibilitando seguridad, rapidez y control de las evidencias y autoevaluaciones por parte de los estudiantes.

1 CAPÍTULO I. ESTUDIO DEL ESTADO DEL ARTE

En este capítulo se definen los principales conceptos asociados al tema, se realiza un estudio a los sistemas homólogos, se define la metodología y las herramientas a emplear.

1.1 CONCEPTOS ASOCIADOS AL TEMA

Integralidad: Puede definirse como una cualidad de la persona que la faculta para tomar decisiones sobre su comportamiento por sí misma.

Evaluación: Es la determinación sistemática del mérito, el valor y el significado de algo en función de unos criterios respecto a un conjunto de normas.

Evaluación integrada: Puede definirse como modo de lograr establecer a través de un conjunto de requerimientos el grado de integralidad de un estudiante para lograr llegar a ser un profesional capacitado con un desempeño profesional que clara y consistentemente sobresale con respecto a lo que se espera en el indicador evaluado y que cumple con lo requerido para ejercer profesionalmente la carrera.

Formación integral: Podemos definirla como el proceso continuo, permanente y participativo que busca desarrollar armónica y coherentemente todas y cada una de las dimensiones del ser humano, a fin de lograr su realización plena en la sociedad. La formación integral en la Universidad de Ciencias Informáticas influye a que los estudiantes tengan una integración a la sociedad como futuros profesionales.

Rendimiento académico: Hace referencia a la evaluación del conocimiento adquirido en el ámbito escolar, terciario o universitario. Un estudiante con buen rendimiento académico es aquél que obtiene calificaciones positivas en los exámenes que debe rendir a lo largo de una cursada. En otras palabras, el rendimiento académico es una

medida de las capacidades del alumno, que expresa lo que este ha aprendido a lo largo del proceso formativo. También supone la capacidad del alumno para responder a los estímulos educativos. En este sentido, el rendimiento académico está vinculado a la aptitud. El rendimiento académico está asociado conjuntamente a la evaluación integral de los estudiantes tomándolo en cuenta para este proceso.

Gestión: Es la acción o el efecto de gestionar y administrar. De una forma más específica, una gestión es una diligencia, entendida como un trámite necesario para conseguir algo o resolver un asunto, habitualmente de carácter administrativo o que conlleva a documentación.

1.2 ANÁLISIS DE SISTEMAS HOMÓLOGOS

Como parte de la investigación se realizó un estudio de los sistemas informáticos para la evaluación de integralidad existentes, tanto en el país, en la universidad, como en el resto del mundo, que pudieran ser la solución semejante al problema planteado en función de cómo estos manejan las evidencias de sus usuarios.

Internacionales:

Sistema de la Facultad de Ciencias Exactas de la Universidad Nacional del Centro de la provincia de Buenos Aires:

El sistema se centra en las caracterizaciones de las investigaciones de las ciencias exactas, donde se incluyen las diferentes informaciones referidas a los estudiantes y graduados. Está desarrollado con la combinación de los lenguajes del lado del cliente: HTML, JavaScript y CSS, aunque algunas de sus páginas están hechas en XML. Presenta una funcionalidad referida a la gestión de los planes de trabajo. En la misma se muestra información sobre las acciones y actividades que se realizan en la universidad durante un período determinado, pero a dichas acciones no se le puede

dar seguimiento, ni se pueden generar reportes de las mismas para ser almacenados en formato duro.

Nacionales:

Sistema informático para la evaluación de desempeño laboral basado en el módulo “Action Logs” del sistema ROC FM.

El sistema informático permite capturar el fichero que genera el módulo Action Logs del sistema ROC FM, traducirlo, mostrar la información, proponer una evaluación, además, obtener reportes en hoja de cálculo con extensión xlsx, gráfico de línea y de barra. Con el uso de este sistema además del jefe de departamento, los trabajadores, podrán consultar su estado laboral e implementar soluciones de mejora en caso de que su desempeño sea deficiente. Está desarrollado con varias tecnologías y herramientas teniendo en cuenta cada una de sus características esenciales; Python como lenguaje de programación, Xilema base Web como marco de trabajo web, PostgreSQL como SGBD y Apache como servidor.

Sistema de Gestión Académica AKADEMOS.

La eficiencia en el proceso de gestión académica en la Universidad de las Ciencias Informáticas se logra a través de Akademos, un sistema automatizado para la gestión académica desarrollado por un equipo de trabajo de la Dirección de Informatización. Akademos posibilita un mejor control e incidencia en el aprendizaje de los estudiantes, pues con la información que los profesores introducen en tiempo real, es posible determinar el estado de los estudiantes tanto de forma horizontal como vertical. Además, sus reportes pueden servir para efectuar investigaciones sociales importantes, dada la variedad en términos de proveniencia geográfica, social y académica de la masa estudiantil en la UCI. El sistema informático Akademos ha posibilitado la participación de directivos, docentes y estudiantes para agilizar los mecanismos de la gestión académica y disminuir la ocurrencia de errores. Aun cuando

existan restricciones legales en cuanto a la documentación dura que se maneja en la gestión académica, la posibilidad de que toda la información que se gestiona en Akademos pueda consultarse en línea hace que se incurra en un ahorro de materiales de oficina. Este sistema Web está desarrollado en la plataforma .NET, que utiliza SQL Server 2000 para el almacenamiento de los datos, IIS (Internet Information Services) como servidor Web y Servicios WEB XML para el intercambio con otras aplicaciones. Este sistema posee un apartado científico en el cual el estudiante puede acceder a un registro de evidencias de participación en premios y actividades curriculares a lo largo del curso, pero este no puede añadirlas por sí mismo. (Infante Costa, 2019)

Resultado del análisis

A partir del análisis de los sistemas informáticos homólogos antes descrito se puede arribar a la conclusión que tiene cierta similitud con el sistema para la gestión de la evaluación integrada de los estudiantes a desarrollar, de ellos pudimos determinar varios aspectos que nos servirán de guía para desarrollar la aplicación contando con varias funcionalidades, entre ellas están, la evaluación de desempeño permitiéndonos mostrar la información a partir de un análisis de las tareas realizadas y rendimiento general que se podrán consultar en su estado de evaluación, además de la funcionalidad de que el estudiante pueda añadir sus eventos. Se adoptará el estilo de colores naranja y blanco y se agregará la funcionalidad de exportar esta autoevaluación una vez aprobada a un documento en formato PDF. Por lo que se desarrollará un nuevo sistema teniendo en cuenta estas propuestas identificadas.

1.3 METODOLOGÍA PARA EL DESARROLLO DEL SOFTWARE

El equipo de desarrollo se decantó por elegir una metodología de desarrollo ágil, estas metodologías nacieron en la industria del desarrollo de 'software', cuando las compañías de este sector comprendieron que la forma tradicional de trabajo retrasaba

mucho la entrega del producto final. Unos procesos basados normalmente en un contrato cerrado, con escasa comunicación de los trabajadores, que conducían a entregables de mala calidad. (Tena)

Las principales ventajas del 'agile' son:

1. **Mejora la calidad:** Minimiza los errores en los entregables y mejora la experiencia y las funcionalidades para el cliente.
2. **Mayor compromiso:** Mejora la satisfacción del empleado y genera conciencia de equipo.
3. **Rapidez:** Acorta los ciclos de producción y minimiza los tiempos de reacción y toma de decisiones.
4. **Aumento de la productividad:** Al asignar mejor los recursos, y de forma más dinámica, mejora la producción según las prioridades que tenga la empresa.

Antes de 'agile', cuando una empresa quería desarrollar un proyecto nuevo, se ponía en marcha un **proceso lineal** que podía tardar uno o dos años en entregar un producto, con un alto riesgo de no adaptarse a la demanda final del cliente. Cuando esto sucedía, se optaba por soluciones de urgencia, o incluso empezar de cero el proyecto. Otro de los rasgos más característicos de las metodologías ágiles es el de emplear equipos multidisciplinares que trabajen juntos, codo con codo, durante todo el proceso. De esta manera, y junto con las entregas más rápidas, tempranas y frecuentes, el producto resultante es exactamente lo que el mercado está demandando. (Tena)

Entre las metodologías ágiles existentes se encuentran: Scrum, Programación extrema o eXtreme Programming (XP), Kanban. De ellas el equipo de trabajo decidió elegir XP.

Programación Extrema (XP)

Nace de la mano de Kent Beck en el verano de 1996, cuando trabajaba para Chrysler Corporation. Él tenía varias ideas de metodologías para la realización de programas que eran cruciales para el buen desarrollo de cualquier sistema. Las ideas primordiales de sus sistemas las comunico en las revistas C++ Magazine en una entrevista que esta le hizo el año 1999. (SINTYA MILENA MELÉNDEZ VALLADAREZ, 2016)

¿Qué es la programación extrema?

Es una Metodología ligera de desarrollo de aplicaciones que se basa en la simplicidad, la comunicación y la realimentación del código desarrollado.

Objetivos de XP

- La Satisfacción del cliente.
- Potenciar el trabajo en grupo.
- Minimizar el riesgo actuando sobre las variables del proyecto: costo, tiempo, calidad, alcance.

Valores de XP

- Simplicidad: Este valor se centra en reducir la complejidad, características extra, y pérdidas. El equipo debe tener la siguiente frase en mente “Buscar la manera más simple de que funcione”, y buscar esa solución en primer lugar.
- Comunicación: Este valor se centra en asegurarse de que todos los miembros del equipo sepan que se espera de ellos, y que otras personas están trabajando en ello. La reunión diaria es un componente de comunicación clave.
- Retroalimentación: El equipo debe obtener impresiones de la idoneidad de su trabajo lo antes posible. Fallar pronto puede ser útil, especialmente si al hacerlo obtenemos nueva información, mientras aún tenemos tiempo de mejorar el producto.

Características de XP

- Metodología basada en prueba y error para obtener un software que funcione realmente.
- Fundamentada en principios.
- Está orientada hacia quien produce y usa software (el cliente participa muy activamente).
- Reduce el costo del cambio en todas las etapas del ciclo de vida del sistema.
- Combina las que han demostrado ser las mejores prácticas para desarrollar software, y las lleva al extremo.
- Cliente bien definido.
- Los requisitos pueden cambiar.
- Grupo pequeño y muy integrado (2-12 personas).
- Equipo con formación elevada y capacidad de aprender

Fases XP

Planeación:

La Metodología XP plantea la planificación como un diálogo continuo entre las partes involucradas en el proyecto, incluyendo al cliente, a los programadores y a los coordinadores. El proyecto comienza recopilando las historias de usuarios, las que constituyen a los tradicionales casos de uso. Una vez obtenidas estas historias de usuarios, los programadores evalúan rápidamente el tiempo de desarrollo de cada una.

Diseño:

La Metodología XP hace especial énfasis en los diseños simples y claros. Los conceptos más importantes de diseño en esta metodología son los siguientes:

- Simplicidad
- Recodificación
- Metáforas
- Soluciones “Spike”

Codificación:

- Disponibilidad del Cliente
- Uso de Estándares
- Programación Dirigida por las Pruebas
- Programación en Pares
- Integraciones Permanentes
- Propiedad Colectiva del Código

Pruebas:

- Pruebas Unitarias
- Pruebas de Aceptación

1.4 HERRAMIENTAS Y TECNOLOGÍAS

1.4.1 Lenguaje de programación

Java

Desde su creación a mediados de la década de 1990, Java siempre ha estado entre los lenguajes de programación más usados y 30 años después sigue siendo un lenguaje de programación líder. Java es, de hecho, el lenguaje nativo de Android, la plataforma móvil más utilizada en el mundo. Una de sus principales características es que ofrece una gran portabilidad y puede ejecutarse en casi cualquier sistema.

También es muy escalable, lo que lo hace ser demandado entre las grandes empresas y las emergentes. Es un lenguaje de tipo estático, por lo que es rápido y fácil de mantener, con pocos errores. También es compatible con versiones anteriores y esto ayuda a mantener los costes de una organización; no hay necesidad de reescribir constantemente el código cada vez que se lanza una nueva versión. (Universia, 2022)

Características del lenguaje JAVA:

- Orientado a objetos: es uno de los estilos de programación más populares. Permite diseñar el software de forma que los distintos tipos de datos que se usen estén unidos a sus operaciones.
- Es distribuido.
- Java proporciona una gran biblioteca estándar y herramientas para que los programas puedan ser distribuidos.
- Es multihilo.
- Java logra llevar a cabo varias tareas simultáneamente dentro del mismo programa. Esto permite mejorar el rendimiento y la velocidad de ejecución.
- Independiente a la plataforma.
- Esto significa que programas escritos en el lenguaje Java pueden ejecutarse en cualquier tipo de hardware, lo que lo hace portable.

PHP

Es un lenguaje de programación que favorece la conexión entre servidores e interfaz de usuario. Entre los factores que lo destacan se encuentra el hecho de que es de código abierto esto significa que no hay restricciones de uso vinculadas a los derechos y que se encuentra en constante perfeccionamiento por parte de la comunidad. La simplicidad para aprender a usarlo y el desarrollo de código abierto le facilita el trabajo a profesionales que eligen estructurar sitios web utilizando la plataforma pues a

medida que avanzan las configuraciones y ediciones se simplifica cada vez más. Una de sus principales características es que es un lenguaje más dinámico que otras opciones por lo tanto es esencial para desarrollar aplicaciones complejas. La idea de usar este lenguaje es disminuir el tiempo de carga de las páginas para que el servidor trabaje con más suavidad. (Souza, 2020)

Entre sus principales ventajas tenemos:

- Aprendizaje intuitivo simplificado.
- Código abierto.
- Admite gran cantidad de datos.
- Compatible con la gran mayoría de base datos.

Python

Python es un lenguaje de programación de alto nivel que se utiliza para desarrollar aplicaciones de todo tipo. A diferencia de otros lenguajes como Java o .NET, se trata de un lenguaje interpretado, es decir, que no es necesario compilarlo para ejecutar las aplicaciones escritas en Python, sino que se ejecutan directamente por el ordenador utilizando un programa denominado interpretador, por lo que no es necesario “traducirlo” a lenguaje máquina. Es un lenguaje sencillo de leer y escribir debido a su alta similitud con el lenguaje humano, además, se trata de un lenguaje multiplataforma de código abierto y, por lo tanto, gratuito, lo que permite desarrollar software sin límites. Con el paso del tiempo, Python ha ido ganando adeptos gracias a su sencillez y a sus amplias posibilidades, sobre todo en los últimos años por su facilidad al trabajar con inteligencia artificial, aprendizaje automático, entre muchos otros campos en auge. (Santander Universidades)

Características de Python:

- Programación Orientada a objetos.

- Lenguaje interpretado.
- Multiplataforma.
- Tipado dinámico.
- Lenguaje de código abierto.
- Ampliamente respaldado.
- Es polivalente.

Resultado del análisis

Se decidió el uso de Python en su versión 3.7.2 como lenguaje de programación debido a que entre sus ventajas podemos destacar que, Python es un lenguaje de alto nivel por lo que su curva de dificultad no es complicada, está sustentado en una gran comunidad de usuarios que explora sus posibilidades frecuentemente, esta comunidad tan activa permite que usuarios de todos los niveles encuentren los mejores tutoriales, consejos y claves para empezar a utilizarlo. Puedes empezar un programa en Python sin tener que preocuparte por la difícil tarea de reescribir o adaptar el código a otras plataformas a medida que vas creciendo, es decir, este lenguaje te permitirá aumentar la complejidad de tus programas a lo largo del tiempo.

1.4.2 Marco de trabajo para el desarrollo de la solución informática

Django

Django es un marco de trabajo web de alto nivel que permite el desarrollo rápido de sitios web seguros y mantenibles. Desarrollado por programadores experimentados, Django se encarga de gran parte de las complicaciones del desarrollo web, por lo que puedes concentrarte en escribir tu aplicación sin necesidad de reinventar la rueda. Es gratuito y de código abierto, tiene una comunidad próspera y activa, una gran documentación y muchas opciones de soporte gratuito y de pago. (Mozilla Foundation)

Algunas de las características de Django son:

- Seguridad: Django tiene activados mecanismos incluidos para proteger tu base de datos, formularios y JavaScript.
- Escalabilidad: puedes utilizar el marco de trabajo para un desarrollo sencillo, hasta uno mucho más complejo, ambos casos funcionarán de manera estable y con rapidez.
- Interfaz: Su interfaz para acceso a la base de datos y hacer consultas es sumamente buena.
- Portable: Al estar escrito en Python, se puede ejecutar en muchas plataformas como Windows, OS X, entre otras, dándole muchísima libertad al programador al momento de ejecutar las aplicaciones.

Flask

Flask es un “micro” marco de trabajo web escrito en Python y concebido para facilitar el desarrollo de Aplicaciones Web bajo el patrón MVC. Su principal intención es ser simple y pequeño; todo el marco de trabajo consiste en un grupo de módulos. No hay un esqueleto o una estructura de la cual partir, todo se empieza con una página en blanco. Flask no proporciona grandes funcionalidades, pero hay extensiones Flask disponibles para agregar ORM, validación de formularios, manejo de carga, etc.

Flask es ideal, entre otras cosas, para aprender a programar y para ser utilizado por desarrolladores que se preocupan por las buenas prácticas y el código “elegante”, los que quieren crear prototipos de forma rápida y aquellos que necesitan una aplicación independiente.

Algunas de las características de Flask son:

- Incluye un servidor web de desarrollo
- Tiene un depurador y soporte integrado para pruebas unitarias

- Buen manejo de rutas
- Sirve para construir servicios web (como APIs REST) o aplicaciones de contenido estático.
- Flask es código abierto y está amparado bajo una licencia BSD.

Resultado del análisis

Se decidió usar DJANGO en su versión 3.2.9 debido a su capacidad para procesar gran cantidad de información, proporciona una estructura de código autogenerado, además de trabajar bajo un patrón MVC (Modelo Vista Controlador), lo que permite un desarrollo ágil y reutilizable.

Bootstrap

Bootstrap es una librería CSS desarrollado por Twitter en 2010, para estandarizar las herramientas de la compañía. Inicialmente, se llamó Twitter Blueprint y, un poco más tarde, en 2011, se transformó en código abierto y su nombre cambió para Bootstrap, esta combina CSS y JavaScript para estilizar los elementos de una página HTML. Permite mucho más que, simplemente, cambiar el color de los botones y los enlaces. (rockcontent blog, 2020)

Esta es una herramienta que proporciona interactividad en la página, por lo que ofrece una serie de componentes que facilitan la comunicación con el usuario, como menús de navegación, controles de página, barras de progreso y más.

Características de Bootstrap:

- Permite crear interfaces que se adapten a los diferentes navegadores, tanto de escritorio como tabletas y móviles a distintas escalas y resoluciones.
- Ofrece un diseño sólido usando LESS y estándares como CSS3/HTML5.

- Funciona con todos los navegadores, incluido Internet Explorer usando HTML Shim para que reconozca los tags HTML5.
- Permite utilizar muchos elementos web: desde iconos a desplegables, combinando HTML5, CSS y JavaScript.

Tailwind CSS

Tailwind CSS es una potente herramienta para el desarrollo frontend. Está dentro de la clasificación de los marcos de trabajo CSS o también llamados marcos de diseño. Permite a los desarrolladores y diseñadores aplicar estilos a los sitios web de una manera ágil y optimizada. Las hermosas interfaces de usuario personalizadas se pueden construir de manera efectiva usando CSS sin realmente poner mucho esfuerzo de codificación. Tailwind CSS ofrece la ventaja de diseñar cada componente de manera distintiva, de la manera que deseas. (Rodríguez, 2021)

Características de Tailwind:

- Tailwind CSS es una librería CSS altamente personalizable: Aunque viene con una configuración predeterminada, es simple sobrescribirla gracias al fichero de configuración `tailwind.config.js`. Este archivo de configuración permite una fácil personalización de paletas de colores, estilizado, espaciado, temas, etc.
- Tiene patrones de utilidad comunes: Elimina la molestia de nombrar clases con Tailwind CSS. La disponibilidad de patrones de utilidad comunes resuelve numerosos problemas como especificar clases, organizarlas, ponerlas en cascada y mucho más. Las clases de utilidades simplifican el proceso de creación de componentes personalizados.
- Permite la creación de diseños complejos y responsive de forma libre: Tailwind CSS utiliza un enfoque predeterminado de dispositivos móviles. La disponibilidad

de clases de utilidades facilita la creación de diseños complejos con capacidad de respuesta de manera libre.

Foundation

Foundation es una librería de interfaz de usuario responsive. Este proporciona una cuadrícula responsive e incluye componentes de interfaz de usuario HTML y CSS, plantillas, y fragmentos de código, incluyendo tipografía, formularios, botones, barras de navegación y otros componentes de interfaz usuario, así como extensiones de JavaScript opcionales. Foundation está mantenida por zurb.com y es un proyecto de código abierto, fue diseñado y probado en numerosos dispositivos y navegadores. Es la primera librería de móvil responsive construido con Sass/SCSS dando buenas prácticas a diseñadores para el desarrollo rápido. Esta librería incluye los patrones necesarios más comunes para rápidamente maquetar un sitio responsive. A través del uso de Sass mixins, los componentes de Foundation son fácilmente estilizados y sencillos de extender.

Estas son las características de Foundation:

- Flexibilidad extrema: Foundation se creó para dar al desarrollador front-end un control total sobre sus interfaces de usuario. Como resultado, Foundation se sentirá suave y enormemente complejo para el recién llegado. Sin embargo, la razón es que Foundation no te impone ningún lenguaje de estilo, sino que pretende ser exactamente lo que es: un marco CSS excelente.
- Sistema de cuadrícula y diseño responsive: Foundation viene por defecto con un diseño de cuadrícula flexible de 940 píxeles de ancho. El conjunto de herramientas es totalmente responsive permitiendo hacer uso de diferentes resoluciones y tipos de dispositivos: teléfonos móviles, formato vertical y horizontal, tabletas y PC.
- Entender la hoja de estilos CSS: Foundation ofrece un conjunto de hojas de estilo que proporcionan definiciones de estilo básicas para todos los componentes HTML

clave. Estos proporcionan un navegador y todo un sistema uniforme, aspecto moderno para el formato de textos, tablas y elementos de formulario. (Ankush)

Resultado del análisis

Se decidió elegir como librería CSS Bootstrap en su versión 5.0, debido a su bajo nivel de aprendizaje, fácil acceso a su documentación, acoplamiento con nuestro marco de trabajo de desarrollo web Django e integración con todos los navegadores web.

1.4.3 Herramienta CASE

Lucidchart

Lucidchart es una herramienta UML a la que se puede acceder en el navegador. La cuenta gratuita te da paso a un paquete de herramientas UML muy completo. Incluye 7 tipos de diagramas UML y lenguajes de modelado de procesos de negocio como BPMN 2.0, plantillas de iconos de red, maquetas de dispositivos móviles e integración de vídeo. Una de las ventajas de Lucidchart es su funcionamiento intuitivo. También permite compartir y editar simultáneamente diagramas en equipo e integrar comentarios directamente en la herramienta. Entre sus principales ventajas cabe destacar: (IONOS Cloud S.L.U.)

- Muchas características de trabajo en equipo
- Amplia biblioteca de plantillas
- Las marcas de UML aceleran el flujo de trabajo
- Escalable a través del almacenamiento en la nube
- Ahorro de espacio en disco y diseño claro

GitMind

GitMind es un software gratuito de mapas mentales en línea que se utiliza para organizar datos en mapas junto con la realización de lluvias de ideas para tomar

decisiones inteligentes. Ayuda a las empresas a medir el rendimiento mediante el análisis de marketing. El software de mapas mentales GitMind también realiza funciones como la gestión de proyectos, la organización de la empresa, la planificación y el desarrollo de ideas mediante el uso de mapas mentales, organigramas, diagramas UML, carriles, diagramas de flujo y gráficos de análisis. (IONOS Cloud S.L.U.)

Entre sus principales características podemos destacar:

- Gratis y en línea
- Fácil de usar.
- Convenientes teclas de acceso rápido
- Plantillas diversificadas y estilo de diseño múltiple
- Almacenamiento seguro en la nube

Visual Paradigm

La herramienta CASE Visual Paradigm para UML 8.0 la cual soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue y contribuye a una rápida construcción de aplicaciones de calidad, mejores y a un menor costo. Se caracteriza por el uso de un lenguaje estándar común al equipo de trabajo, que facilita la comunicación entre sus integrantes, es una herramienta fácil de instalar y actualizar, genera código para varios lenguajes de programación y exporta en formato HTML, es una tecnología libre y está disponible en varios idiomas. Uno de los elementos importantes que le aportan distinción es la posibilidad que brinda de soportar aplicaciones web (Pressman, 2002).

Esta herramienta permite aumentar la calidad del software, a través de la mejora en el desarrollo y mantenimiento del mismo, de igual forma potencia la reutilización del software y estandarización de la documentación, además del uso de las distintas

metodologías propias de la Ingeniería del Software. Entre sus principales características se encuentran:

- Entorno de creación de diagramas para UML 2.0.
- Diseño centrado en Casos de Uso y enfocado al negocio que genera un software de mayor calidad.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Disponibilidad de múltiples versiones, para cada necesidad.
- Modelamiento de Base de Datos.
- Ingeniería de Código.
- Disponibilidad en múltiples plataformas.

Resultado del análisis

Se utilizó como herramienta Visual Paradigm en su versión 5.0 debido no solo a ser la herramienta con la que más afinidad posee el equipo de trabajo, también podemos destacar que posee capacidades de ingeniería directa e inversa, licencia gratuita y comercial, exportación como HTML, diseño centrado en casos de uso y enfocado al negocio que genera un software de mayor calidad, también el uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.

1.4.4 Lenguaje de modelado

UML

UML es un lenguaje de modelado visual que es usado para especificar, visualizar, construir y documentar documentos y artefactos de un sistema de software. Captura decisiones y conocimiento sobre los sistemas que se deben construir. Se usa para entender, diseñar, hojear, mantener, y controlar la información sobre tales sistemas.

Está pensado para usarse con todos los métodos de desarrollo, etapas de ciclo de vidas, dominios de aplicación y medios. Incluye conceptos semánticos, notación y principios generales. Es usado para describir grandes sistemas con la calidad requerida y que estos puedan ser entendidos por los usuarios.

Es comparable a los planos usados en otros campos y consiste en diferentes tipos de diagramas. En general, los diagramas UML describen los límites, la estructura y el comportamiento del sistema y los objetos que contiene. UML no es un lenguaje de programación, pero existen herramientas que se pueden usar para generar código en diversos lenguajes usando los diagramas UML. UML guarda una relación directa con el análisis y el diseño orientados a objetos. (Lucid Software Inc.)

UML cumple con los siguientes requerimientos:

- Establecer una definición formal de un metamodelo común basado en el estándar MOF (Meta-Object Facility) que especifique la sintaxis abstracta del UML. La sintaxis abstracta define el conjunto de conceptos de modelado UML, sus atributos y sus relaciones, así como las reglas de combinación de estos conceptos para construir modelos UML parciales o completos.
- Brindar una explicación detallada de la semántica de cada concepto de modelado UML. La semántica define, de manera independiente a la tecnología, cómo los conceptos UML se habrán de desarrollar por las computadoras.
- Especificar los elementos de notación de lectura humana para representar los conceptos individuales de modelado UML, así como las reglas para combinarlos en una variedad de diferentes tipos de diagramas que corresponden a diferentes aspectos de los sistemas modelados.
- Definir formas que permitan hacer que las herramientas UML cumplan con esta especificación. Esto se apoya (en una especificación independiente) con una

especificación basada en XML de formatos de intercambio de modelos correspondientes que deben ser concretados por herramientas compatibles.

1.4.5 Entorno de desarrollo integrado

Visual Studio Code

Este es un editor de código fuente desarrollado por Microsoft para Windows, Linux, macOS y Web. Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código. Visual Studio Code es rápido, ligero y muy fácil de utilizar. Un aspecto fundamental es que puede ser utilizado con los lenguajes de programación que se trabaja a diario. Visual Studio Code es una herramienta que tiene soporte nativo para gran variedad de lenguajes, entre ellos se puede destacar los principales del desarrollo Web: HTML, CSS, y JavaScript, entre otros. Otra ventaja interesante es la posibilidad de configurar la vista a nuestro gusto, de esta forma, se puede tener más de un código visible al mismo tiempo, así como las carpetas del proyecto y también acceso a la terminal o un detalle de problemas, entre otras posibilidades. Esta herramienta se destaca por ofrecer las opciones esenciales integradas en el programa. Además, brinda la posibilidad de potenciar sus características mediante la instalación de manera sencilla de extensiones. Para quienes tienen experiencia en el mundo del desarrollo, Visual Studio Code se presenta como una alternativa que puede aprender a utilizar sin grandes complicaciones. (Luca)

Eclipse

Eclipse es un software veterano, y como tal ha ido afinando mucho su funcionalidad. Siendo software libre, es una opción muy buena para los que desea ir más allá de los límites que impone un entorno como Processing. Os aportará todo lo que aportan los entornos de desarrollo integrados más completos: autocompletar código, acceso muy

eficiente a los archivos del proyecto, facilidad para probar el código, corrección de sintaxis, etc. Uno de sus puntos fuertes, además de una comunidad de usuarios muy extensa, es el gran número de complementos que posee, por lo que su funcionalidad se puede ampliar cuando las necesidades específicas del proyecto lo soliciten. (Soler-Adillo, 2017)

Principales ventajas

- Una plataforma muy versátil, con muchísimas extensiones.
- Un proyecto muy consolidado, y por tanto con un ecosistema importante alrededor.

Pycharm

Creado por la compañía JetBrains es un editor de pago que también tiene una versión gratuita mantenida por la comunidad. Es quizás el mejor editor de código para Python gratuito que se puede encontrar en el mercado debido a la gran cantidad de atajos al momento de programar. PyCharm no solo funciona correctamente con archivos de Python, sino que también admite otros lenguajes de programación como JavaScript, Kotlin o CoffeeScript y otras herramientas como HTML o CSS. Esto hace que por lo menos sea un IDE que merezca probarse para aprender un lenguaje de programación tan vinculado a GNU/Linux. (García)

Funcionalidades de PyCharm

- Asistencia inteligente a la codificación.
- Herramientas de desarrollo integradas.
- Desarrollo web.
- Herramientas científicas.
- IDE personalizable y multiplataforma.

Resultado del análisis

Se decidió usar Visual Studio Code en su versión 1.42 debido a sus ventajas como son la gran variedad de complementos que se le pueden agregar fácilmente transformándolo en un editor sumamente completo y fácil de emplear, es multiplataforma y se integra la mayoría de lenguajes actuales, además de poseer una interfaz adaptable.

1.4.6 Gestor de base de datos

PostgreSQL

Potente sistema de base de datos relacional de objetos de código abierto que utiliza y amplía el lenguaje SQL combinado con muchas características que almacenan y escalan de forma segura las cargas de trabajo de datos más complicadas. PostgreSQL se ha ganado una sólida reputación por su arquitectura comprobada, confiabilidad, integridad de datos, conjunto sólido de funciones, extensibilidad y la dedicación de la comunidad de código abierto detrás del software para ofrecer soluciones innovadoras y de alto rendimiento. PostgreSQL se ejecuta en todos los principales sistemas operativos, cumple con ACID desde 2001 y tiene potentes complementos. PostgreSQL tiene dos ventajas fundamentales, primero en lo que respecta a su funcionalidad y capacidad de trabajar con mayores cantidades de datos, pero también en lo que respecta a su licencia. MySQL tiene una licencia dual, lo que significa que para proyectos comerciales habría que pagar por su uso. Sin embargo, PostgreSQL tiene una única licencia totalmente abierta para cualquier uso. (Arsys blog, 2018)

Algunas de sus principales características son:

- **Alta concurrencia:** Es capaz de atender a muchos clientes al mismo tiempo y entregar la misma información de sus tablas, sin bloqueos.

- **Soporte para múltiples tipos de datos de manera nativa:** Ofrece los tipos de datos habituales en los sistemas gestores, pero además muchos otros que no están disponibles en otros competidores, como direcciones IP, direcciones MAC, arreglos, números decimales con precisión configurable, figuras geométricas, etc.
- **Soporte a triggers:** Permite definir eventos y generar acciones cuando estos se disparan.
- **Trabajo con vistas:** Esto quiere decir que pueden consultar los datos de manera diferente al modo en el que se almacenan.
- **Objeto-relacional:** Otra de sus principales características, que permite trabajar con sus datos como si fueran objetos y ofrece mecanismos de la orientación a objetos, como herencia de tablas.
- **Soporte para bases de datos distribuidas:** Donde el trabajo con transacciones asegura que estas tendrán éxito cuando han podido realizarse en todos los sistemas involucrados.
- **Soporte para gran cantidad de lenguajes:** PostgreSQL es capaz de trabajar con funciones internas, que se ejecutan en el servidor, escritas en diversos lenguajes como C, C++, Java, PHP, Python o Ruby. Además, ofrece interfaces para ODBC y JDBC, así como interfaces de programación para infinidad de lenguajes de programación.

SQLite

SQLite es una herramienta de software libre, que permite almacenar información en dispositivos empotrados de una forma sencilla, eficaz, potente, rápida y en equipos con pocas capacidades de hardware, como puede ser una PDA o un teléfono celular. SQLite implementa el estándar SQL92 y también agrega extensiones que facilitan su uso en cualquier ambiente de desarrollo. Esto permite que SQLite soporte desde las consultas más básicas hasta las más complejas del lenguaje SQL, y lo más importante

es que se puede usar tanto en dispositivos móviles como en sistemas de escritorio, sin necesidad de realizar procesos complejos de importación y exportación de datos, ya que existe compatibilidad al 100% entre las diversas plataformas disponibles, haciendo que la portabilidad entre dispositivos y plataformas sea transparente. (Rómmel)

Estas son algunas de las características principales de SQLite:

- La base de datos completa se encuentra en un solo archivo.
- Puede funcionar enteramente en memoria, lo que la hace muy rápida.
- Es totalmente autocontenida (sin dependencias externas).
- Cuenta con librerías de acceso para muchos lenguajes de programación.
- El código fuente es de dominio público y se encuentra muy bien documentado.

(Rómmel)

MySQL

MySQL es un sistema de gestión de bases de datos relacionales (RDBMS) de código abierto respaldado por Oracle y basado en el lenguaje de consulta estructurado (SQL). MySQL funciona prácticamente en todas las plataformas, incluyendo Linux, UNIX y Windows. Aunque puede utilizarse en una amplia gama de aplicaciones, MySQL se asocia más a menudo con las aplicaciones web y la publicación en línea. MySQL se basa en un modelo cliente-servidor. El núcleo de MySQL es el servidor MySQL, que maneja todas las instrucciones (o comandos) de la base de datos. El servidor MySQL está disponible como un programa independiente para su uso en un entorno de red cliente-servidor y como una biblioteca que puede ser incrustada (o enlazada) en aplicaciones independientes. (TechTarget)

Estas son algunas de las características principales:

- Arquitectura cliente y servidor.

- Compatibilidad con SQL.
- Procedimientos almacenados. MySQL posee la característica de no procesar las tablas directamente, sino que a través de procedimientos almacenados.
- Desencadenantes. MySQL permite además poder automatizar ciertas tareas dentro de nuestra base de datos.

Resultado del análisis

Se decidió elegir PostgreSQL en su versión 9.4 debido a su bajo nivel y fácil empleo además de ser el motor de base de datos de la actualidad y el más usado, siendo este el sistema gestor de bases de datos de código libre más potente y robusto del mercado, además de poseer gran compatibilidad con las otras herramientas usadas en el sistema actual.

1.5 CONCLUSIONES PARCIALES

- Con el desarrollo de este capítulo se estableció una panorámica de los conceptos relacionados con el tema, la fundamentación de herramientas usadas las cuales fueron Python v3.7.2 como lenguaje de programación, Django v3.2.9 como marco de trabajo web, PostgreSQL v9.4 como gestor de base datos, Visual Paradigm como herramienta CASE, Bootstrap v5.0 como librería CSS y Visual Studio Code v1.42 como entorno de desarrollo cumpliendo con los requisitos fundamentales para crear un prototipo de software de fácil uso para los usuarios y métodos a utilizar para desarrollar el sistema informático en vista a resolver, teniendo previamente establecido el enfoque ágil en que se va a desarrollar el proyecto, siendo la metodología usada programación extrema.

2 CAPÍTULO II. PROPUESTA DE SOLUCIÓN

La metodología XP plantea la planificación como un diálogo continuo entre las partes involucradas en el proyecto, incluyendo al cliente, a los programadores y a los coordinadores. El proyecto comienza recopilando las historias de usuarios, una vez obtenidas estas historias de usuarios, los programadores evalúan rápidamente el tiempo de desarrollo de cada una y las dividen por iteraciones para luego definir el plan de entrega.

2.1 REQUISITOS FUNCIONALES

Los requisitos funcionales describen la interacción entre el sistema y su ambiente independientemente de su implementación. El ambiente incluye al usuario y cualquier otro sistema externo que interactúa con el sistema. Los requisitos funcionales definen funciones del sistema de software o sus componentes. Una función es descrita como un conjunto de entradas, comportamientos y salidas. (Luis Ruben Lima Mateo, Plataforma SOFIA para la distribución electrónica de libros digitales en Cuba, 2019)

No.	Nombre	Descripción	Prioridad	Complejidad
RF 1	Registrar autoevaluación	El sistema debe crear una autoevaluación del estudiante una vez que este accede a este panel en caso de no tenerla.	Alta	Alta
RF 2	Modificar autoevaluación	El sistema debe permitir a los estudiantes modificar los	Alta	Alta

		datos de la autoevaluación a partir de político ideológico, académico, investigativa, extensión universitaria, distinciones, misiones, sanciones, señalamientos, recomendaciones.		
RF 3	Registrar evidencia	El sistema debe permitir que el estudiante registre sus evidencias a partir de .jpg, .mp4, .pdf.	Alta	Alta
RF 4	Eliminar evidencia	El sistema debe permitir que el estudiante elimine sus evidencias.	Alta	Alta
RF 5	Visualizar evidencias	El sistema debe permitir al profesor guía, presidente de brigada, representante de la FEU y estudiantes ver las evidencias subidas por los estudiantes	Baja	Alta
RF 6	Asignar evaluación	El sistema debe permitir al profesor guía una evaluación a los estudiantes de satisfactoria o no satisfactoria.	Alta	Alta

RF 7	Modificar evaluación	El sistema debe permitir el profesor guía modificar los datos de la evaluación del estudiante: satisfactoria o no satisfactoria.	Alta	Alta
RF 8	Registrar Profesor	El sistema debe permitir registrar los datos del profesor a partir de los campos: nombre, apellidos, usuario, provincia, municipio, teléfono, CI, sexo.	Alta	Alta
RF 9	Registrar usuario	El sistema debe permitir registrar los datos del usuario: usuario, contraseña, rol.	Alta	Alta
RF 10	Eliminar usuario	El sistema debe permitir al administrador eliminar los datos de los usuarios.	Alta	Alta
RF 11	Buscar usuario	El sistema debe permitir al administrador buscar los datos de los usuarios.	Media	Alta
RF 12	Registrar grupo docente	El sistema debe permitir registrar los datos del grupo docente a partir de: facultad,	Alta	Alta

		año académico, número grupo.		
RF 13	Ver autoevaluación aprobada	El sistema debe permitir visualizar la evaluación aprobada.	Baja	Alta
RF 14	Exportar autoevaluación aprobada	El sistema debe permitir exportar a un .pdf la autoevaluación de la estudiante una vez aprobada.	Alta	Alta
RF 15	Autenticar Usuario	El sistema debe permitir autenticar usuario a partir de los campos: usuario, contraseña.	Alta	Alta
RF 16	Registrar estudiante	El sistema debe permitir registrar los datos del estudiante a partir de: nombre, solapín, usuario, cargo, número de expediente, carrera, facultad, grupo, provincia, municipio, teléfono, carnet, sexo.	Alta	Alta

RF 17	Añadir nota	El sistema debe crear el campo nota al estudiante crear una autoevaluación.	Baja	Baja
RF 18	Modificar nota	El sistema debe permitir al profesor guía modificar la nota de la autoevaluación del estudiante.	Baja	Baja
RF 19	Crear comisión	El sistema debe permitir al profesor principal crear una comisión de evaluación.	Alta	Media
RF 20	Asignar profesor principal	El sistema debe permitir al decano asignar profesor principal.	Baja	Baja
RF 21	Asignar profesor guía	El sistema debe permitir al profesor principal asignar los profesores guías.	Baja	Baja
RF 22	Crear evento	El sistema debe permitir al administrador crear un evento a partir de los campos: nombre, descripción, categoría	Media	Baja
RF 23	Modificar evento	El sistema debe permitir al administrador modificar los	Media	Baja

		eventos a partir de los campos: nombre, categoría		
RF 24	Eliminar evento	El sistema debe permitir al administrador eliminar los eventos.	Media	Baja
RF 25	Buscar eventos	El sistema debe permitir al administrador listar los eventos.	Baja	Baja
RF 26	Registrar evento	El sistema debe permitir al estudiante registrar un evento en su autoevaluación.	Media	Media
RF 27	Eliminar evento registrado	El sistema debe permitir al estudiante eliminar el o los eventos seleccionados de su autoevaluación.	Media	Media
RF 28	Validar autoevaluación	El sistema debe permitir al profesor guía, presidente de brigada y representante de la FEU, establecer en validado o invalidado la autoevaluación del estudiante.	Alta	Media

RF 29	Modificar validar autoevaluación	El sistema debe permitir al profesor guía, presidente de brigada y representante de la FEU modificar la validación o invalidación de la autoevaluación del estudiante.	Media	Media
RF 30	Ver historial	El sistema debe permitir al profesor guía visualizar la información de todas las autoevaluaciones registradas por cada estudiante de su grupo.	Media	Media
RF 31	Asignar evaluación integral	El sistema debe permitir al profesor guía asignar una evaluación integral de Satisfactoria o Insatisfactoria a cada estudiante de su grupo.	Baja	Alta
RF 32	Modificar evaluación integral	El sistema debe permitir al profesor guía modificar la evaluación integral de Satisfactoria o Insatisfactoria a cada estudiante de su grupo.	Baja	Alta

RF 33	Sugerencia del sistema	El sistema debe generar una recomendación de evaluación integral al profesor guía en dependencia del historial de cada estudiante de su grupo.	Alta	Media
-------	------------------------	--	------	-------

2.2 REQUISITOS NO FUNCIONALES

Usabilidad

- RnF1 La interfaz web debe ser sencilla con colores anaranjado y blanco, sin cúmulo de imágenes u objetos que distraigan al usuario de su objetivo.

Eficiencia del desempeño

- RnF2 El sistema debe ejecutarse sobre un procesador dual Core o superior.
- RnF3 El sistema debe ejecutarse con una memoria RAM de 2 GB o superior.
- RnF4 El sistema debe almacenarse en un disco duro de al menos 50gb.

Fiabilidad

- RnF5. La aplicación debe estar disponible las 24 horas del día, para lo cual es preciso una aplicación web y un servidor.

Portabilidad

- RnF6 El sistema debe permitir el acceso al usuario desde cualquier sistema operativo.
- RnF7 El sistema debe permitir el acceso al usuario desde cualquier navegador web.

Restricciones del diseño y la implementación

- RnF8 El sistema debe emplear Python v3.7.2 como lenguaje de programación.
- RnF9 El sistema debe utilizar PostgreSQL v9.4 como gestor de base datos.
- RnF10 El sistema debe utilizar Visual Studio Code v1.42 como entorno de desarrollo.
- RnF11 El sistema debe utilizar Django v3.2.9 como marco de trabajo de desarrollo web.
- RnF12 El sistema debe utilizar Bootstrap v5.0 como librería CSS.

Seguridad

- RnF13 El sistema solo permitirá el acceso a usuarios registrados.
- RnF14 Los usuarios deben acceder solo a aquellas funcionalidades que les corresponde.
- RnF15 El sistema debe permitir que la información sea modificada solo por el personal autorizado.

2.3 HISTORIAS DE USUARIO

Representan una breve descripción del comportamiento del sistema, se realizan por cada característica principal del sistema y son utilizadas para cumplir estimaciones de tiempo y el plan de lanzamientos y presiden la creación de las pruebas de aceptación. Cada historia de usuario debe ser lo suficientemente comprensible y delimitada para que los programadores puedan implementarlas en unas semanas. Las iteraciones duran unas dos semanas, y los desarrolladores trabajan en pareja para escribir el código durante esas iteraciones. Todo desarrollo de software está sujeto a rigurosos y frecuentes testeos. Entonces, con la aprobación del cliente, el software es entregado en pequeñas entregas. (SINTYA MILENA MELÉNDEZ VALLADAREZ, 2016)

Tabla 1HU1

Historia de usuario	
Número: 1	Nombre del requisito: Registrar autoevaluación
Programador: Javier Mancha Cabrera	Iteración asignada: 1
Prioridad: Alta	Tiempo estimado: 3 días
Riesgo en desarrollo: N/A	Tiempo real: 3 días
Descripción: El sistema debe crear una autoevaluación del estudiante una vez que este accede a este panel en caso de no tenerla.	
Observaciones: N/A	

Tabla 2HU2

Historias de usuario	
Número: 2	Nombre del requisito: Modificar autoevaluación
Programador: Javier Mancha Cabrera	Iteración asignada: 1
Prioridad: Alta	Tiempo estimado: 2 días
Riesgo en desarrollo: N/A	Tiempo real: 2 días
Descripción: El sistema debe permitir a los estudiantes modificar los datos de la autoevaluación a partir de político ideológico, académico, investigativa, extensión universitaria, distinciones, misiones, sanciones, señalamientos, recomendaciones.	

Observaciones: N/A

Tabla 3HU3

Historias de usuario	
Número: 3	Nombre del requisito: Registrar evidencia
Programador: Javier Mancha Cabrera	Iteración asignada: 1
Prioridad: alta	Tiempo estimado: 5 días
Riesgo en desarrollo: N/A	Tiempo real: 5 días
Descripción: El sistema debe permitir que el estudiante registre sus evidencias a partir de .jpg, .mp4, .pdf.	
Observaciones: N/A	

- ✓ Las demás historias de usuario serán colocadas en la sección de [Anexos](#).

REQUISITO	Reg. autoevaluaci	Mod. autoevaluaci	Eliminar evidencia	Visualizar	Asignar eval	Modif. eva	Reg profesor	Reg usuario	Elim usuario	Buscar usuario	Reg GD	Ver autoeval	Exportar autoeval	Autenticar	Reg estud	Añadir	Modif. nota	Crear transmisión	Asig. Prof.	Crear	Modif	Elim	Buscar	Selec evento	Elim selec evento	Validar	Modif historial	Asignar	Sugerenci	
1	X																													
2		X																												
3			X																											
4				X																										
5					X																									
6						X																								
7							X																							
8								X																						
9									X																					
10										X																				
11											X																			
12												X																		
13													X																	
14														X																
15															X															
16																X														
17																	X													
18																		X												
19																			X											
20																				X										
21																					X									
22																						X								
23																							X							
24																								X						
25																									X					
26																										X				
27																											X			
28																												X		
29																													X	
30																													X	
31																													X	
32																													X	

Ilustración 1 Matriz de trazabilidad

2.4 ESTIMACIÓN DE TIEMPO POR HISTORIA DE USUARIO

La programación bajo la metodología XP (programación extrema) basa sus procesos de planificación en estimaciones temporales de las historias de usuario, las cuáles deben ser realizadas por los desarrolladores durante las diversas reuniones de planificación. Una historia de usuario es lo suficientemente pequeña como para que el equipo la desarrolle durante una entrega de una a tres semanas; más de tres semanas implica que se debe señalar al cliente que debe dividir una historia de usuario y menos de una semana implica que la historia es demasiado sencilla y por lo que se deben unir dos o más de ellas para su mejor interpretación. (Luis Ruben Lima Mateo, Plataforma SOFIA para la distribución electrónica, 2019)

Denotación de punto de estimación	Interpretación
1 punto	1 día (8 horas laborales)

Tabla 4 Estimación de tiempo

<u>Historia de usuario</u>	<u>Punto estimado</u>
Registrar autoevaluación	3
Modificar autoevaluación	2
Registrar evidencia	5
Eliminar evidencia	3
Visualizar evidencia	2
Asignar evaluación	3
Modificar evaluación	2
Registrar Profesor	1
Registrar usuario	3
Eliminar usuario	4
Buscar usuario	3
Registrar grupo docente	3
Ver autoevaluación aprobada	3
Exportar autoevaluación aprobada	2
Autenticar Usuario	1
Registrar estudiante	3

Añadir nota	2
Modificar nota	2
Crear comisión	3
Asignar profesor principal	1
Asignar profesor guía	1
Crear evento	2
Modificar evento	3
Eliminar evento	2
Buscar eventos	2
Registrar evento	2
Eliminar evento registrado	3
Validar autoevaluación	2
Modificar validar autoevaluación	1
Ver historial	2
Asignar evaluación integral	1
Modificar evaluación integral	1
Sugerencia del sistema	3

Las estimaciones realizadas permitieron confeccionar una evaluación puntual del tiempo de implementación de cada historia de usuario para la posterior elaboración del plan de iteración.

2.5 PLAN DE ITERACIONES

Un plan de iteración está constituido por un conjunto secuencial de actividades y tareas, cada una tiene recursos asignados y pueden depender a su vez de otras tareas. El plan de iteración se realiza una vez por cada iteración.

Tabla 5 Plan de iteraciones

No. Iteración	Historia de Usuario	Prioridad	Esfuerzo Estimado	
1	Registrar autoevaluación	Alta	3	20
	Modificar autoevaluación	Alta	2	
	Registrar evidencia	Alta	5	
	Eliminar evidencia	Alta	3	
	Visualizar evidencia	Baja	2	
	Asignar evaluación	Alta	3	
	Modificar evaluación	Alta	2	
2	Registrar Profesor	Alta	1	14
	Registrar usuario	Alta	3	
	Eliminar usuario	Alta	4	
	Buscar usuario	Media	3	
	Registrar grupo docente	Alta	3	
3	Ver autoevaluación aprobada	Baja	3	42
	Exportar autoevaluación aprobada	Alta	2	
	Autenticar Usuario	Alta	1	

Registrar estudiante	Alta	3	
Añadir nota	Baja	2	
Modificar nota	Baja	2	
Crear comisión	Alta	3	
Asignar profesor principal	Baja	1	
Asignar profesor guía	Baja	1	
Crear evento	Media	2	
Modificar evento	Media	3	
Eliminar evento	Media	2	
Buscar eventos	Baja	2	
Registrar evento	Media	2	
Eliminar evento registrado	Media	3	
Validar autoevaluación	Alta	2	
Modificar validar autoevaluación	Media	1	
Ver historial	Media	2	
Asignar evaluación integral	Baja	1	
Modificar evaluación integral	Baja	1	

	Sugerencia del sistema	Alta	3	
--	------------------------	------	---	--

Tareas de la Iteración 1

1. Registrar autoevaluación
2. Modificar autoevaluación
3. Registrar evidencia
4. Eliminar evidencia
5. Visualizar evidencia
6. Asignar evaluación
7. Modificar evaluación

Tabla 6 Tarea de ingeniería 1

Tarea de ingeniería	
Número de Tarea: 1	Número de la Historia de Usuario: 1
Nombre de la Tarea: Registrar autoevaluación	
Tipo de Tarea: Desarrollo	Estimación: 3 días
Fecha de inicio: 16/8/2022	Fecha de fin: 18/8/2022
Programador responsable: Javier Mancha Cabrera	
Descripción: Implementar una vista que permita registrar una autoevaluación al usuario estudiante.	

Tabla 7 Tarea de ingeniería 2

Tarea de ingeniería	
Número de Tarea: 2	Número de la Historia de Usuario: 2
Nombre de la Tarea: Modificar autoevaluación	

Tipo de Tarea: Desarrollo	Estimación: 2 días
Fecha de inicio: 19/8/2022	Fecha de fin: 20/8/2022
Programador responsable: Javier Mancha Cabrera	
Descripción: Implementar una vista que permita modificar una autoevaluación al usuario estudiante.	

Tabla 8 Tarea de ingeniería 3

Tarea de ingeniería	
Número de Tarea: 3	Número de la Historia de Usuario: 3
Nombre de la Tarea: Registrar evidencia	
Tipo de Tarea: Desarrollo	Estimación: 5 días
Fecha de inicio: 21/8/2022	Fecha de fin: 25/8/2022
Programador responsable: Javier Mancha Cabrera	
Descripción: Implementar una vista que permita registrar una evidencia al usuario administrador.	

Tareas de la Iteración 2

8. Registrar Profesor
9. Registrar usuario
10. Eliminar usuario
11. Buscar usuario
12. Registrar grupo docente

Tabla 9 Tarea de ingeniería 8

Tarea de ingeniería	
Número de Tarea: 8	Número de la Historia de Usuario: 8
Nombre de la Tarea: Registrar Profesor	
Tipo de Tarea: Desarrollo	Estimación: 1 día

Fecha de inicio: 6/9/2022	Fecha de fin: 6/9/2022
Programador responsable: Asael Caraballo Oquendo	
Descripción: Implementar una vista que permita registrar un profesor.	

Tabla 10 Tarea de ingeniería 9

Tarea de ingeniería	
Número de Tarea: 9	Número de la Historia de Usuario: 9
Nombre de la Tarea: Registrar usuario	
Tipo de Tarea: Desarrollo	Estimación: 3 días
Fecha de inicio: 7/9/2022	Fecha de fin: 9/9/2022
Programador responsable: Asael Caraballo Oquendo	
Descripción: Implementar una vista que permita modificar los profesores.	

Tabla 11 Tarea de ingeniería 10

Tarea de ingeniería	
Número de Tarea: 10	Número de la Historia de Usuario: 10
Nombre de la Tarea: Eliminar usuario	
Tipo de Tarea: Desarrollo	Estimación: 4 días
Fecha de inicio: 10/9/2022	Fecha de fin: 14/9/2022
Programador responsable: Asael Caraballo Oquendo	
Descripción: Implementar una vista que permita eliminar los profesores	

Tareas de la Iteración 3

- 13. Ver autoevaluación aprobada
- 14. Exportar autoevaluación aprobada
- 15. Autenticar Usuario
- 16. Registrar estudiante
- 17. Añadir nota
- 18. Modificar nota

- 19. Crear comisión
- 20. Asignar profesor principal
- 21. Asignar profesor guía
- 22. Crear evento
- 23. Modificar evento
- 24. Eliminar evento
- 25. Buscar eventos
- 26. Registrar evento
- 27. Eliminar evento registrado
- 28. Validar autoevaluación
- 29. Modificar validar autoevaluación
- 30. Ver historial
- 31. Asignar evaluación integral
- 32. Modificar evaluación integral
- 33. Sugerencia del sistema

Tabla 12 Tarea de ingeniería 13

Tarea de ingeniería	
Número de Tarea: 13	Número de la Historia de Usuario: 13
Nombre de la Tarea: Ver autoevaluación aprobada	
Tipo de Tarea: Desarrollo	Estimación: 3 días
Fecha de inicio: 21/9/2022	Fecha de fin: 23/9/2022
Programador responsable: Asael Caraballo Oquendo	
Descripción: Implementar una vista que permita visualizar las autoevaluaciones aprobadas.	

Tabla 13 Tarea de ingeniería 14

Tarea de ingeniería

Número de Tarea: 14	Número de la Historia de Usuario: 14
Nombre de la Tarea: Exportar autoevaluación aprobada	
Tipo de Tarea: Desarrollo	Estimación: 2 días
Fecha de inicio: 24/9/2022	Fecha de fin: 25/9/2022
Programador responsable: Asael Caraballo Oquendo	
Descripción: Implementar una vista que permita imprimir una autoevaluación aprobada.	

Tabla 14 Tarea de ingeniería 15

Tarea de ingeniería	
Número de Tarea: 15	Número de la Historia de Usuario: 15
Nombre de la Tarea: Autenticar usuario	
Tipo de Tarea: Desarrollo	Estimación: 1 día
Fecha de inicio: 26/9/2022	Fecha de fin: 26/9/2022
Programador responsable: Asael Caraballo Oquendo	
Descripción: Implementar una vista que permita autenticarse a los usuarios.	

2.6 PLAN DE ENTREGA

Tabla 15 Plan de entrega

	Iteración No.1	Iteración No.2	Iteración No.3
Cantidad de HU	7	5	17
Fecha de inicio	16/8/2022	6/9/2022	21/9/2022
Fecha de entrega	5/9/2022	20/9/2022	2/11/2022

Una vez realizado el plan de entrega se puede confirmar con el cliente que el proyecto durará 10 semanas aproximadamente.

2.7 TARJETAS CRC

Las tarjetas CRC se elaboran durante la fase de diseño de la metodología XP para describir las entidades existentes en la aplicación. El uso de este tipo de tarjetas es una técnica de modelado que permite identificar las clases, responsabilidades y colaboraciones. El objetivo es obtener un diseño simple, elegante y fácil de comprender por parte de los programadores.

Tabla 16 Tarjeta CRC 1

<u>TARJETA CRC</u>	
NOMBRE DE LA CLASE: Integralidad_estudiante	
RESPONDABILIDADES: Se encarga de crear una autoevaluación, subir evidencias y validar autoevaluación	COLABORADORES: Integralidad_usuario Integralidad_gd Integralidad_autoeval

Tabla 17 Tarjeta CRC 2

<u>TARJETA CRC</u>	
NOMBRE DE LA CLASE: Integralidad_profesor	
RESPONDABILIDADES: Se encarga de evaluar la autoevaluación, asignar comisión, asignar profesor principal,	COLABORADORES: Integralidad_usuario

asignar profesor guía, validar autoevaluación	Integralidad_gd Integralidad_autoevalnotas
---	---

Tabla 18 Tarjeta CRC 3

<u>TARJETA CRC</u>	
NOMBRE DE LA CLASE: Integralidad_gd	
RESPONDABILIDADES: Almacenar información de los usuarios	COLABORADORES: Integralidad_estudiante Integralidad_profesor Integralidad_comision

✓ Las demás tarjetas CRC serán colocadas en la sección de [Anexos](#).

2.8 ARQUITECTURA DEL SISTEMA

Aunque Django está fuertemente inspirado en la filosofía de desarrollo Modelo-Vista-Controlador, sus desarrolladores declaran públicamente que no se sienten especialmente atados a observar estrictamente ningún paradigma particular, y en cambio prefieren hacer "lo que les parece correcto". Como resultado, por ejemplo, lo que se llamaría "controlador" en un "verdadero" marco de trabajo MVC se llama en Django "vista", y lo que se llamaría "vista" se llama "plantilla", dando como resultado el patrón Modelo-Vista-Plantilla. (Muñiz, 2013)

- Vista: la capa de presentación se basa en plantillas HTML. Django presenta un template engine y un template loader muy potente que permite presentar al usuario diversas páginas HTML usando una base como plantilla. Esto es posible porque en

cada una de las plantillas se pueden introducir determinadas etiquetas Django que el template loader se encargará de interpretar.

- Controlador: Es lo que en Django se llama views. Puede llevar a la confusión, aunque Django lo llame views, éstas son las que actúan como controlador. Escritas en puro código Python cada view atenderá una petición HTML según el mapeo de URL del que ya se hablará más adelante.

- Modelo: Una de las partes más potentes de Django, su modelo de datos. Cada uno de los modelos creados se mapean en diferentes tablas en la base de datos. Esto permite aislar la base de datos del código y olvidarte de los diferentes select y updates a veces tan tediosos. (Pita, 2009)

2.9 DISEÑO DE LA BASE DATOS DEL SISTEMA

La imagen a continuación representa el diagrama entidad-relación de la base datos, generada desde el sistema gestor de base datos PostgreSQL a partir de los modelos utilizados en el marco de trabajo web Django:

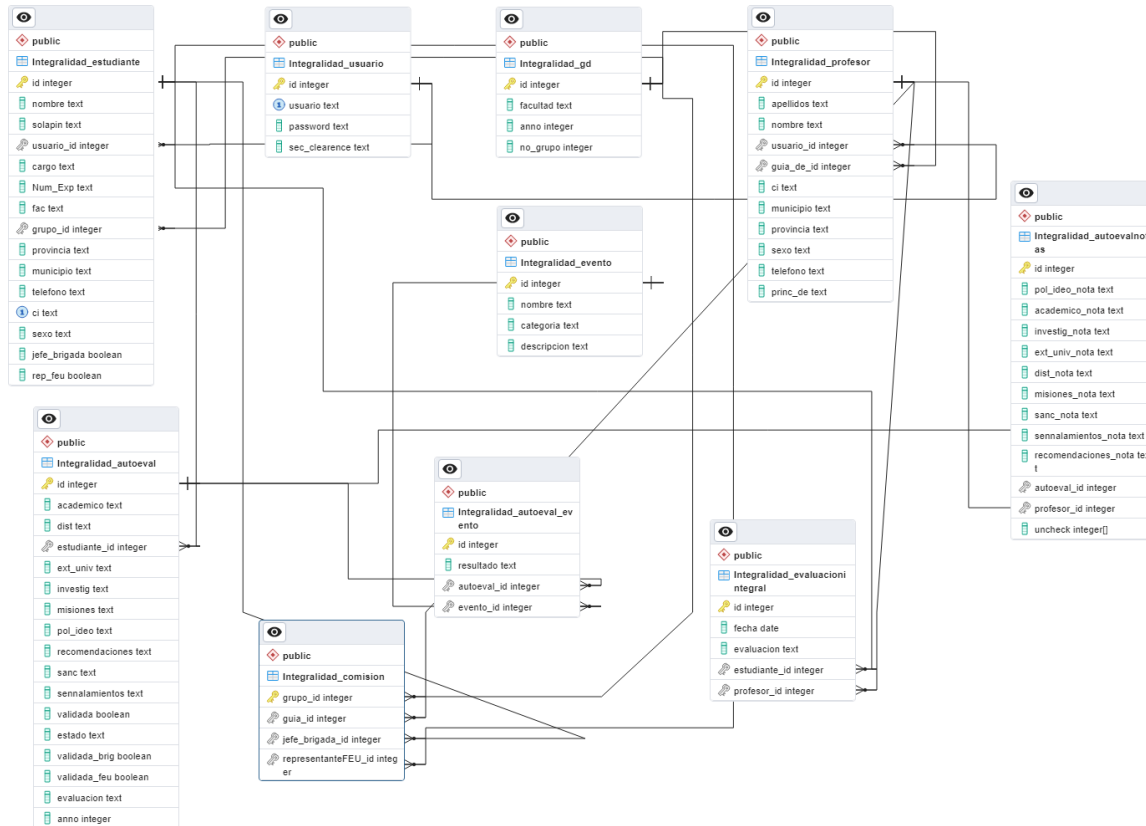


Ilustración 2 Diagrama entidad-relación

2.10 PATRONES DE DISEÑO

La utilización de patrones de diseño demuestra la madurez de un programador de software, este utiliza soluciones probadas para problemas concretos que han sido empleados en el pasado. Toma en cuenta que el dominio de patrones de diseño es una práctica que se tiene que perfeccionar y practicar. Lo más importante de estos es que evitan tener que reinventar, debido a que son escenarios identificados y su solución está documentada y probada por lo que no es necesario comprobar su efectividad. (Blancarte, 2016) El patrón fundamental empleado en el sistema es el modelo-vista-plantilla heredado de Django, en el cual se refleja el uso de una alta

cohesión y bajo acoplamiento, no obstante en la implementación se rescatan otros patrones clásicos de GRASP y GOF.

Patrones GRASP

Alta cohesión y bajo acoplamiento: La alta cohesión la tiene un objeto si todo lo que hace está bien delimitado dentro del mismo objeto. Además, para tener alta cohesión, debe de ser un objeto cuanto más pequeño mejor. Este concepto está relacionado inversamente con el acoplamiento, es decir, si aumenta la cohesión, suele disminuir el acoplamiento, y viceversa. El acoplamiento entre objetos se refiere a cuánta relación tienen los objetos entre sí. (Niñoles, 2022)

- Se evidencia en la vista G_User, todos sus métodos se encargan de la manipulación de datos de la clase usuario a la vez que recibe las peticiones de las plantillas y renderiza la respuesta del servidor hacia las mismas. Sirviendo de puente entre el servidor y las plantillas garantizando el bajo acoplamiento y una alta cohesión.

Experto: es el principio básico de asignación de responsabilidades. De este modo se obtiene un diseño más coherente y se disminuye el acoplamiento.

- Se evidencia en las clases Estudiante y AutoEval que mantienen una relación de 1 a muchos, mientras que cada una se encarga de almacenar la información pertinente a sí misma.

Creador: Identifica quién debe ser el responsable de la creación de objetos o clases.

- Todas las clases Form se encargan de la creación de los objetos del modelo.

Patrones GOF

Decorator: Añade dinámicamente nuevas responsabilidades a un objeto,

proporcionando una alternativa flexible a la herencia para extender la funcionalidad. (Gracia, 2013)

- Se evidencia en la clase `hybrid_manager` que se utiliza para añadir dinámicamente nuevas responsabilidades a las clases del modelo extendiendo así su funcionalidad

Mediator: Define un objeto que encapsula cómo interactúan un conjunto de objetos. Promueve un bajo acoplamiento al evitar que los objetos se refieran unos a otros explícitamente, y permite variar la interacción entre ellos de forma independiente.

- Se evidencia en la clase `Autoeval_Evento` que mapea la relación de mucho a mucho de la clase `AutoEval` y la clase `Evento`

2.11 PROTOTIPO DE INTERFAZ DE USUARIO

Los prototipos ayudan a identificar, comunicar y probar un producto antes de crearlo.

Prototipo de interfaz de acceso

<p>Acceder</p> <p>Usuario:</p> <input type="text"/>	<p>Bienvenido al sistema de Gestión de la Integralidad Estudiantil</p>
<p>Contraseña:</p> <input type="text"/>	
<input type="button" value="Entrar"/>	<p>UCI 2022</p>

Ilustración 3 Interfaz de acceso

Prototipo interfaz página principal

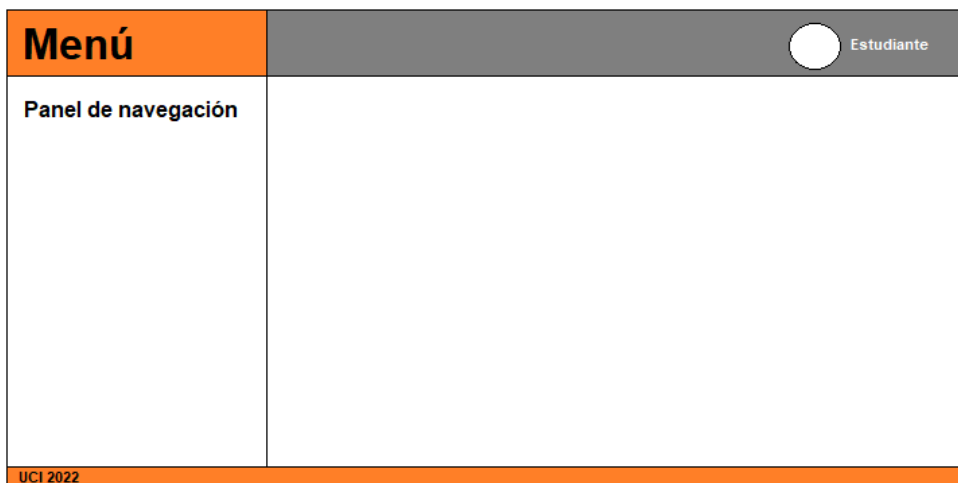


Ilustración 4 Interfaz prototipo de menú principal

Como podemos observar los prototipos se ajustan al diseño propuesto de colores anaranjado y blanco.

2.12 CONCLUSIONES PARCIALES

- Con la realización de este capítulo se establecieron los requisitos no funcionales y funcionales contando el sistema actual con un total de 15 y 29 respectivamente, se conformaron las historias de usuario por cada requisito funcional del sistema, las tarjetas CRC, el plan y estimación de las historias de usuario, se aplicó un plan de entrega al cliente el cual será de aproximadamente 10 semanas. Se especificó y abordó la arquitectura Modelo-Vista-Plantilla, así como los patrones de diseño, finalizando el capítulo con los prototipos de interfaz a elaborar.

3 CAPÍTULO III. IMPLEMENTACIÓN Y PRUEBAS

En este capítulo se aborda sobre los estándares de codificación, las diferentes pruebas realizadas al software y que son exigidas por la metodología XP para el buen funcionamiento y calidad de la propuesta desarrollada. Se detalla con exactitud los diferentes resultados alojados por cada prueba.

3.1 IMPLEMENTACIÓN

Las funcionalidades son desarrolladas en esta fase, generando al final de cada una un entregable funcional que implementa las historias de usuario asignadas a la iteración. Como las historias de usuario no tienen suficiente detalle como para permitir su análisis y desarrollo, al principio de cada iteración se realizan las tareas necesarias de análisis, recabando con el cliente todos los datos que sean necesarios.

El cliente, por lo tanto, también debe participar activamente durante esta fase del ciclo. Las iteraciones son también utilizadas para medir el progreso del proyecto. Una iteración terminada sin errores es una medida clara de avance.

3.1.1 Diagrama de despliegue

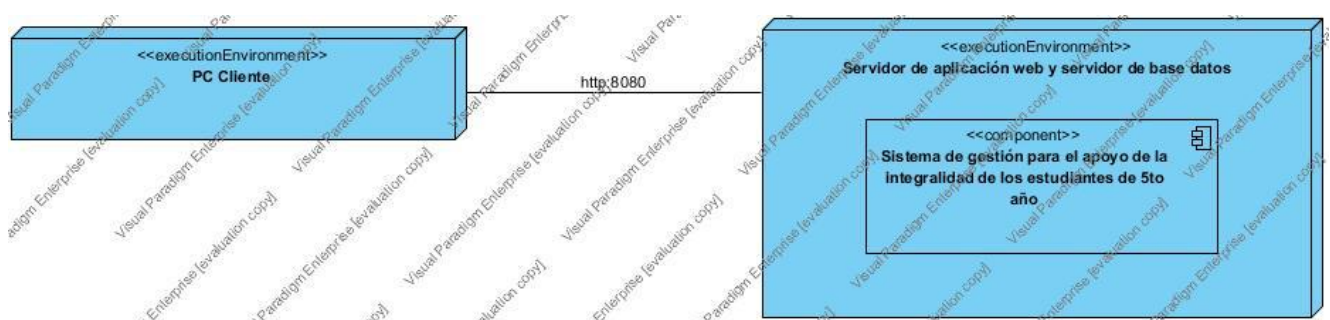


Ilustración 5 Diagrama de despliegue

3.1.2 Estándares de codificación

Los estándares de codificación o programación son términos que describen convenciones para escribir código fuente en los lenguajes de programación. En este caso se utilizó el estándar de codificación de Django.

- Grosor de línea: Cada línea de código no debe exceder los 80 caracteres en la medida de lo posible (en circunstancias especiales, puede exceder ligeramente los 80, pero la más larga no puede exceder los 120)

```
class EvaluacionIntegral(models.Model):
    estudiante = models.ForeignKey(Estudiante, on_delete=models.CASCADE)
    profesor = models.ForeignKey(Profesor, on_delete=models.CASCADE)
    fecha = models.DateField()
    evaluacion = models.TextField(
        max_length=30, choices=evaluacion_choices.choices,
        default=evaluacion_choices.unknown)
```

- Comillas: En pocas palabras, el lenguaje natural usa comillas dobles y las etiquetas de máquina usan comillas simples, por lo que la mayor parte del código debe usar comillas simples

```
context['autoeval'] = ae
context['notas'] = notas
context['uncheck'] = uncheck
context['pol_saved'] = pol_saved
context['aca_saved'] = aca_saved
context['inv_saved'] = inv_saved
context['ext_saved'] = ext_saved
```

- Declaración de importación: La declaración de importación debe escribirse en líneas separadas

```
from datetime import date
from django.shortcuts import render, redirect
from django.http import HttpResponse
```

```
from Integralidad.models import AutoEval, Evento, EvaluacionIntegral,
Autoeval_Evento
from Integralidad.forms import AutoEval_Form, AutoEvalNotas_Form
from Integralidad import DB.
```

- Espacio: Un espacio a cada lado del operador binario [=, -, +=, ==,>, in,is not, and]. En la lista de parámetros de la función, no agregue espacios a ambos lados del signo igual predeterminado.

```
if(not autoevals):
    autoeval = AutoEval(estudiante=estudiante, anno=estudiante.grupo.anno)
    autoeval.save()
    context["autoeval"] = autoeval
    return render(request,
                  "Integralidad/Panel_Estudiante/Inspect_autoeval.html", context)
else:
    new = True
    for x in autoevals:
        if(x.anno == estudiante.grupo.anno):
            new = False
            autoeval = x
```

- En la lista de parámetros de la función, debe haber un espacio después.

```
def CheckPersistence(our_user, user_info):
    updated = False
    try:
        uci_grupo = int(user_info['area']['nombreArea'][-4:])
    except:
        our_user.delete()
        return False
```

3.2 ESTRATEGIA DE PRUEBAS

La Metodología XP propone que las pruebas de software sean realizadas al término de cada iteración, garantizando el funcionamiento deseado y la aceptabilidad por el

cliente para realizar una entrega funcional y acorde a las exigencias de un producto con calidad.

Las dos pruebas exigidas por la metodología, por su importancia y agilidad en el proceso; son las pruebas unitarias y de aceptación. Se hicieron las pruebas unitarias al código al finalizar cada iteración e igualmente se realizaron las pruebas de aceptación.

3.2.1 Pruebas unitarias

Las pruebas unitarias son pruebas de caja blanca donde los componentes individuales del software se someten a pruebas. El propósito de estas es asegurar que cada unidad de trabajo funciona de forma correcta individualmente, responde como se espera que deba responder, o falle como y cuando se supone que debe fallar. Se estima que las pruebas unitarias deben probar la unidad mínima de trabajo de un programa que es aquella que devuelve un valor o produce un cambio en el estado del programa, generalmente hablaríamos de un solo método o una función. (Alvarado, 2020)

Todos los módulos deben de pasar las pruebas unitarias antes de ser liberados o publicados. Todo código liberado que pase correctamente las pruebas unitarias, es lo que habilita que funcione la propiedad colectiva del código. (SINTYA MILENA MELÉNDEZ VALLADAREZ, 2016)

Para la realización de dichas pruebas se utilizó el marco de trabajo Django, haciendo uso de su librería Testcase, en el siguiente código se refleja un segmento realizado a las funcionalidades de los registros en la base datos encontrado en el archivo test.py:

```
from django.test import TestCase, Client
from django.urls import reverse
from .models import usuario, Estudiante, Profesor, AutoEval, GD, EvaluacionIntegral, Evento
```



```
import datetime

from django.utils.encoding import smart_bytes

class LoginTests(TestCase):

    def setUp(self):

        self.client = Client()

        self.login_url = reverse('login')

    def test_user_login_GET(self):

        response = self.client.get(self.login_url)

        self.assertEqual(response.status_code, 200)

        self.assertTemplateUsed(

            response, 'Integralidad/Login.html')

    def test_estudiante_login_POST(self):

        response = self.client.post(self.login_url, {

            'user': 'danielcp',

            'pass': 'a'
```

```
    })

    self.assertEqual(response.status_code, 200)

    self.assertTemplateUsed(
        response, 'Integralidad/estindex.html')

    # comprobando la creacion del usuario
    usuario_creado = usuario.objects.get(usuario='danielcp')
    exists = usuario.objects.filter(usuario='danielcp').exists()
    self.assertEqual(exists, True)

    # comprobando la creacion del estudiante
    exists = Estudiante.objects.filter(usuario=usuario_creado).exists()
    self.assertEqual(exists, True)

    # comprobando la creacion de un nuevo grupo docente para el usuario
    estudiante_creado = Estudiante.objects.get(usuario=usuario_creado)
    exists = GD.objects.filter(id=estudiante_creado.grupo_id).exists()
    self.assertEqual(exists, True)

def test_estudiante_login_POST_wrong_credenciales(self):
```

```
response = self.client.post(self.login_url, {
    'user': 'danielcpasd',
    'pass': 'a'
})

self.assertEqual(response.status_code, 200)

self.assertTemplateUsed(
    response, 'Integralidad/Login.html')

# comprobando que no se creo un nuevo usuario
exists = usuario.objects.filter(usuario='danielcp').exists()
self.assertEqual(exists, False)

def test_normal_profesor_login_POST(self):

    response = self.client.post(self.login_url, {
        'user': 'yluguen',
        'pass': 'a'
    })

    self.assertEqual(response.status_code, 200)

    self.assertTemplateUsed(
```

```
response, 'Integralidad/testeee.html')

# comprobando la creacion del usuario
usuario_creado = usuario.objects.get(usuario='yluguen')
exists = usuario.objects.filter(usuario='yluguen').exists()
self.assertEqual(exists, True)

# comprobando el credencial de seguridad User
self.assertEqual(usuario_creado.sec_clearance, "User")

# comprobando la creacion del profesor
exists = Profesor.objects.filter(usuario=usuario_creado).exists()
self.assertEqual(exists, True)

def test_decano_login_POST(self):

    response = self.client.post(self.login_url, {
        'user': 'yaimi',
        'pass': 'a'
    })

    self.assertEqual(response.status_code, 200)
    self.assertTemplateUsed(
```

```
response, 'Integralidad/decanoindex.html')

# comprobando la creacion del usuario
usuario_creado = usuario.objects.get(usuario='yaimi')
exists = usuario.objects.filter(usuario='yaimi').exists()
self.assertEqual(exists, True)

# comprobando el credencial de seguridad Decano
self.assertEqual(usuario_creado.sec_clearance, "Decano")

# comprobando la creacion del decano
exists = Profesor.objects.filter(usuario=usuario_creado).exists()
self.assertEqual(exists, True)

def test_principal_login_POST(self):

    response = self.client.post(self.login_url, {
        'user': 'lsgomez',
        'pass': 'a'
    })

    self.assertEqual(response.status_code, 200)
    self.assertTemplateUsed(
```

```
response, 'Integralidad/index.html')

# comprobando la creacion del usuario
usuario_creado = usuario.objects.get(usuario='lsgomez')
exists = usuario.objects.filter(usuario='lsgomez').exists()
self.assertEqual(exists, True)

# comprobando el credencial de seguridad Principal
self.assertEqual(usuario_creado.sec_clearance, "Principal")

# comprobando la creacion del decano
exists = Profesor.objects.filter(usuario=usuario_creado).exists()
self.assertEqual(exists, True)
```

Además, se puede observar en la siguiente figura los resultados arrojados a través de la consola del IDE Visual Studio Code donde fueron ejecutados los tests o pruebas unitarias. La misma muestra la admisión de todos los scripts de pruebas realizados.

```

System check identified 12 issues (0 silenced).
test_asig_event_to_student (Integralidad.tests.Asignacion_de_eventos) ... ok
test_asig_event_to_student_missing_autoeval (Integralidad.tests.Asignacion_de_eventos) ... ok
test_G_Autoeval_Insertar (Integralidad.tests.AutoevalViews) ... ok
test_G_Autoeval_Modificar_GET (Integralidad.tests.AutoevalViews) ... ok
test_G_Autoeval_Modificar_POST (Integralidad.tests.AutoevalViews) ... ok
test_evaluar_autoeval (Integralidad.tests.EvaluacionTests) ... ok
test_evaluar_autoeval_modificar (Integralidad.tests.EvaluacionTests) ... ok
test_create_new_event_GET (Integralidad.tests.EventosTests) ... ok
test_listar_eventos (Integralidad.tests.EventosTests) ... ok
test_decano_login_POST (Integralidad.tests.LoginTests) ... ok
test_estudiante_login_POST (Integralidad.tests.LoginTests) ... ok
test_estudiante_login_POST_wrong_credenciales (Integralidad.tests.LoginTests) ... ok
test_normal_profesor_login_POST (Integralidad.tests.LoginTests) ... ok
test_principal_login_POST (Integralidad.tests.LoginTests) ... ok
test_user_login_GET (Integralidad.tests.LoginTests) ... ok

-----
Ran 15 tests in 15.974s
    
```

Ilustración 6 Resultados pruebas unitarias

3.2.2 Pruebas de aceptación

Las pruebas de aceptación son una parte integral del desarrollo incremental. Todas las historias de usuarios están respaldadas por pruebas de aceptación, que son definidas por el cliente in situ y significaban la satisfacción del mismo con el producto desarrollado. Estas pruebas obligan al cliente a profundizar en el conocimiento de su dominio y declarar con precisión qué debe hacer la aplicación en circunstancias específicas, por esto, el cliente es la persona adecuada para diseñarlas. En efecto, las pruebas de aceptación marcan el camino a seguir en cada iteración, indicándole al equipo de desarrollo la dirección a seguir, así como los puntos o funcionalidades en que se debe poner el mayor esfuerzo y atención. (SINTYA MILENA MELÉNDEZ VALLADAREZ, 2016)

Como técnica para las pruebas de aceptación se utilizó:

- Técnica de la Partición de Equivalencia: esta técnica divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.

La partición equivalente es un método de prueba de caja negra que divide el dominio de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Un caso de prueba ideal descubre de forma inmediata una clase de error que de otro modo requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar. (Patton, Software Testing, 2005)

A continuación, se especifican algunas de las pruebas de aceptación realizadas al sistema:

Pruebas de aceptación para la Iteración 1

Tabla 19 Caso de prueba de aceptación 1

Caso de prueba de aceptación	
Código: HU1_P1	Historia de usuario:1
Nombre: Registrar autoevaluación.	
Condiciones de ejecución: El usuario tiene que estar autenticado en el sistema como estudiante	
Pasos de ejecución: el usuario selecciona el menú de autoevaluación	
Resultados esperados: el sistema registra una autoevaluación del estudiante.	
Evaluación de la prueba: Satisfactorio	

Tabla 20 Caso de prueba de aceptación 2

Caso de prueba de aceptación	
Código: HU2_P1	Historia de usuario:2
Nombre: Modificar autoevaluación.	

Condiciones de ejecución: El usuario tiene que estar autenticado en el sistema como estudiante.
Pasos de ejecución: el usuario rellena los espacios de la autoevaluación.
Resultados esperados: se modifica la autoevaluación del estudiante.
Evaluación de la prueba: Satisfactorio

Tabla 21 Caso de prueba de aceptación 3

Caso de prueba de aceptación	
Código: HU2_P2	Historia de usuario: 2
Nombre: Modificar autoevaluación	
Condiciones de ejecución: El usuario tiene que estar autenticado en el sistema como estudiante.	
Pasos de ejecución: el usuario rellena los espacios de la autoevaluación y deja espacios en blanco.	
Resultados esperados: se modifica la autoevaluación del estudiante.	
Evaluación de la prueba: Satisfactorio	

Pruebas de aceptación para la Iteración 2

Tabla 22 Caso de prueba de aceptación 4

Caso de prueba de aceptación	
Código: HU8_P1	Historia de usuario: 8
Nombre: Registrar Profesor	
Condiciones de ejecución: las credenciales del usuario deben estar registradas en la base datos.	
Pasos de ejecución: el usuario se autentica con sus datos.	
Resultados esperados: Se registra un profesor.	

Evaluación de la prueba: Satisfactorio

Tabla 23 Caso de prueba de aceptación 5

Caso de prueba de aceptación	
Código: HU9_P1	Historia de usuario:9
Nombre: Registrar usuario	
Condiciones de ejecución: las credenciales del usuario deben estar registradas en la base datos.	
Pasos de ejecución: el usuario se autentica con sus datos.	
Resultados esperados: Se registra un profesor.	
Evaluación de la prueba: Satisfactorio	

Tabla 24 Caso de prueba de aceptación 6

Caso de prueba de aceptación	
Código: HU10_P1	Historia de usuario:10
Nombre: Eliminar usuario	
Condiciones de ejecución: El usuario tiene que estar autenticado en el sistema como administrador.	
Pasos de ejecución: el usuario accede al panel de usuario y selecciona los usuarios a eliminar, luego presiona el botón de eliminar y confirma la interfaz emergente.	
Resultados esperados: se elimina o eliminan los usuarios seleccionados.	
Evaluación de la prueba: Satisfactorio	

Pruebas de aceptación para la Iteración 3

Tabla 25 Caso de prueba de aceptación 7

Caso de prueba de aceptación	
Código: HU10_P2	Historia de usuario:10
Nombre: Eliminar usuario	

Condiciones de ejecución: El usuario tiene que estar autenticado en el sistema como administrador.
Pasos de ejecución: el usuario accede al panel de usuario y selecciona los usuarios a eliminar, luego presiona el botón de eliminar y cancela la interfaz emergente.
Resultados esperados: no se modifican los datos.
Evaluación de la prueba: Satisfactorio

Tabla 26 Caso de prueba de aceptación 8

Caso de prueba de aceptación	
Código: HU14_P1	Historia de usuario: 14
Nombre: Exportar autoevaluación aprobada	
Condiciones de ejecución: el usuario debe tener visible el botón de exportar PDF.	
Pasos de ejecución: el usuario selecciona el botón descargar PDF.	
Resultados esperados: se descarga el PDF.	
Evaluación de la prueba: Satisfactorio	

Tabla 27 Caso de prueba de aceptación 9

Caso de prueba de aceptación	
Código: HU15_P1	Historia de usuario: 15
Nombre: Autenticar Usuario.	
Condiciones de ejecución: las credenciales del usuario deben estar registradas en la base datos.	
Pasos de ejecución: el usuario introduce usuario y contraseña.	
Resultados esperados: Se autentica en el sistema con el rol correspondiente.	
Evaluación de la prueba: Satisfactorio.	

- ✓ Los demás casos de prueba de aceptación serán colocados en la sección de [Anexos](#).

3.3 RESULTADOS DE LAS PRUEBAS

Al realizarse las pruebas se pudieron identificar una serie de no conformidades. Estas se pueden observar en la siguiente tabla.

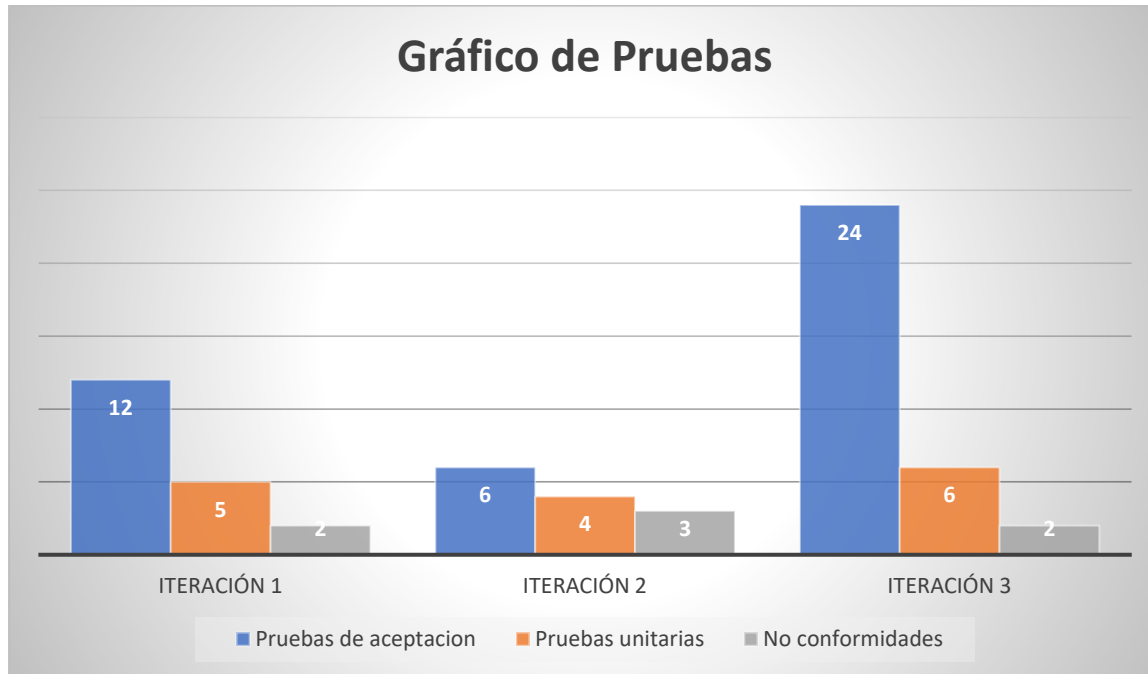
Tabla 28 Resultados de las pruebas

No. NC	Requisito funcional	Descripción	Complejidad	Estado
1	RF 2	Al escribir la descripción de cada campo se salía de los límites del diseño de la web.	Baja	Resuelto
2	RF 3	Se podía subir archivos en cualquier formato.	Media	Resuelto
3	RF 14	Los datos no salían completos en el archivo PDF descargado.	Alta	Resuelto

4	RF 18	Al escribir la descripción en el campo se salía de los límites del diseño de la web.	Media	Resuelto
5	RF 2	Los cuadros de texto no aceptaban saltos de línea	Media	Resuelto
6	RF 18	El cuadro de texto no aceptaba saltos de línea	Media	Resuelto
7	-	Se detecto un error ortográfico en el panel principal	Baja	Resuelto

En la siguiente figura se muestran los datos correspondientes a cada iteración de prueba por las que transitó el módulo.

Tabla 29 Gráfico de pruebas



En resumen: fueron realizadas tres iteraciones de prueba, de las cuales se obtuvieron un total de 7 no conformidades (NC) significativas: 4 con complejidad media, 2 bajas y una de complejidad alta. En la primera iteración fueron realizados 12 casos de prueba de caja negra y 5 casos de pruebas unitarias, dando como resultado 2 no conformidades y se resolvieron. Luego se realizó una segunda iteración donde se desarrollaron 6 casos de prueba de caja negra y 4 casos de pruebas unitarias, dando como resultado 3 no conformidades y se resolvieron. En la tercera iteración se desarrollaron 24 casos de caja negra y 6 casos de pruebas unitarias, se detectaron 2 no conformidades y se resolvieron.

3.4 CONCLUSIONES PARCIALES

- Se establecieron los estándares de código a usar basados en estándares de Django. Se realizaron las pruebas correspondientes al software haciendo uso de las herramientas dadas por nuestro marco de trabajo web y metodología, contando 57 pruebas en total ejecutadas sobre los requisitos funcionales establecidos anteriormente permitiendo comprobar los errores existentes y mejorar la calidad de los resultados. Se utilizó la técnica de partición de equivalencia para las pruebas de aceptación y flujo de datos para unitarias, gracias a estas se pudo captar varias no conformidades en el sistema y se logró la aceptabilidad de los requisitos descritos en cada iteración.

4 CONCLUSIONES

Considerando los resultados arribados luego de realizar este informe se llegó a la conclusión de:

- El análisis sistematizado de los referentes teóricos y plataformas homólogas de integralidad determinó que estas últimas no pueden ser propuestas como solución emergente a la problemática. Sin embargo, fueron seleccionados como solución parcial para el desarrollo del sistema de gestión de la integralidad estudiantil.
- Los resultados de la investigación que describe este informe demostraron la viabilidad de la solución propuesta para la integralidad estudiantil, cuyo desarrollo contribuye al proceso de evidencias que se tienen en cuenta en la evaluación del estudiante.

5 RECOMENDACIONES

- Implementar un módulo de reportes.

6 REFERENCIAS BIBLIOGRÁFICAS

(s.f.).

(21 de septiembre de 2021). Obtenido de <https://www.aden.org/business-magazine/metodologias-agiles/>

Alvarado, I. (31 de marzo de 2020). *LA IMPORTANCIA DE LAS PRUEBAS UNITARIAS PARA COMPROBAR FRAGMENTOS DE CÓDIGO*. Obtenido de LA IMPORTANCIA DE LAS PRUEBAS UNITARIAS PARA COMPROBAR FRAGMENTOS DE CÓDIGO: <https://ceroideas.es/la-importancia-de-las-pruebas-unitarias-para-comprobar-fragmentos-de-codigo/>

Álvarez, J. (s.f.). *LA INTEGRALIDAD DE LA EDUCACIÓN: EN BUSCA DE UN MODELO AXIOLÓGICO*.

Ankush. (s.f.). *Los 14 mejores marcos/bibliotecas CSS para desarrolladores front-end*. Obtenido de Los 14 mejores marcos/bibliotecas CSS para desarrolladores front-end : <https://geekflare.com/es/best-css-frameworks/>

Arsys blog. (13 de junio de 2018). *Qué es PostgreSQL y por que llevarlo a Cloud? | Arsys*. Obtenido de Qué es PostgreSQL y por que llevarlo a Cloud?: <https://www.arsys.es/blog/soluciones/postgresql-servidores>

Blancarte, O. (octubre de 2016). *Introducción a los patrones de diseño: Un enfoque práctico*. México.

Carmona, D. S.-M. (20 de Septiembre de 2019). *Scrum: cuáles son sus características | OpenWebinars*. Obtenido de Scrum: cuáles son sus características: <https://openwebinars.net/blog/scrum-caracteristicas/>

Dorta, D. A. (2020). *Framework para agilizar la aplicación de técnicas basadas en Deep Learning*.

Ferré, A. (03 de diciembre de 2018). *Sublime Text: Información y trucos para empezar desde cero*. Obtenido de Sublime Text: Información y trucos para empezar desde cero: <https://cipsa.net/sublime-text-informacion-y-trucos-para-empezar-desde-cero/>

García, J. (s.f.). *PyCharm, un potente IDE para crear programas con Python | Linux Adictos*. Obtenido de PyCharm, un potente IDE para crear programas con Python: <https://www.linuxadictos.com/pycharm-un-potente-ide-para-crear-programas-con-python.html>

Gracia, L. (2 de enero de 2013). *Un poco de Patrones de Diseño GoF (Gang of Four)*. Obtenido de Un poco de Patrones de Diseño GoF (Gang of Four) : <https://unpocodejava.com/2013/01/02/un-poco-de-patrones-de-diseno-gof-gang-of-four/>

- Infante Costa, G. (23 de mayo de 2019). *Akados, un Sistema Automatizado para la Gestión Académica*. Obtenido de Serie Científica De La Universidad De Las Ciencias Informáticas: <https://publicaciones.uci.cu/index.php/serie/article/view/253/149>
- IONOS Cloud S.L.U. (s.f.). *Las 6 mejores herramientas UML - IONOS*. Obtenido de 6 herramientas UML para cualquier ocasión: <https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/las-mejores-herramientas-uml/>
- Kanbanize. (s.f.). *Qué es Kanban: Definición, Características y Ventajas*. Obtenido de Qué es Kanban: Definición, Características y Ventajas: <https://kanbanize.com/es/recursos-de-kanban/primeros-pasos/que-es-kanban>
- Luca, D. D. (s.f.). *Visual Studio Code: características principales - Damián de Luca*. Obtenido de Visual Studio Code: características principales: <https://damiandeluca.com.ar/visual-studio-code-caracteristicas-principales>
- Lucid Software Inc. (s.f.). *¿Qué es el lenguaje unificado de modelado (UML)? | Lucidchart*. Obtenido de Qué es el lenguaje unificado de modelado (UML): <https://www.lucidchart.com/pages/es/que-es-el-lenguaje-unificado-de-modelado-uml>
- Luis Ruben Lima Mateo, R. E. (junio de 2019). *Plataforma SOFIA para la distribución electrónica*. La Habana, La Habana, Cuba.
- Luis Ruben Lima Mateo, R. E. (junio de 2019). *Plataforma SOFIA para la distribución electrónica de libros digitales en Cuba*. La Habana.
- Mozilla Foundation. (s.f.). *Introducción a Django - Aprende sobre desarrollo web | MDN*. Obtenido de Introducción a Django: <https://developer.mozilla.org/es/docs/Learn/Server-side/Django/Introduction>
- Muñiz, A. (18 de julio de 2013). *Django: Desarrollo web en Python*. Obtenido de Django: Desarrollo web en Python: <https://dslab.es/desarrollo-web/django-desarrollo-web-en-python/>
- Muradas, Y. (08 de Marzo de 2018). *Conoce las 3 metodologías ágiles más usadas | OpenWebinars*. Obtenido de Conoce las 3 metodologías ágiles más usadas: <https://openwebinars.net/blog/conoce-las-3-metodologias-agiles-mas-usadas/>
- Niños, J. (3 de febrero de 2022). *Los patrones del diseño software GRASP*. Obtenido de Los patrones del diseño software GRASP: <https://jnjsite.com/los-patrones-del-diseno-software-grasp/>
- Patton, R. (2005). *Software Testing*. Sams Publishing.
- Patton, R. (2005). *Software Testing*. Sams Publishing.

- Pita, F. A. (septiembre de 2009). *McLibre.org*. Obtenido de <https://www.mclibre.org/descargar/docs/revistas/linvix/linvix-04-es-200909.pdf>
- Robledano, A. (22 de Julio de 2019). *Qué es C++: Características y aplicaciones | OpenWebinars*. Obtenido de Qué es C++: Características y aplicaciones: <https://openwebinars.net/blog/que-es-cpp/>
- rockcontent blog. (12 de Abril de 2020). *¿Bootstrap qué es, para qué sirve y cómo instalarlo?* Obtenido de Bootstrap: guía para principiantes de qué es, por qué y cómo usarlo: <https://rockcontent.com/es/blog/bootstrap/>
- Rodriguez, J. L. (26 de enero de 2021). *Qué es Tailwind y por qué usarlo - Las Grandes Ventajas*. Obtenido de Qué es Tailwind y por qué usarlo: <https://www.atsistemas.com/es/blog/que-es-tailwind>
- Rómmel, F. (s.f.). *SQLite: La Base de Datos Embebida | SG Buzz*. Obtenido de SQLite: La Base de Datos Embebida: <https://sg.com.mx/revista/17/sqlite-la-base-datos-embebida>
- Santander Universidades . (s.f.). *¿Qué es Python? | Blog Becas Santander* . Obtenido de Python: qué es y por qué deberías aprender a utilizarlo : <https://www.becas-santander.com/es/blog/python-que-es.html>
- SINTYA MILENA MELÉNDEZ VALLADAREZ, M. E. (28 de enero de 2016). METODOLOGIA ÁGIL DE DESARROLLO DE SOFTWARE PROGRAMACION EXTREMA. MANAGUA, NICARAGUA.
- Soler-Adillo, J. (24 de mayo de 2017). *Tres IDEs para programar: Processing, Eclipse y Visual Studio*. Obtenido de Tres IDEs para programar: Processing, Eclipse y Visual Studio: <https://mosaic.uoc.edu/2017/05/24/analisis-de-distintos-ides-para-programar/>
- Souza, I. d. (9 de marzo de 2020). Obtenido de <https://rockcontent.com/es/blog/php/>
- TechTarget. (s.f.). *¿Qué es MySQL? - Definición en WhatIs.com*. Obtenido de MySQL: <https://www.computerweekly.com/es/definicion/MySQL>
- Tena, M. (s.f.). *¿Qué es la metodología 'agile'?* Obtenido de ¿Qué es la metodología 'agile'? : <https://www.bbva.com/es/metodologia-agile-la-revolucion-las-formas-trabajo/>
- Universia. (9 de marzo de 2022). *Los lenguajes de programación más usados en la actualidad*. Obtenido de Los lenguajes de programación más usados en la actualidad: <https://www.universia.net/es/actualidad/empleo/lenguajes-programacion-mas-usados-actualidad-1136443.html>

VIEWNEXT. (29 de noviembre de 2018). *¿Qué ventajas aporta Extreme Programming?* Obtenido de *¿Qué ventajas aporta Extreme Programming?*: <https://www.viewnext.com/ventajas-extreme-programming/>

7 ANEXOS

Tabla 30 HU4

Historias de usuario	
Número: 4	Nombre del requisito: Eliminar evidencia.
Programador: Javier Mancha Cabrera	Iteración asignada: 1
Prioridad: alta	Tiempo estimado: 3 días
Riesgo en desarrollo: N/A	Tiempo real: 3 días
Descripción: El sistema debe permitir que el estudiante elimine sus evidencias.	
Observaciones: N/A	

Tabla 31 HU5

Historias de usuario	
Número: 5	Nombre del requisito: Visualizar evidencia.

Programador: Asael Caraballo Oquendo	Iteración asignada: 1
Prioridad: baja	Tiempo estimado: 2 días
Riesgo en desarrollo: N/A	Tiempo real: 2 días
Descripción: El sistema debe permitir al profesor guía, presidente de brigada, representante de la FEU y estudiantes ver las evidencias subidas por los estudiantes.	
Observaciones: N/A	

Tabla 32 HU6

Historias de usuario	
Número: 6	Nombre del requisito: Asignar evaluación
Programador: Asael Caraballo Oquendo	Iteración asignada: 1
Prioridad: Alta	Tiempo estimado: 3 días
Riesgo en desarrollo: N/A	Tiempo real: 3 días
Descripción: El sistema debe permitir al profesor guía dar una evaluación a los estudiantes de satisfactoria o no satisfactoria.	
Observaciones: N/A	

Tabla 33 HU7

Historias de usuario

Número: 7	Nombre del requisito: Modificar evaluación
Programador: Asael Caraballo Oquendo	Iteración asignada: 1
Prioridad: Alta	Tiempo estimado: 2 días
Riesgo en desarrollo: N/A	Tiempo real: 2 días
Descripción: El sistema debe permitir el profesor guía modificar los datos de la evaluación del estudiante: satisfactoria o no satisfactoria.	
Observaciones: N/A	

Tabla 34 HU8

Historias de usuario	
Número: 8	Nombre del requisito: Registrar Profesor
Programador: Asael Caraballo Oquendo	Iteración asignada: 2
Prioridad: alta	Tiempo estimado: 1 días
Riesgo en desarrollo: N/A	Tiempo real: 1 días
Descripción: El sistema debe permitir registrar los datos del profesor a partir de los campos: nombre, apellidos, usuario, provincia, municipio, teléfono, CI, sexo.	
Observaciones: N/A	

Tabla 35 HU9

Historias de usuario	
Número: 9	Nombre del requisito: Registrar usuario
Programador: Asael Caraballo Oquendo	Iteración asignada: 2
Prioridad: Alta	Tiempo estimado: 3 días
Riesgo en desarrollo: N/A	Tiempo real: 3 días
Descripción: El sistema debe permitir registrar los datos del usuario: usuario, contraseña, rol.	
Observaciones: N/A	

Tabla 36 HU10

Historia de usuario	
Número: 10	Nombre del requisito: Eliminar usuario
Programador: Javier Mancha Cabrera	Iteración asignada: 2
Prioridad: alta	Tiempo estimado: 4 días
Riesgo en desarrollo: N/A	Tiempo real: 4 días
Descripción: El sistema debe permitir eliminar los datos del usuario.	
Observaciones: N/A	

Tabla 37 HU11

Historia de usuario	
Número: 11	Nombre del requisito: Buscar usuario
Programador: Javier Mancha Cabrera	Iteración asignada: 2
Prioridad: media	Tiempo estimado: 3 días
Riesgo en desarrollo: N/A	Tiempo real: 3 días
Descripción: El sistema debe permitir buscar los datos del usuario	
Observaciones: N/A	

Tabla 38 HU12

Historia de usuario	
Número: 12	Nombre del requisito: Registrar grupo docente
Programador: Javier Mancha Cabrera	Iteración asignada: 2
Prioridad: Alta	Tiempo estimado: 3 días
Riesgo en desarrollo: N/A	Tiempo real: 3 días
Descripción: El sistema debe permitir registrar los datos del grupo docente a partir de: facultad, año académico, número grupo.	
Observaciones: N/A	

Tabla 39 HU13

Historia de usuario	
Número: 13	Nombre del requisito: Ver autoevaluación aprobada
Programador: Javier Mancha Cabrera	Iteración asignada: 3
Prioridad: baja	Tiempo estimado: 3 días
Riesgo en desarrollo: N/A	Tiempo real: 3 días
Descripción: El sistema debe permitir visualizar la evaluación aprobada.	
Observaciones: N/A	

Tabla 40 HU14

Historia de usuario	
Número: 14	Nombre del requisito: Exportar autoevaluación aprobada
Programador: Javier Mancha Cabrera	Iteración asignada: 3
Prioridad: Alta	Tiempo estimado: 2 días
Riesgo en desarrollo: N/A	Tiempo real: 2 días
Descripción: El sistema debe permitir exportar a un .pdf la autoevaluación de la estudiante una vez aprobada.	
Observaciones: N/A	

Tabla 41 HU15

Historia de usuario	
Número: 15	Nombre del requisito: Autenticar Usuario
Programador: Javier Mancha Cabrera	Iteración asignada: 3
Prioridad: alta	Tiempo estimado: 1 días
Riesgo en desarrollo: N/A	Tiempo real: 1 días
Descripción: El sistema debe permitir autenticar usuario a partir de los campos: usuario, contraseña.	
Observaciones: N/A	

Tabla 42 HU16

Historia de usuario	
Número: 16	Nombre del requisito: Registrar estudiante
Programador: Javier Mancha Cabrera	Iteración asignada: 3
Prioridad: alta	Tiempo estimado: 3 días
Riesgo en desarrollo: N/A	Tiempo real: 3 días
Descripción: El sistema debe permitir registrar los datos del estudiante a partir de: nombre, solapín, usuario, cargo, número de expediente, carrera, facultad, grupo, provincia, municipio, teléfono, carnet, sexo.	
Observaciones: N/A	

Tabla 43 HU17

Historia de usuario	
Número: 17	Nombre del requisito: Añadir nota
Programador: Javier Mancha Cabrera	Iteración asignada: 3
Prioridad: baja	Tiempo estimado: 2 días
Riesgo en desarrollo: N/A	Tiempo real: 2 días
Descripción: El sistema debe crear un campo nota al estudiante crear una autoevaluación.	
Observaciones: N/A	

Tabla 44 HU18

Historia de usuario	
Número: 18	Nombre del requisito: Modificar nota
Programador: Javier Mancha Cabrera	Iteración asignada: 3
Prioridad: baja	Tiempo estimado: 2 días
Riesgo en desarrollo: N/A	Tiempo real: 4 días
Descripción: El sistema debe permitir al profesor guía modificar la nota de la autoevaluación de los estudiantes.	
Observaciones: N/A	

Tabla 45 HU19

Historia de usuario	
Número: 19	Nombre del requisito: Crear comisión
Programador: Javier Mancha Cabrera	Iteración asignada: 3
Prioridad: alta	Tiempo estimado: 3 días
Riesgo en desarrollo: N/A	Tiempo real: 3 días
Descripción: El sistema debe permitir al profesor principal crear una comisión de evaluación.	
Observaciones: N/A	

Tabla 46 HU20

Historia de usuario	
Número: 20	Nombre del requisito: Asignar profesor principal
Programador: Javier Mancha Cabrera	Iteración asignada: 3
Prioridad: Baja	Tiempo estimado: 1 días
Riesgo en desarrollo: N/A	Tiempo real: 1 días
Descripción: El sistema debe permitir al decano asignar profesor principal.	
Observaciones: N/A	

Tabla 47 HU21

Historia de usuario	
Número: 21	Nombre del requisito: Asignar profesor guía
Programador: Javier Mancha Cabrera	Iteración asignada: 3
Prioridad: Baja	Tiempo estimado: 1 días
Riesgo en desarrollo: N/A	Tiempo real: 1 días
Descripción: El sistema debe permitir al profesor principal asignar los profesores guías.	
Observaciones: N/A	

Tabla 48 HU22

Historia de usuario	
Número: 22	Nombre del requisito: Crear evento
Programador: Javier Mancha Cabrera	Iteración asignada: 3
Prioridad: Media	Tiempo estimado: 2 días
Riesgo en desarrollo: N/A	Tiempo real: 2 días
Descripción: El sistema debe permitir al decano crear un evento a partir de los campos: nombre, tipo, valor.	
Observaciones: N/A	

Tabla 49 HU23

Historia de usuario	
Número: 23	Nombre del requisito: Modificar evento
Programador: Javier Mancha Cabrera	Iteración asignada: 3
Prioridad: Media	Tiempo estimado: 3 días
Riesgo en desarrollo: N/A	Tiempo real: 3 días
Descripción: El sistema debe permitir al decano modificar los eventos a partir de los campos: nombre, tipo, valor.	
Observaciones: N/A	

Tabla 50 HU24

Historia de usuario	
Número: 24	Nombre del requisito: Eliminar evento
Programador: Javier Mancha Cabrera	Iteración asignada: 3
Prioridad: Media	Tiempo estimado: 2 días
Riesgo en desarrollo: N/A	Tiempo real: 2 días
Descripción: El sistema debe permitir al decano eliminar los eventos.	
Observaciones: N/A	

Tabla 51 HU25

Historia de usuario	
Número: 25	Nombre del requisito: Buscar eventos
Programador: Javier Mancha Cabrera	Iteración asignada: 3
Prioridad: Baja	Tiempo estimado: 2 días
Riesgo en desarrollo: N/A	Tiempo real: 2 días
Descripción: El sistema debe permitir al decano listar los eventos.	
Observaciones: N/A	

Tabla 52 HU26

Historia de usuario	
Número: 26	Nombre del requisito: registrar evento
Programador: Javier Mancha Cabrera	Iteración asignada: 3
Prioridad: media	Tiempo estimado: 2 días
Riesgo en desarrollo: N/A	Tiempo real: 2 días
Descripción: El sistema debe permitir al estudiante seleccionar un evento.	
Observaciones: N/A	

Tabla 53 HU27

Historia de usuario

Número: 27	Nombre del requisito: eliminar evento registrado
Programador: Javier Mancha Cabrera	Iteración asignada: 3
Prioridad: media	Tiempo estimado: 3 días
Riesgo en desarrollo: N/A	Tiempo real: 3 días
Descripción: El sistema debe permitir al estudiante eliminar el o los eventos seleccionados.	
Observaciones: N/A	

Tabla 54 HU28

Historia de usuario	
Número: 28	Nombre del requisito: validar autoevaluación
Programador: Javier Mancha Cabrera	Iteración asignada: 3
Prioridad: alta	Tiempo estimado: 2 días
Riesgo en desarrollo: N/A	Tiempo real: 2 días
Descripción: El sistema debe permitir al profesor guía, presidente de brigada y representante de la FEU, establecer en validado o invalidado la autoevaluación del estudiante.	
Observaciones: N/A	

Tabla 55 HU29

Historia de usuario	
Número: 29	Nombre del requisito: Modificar validar autoevaluación
Programador: Javier Mancha Cabrera	Iteración asignada: 3
Prioridad: media	Tiempo estimado: 1 día
Riesgo en desarrollo: N/A	Tiempo real: 1 día
Descripción: El sistema debe permitir al profesor guía, presidente de brigada y representante de la FEU modificar la validación o invalidación de la autoevaluación del estudiante.	
Observaciones: N/A	

Tabla 56 HU30

Historia de usuario	
Número: 30	Nombre del requisito: Ver historial
Programador: Javier Mancha Cabrera	Iteración asignada: 3
Prioridad: media	Tiempo estimado: 2 día
Riesgo en desarrollo: N/A	Tiempo real: 2 día
Descripción: El sistema debe permitir al profesor guía visualizar la información de todas las autoevaluaciones registradas por cada estudiante de su grupo.	
Observaciones: N/A	

Tabla 57 HU31

Historia de usuario	
Número: 31	Nombre del requisito: Asignar evaluación integral
Programador: Javier Mancha Cabrera	Iteración asignada: 3
Prioridad: Baja	Tiempo estimado: 1 día
Riesgo en desarrollo: N/A	Tiempo real: 1 día
Descripción: El sistema debe permitir al profesor guía asignar una evaluación integral de Satisfactoria o Insatisfactoria a cada estudiante de su grupo.	
Observaciones: N/A	

Tabla 58 HU32

Historia de usuario	
Número: 32	Nombre del requisito: Modificar evaluación integral
Programador: Javier Mancha Cabrera	Iteración asignada: 3
Prioridad: baja	Tiempo estimado: 1 día
Riesgo en desarrollo: N/A	Tiempo real: 1 día
Descripción: El sistema debe permitir al profesor guía modificar la evaluación integral de Satisfactoria o Insatisfactoria a cada estudiante de su grupo.	
Observaciones: N/A	

Tabla 59 HU33

Historia de usuario	
Número: 33	Nombre del requisito: Sugerencia del sistema
Programador: Javier Mancha Cabrera	Iteración asignada: 3
Prioridad: alta	Tiempo estimado: 3 día
Riesgo en desarrollo: N/A	Tiempo real: 3 día
Descripción: El sistema debe generar una recomendación de evaluación integral al profesor guía en dependencia del historial de cada estudiante de su grupo.	
Observaciones: N/A	

Tabla 60 Tarjeta CRC 4

<u>TARJETA CRC</u>	
NOMBRE DE LA CLASE: Integralidad_usuario	
RESPONDABILIDADES: Se registra en el sistema para que se le asigne un rol	COLABORADORES: Integralidad_estudiante Integralidad_profesor

Tabla 61 Tarjeta CRC 5

<u>TARJETA CRC</u>	
NOMBRE DE LA CLASE: Integralidad_autoevalnotas	
RESPONDABILIDADES: Se encarga de dar retroalimentación al estudiante	COLABORADORES: Integralidad_autoeval Integralidad_profesor

Tabla 62 Tarjeta CRC 6

<u>TARJETA CRC</u>	
NOMBRE DE LA CLASE: Integralidad_autoeval	
RESPONDABILIDADES: Se encarga de mostrar los eventos participados	COLABORADORES: Integralidad_autoevalnotas Integralidad_estudiante Integralidad_autoeval_eventos

Tabla 63 Tarjeta CRC 7

<u>TARJETA CRC</u>	
NOMBRE DE LA CLASE: Integralidad_comision	
RESPONDABILIDADES: Se encarga de asignar nota, validar autoevaluación	COLABORADORES: Integralidad_gd

Tabla 64 Tarjeta CRC 8

<u>TARJETA CRC</u>	
NOMBRE DE LA CLASE: Integralidad_autoeval_eventos	

RESPONDABILIDADES: Se encarga de mostrar los eventos participados	COLABORADORES: Integralidad_evento Integralidad_autoeval
---	---

Tabla 65 Tarjeta CRC 9

TARJETA CRC	
NOMBRE DE LA CLASE: Integralidad_evento	
RESPONDABILIDADES: Se encarga de almacenar los eventos del sistema	COLABORADORES: Integralidad_autoeval_eventos

Tabla 66 CP Aceptación 10

Caso de prueba de aceptación	
Código: HU4_P1	Historia de usuario:4
Nombre: Eliminar evidencia	
Condiciones de ejecución: El usuario tiene que estar autenticado en el sistema como estudiante, el estudiante debe tener una evidencia.	
Pasos de ejecución: el usuario selecciona el botón X para eliminar y acepta la ventana emergente.	
Resultados esperados: Se elimina una evidencia.	
Evaluación de la prueba: Satisfactorio.	

Tabla 67 CP Aceptación 11

Caso de prueba de aceptación	
Código: HU4_P2	Historia de usuario:4
Nombre: Eliminar evidencia	

Condiciones de ejecución: El usuario tiene que estar autenticado en el sistema como estudiante, el usuario debe tener una evidencia.
Pasos de ejecución: el usuario selecciona el botón X para eliminar y cancela la ventana emergente.
Resultados esperados: no se ejecutan cambios.
Evaluación de la prueba: Satisfactorio.

Tabla 68 CP Aceptación 12

Caso de prueba de aceptación	
Código: HU15_P1	Historia de usuario:15
Nombre: Autenticar Usuario.	
Condiciones de ejecución: las credenciales del usuario deben estar registradas en la base datos UCI	
Pasos de ejecución: el usuario introduce un usuario o contraseña incorrecta	
Resultados esperados: se alerta que el usuario o contraseña son incorrectos.	
Evaluación de la prueba: Satisfactorio.	

Tabla 69 CP Aceptación 13

Caso de prueba de aceptación	
Código: HU7_P7	Historia de usuario:7
Nombre: Modificar evaluación	
Condiciones de ejecución: El usuario tiene que estar autenticado en el sistema como profesor guía.	
Pasos de ejecución: el usuario selecciona el botón de satisfactoria o no satisfactoria	
Resultados esperados: Se modifica la evaluación del estudiante.	
Evaluación de la prueba: Satisfactorio.	

Tabla 70 CP Aceptación 14

Caso de prueba de aceptación	
Código: HU18_P1	Historia de usuario:18
Nombre: Modificar nota	
Condiciones de ejecución: El usuario tiene que estar autenticado en el sistema como profesor guía.	
Pasos de ejecución: el usuario accede a la autoevaluación del estudiante, añade la nota y guarda los cambios.	
Resultados esperados: se guardan los cambios efectuados.	
Evaluación de la prueba: Satisfactorio.	

Tabla 71 CP Aceptación 15

Caso de prueba de aceptación	
Código: HU18_P2	Historia de usuario:18
Nombre: Modificar nota	
Condiciones de ejecución: El usuario tiene que estar autenticado en el sistema como profesor guía.	
Pasos de ejecución: el usuario accede a la autoevaluación del estudiante, añade una nota en blanco.	
Resultados esperados: se guardan los cambios efectuados.	
Evaluación de la prueba: Satisfactorio.	

Tabla 72 CP Aceptación 16

Caso de prueba de aceptación	
Código: HU3_P1	Historia de usuario:3
Nombre: registrar evidencia	
Condiciones de ejecución: El usuario tiene que estar autenticado en el sistema como estudiante, debe tener al menos un evento	

Pasos de ejecución: el usuario marca la opción de evidencia, sube una evidencia al sistema a través del botón examinar, selecciona el archivo y luego acepta los cambios.
Resultados esperados: Se crea una evidencia de un evento.
Evaluación de la prueba: Satisfactorio.

Tabla 73 CP Aceptación 17

Caso de prueba de aceptación	
Código: HU3_P2	Historia de usuario:3
Nombre: registrar evidencia	
Condiciones de ejecución: El usuario tiene que estar autenticado en el sistema como estudiante, debe tener al menos un evento	
Pasos de ejecución: el usuario marca la opción de evidencia, sube una evidencia al sistema a través del botón examinar, selecciona el archivo y luego cancela los cambios.	
Resultados esperados: no se ejecutan cambios	
Evaluación de la prueba: Satisfactorio.	

Tabla 74 CP Aceptación 18

Caso de prueba de aceptación	
Código: HU3_P3	Historia de usuario:3
Nombre: registrar evidencia	
Condiciones de ejecución: El usuario tiene que estar autenticado en el sistema como estudiante, debe tener al menos un evento	
Pasos de ejecución: el usuario marca la opción de evidencia, no sube una evidencia al sistema y luego acepta los cambios.	
Resultados esperados: se crea un evento sin evidencia.	
Evaluación de la prueba: Satisfactorio.	

Tabla 75 CP Aceptación 19

Caso de prueba de aceptación	
Código: HU5_P1	Historia de usuario:5
Nombre: visualizar evidencias	
Condiciones de ejecución: El usuario tiene que estar autenticado en el sistema como profesor guía o estudiante.	
Pasos de ejecución: el usuario selecciona el botón evidencias.	
Resultados esperados: se visualizan las evidencias del evento.	
Evaluación de la prueba: Satisfactorio.	

Tabla 76 CP Aceptación 20

Caso de prueba de aceptación	
Código: HU20_P1	Historia de usuario:20
Nombre: Asignar profesor principal	
Condiciones de ejecución: El usuario tiene que estar autenticado en el sistema como decano.	
Pasos de ejecución: el usuario selecciona el botón asignar, selecciona los profesores principales a sus respectivos años y acepta los cambios	
Resultados esperados: se asignan los profesores principales a sus años.	
Evaluación de la prueba: Satisfactorio.	

Tabla 77 CP Aceptación 21

Caso de prueba de aceptación	
Código: HU21_P21	Historia de usuario:21
Nombre: Asignar profesor guía	
Condiciones de ejecución: El usuario tiene que estar autenticado en el sistema como profesor principal.	
Pasos de ejecución: el usuario selecciona el botón asignar, selecciona los profesores guía a sus respectivos grupos y acepta los cambios	

Resultados esperados: se asignan los profesores principales a sus grupos.

Evaluación de la prueba: Satisfactorio.

Tabla 78 CP Aceptación 22

Caso de prueba de aceptación

Código: HU24_P1

Historia de usuario:24

Nombre: Eliminar evento

Condiciones de ejecución: El usuario tiene que estar autenticado en el sistema como administrador.

Pasos de ejecución: el usuario accede al panel de evento, selecciona los eventos a eliminar, presiona el botón eliminar y acepta la ventana emergente.

Resultados esperados: se eliminan los eventos seleccionados

Evaluación de la prueba: Satisfactorio.

Tabla 79 CP Aceptación 23

Caso de prueba de aceptación

Código: HU24_P2

Historia de usuario:24

Nombre: Eliminar evento

Condiciones de ejecución: El usuario tiene que estar autenticado en el sistema como administrador.

Pasos de ejecución: el usuario accede al panel de evento, selecciona los eventos a eliminar, presiona el botón eliminar y cancela la ventana emergente.

Resultados esperados: no se ejecutan cambios.

Evaluación de la prueba: Satisfactorio.

Tabla 80 CP Aceptación 24

Caso de prueba de aceptación

Código: HU6_P1

Historia de usuario:6

Nombre: asignar evaluación

Condiciones de ejecución: El usuario tiene que estar autenticado en el sistema como profesor guía
Pasos de ejecución: el usuario accede al botón historial, selecciona el botón satisfactoria o no satisfactoria
Resultados esperados: se le asigna la evaluación del estudiante
Evaluación de la prueba: Satisfactorio.

Tabla 81 CP Aceptación 25

Caso de prueba de aceptación	
Código: HU11_P1	Historia de usuario:11
Nombre: buscar usuario	
Condiciones de ejecución: El usuario tiene que estar autenticado en el sistema como administrador	
Pasos de ejecución: el usuario accede al panel gestionar usuario y selecciona el botón Todos	
Resultados esperados: se listan todos los usuarios del sistema.	
Evaluación de la prueba: Satisfactorio.	

Tabla 82 CP Aceptación 26

Caso de prueba de aceptación	
Código: HU12_P1	Historia de usuario:12
Nombre: registrar grupo docente	
Condiciones de ejecución: el grupo docente debe existir en la base datos UCI.	
Pasos de ejecución: el usuario se registra en el sistema	
Resultados esperados: se registra un grupo docente en la base datos local en caso de no existir en el sistema.	
Evaluación de la prueba: Satisfactorio.	

Tabla 83 CP Aceptación 27

Caso de prueba de aceptación	
Código: HU16_P1	Historia de usuario:16
Nombre: registrar estudiante	
Condiciones de ejecución: las credenciales del estudiante deben estar registradas en la base datos UCI.	
Pasos de ejecución: el usuario se autentica en el sistema	
Resultados esperados: se registra un estudiante en la base datos local en caso de no existir en el sistema.	
Evaluación de la prueba: Satisfactorio.	

Tabla 84 CP Aceptación 28

Caso de prueba de aceptación	
Código: HU19_P1	Historia de usuario:19
Nombre: crear comisión	
Condiciones de ejecución: El usuario tiene que estar autenticado en el sistema como profesor principal	
Pasos de ejecución: el usuario accede al panel de comisión, selecciona los participantes de la comisión	
Resultados esperados: se crea una comisión	
Evaluación de la prueba: Satisfactorio.	

Tabla 85 CP Aceptación 29

Caso de prueba de aceptación	
Código: HU22_P1	Historia de usuario:22
Nombre: crear evento	
Condiciones de ejecución: El usuario tiene que estar autenticado en el sistema como administrador	

Pasos de ejecución: el usuario accede al panel evento, selecciona el botón insertar evento, rellena los campos y acepta.

Resultados esperados: se crea un evento

Evaluación de la prueba: Satisfactorio.

Tabla 86 CP Aceptación 30

Caso de prueba de aceptación	
Código: HU22_P2	Historia de usuario:22
Nombre: crear evento	
Condiciones de ejecución: El usuario tiene que estar autenticado en el sistema como administrador	
Pasos de ejecución: el usuario accede al panel evento, selecciona el botón insertar evento, deja campos en blanco y acepta.	
Resultados esperados: alerta al usuario que debe rellenar el campo.	
Evaluación de la prueba: Satisfactorio.	

Tabla 87 CP Aceptación 31

Caso de prueba de aceptación	
Código: HU22_P3	Historia de usuario:22
Nombre: crear evento	
Condiciones de ejecución: El usuario tiene que estar autenticado en el sistema como administrador	
Pasos de ejecución: el usuario accede al panel evento, selecciona el botón insertar evento, rellena los campos y cancela.	
Resultados esperados: no se ejecutan cambios.	
Evaluación de la prueba: Satisfactorio.	

Tabla 88 CP Aceptación 32

Caso de prueba de aceptación

Código: HU23_P1	Historia de usuario:23
Nombre: modificar evento	
Condiciones de ejecución: El usuario tiene que estar autenticado en el sistema como administrador	
Pasos de ejecución: el usuario accede al panel evento, selecciona el botón Todos, selecciona el botón modificar, modifica los datos y acepta.	
Resultados esperados: se modifica el evento	
Evaluación de la prueba: Satisfactorio.	

Tabla 89 CP Aceptación 33

Caso de prueba de aceptación	
Código: HU23_P2	Historia de usuario:23
Nombre: modificar evento	
Condiciones de ejecución: El usuario tiene que estar autenticado en el sistema como administrador	
Pasos de ejecución: el usuario accede al panel evento, selecciona el botón Todos, selecciona el botón modificar, modifica los datos y cancela.	
Resultados esperados: no se ejecutan los cambios	
Evaluación de la prueba: Satisfactorio.	

Tabla 90 CP Aceptación 34

Caso de prueba de aceptación	
Código: HU23_P3	Historia de usuario:23
Nombre: modificar evento	
Condiciones de ejecución: El usuario tiene que estar autenticado en el sistema como administrador	

Pasos de ejecución: el usuario accede al panel evento, selecciona el botón Todos, selecciona el botón modificar, modifica los datos, deja espacios en blanco acepta.

Resultados esperados: alerta al usuario que debe rellenar el campo.

Evaluación de la prueba: Satisfactorio.

Tabla 91 CP Aceptación 35

Caso de prueba de aceptación

Código: HU25_P1

Historia de usuario:25

Nombre: buscar evento

Condiciones de ejecución: El usuario tiene que estar autenticado en el sistema como administrador

Pasos de ejecución: el usuario accede al panel evento, selecciona el botón Todos.

Resultados esperados: se listan los eventos

Evaluación de la prueba: Satisfactorio.

Tabla 92 CP Aceptación 36

Caso de prueba de aceptación

Código: HU26_P1

Historia de usuario:26

Nombre: seleccionar evento

Condiciones de ejecución: El usuario tiene que estar autenticado en el sistema como estudiante

Pasos de ejecución: el usuario accede al panel evidencia, selecciona el evento y acepta

Resultados esperados: se añade un evento

Evaluación de la prueba: Satisfactorio.

Tabla 93 CP Aceptación 37

Caso de prueba de aceptación	
Código: HU27_P1	Historia de usuario:5
Nombre: eliminar evento seleccionado	
Condiciones de ejecución: El usuario tiene que estar autenticado en el sistema como estudiante	
Pasos de ejecución: el usuario accede al panel inspeccionar autoevaluación, , selecciona editar, selecciona el botón eliminar y acepta la ventana emergente.	
Resultados esperados: se elimina el evento del estudiante.	
Evaluación de la prueba: Satisfactorio.	

Tabla 94 CP Aceptación 39

Caso de prueba de aceptación	
Código: HU28_P1	Historia de usuario:28
Nombre: validar autoevaluación	
Condiciones de ejecución: El usuario tiene que estar autenticado en el sistema como estudiante o profesor guía	
Pasos de ejecución: el usuario accede al botón historial, selecciona el botón validar o no validar	
Resultados esperados: se valida o invalida la autoevaluación del estudiante.	
Evaluación de la prueba: Satisfactorio.	

Tabla 95 CP Aceptación 40

Caso de prueba de aceptación	
Código: HU29_P1	Historia de usuario:29
Nombre: Modificar validar autoevaluación	
Condiciones de ejecución: El usuario tiene que estar autenticado en el sistema como estudiante o profesor guía.	

Pasos de ejecución: el usuario accede al botón historial, selecciona el botón validar o no validar
Resultados esperados: se valida o invalida la autoevaluación del estudiante.
Evaluación de la prueba: Satisfactorio.

Tabla 96 CP Aceptación 41

Caso de prueba de aceptación	
Código: HU27_P1	Historia de usuario:5
Nombre: eliminar evento seleccionado	
Condiciones de ejecución: El usuario tiene que estar autenticado en el sistema como estudiante	
Pasos de ejecución: el usuario accede al panel inspeccionar autoevaluación, , selecciona editar, selecciona el botón eliminar y cancela la ventana emergente.	
Resultados esperados: no se ejecutan cambios.	
Evaluación de la prueba: Satisfactorio.	

Tabla 97 CP Aceptación 42

Caso de prueba de aceptación	
Código: HUD_P1	Historia de usuario: -
Nombre: Desconectarse	
Condiciones de ejecución: debe estar autenticado en el sistema	
Pasos de ejecución: seleccionar en el panel superior izquierdo, seleccionar el botón desconectarse	
Resultados esperados: se desconecta del sistema	
Evaluación de la prueba: Satisfactorio.	

Tabla 98 Tarea de ingeniería 4

Tarea de ingeniería

Número de Tarea: 4	Número de la Historia de Usuario: 4
Nombre de la Tarea: Eliminar evidencia	
Tipo de Tarea: Desarrollo	Estimación: 3 días
Fecha de inicio: 26/8/2022	Fecha de fin: 28/8/2022
Programador responsable: Javier Mancha Cabrera	
Descripción: Implementar una vista para el estudiante que permita eliminar las evidencias añadidas	

Tabla 99 Tarea de ingeniería 5

Tarea de ingeniería	
Número de Tarea: 5	Número de la Historia de Usuario: 5
Nombre de la Tarea: Visualizar evidencia	
Tipo de Tarea: Desarrollo	Estimación: 2 días
Fecha de inicio: 29/8/2022	Fecha de fin: 30/8/2022
Programador responsable: Javier Mancha Cabrera	
Descripción: Implementar una vista que permita visualizar las evidencias subidas por el estudiante	

Tabla 100 Tarea de ingeniería 6

Tarea de ingeniería	
Número de Tarea: 6	Número de la Historia de Usuario: 6
Nombre de la Tarea: Asignar evaluación	
Tipo de Tarea: Desarrollo	Estimación: 3 días
Fecha de inicio: 31/8/2022	Fecha de fin: 2/9/2022
Programador responsable: Javier Mancha Cabrera	
Descripción: Implementar una vista que permita al profesor guía asignar evaluación a los estudiantes	

Tabla 101 Tarea de ingeniería 7

Tarea de ingeniería	
Número de Tarea: 7	Número de la Historia de Usuario: 7
Nombre de la Tarea: Modificar evaluación	
Tipo de Tarea: Desarrollo	Estimación: 2 días
Fecha de inicio: 3/9/2022	Fecha de fin: 4/9/2022
Programador responsable: Javier Mancha Cabrera	
Descripción: Implementar una vista que permita al profesor guía modificar los datos de la evaluación del estudiante.	

Tabla 102 Tarea de ingeniería 11

Tarea de ingeniería	
Número de Tarea: 11	Número de la Historia de Usuario: 11
Nombre de la Tarea: Buscar usuario	
Tipo de Tarea: Desarrollo	Estimación: 3 días
Fecha de inicio: 15/9/2022	Fecha de fin: 17/9/2022
Programador responsable: Javier Mancha Cabrera	
Descripción: Implementar una vista que permita al administrador buscar los datos de los usuarios.	

Tabla 103 Tarea de ingeniería 12

Tarea de ingeniería	
Número de Tarea: 12	Número de la Historia de Usuario: 12
Nombre de la Tarea: Registrar grupo docente	
Tipo de Tarea: Desarrollo	Estimación: 3 días
Fecha de inicio: 18/9/2022	Fecha de fin: 20/9/2022
Programador responsable: Javier Mancha Cabrera	
Descripción: Implementar una vista que permita registrar los datos del grupo docente.	

Tabla 104 Tarea de ingeniería 16

Tarea de ingeniería	
Número de Tarea: 16	Número de la Historia de Usuario: 16
Nombre de la Tarea: Registrar estudiante	
Tipo de Tarea: Desarrollo	Estimación: 3 días
Fecha de inicio: 27/9/2022	Fecha de fin: 30/9/2022
Programador responsable: Javier Mancha Cabrera	
Descripción: Implementar una vista que permita registrar los datos del estudiante.	

Tabla 105 Tarea de ingeniería 17

Tarea de ingeniería	
Número de Tarea: 17	Número de la Historia de Usuario: 17
Nombre de la Tarea: Añadir nota	
Tipo de Tarea: Desarrollo	Estimación: 2 días
Fecha de inicio: 1/10/2022	Fecha de fin: 2/10/2022
Programador responsable: Javier Mancha Cabrera	
Descripción: Implementar una vista que permita crear el campo nota al estudiante crear una autoevaluación.	

Tabla 106 Tarea de ingeniería 18

Tarea de ingeniería	
Número de Tarea: 18	Número de la Historia de Usuario: 18
Nombre de la Tarea: Modificar nota	
Tipo de Tarea: Desarrollo	Estimación: 2 días
Fecha de inicio: 3/10/2022	Fecha de fin: 4/10/2022
Programador responsable: Javier Mancha Cabrera	
Descripción: Implementar una vista que permita al profesor guía modificar la nota de la autoevaluación del estudiante.	

Tabla 107 Tarea de ingeniería 19

Tarea de ingeniería	
Número de Tarea: 19	Número de la Historia de Usuario: 19
Nombre de la Tarea: Crear comisión	
Tipo de Tarea: Desarrollo	Estimación: 3 días
Fecha de inicio: 5/10/2022	Fecha de fin: 7/10/2022
Programador responsable: Javier Mancha Cabrera	
Descripción: Implementar una vista que permita al profesor principal crear una comisión de evaluación.	

Tabla 108 Tarea de ingeniería 20

Tarea de ingeniería	
Número de Tarea: 20	Número de la Historia de Usuario: 20
Nombre de la Tarea: Asignar profesor principal	
Tipo de Tarea: Desarrollo	Estimación: 1 día
Fecha de inicio: 8/10/2022	Fecha de fin: 8/10/2022
Programador responsable: Javier Mancha Cabrera	
Descripción: Implementar una vista que permita al decano asignar profesor principal.	

Tabla 109 Tarea de ingeniería 21

Tarea de ingeniería	
Número de Tarea: 21	Número de la Historia de Usuario: 21
Nombre de la Tarea: Asignar profesor guía	
Tipo de Tarea: Desarrollo	Estimación: 1 día
Fecha de inicio: 9/10/2022	Fecha de fin: 9/10/2022
Programador responsable: Javier Mancha Cabrera	
Descripción: Implementar una vista que permita al profesor principal asignar los profesores guías.	

Tabla 110 Tarea de ingeniería 22

Tarea de ingeniería	
Número de Tarea: 22	Número de la Historia de Usuario: 22
Nombre de la Tarea: Crear evento	
Tipo de Tarea: Desarrollo	Estimación: 2 días
Fecha de inicio: 10/10/2022	Fecha de fin: 11/10/2022
Programador responsable: Javier Mancha Cabrera	
Descripción: Implementar una vista que permita al administrador crear un evento	

Tabla 111 Tarea de ingeniería 23

Tarea de ingeniería	
Número de Tarea: 23	Número de la Historia de Usuario: 23
Nombre de la Tarea: Modificar evento	
Tipo de Tarea: Desarrollo	Estimación: 3 días
Fecha de inicio: 12/10/2022	Fecha de fin: 14/10/2022
Programador responsable: Javier Mancha Cabrera	
Descripción: Implementar una vista que permita al administrador modificar los eventos	

Tabla 112 Tarea de ingeniería 24

Tarea de ingeniería	
Número de Tarea: 24	Número de la Historia de Usuario: 24
Nombre de la Tarea: Eliminar evento	
Tipo de Tarea: Desarrollo	Estimación: 2 días
Fecha de inicio: 15/10/2022	Fecha de fin: 17/10/2022
Programador responsable: Javier Mancha Cabrera	
Descripción: Implementar una vista que permita al administrador eliminar los eventos.	

Tabla 113 Tarea de ingeniería 25

Tarea de ingeniería	
Número de Tarea: 25	Número de la Historia de Usuario: 25
Nombre de la Tarea: Buscar eventos	
Tipo de Tarea: Desarrollo	Estimación: 2 días
Fecha de inicio: 18/10/2022	Fecha de fin: 20/10/2022
Programador responsable: Javier Mancha Cabrera	
Descripción: Implementar una vista que permita al administrador listar los eventos.	

Tabla 114 Tarea de ingeniería 26

Tarea de ingeniería	
Número de Tarea: 26	Número de la Historia de Usuario: 26
Nombre de la Tarea: registrar evento	
Tipo de Tarea: Desarrollo	Estimación: 2 días
Fecha de inicio: 21/10/2022	Fecha de fin: 22/10/2022
Programador responsable: Javier Mancha Cabrera	
Descripción: Implementar una vista que permita al estudiante registrar un evento en su autoevaluación.	

Tabla 115 Tabla 115 Tarea de ingeniería 27

Tarea de ingeniería	
Número de Tarea: 27	Número de la Historia de Usuario: 27
Nombre de la Tarea: Eliminar evento registrado	
Tipo de Tarea: Desarrollo	Estimación: 3 días
Fecha de inicio: 23/10/2022	Fecha de fin: 25/10/2022
Programador responsable: Javier Mancha Cabrera	
Descripción: Implementar una vista que permita al estudiante eliminar el o los eventos seleccionados de su autoevaluación.	

Tabla 116 Tabla 115 Tarea de ingeniería 28

Tarea de ingeniería	
Número de Tarea: 28	Número de la Historia de Usuario: 28
Nombre de la Tarea: Validar autoevaluación	
Tipo de Tarea: Desarrollo	Estimación: 2 días
Fecha de inicio: 26/10/2022	Fecha de fin: 27/10/2022
Programador responsable: Javier Mancha Cabrera	
Descripción: Implementar una vista que permita al profesor guía, presidente de brigada y representante de la FEU, establecer una autoevaluación al estudiante.	

Tabla 117 Tabla 115 Tarea de ingeniería 29

Tarea de ingeniería	
Número de Tarea: 29	Número de la Historia de Usuario: 29
Nombre de la Tarea: Modificar validar autoevaluación	
Tipo de Tarea: Desarrollo	Estimación: 1 días
Fecha de inicio: 28/10/2022	Fecha de fin: 28/10/2022
Programador responsable: Javier Mancha Cabrera	
Descripción: Implementar una vista que permita al profesor guía, presidente de brigada y representante de la FEU modificar la autoevaluación del estudiante.	

Tabla 118 Tabla 115 Tarea de ingeniería 30

Tarea de ingeniería	
Número de Tarea: 30	Número de la Historia de Usuario: 30
Nombre de la Tarea: Ver historial	
Tipo de Tarea: Desarrollo	Estimación: 2 días
Fecha de inicio: 29/10/2022	Fecha de fin: 30/10/2022
Programador responsable: Javier Mancha Cabrera	

Descripción: Implementar una vista que permita al profesor guía visualizar la información de todas las autoevaluaciones registradas por cada estudiante de su grupo.

Tabla 119 Tabla 115 Tarea de ingeniería 31

Tarea de ingeniería	
Número de Tarea: 31	Número de la Historia de Usuario: 31
Nombre de la Tarea: Asignar evaluación integral	
Tipo de Tarea: Desarrollo	Estimación: 1 día
Fecha de inicio: 31/10/2022	Fecha de fin: 31/10/2022
Programador responsable: Javier Mancha Cabrera	
Descripción: Implementar una vista que permita al profesor guía asignar una evaluación integral a cada estudiante de su grupo.	

Tabla 120 Tabla 115 Tarea de ingeniería 32

Tarea de ingeniería	
Número de Tarea: 32	Número de la Historia de Usuario: 32
Nombre de la Tarea: Modificar evaluación integral	
Tipo de Tarea: Desarrollo	Estimación: 1 día
Fecha de inicio: 1/11/2022	Fecha de fin: 1/11/2022
Programador responsable: Javier Mancha Cabrera	
Descripción: Implementar una vista que permita al profesor guía modificar la evaluación integral a cada estudiante de su grupo.	

Tabla 121 Tabla 115 Tarea de ingeniería 33

Tarea de ingeniería	
Número de Tarea: 33	Número de la Historia de Usuario: 33
Nombre de la Tarea: Sugerencia del sistema	
Tipo de Tarea: Desarrollo	Estimación: 3 días

Fecha de inicio: 2/11/2022	Fecha de fin: 4/11/2022
Programador responsable: Javier Mancha Cabrera	
Descripción: Implementar una vista que permita generar una recomendación de evaluación integral al profesor guía en dependencia del historial de cada estudiante de su grupo	

