SISTEMA PARA LA AUTOMATIZACIÓN DE LOS PROCESOS DE LA ESTRATEGIA DE SUPERACIÓN PEDAGÓGICA

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

Autor:
Jorge Javier Tamayo Muñiz

Tutores:
Ing. Eric Rodríguez Ochoa
M.Sc. Eylín Hernández Luque

La Habana, julio de 2016
"Año 58 de la Revolución"

"Todos y cada uno de nosotros paga puntualmente su cuota de sacrificio consciente de recibir el premio en la satisfacción del deber cumplido, conscientes de avanzar con todos hacia el Hombre Nuevo que se vislumbra en el horizonte."

Ernesto Che Guevara

DECLARACIÓN DE AUTORÍA

Declaro por este medio que yo Jorge Javier Tamayo Muñiz, con carné de identidad 91072648184, soy el autor principal de la tesis “Sistema para la Automatización de los Procesos de la Estrategia de Superación Pedagógica”, y autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

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

_____

Autor: Jorge Javier Tamayo Muñiz

_____ _____

Tutor: M. Sc. Eylín Hernández Luque Tutor: Ing. Eric Rodríguez Ochoa

DATOS DE CONTACTO

Tutor: Ing. Eric Rodríguez Ochoa

Graduado de Ingeniería en Ciencias Informáticas en la Universidad de las Ciencias Informáticas en julio de 2014.

Profesor de Programación en la Facultad Introductoria de Ciencias Informáticas.

Centro de Trabajo: Universidad de las Ciencias Informáticas (UCI). Centro de Innovación y Calidad de la Educación (CICE).

Cargo: Profesor del Centro de Innovación y Calidad de la Educación (CICE). Vicerrectoría de Formación.

Teléfono Oficina: +53 – 7 – 837 – 8126.

Correo electrónico: erochoa@uci.cu.

Tutora: Ms.C. Eylín Hernández Luque

Graduado de Ingeniería en Ciencias Informáticas en la Universidad de las Ciencias Informáticas en julio de 2007.

Máster en Gestión de Proyectos Informáticos en el año 2013.

Categoría docente: Asistente.

Profesor de las asignaturas de pregrado: Metodología de la Investigación Científica, Ingeniería y Gestión de Software, Base de Datos 1 y 2, Introducción a las Ciencias Informáticas, Proyecto de Investigación de desarrollo. Imparte en postgrado el curso de la formación del profesional en la Universidad Cubana.

Centro de Trabajo: Universidad de las Ciencias Informáticas (UCI). Centro de Innovación y Calidad de la Educación (CICE).

Cargo: Metodóloga de investigación del Centro de Innovación y Calidad de la Educación (CICE). Vicerrectoría de Formación.

Teléfono Oficina: +53 – 7 – 837 – 2522.

Correo electrónico: ehernandezl@uci.cu.

DEDICATORIA

Dedico este trabajo en primer lugar a mi madre, por su amor incondicional, por su temple y fuerza, por estar presente en los momentos más difíciles de mi vida y por todos los sacrificios realizados en aras de cumplir mis metas.

A mi familia, por apoyarme y ayudarme a cumplir mis sueños.

AGRADECIMIENTOS

Toda obra humana, por humilde que sea, necesita del concurso de varios. Jamás el hombre alcanzó la meta por sí solo, siempre fue de alguna manera asistido en el empeño, y si algo lo enaltece es el reconocimiento.

A dios.

Por haberme acompañado y guiado a lo largo de mi carrera, por haberme dado salud para lograr mis objetivos y por brindarme una vida llena de aprendizajes, experiencias y sobre todo felicidad.

A mi madre Deisy.

Por ser el pilar fundamental de mi vida, por estar ahí cuando más lo necesitaba, por su esfuerzo, amor, dedicación y apoyo a lo largo de toda mi trayectoria estudiantil. Por ser la mejor madre del mundo. Por darme uno de los tesoros más grandes de mi vida a mi hermano Liuber. Por su sacrificio de madre, por enseñarme a salir adelante, a triunfar, por protegerme y por guiar cada paso de mi vida.

A mi padre Javier.

Por brindarme amor y cariño. Por su apoyo incondicional en cada etapa de mi vida. Por ayudarme a ser un hombre de bien. Por esperarme cada diciembre para que juntos disfrutáramos del puerco de fin de año.

A Hubert.

Por estar ahí en cada una de las batallas. Por ser ejemplo de padre incondicional. Por enseñarme a crecer y poder resolver todos los problemas que nos prepara la vida.

A mis abuelos Mirian y Gerardo.

Por haberme criado como un hijo. Por su amor y cariño en todas las etapas de mi vida. Por esperarme con una buena comida cada vez que llegaba a la casa, con lágrimas y con los brazos abiertos.

A mi familia.

A mi hermano Liuber por quererme tanto y abrazarme cada vez que me veía llegar de vacaciones. A mi tía Leyanis por apoyarme cuando lo necesité, por demostrarme que se pueden alcanzar todas las metas que nos propongamos. A mi tío Lázaro por demostrarme en todos estos años que podía contar con él y ser parte de la familia. A mis primos Andro y Leandro por quererme y estar a mi lado cada vez que podían en el día.

A Yuri.

Por ser mi amigo y contar con su apoyo incondicional en todos los momentos a lo largo de mi carrera.

A Reinier.

Por su apoyo y dedicación en el transcurso de esta tesis. Por ayudarme cuando más lo necesité para alcanzar mis metas.

A One.

Por quererme y apoyarme desde el momento en que nos conocimos.

A Eduardo.

Por apoyarme y trasmitirme sus experiencias y conocimientos para lograr mis metas.

A mis tutores Eylín y Eric.

Por su asesoría metodológica y sabios consejos. Por su apoyo y dedicación en cada una de las etapas por las que transcurrió esta tesis. Por ayudarme a llegar a la meta y graduarme como ingeniero.

A mis amistades.

Wilbia, Roberto, Darlin, Danilo, Arian, Lisbet, Policarpo, Raiza, Ramón, Andy, Osniel, Adniel, Jany, Clara, Yohandra, Rafa, Maylin, Yoandry, Lyen. Por confiar y creer en mí y haber hecho de mi etapa universitaria un trayecto de vivencias que nunca olvidaré.

A todos los profesores que me apoyaron y formaron en el trayecto de la carrera. A la profe Maydelis, al profe Hubert. Al tribunal integrado por los profesores: Neriada, Niurvis, Harold y Yadier; por permitirme esforzarme y ser ejemplo de responsabilidad y conocimiento.

Gracias a todas aquellas personas que de una forma u otra han contribuido a lograr mi objetivo.

A todos Gracias.

RESUMEN

En el Centro de Innovación y Calidad de la Educación, se gestiona la caracterización del claustro de la Universidad de las Ciencias Informáticas, para contribuir a la calidad del proceso de formación integral en el área de las ciencias pedagógicas y ciencias de la educación. El procesamiento de la información individual de cada profesional es complejo y depende de las habilidades y experiencias adquiridas por los especialistas. El objetivo que se trazó en la investigación fue desarrollar un sistema para la automatización de los procesos de la Estrategia de Superación Pedagógica, que contribuya a los niveles de escalabilidad del software.

Los fundamentos teóricos de los principales conceptos del proceso de superación individual de cada profesional, las tendencias actuales de los sistemas que automatizan los procesos de la Estrategia de Superación Pedagógica en el ámbito extranjero y nacional y el estudio de las herramientas, tecnologías, y lenguajes utilizados, permitió la selección del lenguaje PHP, el *framework Symfony2* y *PostgreSQL* como gestor de base de datos. Para orientar el proceso de desarrollo, se utilizó la metodología XP. Lo que permite la generación de los artefactos correspondientes en cada una de sus fases. Las pruebas de unidad, de sistema y de validación, corroboraron que la propuesta es fiable para la gestión de la información de la caracterización del claustro, en cuanto a los datos de los controles a clases, la evaluación profesoral, los grupos de trabajo de formación pedagógica y la estrategia individual de superación pedagógica.

Palabras claves: estrategia, superación pedagógica, sistema, automatización y escalabilidad.

CONTENIDO DE LA MEMORIA ESCRITA DE LA INVESTIGACIÓN

INTRODUCCIÓN 1

CAPÍTULO 1 FUNDAMENTOS TEÓRICOS-METODOLÓGICOS PARA EL DESARRROLLO DEL SISTEMA 7

1.1 Introducción 7
1.2 Conceptos relacionados con el proceso de gestión de estrategias de superación pedagógica 7
1.3 Conceptos relacionados con los sistemas de automatización de superación pedagógica 8
1.4 Sistemas existentes relacionados con la superación pedagógica 9
1.5 Tecnologías y lenguaje de programación 13
1.6 Marcos de Trabajo 14
1.7 Lenguaje de Modelado 15
1.8 Herramientas para el desarrollo 16
1.8.1 Herramienta para el modelado de diagramas 16
1.8.2 Entorno Integrado de Desarrollo (IDE por sus siglas en inglés) 16
1.8.3 Sistema gestor de bases de datos (SGBD) 17
1.8.4 Herramienta para la gestión de SGBD 19
1.8.5 Herramienta de pruebas 19
1.8.6 Herramientas de Mapeo Objeto-Relacional (ORM) 20
1.9 Metodologías de desarrollo de software 21
1.9.1 Programación extrema 21
1.9.2 Scrum 22
1.9.3 OpenUP 22
1.10 Conclusiones parciales 24

CAPÍTULO 2 CARACTERÍSTICAS DE LA PROPUESTA DE SOLUCIÓN 25

2. 1 Introducción 25
2. 2 Caracterización del sistema actual 25
2. 3 Propuesta de solución 26
2. 4 Funcionalidades del sistema 27
2. 5 Fase 1: Exploración. 30
2. 6 Fase 2: Planificación 33
2. 7 Fase 3: Diseño 38
2.8 Arquitectura del sistema 41
2.9 Patrones de diseño 42
2.10 Conclusiones parciales 47

CAPÍTULO 3 IMPLEMENTACIÓN Y PRUEBA 48

3.1 Introducción 48
3.2 Fase de implementación 48
3.2.1 Estándar de codificación 48
3.3 Diagrama de despliegue 49
3.4 Estrategia de pruebas 50
3.4.1 Pruebas unitarias 50
3.5 Resultados de las pruebas 55
3.6 Conclusiones parciales 59

CONCLUSIONES GENERALES 60

RECOMENDACIONES 61

BIBLIOGRAFÍA 62

ANEXOS QUE COMPLEMENTAN LA MEMORIA ESCRITA 65

ANEXO 1: CATEGORÍA Y RASGOS DEL PROFESOR 65
ANEXO 2: ENTREVISTA. INSTRUMENTO PARA COMPROBAR EL CONOCIMIENTO PREVIO QUE PRESENTA EL PROCESO DE GESTIÓN DE ESTRATEGIAS DE SUPERACIÓN PEDAGÓGICA EN EL CENTRO DE INNOVACIÓN Y CALIDAD DE LA EDUCACIÓN. 66
ANEXO 3: ILUSTRACIÓN DE CADA UNA DE LAS CLASES DEL MODELO DE DOMINIO QUE EXPRESAN LOS CONCEPTOS FUNDAMENTALES DEL DOMINIO DE LA INVESTIGACIÓN 68
ANEXO 4: HISTORIAS DE USUARIOS 69
ANEXO 5: TARJETAS CRC 125
ANEXO 6: MODELO ENTIDAD RELACION (MER) QUE COMPLEMENTA EL MODELO DE DATOS PROPUESTO 137
ANEXO 7: IMPLEMENTACIÓN DE LAS FUNCIONALIDADES PARA LAS PRUEBAS DE CAJA BLANCA UTILIZANDO LA TÉCNICA DEL CAMINO BÁSICO. 138
ANEXO 8: CASOS DE PRUEBA DE ACEPTACIÓN 141

Índice de Figuras

Figura 1. Fases de la metodología XP. Fuente: (Escribano, 2002). 22
Figura 2. Modelo de dominio. Fuente: Elaboración propia. 27
Figura 3. Modelo de datos. Fuente: Elaboración propia. 41
Figura 4. Patrón arquitectónico MVC (Modelo Vista Controlador). Fuente: (Fabien Potencier, 2013) 42
Figura 5. Ejemplo de patrón experto. Fuente: Elaboración propia. 44
Figura 6. Ejemplo de patrón creador. Fuente: Elaboración propia. 44
Figura 7. Ejemplo de patrón controlador. Fuente: Elaboración propia. 45
Figura 8. Ejemplo de patrón de alta cohesión. Fuente: Elaboración propia. 45
Figura 9. Ejemplo de patrón de bajo acoplamiento. Fuente: Elaboración propia. 46
Figura 10. Ejemplo del patrón Decorador. Fuente: Elaboración propia. 47
Figura 11. Diagrama de despliegue. Fuente: Elaboración propia. 50
Figura 12. Grafo de flujo y Complejidad ciclomática correspondiente al método personalAction. Fuente: Elaboración propia 52
Figura 13. Grafo de flujo y Complejidad ciclomática correspondiente al método createAction. Fuente: Elaboración propia 53
Figura 14. Resultados de las pruebas unitarias. Fuente: Elaboración propia. 56
Figura 15. Evaluación profesoral. Fuente: Elaboración propia. 56
Figura 16. Diagrama de las pruebas de validación. Fuente: Elaboración propia. 57
Figura 17. Resultados de las pruebas del sistema. Mostrar profesor. Fuente: Elaboración propia. 57
Figura 18. Resultados de las pruebas del sistema. Editar profesor. Fuente: Elaboración propia. 58
Figura 19. Resultados de las pruebas del sistema. Eliminar profesor. Fuente: Elaboración propia. 58

Índice de Tablas

Tabla 1. Sistemas analizados y algunas de sus características. Fuente: Elaboración propia. 13
Tabla 2. Descripción de las características de algunos SGBD. Fuente: Elaboración propia. 17
Tabla 3. HU29: Registrar profesor. Fuente: Elaboración propia 31
Tabla 4. HU46: Registrar curso de superación. Fuente: Elaboración propia 31
Tabla 5. Roles de usuario. Fuente: Elaboración propia. 32
Tabla 6. Plan de estimación de esfuerzo por historias de usuario. Fuente: Elaboración propia. 33
Tabla 7. Plan de entrega. Fuente: Elaboración propia. 35
Tabla 8. Plan de duración de iteraciones. Fuente: Elaboración propia. 36
Tabla 9. Tarjeta CRC_19. Clase Persona. Fuente: Elaboración propia. 39
Tabla 10. Tarjeta CRC_20. Clase PersonaController. Fuente: Elaboración propia. 39
Tabla 11. Caso de prueba: Registrar profesor. Fuente: Elaboración propia. 53
Tabla 12. Caso de prueba: Registrar curso de superación. Fuente: Elaboración propia. 54
Tabla 13. HU1 Registrar categoría docente. Fuente: Elaboración propia. 69
Tabla 14. HU2 Listar categoría docente. Fuente: Elaboración propia. 69
Tabla 15. HU3 Editar categoría docente. Fuente: Elaboración propia. 70
Tabla 16. HU4 Eliminar categoría docente. Fuente: Elaboración propia. 71
Tabla 17. HU5 Registrar categoría científica. Fuente: Elaboración propia. 71
Tabla 18. HU6 Listar categorías científicas. Fuente: Elaboración propia. 72
Tabla 19. HU7 Editar categoría científica. Fuente: Elaboración propia 73
Tabla 20. HU8 Eliminar categoría científica. Fuente: elaboración propia. 73
Tabla 21. HU9 Registrar área. Fuente: Elaboración propia. 74
Tabla 22. HU10 Listar áreas. Fuente: Elaboración propia. 75
Tabla 23. HU11 Editar área. Fuente: Elaboración propia. 76
Tabla 24. HU12 Eliminar área. Fuente: Elaboración propia. 77
Tabla 25. HU13 Registrar cargo. Fuente: Elaboración propia. 77
Tabla 26. HU14 Listar cargos. Fuente: Elaboración propia. 78
Tabla 27. HU15 Editar cargo. Fuente: Elaboración propia. 78
Tabla 28. HU16 Eliminar cargo. Fuente: Elaboración propia. 79
Tabla 29. HU17 Registrar tipo de clase. Fuente: Elaboración propia. 79
Tabla 30. HU18 Listar tipos de clase. Fuente: Elaboración propia. 80
Tabla 31. HU19 Editar tipo de clase. Fuente: Elaboración propia. 80
Tabla 32. HU20 Eliminar tipo de clase. Fuente: Elaboración propia. 81
Tabla 33. HU21 Registrar período. Fuente: Elaboración propia. 81
Tabla 34. HU22 Listar períodos. Fuente: Elaboración propia. 82
Tabla 35. HU23 Editar período. Fuente: Elaboración propia. 82
Tabla 36. HU24 Eliminar período. Fuente: Elaboración propia. 83
Tabla 37. HU25 Registrar grupo. Fuente: Elaboración propia. 83
Tabla 38. HU26 Listar grupos. Fuente: Elaboración propia. 84
Tabla 39. HU27 Editar grupo. Fuente: Elaboración propia. 84
Tabla 40. HU28 Eliminar grupo. Fuente: Elaboración propia. 85
Tabla 41. HU29 Registrar profesor. Fuente: Elaboración propia. 85
Tabla 42. HU30 Listar profesores. Fuente: Elaboración propia. 86
Tabla 43. HU31 Editar profesor. Fuente: Elaboración propia. 86
Tabla 44. HU32 Eliminar profesor. Fuente: Elaboración propia. 87
Tabla 45. HU33 Registrar organización. Fuente: Elaboración propia. 87
Tabla 46. HU34 Listar organizaciones. Fuente: Elaboración propia. 88
Tabla 47. HU35 Editar organización. Fuente: Elaboración propia. 88
Tabla 48. HU36 Eliminar organización. Fuente: Elaboración propia. 89
Tabla 49. HU37 Configurar sistema. Fuente: Elaboración propia. 89
Tabla 50. HU38 Respaldar sistema. Fuente: Elaboración propia. 90
Tabla 51. HU39 Cerrar sesión por inactividad. Fuente: Elaboración propia. 91

Tabla 52. HU40 Consumir estructura. Fuente: Elaboración propia. 91
Tabla 53. HU41 Autenticar usuario. Fuente: Elaboración propia. 92
Tabla 54. HU42 Filtrar profesores. Fuente: Elaboración propia. 93
Tabla 55. HU43 Exportar historial del profesor. 94
Tabla 56. HU44 Asignar rol. Fuente: Elaboración propia. 94
Tabla 57. HU45 Crear estrategia de superación pedagógica. Fuente: Elaboración propia. 95
Tabla 58. HU46 Registrar curso de superación. Fuente: Elaboración propia. 96
Tabla 59. HU47 Listar cursos de superación. Fuente: Elaboración propia. 97
Tabla 60. HU48 Editar curso de superación. Fuente: Elaboración propia. 97
Tabla 61. HU49 Eliminar curso de superación. Fuente: Elaboración propia. 98
Tabla 62. HU50 Mostrar curso de superación. Fuente: Elaboración propia. 98
Tabla 63. HU51 Crear GTFP. Fuente: Elaboración propia. 99
Tabla 64. HU52 Listar GTFP. Fuente: Elaboración propia. 100
Tabla 65. HU53 Editar GTFP. Fuente: Elaboración propia. 100
Tabla 66. HU54 Eliminar GTFP. Fuente: Elaboración propia. 101
Tabla 67. HU55 Registrar control. Fuente: Elaboración propia. 101
Tabla 68. HU56 Listar controles. Fuente: Elaboración propia. 102
Tabla 69. HU57 Editar control. Fuente: Elaboración propia. 103
Tabla 70. HU58 Eliminar control. Fuente: Elaboración propia. 103
Tabla 71. HU59 Mostrar control. Fuente: Elaboración propia. 104
Tabla 72. HU60 Registrar evaluación profesoral. Fuente: Elaboración propia. 105
Tabla 73. HU61 Listar evaluaciones profesorales. Fuente: Elaboración propia. 106
Tabla 74. HU62 Editar evaluación profesoral. Fuente: Elaboración propia. 106
Tabla 75. HU63 Eliminar evaluación profesoral. Fuente: Elaboración propia. 107
Tabla 76. HU64 Mostrar evaluación profesoral. Fuente: Elaboración propia. 107
Tabla 77. HU65 Exportar registro de control a clase. Fuente: Elaboración propia. 108
Tabla 78. HU66 Exportar registro de evaluación profesoral. Fuente: Elaboración propia. 109
Tabla 79. HU67 Exportar información del GTFP. Fuente: Elaboración propia. 110
Tabla 80. HU68 Registrar caso. Fuente: Elaboración propia. 111
Tabla 81. HU69 Listar casos. Fuente: Elaboración propia. 112
Tabla 82. HU70 Editar caso. Fuente: Elaboración propia. 113
Tabla 83. HU71 Eliminar caso. Fuente: Elaboración propia. 113
Tabla 84. HU72 Mostrar caso. Fuente: Elaboración propia. 114
Tabla 85. HU73 Atender peticiones. Fuente: Elaboración propia. 115
Tabla 86. HU74 Registrar categoría. Fuente: Elaboración propia. 116
Tabla 87. HU75 Listar categorías. Fuente: Elaboración propia. 117
Tabla 88. HU76 Editar categoría. Fuente: Elaboración propia. 117
Tabla 89. HU77 Eliminar categoría. Fuente: Elaboración propia. 117
Tabla 90. HU78 Registrar programa. Fuente: Elaboración propia. 118
Tabla 91. HU79 Listar programa. Fuente: Elaboración propia. 118
Tabla 92. HU80 Editar programa. Fuente: Elaboración propia. 119
Tabla 93. HU81 Eliminar programa. Fuente: Elaboración propia. 119
Tabla 94. HU82 Exportar información de los programas. Fuente: Elaboración propia. 120
Tabla 95. HU83 Registrar característica. Fuente: Elaboración propia. 121
Tabla 96. HU84 Listar características. Fuente: Elaboración propia. 121
Tabla 97. HU85 Editar característica. Fuente: Elaboración propia. 122
Tabla 98. HU86 Eliminar característica. Fuente: Elaboración propia. 122
Tabla 99. HU87 Mostrar característica. Fuente: Elaboración propia. 123
Tabla 100. Tarjeta CRC 1. Clase Categoría. Fuente: Elaboración propia. 125
Tabla 101. Tarjeta CRC 2. Clase CategoriaController. Fuente: Elaboración propia. 125
Tabla 102. Tarjeta CRC 3. Clase CategoríaDocente. Fuente: Elaboración propia. 125
Tabla 103. Tarjeta CRC 4. Clase CategoriaDocenteController. Fuente: Elaboración propia. 125
Tabla 104. Tarjeta CRC 5. Clase CategoríaCientífica. Fuente: Elaboración propia. 126

Tabla 105. Tarjeta CRC 6. Clase CategoriaCientificaController. Fuente: Elaboración propia. 126
Tabla 106. Tarjeta CRC 7. Clase Área. Fuente: Elaboración propia. 126
Tabla 107. Tarjeta CRC 8. Clase AreaController. Fuente: Elaboración propia. 127
Tabla 108. Tarjeta CRC 9. Clase Cargo. Fuente: Elaboración propia. 127
Tabla 109. Tarjeta CRC 10. Clase CargoController. Fuente: Elaboración propia. 127
Tabla 110. Tarjeta CRC 11. Clase Programa. Fuente: Elaboración propia. 127
Tabla 111. Tarjeta CRC 12. Clase ProgramaController. Fuente: Elaboración propia. 128
Tabla 112. Tarjeta CRC 13. Clase TipoClase. Fuente: Elaboración propia. 128
Tabla 113. Tarjeta CRC 14. Clase TipoClaseController. Fuente: Elaboración propia. 128
Tabla 114. Tarjeta CRC 15. Clase Período. Fuente: Elaboración propia. 129
Tabla 115. Tarjeta CRC 16. Clase PeríodoController. Fuente: Elaboración propia. 129
Tabla 116. Tarjeta CRC 17. Clase Grupo. Fuente: Elaboración propia. 129
Tabla 117. Tarjeta CRC 18. Clase GrupoController. Fuente: Elaboración propia. 129
Tabla 118. Tarjeta CRC 19. Clase Persona. Fuente: Elaboración propia 130
Tabla 119. Tarjeta CRC 20. Clase PersonaController. Fuente: Elaboración propia. 130
Tabla 120. Tarjeta CRC 21. Clase Organización. Fuente: Elaboración propia. 130
Tabla 121. Tarjeta CRC 22. Clase OrganizaciónController. Fuente: Elaboración propia. 131
Tabla 122. Tarjeta CRC 23. Clase Característica. Fuente: Elaboración propia. 131
Tabla 123. Tarjeta CRC 24. Clase CaracteristicaController. Fuente: Elaboración propia. 131
Tabla 124. Tarjeta CRC 25. Clase CursoSuperacProf. Fuente: Elaboración propia. 132
Tabla 125. Tarjeta CRC 26. Clase CursoSuperacProfController. Fuente: Elaboración propia. 132
Tabla 126. Tarjeta CRC 27. Clase ProfGtfp. Fuente: Elaboración propia. 132
Tabla 127. Tarjeta CRC 28. Clase ProgGtfpController. Fuente: Elaboración propia. 132
Tabla 128. Tarjeta CRC 29. Clase Control. Fuente: Elaboración propia. 133
Tabla 129. Tarjeta CRC 30. Clase ControlController. Fuente: Elaboración propia. 133
Tabla 130. Tarjeta CRC 31. Clase EvalProfesoral. Fuente: Elaboración propia. 134
Tabla 131. Tarjeta CRC 32. Clase EvalProfesoralController. Fuente: Elaboración propia. 134
Tabla 132. Tarjeta CRC 33. Clase Caso. Fuente: Elaboración propia. 134
Tabla 133. Tarjeta CRC 34. Clase CasoController. Fuente: Elaboración propia. 134
Tabla 134. Tarjeta CRC 35. Clase Rol. Fuente: Elaboración propia. 135
Tabla 135. Tarjeta CRC 36. Clase RolController. Fuente: Elaboración propia. 135
Tabla 136. Tarjeta CRC 37. Clase Usuario. Fuente: Elaboración propia. 135
Tabla 137. Tarjeta CRC 38. Clase UsuarioController. Fuente: Elaboración propia. 136
Tabla 138. Caso de prueba: Autenticar usuario. Fuente: Elaboración propia. 141
Tabla 139. Caso de prueba: Consumir estructura. Fuente: Elaboración propia. 141
Tabla 140. Caso de prueba: Registrar profesor. Fuente: Elaboración propia. 142
Tabla 141. Caso de prueba: Registrar curso de superación. Fuente: Elaboración propia. 143

INTRODUCCIÓN

En los centros universitarios se enfatiza en la importancia que tiene la capacitación de los profesores. El trabajo se encamina al logro de un mayor conocimiento de las Ciencias de la Educación profundizando en la Didáctica y en las concepciones actuales de la Pedagogía. El docente necesita habilidades y herramientas que le permitan el desarrollo de una comunicación pedagógica adecuada (Carlderón Ariosa, y otros, 2002). La superación profesoral tiene como objetivo la formación permanente y la actualización sistemática de los profesores, el perfeccionamiento del desempeño de sus actividades docentes, así como el enriquecimiento de su acervo cultural. Para elevar la preparación de los docentes, la Superación Profesional como vertiente de la Educación de Postgrado y el Trabajo Metodológico, constituyen vías eficaces que permiten el logro de este fin (Addine Fernández, y otros, 2007).

La formación constituye uno de los procesos principales de la Gestión de Recursos Humanos. Su trascendencia en la preparación de los miembros de las organizaciones, hace frente a las demandas que impone la época contemporánea, en que el "medio de producción" fundamental es y será el conocimiento. La anterior aseveración es válida para todos los sectores de la vida económica y social, y lo es mucho más para la educación, en la que se plantean nuevos retos y desafíos que son generadores de trascendentes procesos de cambio, de diferentes formas y alcances, a los que identifica como objetivo común el logro de la calidad educacional (Addine Fernández, y otros, 2007).

De acuerdo con lo planteado por los autores, se considera que la formación y la superación pedagógica del claustro, se describe como una de las principales vías para el desarrollo docente metodológico y científico. A su vez, contribuye al desarrollo de habilidades en docentes que tributan a la calidad del proceso educativo, teniendo como objetivo perfeccionar el modelo docente existente y a partir de este fortalecer la superación profesional como tarea fundamental del trabajo metodológico.

La superación pedagógica a los docentes es concebida por el Ministerio de Educación Superior mediante la Educación de Postgrado. Se ha identificado que existen deficiencias en esta área del conocimiento, detectadas en un análisis realizado a diferentes diseños de superación pedagógica en el país por (Bravo, 2013), quien constata que existe:

- Diferencias en el reconocimiento de necesidades de superación en el área pedagógica.
- Variedad de contextos en que se desarrolla la preparación pedagógica.
- Diversidad de objetivos y contenidos declarados en el área pedagógica para organizar la superación profesional y la formación académica.

A partir de las transformaciones económicas y sociales que han surgido en la evolución del hombre, se ha producido un auge en el desarrollo de herramientas para fortalecer el trabajo de forma amena, funcional y económica. Con el incremento de este desarrollo, se han marcado pautas que conllevan al surgimiento de las Tecnologías de la Informática y las Comunicaciones (TIC). El incesante crecimiento de las TIC hace que el conocimiento y los avances científico-técnicos sean enriquecidos constantemente y se produzca un cúmulo de información proveniente de diferentes medios existentes. Es por ello que, en nuestro país, las TIC es un aspecto fundamental para el avance socioeconómico de la Revolución.

La Universidad de las Ciencias Informáticas (UCI) es la primera universidad surgida al calor de la batalla de ideas como bastión tecnológico para el desarrollo científico de Cuba. Su objetivo es desarrollar aplicaciones que brinden servicios informáticos con calidad, seguridad y garantía, que permitan obtener un alto prestigio a nivel nacional y extranjero. El Centro de Innovación y Calidad de la Educación (CICE), perteneciente a la UCI, no está ajeno a este constante desarrollo y contribuye al mismo con varias soluciones informáticas, entre los que se encuentra el Sistema Automatizado para la Superación Pedagógica del claustro UCI (SASPED).

El SASPED tiene como principal objetivo trazar la estrategia individual de superación pedagógica de cada profesor del claustro UCI. Para cumplir este objetivo el sistema permite, a partir de cada profesor, gestionar los controles a clases, tanto recibidos como realizados; las evaluaciones profesorales por periodo y los grupos de trabajos de formación pedagógica al cual pertenece. Además, a partir del subsistema experto integrado al sistema; que utiliza 34 rasgos del profesor (Ver Anexo 1) agrupados en 5 categorías (Socio-Afectiva, Pedagogía y Didáctica General, Conocimientos del Área, Controles a Clases y Evaluación Profesoral) sugiere una estrategia individual de superación pedagógica, que no es más que un conjunto de actividades de superación a realizar por el profesor según las 5 categorías antes mencionadas.

A partir de la utilización de las fuentes de información, su análisis y síntesis, permitió determinar regularidades, coincidir con planteamientos y arribar a contradicciones. Lo cual permitió identificar la siguiente situación:

El SASPED fue desarrollado con el *framework* Seam 2.1, el cual es una plataforma de desarrollo (de código abierto) para construir aplicaciones enriquecidas en Internet (Web 2.0) usando JAVA. Este framework, se empleó en el desarrollo de la solución existente para integrar tecnologías como *JavaScript* y XML (AJAX), *Java Server Faces* (JSF), *Java Persistence* (JPA), *Enterprise Java Beans* (EJB 3.0) y *Business Process Management* (BPM) en una única solución completa, con sofisticadas

herramientas que se integran con eclipse. En la actualidad, se evidencia una evolución del mismo hacia otras versiones (versión 3.1), pero se ha identificado que esta evolución se ha detenido y sus proyectos se han disuelto hacia otras tecnologías y hacia otros proyectos. Esta nueva versión del framework cambió su paradigma de desarrollo, por tanto, no es posible cambiar una aplicación desarrollada en Seam 2.1 a Seam 3.1.

Teniendo en cuenta esta situación, el sistema presenta dificultades en la gestión de la información de la caracterización del claustro, de los datos de los controles a clases, de la evaluación profesoral, de los grupos trabajo de formación pedagógica y de la estrategia individual de superación pedagógica. Además, la tecnología que se utilizó para el desarrollo está orientada a grandes procesos empresariales y su utilización en procesos pequeños dificulta su desarrollo. Hoy día, esta tecnología se encuentra obsoleta y sin soporte técnico, lo que dificulta la incorporación de nuevas funcionalidades, propicia problemas con el tiempo de desarrollo e impide la escalabilidad del sistema.

A continuación, se describe cómo se genera y se procesa el flujo de los procesos en SASPED y se presentan las deficiencias que impiden trazar la estrategia individual de superación pedagógica:

- **Controles a clase:** Actualmente este proceso permite introducir de manera correcta los datos pertenecientes a los controles a clase. Sin embargo, el proceso no se lleva a cabo correctamente, aunque los campos del formulario son llenados con resultados satisfactorios, el sistema no satisface las exigencias actuales de los controles a clases a un profesor. Este problema trae consigo insuficiencias en el proceso general, lo cual impide trazar la estrategia de superación profesoral de cada docente.

- **Evaluación profesoral:** El proceso de evaluación profesoral se gestiona de forma correcta. En la actualidad los indicadores por los que se rige este proceso se encuentran desactualizados, lo cual trae como consecuencia que se introduzcan datos erróneos. Al presentar esta insuficiencia, trae consigo que la estrategia de superación pedagógica trazada no cumpla con las exigencias del docente.

- **Grupos de trabajo de formación pedagógica:** Existe una necesidad de fomentar la realización de diferentes grupos de trabajo, con el objetivo de ampliar el desarrollo educativo y contribuir a la formación profesional y al trabajo educativo-metodológico. En la actualidad, el sistema gestiona los grupos de trabajo, asignando a cada uno un jefe. Para la creación del grupo de trabajo no se tiene en cuenta la pirámide de tutoría, donde el profesor de mayor categoría

docente sería el jefe del grupo. Esta insuficiencia impide que el sistema cumpla con el proceso de la estrategia de superación pedagógica al cual responde.

En consecuencia, con la situación descrita, se identifica que existen insuficiencias e irregularidades en la gestión de la información de la caracterización del claustro y la tecnología con la que fue desarrollado SASPED; así como dificultades para trazar la estrategia individual de superación pedagógica y lograr la escalabilidad del software desarrollado.

Las insuficiencias anteriormente descritas permitieron identificar el siguiente **problema científico**: ¿Cómo perfeccionar la gestión de los procesos del sistema automatizado de la superación pedagógica para que contribuya a los niveles de escalabilidad del software?

El **objeto de estudio** lo constituye el proceso de gestión de estrategias de superación pedagógica en el Centro de Innovación y Calidad de la Educación. Se identifica como **campo de acción** los sistemas automatizados de superación pedagógica.

Para resolver el problema científico se propone el siguiente **objetivo general:** Desarrollar un sistema para la automatización de los procesos de la estrategia de superación pedagógica, que contribuya a la escalabilidad del software, en el Centro de Innovación y Calidad de la Educación.

Objetivos específicos:

1. Analizar los referentes teórico-metodológicos relacionados con los sistemas automatizados de superación pedagógica.
2. Caracterizar el proceso de gestión de estrategias de superación pedagógica en el Centro de Innovación y Calidad de la Educación.
3. Implementar el sistema para la automatización de los procesos de la estrategia de superación pedagógica.
4. Validar la contribución del sistema automatizado propuesto.

Idea a defender: Si se desarrolla el sistema para la automatización de los procesos de la estrategia de superación pedagógica, en el Centro de Innovación y Calidad de la Educación, se contribuye a elevar la escalabilidad del software.

El proceso de investigación se orientó a través de las siguientes **tareas de investigación**:

1. Comprobación de la validez del problema científico planteado.
2. Análisis del estado del arte referente a los sistemas automatizados de superación pedagógica.

3. Caracterización del proceso gestión de estrategias de superación pedagógica en el Centro de Innovación y Calidad de la Educación.
4. Fundamentación de las tecnologías, herramientas, metodología y lenguaje necesarios para el desarrollo del sistema.
5. Modelación del ciclo de desarrollo del software teniendo en cuenta la metodología seleccionada.
6. Implementación del sistema para la automatización de los procesos de la estrategia de superación pedagógica, que contribuya a elevar la escalabilidad del software, en el Centro de Innovación y Calidad de la Educación.
7. Validar la propuesta a partir de las pruebas de unidad, de sistema y de validación.

Se utilizaron los métodos teóricos siguientes:

- **Histórico – lógico**: La utilización de las fuentes de información referentes al proceso de gestión de estrategias de superación pedagógica y los sistemas automatizados de superación pedagógica, permitió determinar los antecedentes, tendencias y regularidades actuales.
- **Analítico – sintético**: Permitió un estudio, análisis y síntesis del proceso de gestión de estrategias de superación pedagógica, a través de la fundamentación y los razonamientos, lo cual significa coincidir con planteamiento y/o plantear contradicciones, para tener criterios en la toma de decisiones.
- **Hipotético – deductivo**: Permitió la verificación de las nuevas hipótesis sobre la realidad, así como las nuevas estrategias de desarrollo para la automatización de los procesos de la estrategia de superación pedagógica y el establecimiento de predicciones a partir de los sistemas automatizados de superación pedagógica. Lo cual permitió arribar a nuevos conocimientos y predicciones, que posteriormente fueron sometidos a verificaciones empíricas.
- **Sistémico**: Permitió favorecer la automatización de los procesos de la estrategia de superación pedagógica y contribuir a la escalabilidad del software. Además de analizar, reproducir e integrar las propiedades del proceso de gestión de estrategias de superación pedagógica.
- **Modelación**: Se utilizó con el objetivo de generar los artefactos necesarios para estudiar y transformar los procesos de la estrategia de superación pedagógica, que contribuya a la escalabilidad del software, en el Centro de Innovación y Calidad de la Educación.

Como métodos empíricos se utilizaron los siguientes:

- **Observación**: Se utilizó para obtener una información más precisa acerca de la gestión de la información en la Superación Pedagógica del claustro de profesores en la Universidad de las Ciencias Informáticas.

- **Encuesta**: Se utilizó para la caracterización inicial del objeto que se investiga, la determinación de las contradicciones y específicamente del problema y el objetivo planteado. A la vez, permitió obtener información que caracteriza el estado de opinión acerca del objeto de estudio de la investigación.
- **Análisis documental**: Se utilizó con el objetivo de extraer la información necesaria para lograr un sistema para la automatización de los procesos de la estrategia de superación pedagógica, mediante la selección y recolección de la documentación asociada al proceso que se estudia y a los sistemas automatizados asociados a la superación pedagógica.
- **Análisis de contenido**: Se utilizó este método con el objetivo de analizar y valorar la información obtenida y de esta forma descubrir una serie de ideas, que no están explícitas en los procesos de la estrategia de superación pedagógica.

Resultados esperados:

- Informe de investigación con los artefactos que se diseñan para el desarrollo de la propuesta.
- Sistema para la automatización de los procesos de la estrategia de superación pedagógica, que contribuya a elevar la escalabilidad del software, en el Centro de Innovación y Calidad de la Educación.

Estructura de la investigación:

El trabajo de diploma consta de introducción, tres capítulos, conclusiones, recomendaciones, bibliografía y anexos. En el primer capítulo se presentan los fundamentos teórico – metodológicos del objeto de estudio y el campo de acción. En el segundo se especifica la caracterización del objeto de estudio y el análisis y el diseño de la propuesta. En el tercero se lleva a cabo la implementación y se valida el sistema.

CAPÍTULO 1 FUNDAMENTOS TEÓRICOS-METODOLÓGICOS PARA EL DESARRROLLO DEL SISTEMA

1.1 Introducción

En el presente capítulo se abordan los principales conceptos relacionados con la problemática a resolver. Se exponen las definiciones asociadas con el proceso de gestión de estrategias de superación pedagógica, los sistemas automatizados de superación pedagógica y se hace un estudio de sistemas homólogos en cuanto a objetivos de uso, su funcionamiento y tecnologías que utilizan. Se realiza un estudio científico del tema que se aborda para conformar el basamento teórico utilizado en la solución del problema científico. Además, se definen las herramientas, metodología y lenguaje a utilizar en la solución propuesta.

1.2 Conceptos relacionados con el proceso de gestión de estrategias de superación pedagógica

Para una mayor comprensión de la presente investigación, se analizan los conceptos asociados al proceso de gestión de estrategias de superación pedagógica, definido como objeto de estudio de la investigación.

El Diccionario de la Real Academia Española (DRAE, 2016) define el concepto de **"proceso"** de la siguiente manera: «conjunto de las fases sucesivas de un fenómeno natural o de operación artificial».

Por su parte (Mira, y otros, 2006) expresan que un **proceso** es un «conjunto de actuaciones, decisiones, actividades y tareas que se encadenan de forma secuencial y ordenada para conseguir un resultado que satisfaga plenamente los requerimientos del cliente al que va dirigido». Se toma en la presente investigación el concepto de los autores (Mira y otros, 2006).

Gestión: Según el Diccionario de la Real Academia Española (DRAE, 2016), se denomina gestión a la acción y efecto de gestionar y administrar. Sin embargo, según (Atehortúa Hurtado , 2005) lo define como actividades coordinadas para dirigir y controlar una organización. Se toma en cuenta la fundamentación de ambas definiciones para la presente investigación.

El Diccionario de la Real Academia Española (DRAE, 2016) define el concepto de "estrategia" de la siguiente manera: **Estrategia: «**proceso regulable, conjunto de las reglas que aseguran una decisión óptima en cada momento**».**

Por su parte (Friedman, 2007)**,** define Estrategia como un proceso sistemático de desarrollo e implementación de planes para alcanzar propósitos u objetivos. Se tiene en cuenta la definición dada por (Friedman, 2007) para la presente investigación.

Estrategia de Superación Pedagógica: Esta concepción supera cualquiera de las variantes históricas de comprensión de las tareas de instruir, enseñar y educar, así como de sus implicaciones metodológicas, como aspectos separados en el proceso de la formación y desarrollo integral de la personalidad (Blanco, y otros, 2002).

En el contexto concreto de la Pedagogía "la estrategia establece la dirección inteligente, y desde una perspectiva amplia y global, de las acciones encaminadas a resolver los problemas detectados en un determinado segmento de la actividad humana. Se entienden como problemas las contradicciones o discrepancias entre el estado actual y el deseado, entre lo que es y debería ser, de acuerdo con determinadas expectativas que dimanan de un proyecto social y/o educativo dado. Su diseño implica la articulación dialéctica entre los objetivos (metas perseguidas) y la metodología (vías instrumentadas para alcanzarlas)" (Cruz, y otros, 2013).

Teniendo en cuenta lo expresado anteriormente, el equipo de desarrollo considera que el proceso de gestión de estrategias de superación pedagógica constituye una de las tareas más significativas dentro del proceso de investigación. A través del análisis realizado se arribaron a conclusiones y se pudieron obtener resultados, permitiendo así profundizar en el conocimiento de la realidad que se investiga, teniendo como punto único entre los investigadores que el proceso de gestión de estrategias de superación pedagógica va encaminado a los pasos que se tomen para su realización, la forma en que se va a controlar y administrar la información, las acciones planificadas que ayudan a tomar decisiones y a conseguir los mejores resultados posibles que tributen al proceso de formación y desarrollo integral de la personalidad.

1.3 Conceptos relacionados con los sistemas de automatización de superación pedagógica

Según el Diccionario de la Real Academia Española (DRAE, 2016), se entiende por **sistema** al «conjunto de reglas o principios sobre una materia racionalmente enlazados entre sí». También se refiere al mismo como el «Conjunto de elementos que relacionados entre sí ordenadamente contribuyen a determinado objeto».

Por su parte, (Alegsa, 2014) expresa que un **sistema** es un conjunto de partes o elementos organizados y relacionados que interactúan entre sí para lograr un objetivo.

A su vez, el autor (Giménez, 2003) opina que un sistema es un plan práctico y completo (usando datos) para generar, coordinar y controlar las actividades de una organización. Se toma en cuenta el criterio abordado por la DRAE y autores anteriores al definir el concepto de sistema para la investigación.

Automatización: Para (Henao, 2009), consiste en tener a mano una información en tiempo real que sea accesible a todo el personal involucrado en la operación; su uso en el proceso provee un conjunto

de técnicas de comunicación, computación y equipamiento de oficina utilizadas con la finalidad de aumentar la productividad y calidad de la gestión de la operación.

Para (Cordanov, 2011) significa “la conversión de procesos manuales, en procesos más rápidos mediante implementos electrónicos”.

Sistema Automatizado: Según el autor (Cordanov, 2011), un Sistema Automatizado es “la automatización de un conjunto de procesos donde se transfieren tareas de producción, realizadas manualmente por operadores humanos a un conjunto de elementos tecnológicos”.

Sin embargo, para (Canto, 2010), sistema automatizado es el conjunto de partes interrelacionadas donde se trasfieren tareas de producción, realizadas habitualmente por operadores humanos a un conjunto de elementos tecnológicos que permite almacenar y procesar información. Ambos conceptos están estrechamente relacionados, y se tienen en cuenta los criterios dados por los autores para la investigación.

Superación pedagógica: Cuando se habla de superación pedagógica, (Ortiz Sánchez, y otros, 2014) lo definen como un proceso de autodesarrollo sistemático y contextual orientado a la auto preparación y formación en ámbitos didácticos-metodológicos, personal-social, científico-investigativo, que le promueva en ellos una identidad propia como personas con capacidad de aprender, de reflexionar sobre la actuación profesional pedagógica, ser responsables de las transformaciones que deben ocurrir en el mismo para el perfeccionamiento de su desempeño, implicarse en la resolución de tareas, estableciendo sus propias estrategias para el logro de la calidad en el aprendizaje de los alumnos.

De la conceptualización anterior se resaltan algunos aspectos importantes:
La superación pedagógica y didáctica adecuada, generará profesionales de la enseñanza críticos, autocríticos y reflexivos, cuya actividad científica les permitirá nutrirse de un conjunto de teorías, de la hermenéutica y de la investigación-acción para superar la informalidad, lo intuitivo, lo informativo o lo exclusivamente tecnológico o instrumentalista de la enseñanza en el aula universitaria. En esta medida se prioriza la transformación de los procesos manuales a procesos informatizados o automáticos con el objetivo de romper esquemas de acción del profesorado universitario sobre la enseñanza.

1.4 Sistemas existentes relacionados con la superación pedagógica

Actualmente en el mundo existen innumerables sistemas que permiten la gestión de la información de los procesos relacionados con la pedagogía y su concepción, así como el perfeccionamiento y formación de los docentes en aras de lograr una transformación educativa en el quehacer universitario. Se hace necesario un estudio con el objetivo de analizar la forma en que cada uno de los sistemas desde su perfil brinda soluciones a las diferentes problemáticas existentes.

En el **contexto extranjero** han sido elaborados y diseñados varios sistemas, con la finalidad de gestionar la información de la superación pedagógica. A continuación, se muestran algunos de estos sistemas:

SAGIT es el Sistema Informático para la gestión académica del Instituto del Teatro que se realizó desde el *inLab* en el año 2006. El objetivo del proyecto se enfoca en llevar a cabo el desarrollo de un sistema informático que permita no sólo informatizar los procesos de gestión académica; sino también ser el núcleo de un sistema de información que permitiera, en un futuro, introducir otras titulaciones o cursos que se desarrollaran en el Instituto del Teatro. Algunas de las características asociadas al sistema son: Facilita las tareas del día a día de la gestión académica de los centros, tener un único archivo de datos, ofrece herramientas para facilitar la toma de decisiones, facilidad de uso de la aplicación, sistema de información en entorno de red que soporta la gestión de los estudios en sus tres vertientes: procesos de planificación, de desarrollo y de evaluación, sistema de gestión, que apoya a determinados procesos de gestión de planes de estudios, estudiantes y expedientes, matrícula y evaluación de la actividad académica (información para la toma de decisiones). Disponible en: **<**http://inlab.fib.upc.edu/es/sagit-sistema-informatico-de-gestion-academica-del-instituto-del-teatro**>**

Teniendo como premisa la necesidad de actualización, organización y almacenamiento de los datos de los programas académicos de Postgrado, la Unidad de Postgrado de la UNEXPO Puerto Ordaz en Venezuela, desarrolló en el 2007 un Sistema para la administración integral de los Programas Académicos de Postgrado (**SAIPAP**), el mismo fue desarrollado con la intención de administrar de manera fácil, rápida y segura los datos que componen las solicitudes realizadas por la Unidad Regional de Postgrado. Se implementó usando un sistema de administración de contenidos (CMS) de código abierto construido con PHP, *MySQL* y el servidor web *Apache*. Permite la gestión de solicitudes de usuarios y el trabajo con envío y recepción de formularios de registros, es multiplataforma, además la base de datos es compatible con la mayoría de los sistemas de gestión de base de datos como *PostgreSQL*, *Firebird, SQLite, Apache Derby*, entre otros. Disponible en: **<**http://www.laccei.org/LACCEI2010-Peru/published/UM036_Farage.pdf**>**

Inika es una aplicación Web para la gestión académica y administrativa de un centro de enseñanza. Fue construida por la empresa española *Ukabi* en el año 2014, la cual está destinada a brindar servicios informáticos integrales, orientada al mercado empresarial y con una amplia experiencia en la implantación de soluciones de gestión, de equipos y redes informáticas. La solución es adaptable a centros de educación primaria, preescolar, bachiller, formación profesional, trabaja online contra la base de datos y es multi-idioma. Permite la gestión de alumnos, profesores, padres, notas, faltas, medidas correctoras, transporte, comedor, actividades extraescolares, listados, boletines, actas oficiales o

estadísticas; además permite generar listados, boletines, actas, estadísticas y para los usuarios avanzados: creador SQL, consulta y gráficos dinámicos. Utiliza SQL Server como motor de Base de Datos, el framework cliente Bootstrap, el framework cliente JQuery, con lo que podremos introducir más controles de usuario que faciliten la entrada y visualización de datos. Actualmente este software requiere la disminución del tiempo de ejecución de los procesos y mejorar la seguridad de la aplicación, reforzando los permisos y accesos. Disponible en: **<**http://www.inika.net/es/inicio**>**

Contexto nacional

En el año 2011, el Centro de Cibernética Aplicada a la Medicina (CECAM) desarrolló un sistema automatizado de gestión de la maestría Informática en Salud (SAGMIS), con el fin de coordinar actividades docentes y gestionar la información de estudiantes, profesores y graduados en aras de agilizar los procesos y obtener, de manera oportuna, la información necesaria a cada uno de ellos y a los directivos, para hacer un uso más eficiente y eficaz de los recursos. El sistema provee tres interfaces: una para estudiantes, otra para profesores y una tercera para presidente del C/A, coordinador de la maestría, secretario docente y controlador de expediente. También permite el control de acceso de los diferentes usuarios mediante permisos y contraseñas. De tal manera, consta de 6 módulos: Administración y seguridad**,** Matrícula, Gestión de información de profesores**,** Planificación y/o control de Actividades Docentes, Información Docente y Gestión de cuentas de correo e Internet. Disponible en: **<**http://www.rcim.sld.cu/revista_23/articulo_pdf/sistema.pdf**>**

En el mes de Febrero del año 2011 se diseñó en la Universidad de Ciencias Médicas de Cienfuegos un sistema informático para la gestión del contenido farmacológico (SIGCF), la cual constituye un soporte durante la ejecución de las actividades de superación profesional del profesorado de la carrera de medicina para enfrentar el perfeccionamiento del proceso de enseñanza aprendizaje de la Farmacología durante la formación del Médico General, permitiendo gestionar los contenidos de esta disciplina para luego contribuir a la superación profesional pedagógica de los profesores de dicha carrera. En esencia, este sistema agrupa sus principales funcionalidades en tres aspectos fundamentales, los cuales se centran en la auto preparación de los estudiantes; la gestión de materiales docentes por parte de los profesores y control administrativo y gestión de datos y privilegios de los profesores que integran el sistema. Disponible en: **<**http://www.odiseo.com.mx/bitacora-educativa/2011/02/herramienta-informatica-para-superacion-profesional-farmacologia-profesor.html**>**

La Universidad de las Ciencias Informáticas (UCI), y específicamente el Centro de Tecnologías para la Formación (FORTES), se dio a la tarea de implementar un Sistema de Gestión de los Contenidos (SGC) del Microdiseño Curricular desde la Plataforma Educativa Zera en el año 2012, que integra los principales conceptos de los Hiper-entornos de Aprendizaje (HEA), las mejores prácticas y elementos

arquitectónicos de soluciones similares, así como las principales especificaciones y estándares educativos desarrollados y utilizados a nivel mundial en plataformas de aprendizaje colaborativo. La Plataforma Educativa Zera cuenta con varios módulos, de manera que se encuentran informatizados muchos procesos que son necesarios en las escuelas o instituciones educativas. Estos procesos son muy factibles tanto para el alumno como para el profesor que, aunque cuente con este medio seguiría jugando un papel muy importante dentro del proceso de enseñanza-aprendizaje. Disponible en: **<**http://publicaciones.uci.cu/index.php/SC/article/view/864**>**

Akademos o Sistema de Gestión Universitaria (SGU), es un software construido por un equipo de trabajo de la Dirección de Informatización de la UCI, se denomina Sistema Automatizado para la Gestión Académica. Es un sistema Web distribuido, 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. Sus funcionalidades se agrupan en siete módulos, de los cuales el Plan de Estudio es la entidad fundamental, pues rige todos los subprocesos. Akademos es un sistema de gestión académica que se utiliza en la UCI. Cuenta con varias funcionalidades y módulos que lo hacen útil en la institución. Uno de los módulos por los cuales está conformado es el Plan de estudio, en dependencia del periodo lectivo le muestra al usuario las materias que cursa el mismo. Este módulo se define como una sucesión de periodos de tiempo llamados niveles y momentos, los cuales mantienen un orden de presencia, lo que permite generar la estructura estática en años y semestres de un centro de estudios. El problema fundamental del módulo consiste en que se utiliza mucho el negocio de la UCI en su implementación, lo cual unido a su configuración estática ofrece poca flexibilidad para su despliegue en otros centros. Otras de las funciones de Akademos es permitir la configuración de las evaluaciones de las asignaturas, estas pueden ser frecuentes, parciales o finales. Disponible en: **<**http://publicaciones.uci.cu/index.php/SC/article/view/4**>**

Los sistemas antes expuestos comparten ventajas que son aprovechadas como referentes para la solución propuesta en la investigación:

- Permiten la gestión y control administrativo de la información que generan las actividades ofertadas (Proceso de inscripción, seguimiento y control de las actividades ofertadas, emisión de resultados, proceso informativo).
- Eliminan el procesamiento manual de toda la información disponible.
- Generación automatizada de informes.
- Favorecen la elevación de la calidad en el diagnóstico.

- Sirven de base para el aprendizaje con auxilio de tecnologías informáticas.

A continuación, se lleva a cabo una comparación entre los sistemas estudiados con vista a profundizar en aspectos de interés para la nueva propuesta de solución, teniendo en cuenta los criterios que se describen en la siguiente tabla:

Tabla 1. Sistemas analizados y algunas de sus características. Fuente: Elaboración propia.

Sistemas	Brinda una Estrategia de superación	Filtros de búsqueda	Sistema multiplataforma	Generar reportes	Escalable	Dominio específico
SAGIT				X	X	X
SAIPAP			X		X	X
Inika		X	X	X	X	
SAGMIS	X	X	X			X
SIGCF	X			X		X
SGC		X	X	X	X	
Akademos		X	X	X	X	X

El análisis realizado sobre los sistemas similares tanto en el ámbito nacional como internacional, arrojó las siguientes conclusiones:

Los sistemas estudiados no gestionan la información necesaria para generar de forma automatizada la estrategia de superación pedagógica, la mayoría de los sistemas poseen funcionalidades que responden solamente al proceso específico para el que está diseñado cada sistema. Además, no cuentan con funcionalidades que gestionen los grupos de trabajos de formación pedagógica a los que pertenecen los profesores. Por lo antes descrito, se presenta una propuesta de sistema para la automatización de los procesos de la estrategia de superación pedagógica del claustro de la UCI.

1.5 Tecnologías y lenguaje de programación

Lenguaje de Programación

Un lenguaje de programación es un lenguaje diseñado para describir un conjunto de acciones consecutivas, permitiendo crear programas mediante instrucciones, operadores y reglas de sintaxis para comunicarse con los dispositivos hardware y software de una máquina (CCM, 2016).

Actualmente existen diferentes lenguajes de programación para desarrollar en la web, estos han ido surgiendo debido a las tendencias y necesidades de las plataformas. El uso de estos lenguajes garantiza un diseño óptimo, eficiente, seguro y dinámico en las aplicaciones web (Valdés, 2007).

Basado en el tiempo limitado que se tiene para el desarrollo del sistema es necesario definir un lenguaje de programación que permita avanzar rápidamente. Para ello se tuvieron en cuenta varios indicadores: conocimiento y experiencia del equipo de desarrollo, integración y seguridad. Los lenguajes analizados fueron Java, PHP, C++ y Python.

Teniendo en cuenta los indicadores antes mencionados, se determinó utilizar PHP para la implementación del sistema teniendo en cuenta que es un lenguaje muy potente que, junto con HTML, permite crear sitios web dinámicos, con potencialidad y robustez, ofreciendo potencialidad y capacidad multiplataforma. Además, el equipo de desarrollo tiene avanzados conocimientos sobre este lenguaje y poseen amplias experiencias, capacidades y habilidades con el desarrollo de aplicaciones en PHP. Otras de las razones que determinó utilizar este lenguaje es la de lograr una completa integración entre las funcionalidades y servicios que conforman la concepción de SASPED.

1.6 Marcos de Trabajo

Un marco de trabajo o *framework* es un diseño abstracto orientado a objetos para un determinado tipo de aplicación, es un patrón arquitectónico que proporciona una plantilla extensible para un tipo específico de aplicaciones. Según (Larman, 2004) un *framework* o "esquema" es un subsistema expandible de un conjunto de servicios, es un conjunto cohesivo de interfaces y clases que colaboran para proporcionar los servicios de la parte central e invariable de un subsistema lógico.

Un Marco de Trabajo del inglés Framework, según (A. Guerrero, y otros, 2014), se define como «un conjunto de componentes físicos y lógicos estructurados de tal forma que permiten ser reutilizados en el diseño y desarrollo de nuevos sistemas de información». Aspectos como la seguridad, la gestión del personal, los reportes, entre otros, son contenidos entre las funcionalidades de la solución. Por lo que permite la obtención de soluciones con un alto grado de robustez, dadas por el número de funcionalidades brindadas durante el desarrollo de soluciones web.

Teniendo en cuenta lo anteriormente abordado por los autores, como marcos de trabajo seleccionados por el equipo de desarrollo se encuentran *Symfony2, jQueryUI y Bootstrap*.

Symfony2 v2.3

Symfony2, marco de trabajo implementado completamente con PHP 5, diseñado para optimizar gracias a sus características, el desarrollo de las aplicaciones web. Symfony2, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Suministra varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, que permiten al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. El resultado de todas estas ventajas es que no se debe reinventar la

rueda cada vez que se crea una nueva aplicación web, haciendo posible la reutilización de código. Ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel. Symfony2 es compatible con la mayoría de los gestores de bases de datos relacionales y no relacionales, como *MySQL, PostgreSQL, Oracle y SQL Server de Microsoft y MongoDB*. Se puede ejecutar tanto en plataformas Unix (Linux, etc) como en plataformas Windows. (Eguiluz, 2012)

JQueryUI versión v1.9

JQuery, es una biblioteca o marco de trabajo de JavaScript, es software libre y de código abierto. Posee un doble licenciamiento bajo las licencias MIT y GPL de GNU v2. Permite ser usada en proyectos libres y privativos. JQuery consiste en un único fichero JavaScript que contiene las funcionalidades comunes de DOM, eventos, efectos y AJAX.

Permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la tecnología AJAX a páginas web. Al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo (Alvarez, 2014).

Bootstrap v3.0.2

Bootstrap es un marco de trabajo desarrollado para facilitar el proceso de diseño de páginas web. Para ello ofrece una serie de plantillas CSS y de ficheros JavaScript que permiten conseguir interfaces web que funcionen en los navegadores actuales; un diseño que pueda ser visualizado de forma correcta en distintos dispositivos y a distintas escalas y resoluciones; una mejor integración con las librerías usadas habitualmente, como por ejemplo JQuery; un diseño sólido basado en herramientas actuales y potentes, o estándares como CSS3/HTML5 (Otto, y otros, 2012).

1.7 Lenguaje de Modelado

Al realizar un software debe establecerse una forma estándar de comunicación entre todos los involucrados al software. Los lenguajes de modelado permiten representar un software y describir las actividades a realizar para su desarrollo. En la actualidad existen dos lenguajes que son los más

populares y ampliamente utilizados, BPMN y UML. BPMN toma un enfoque centrado en los procesos y UML en general ofrece un enfoque orientado a objetos para modelar aplicaciones. Se toma como paradigma lo planteado por (Magaña, 2009) al referirse que «UML está dirigido a los arquitectos e ingenieros de software, como un medio para mejorar el proceso de desarrollo de software». Por lo que

en la investigación se utiliza para estandarizar la documentación el Lenguaje Unificado de Modelado (UML).

UML es un lenguaje de propósito general, se adapta a situaciones o necesidades específicas, permite a los usuarios extender o incluso modificar el lenguaje, además es utilizado por varias herramientas CASE como son: *Visual Paradigm, Rational Rose y Enterprise Architect*. UML posee gran documentación fundamentada y actualizada, además, el equipo de desarrollo posee amplias experiencias y habilidades en el trabajo con este lenguaje.

Por todas las bondades funcionales que brinda este lenguaje, el equipo de desarrollo decidió la utilización del mismo para obtener una especificación lo más fiable posible de los procesos y visualizar con claridad el diseño del sistema propuesto.

En aras de lograr un software flexible, de costo mínimo y robusto se definen las siguientes herramientas que conforman el ambiente de desarrollo del sistema:

- Entorno de desarrollo: *NetBeans*.
- Gestor de base datos: *PostgreSQL*.
- Herramienta de Modelado: Visual Paradigm.
- Marco de trabajo: *Symfony*.
- Validar las pruebas de rendimiento: *JMeter*

1.8 Herramientas para el desarrollo

1.8.1 Herramienta para el modelado de diagramas

Escoger una herramienta de modelado con UML, puede ser un factor importante para lograr la calidad de un proyecto. Es por ello que el equipo de desarrollo definió una serie de indicadores para la selección de la herramienta, siendo estos: la plataforma y tipo de software, modelado de datos y manejo de diagramas, apoyo metodológico y soporte completo del UML. Para ello se estudiaron las siguientes herramientas: *Visual Paradigm, Rational Rose y Enterprise Architect.*

Teniendo en cuenta estos indicadores y otros como los conocimientos, experiencias y habilidades del propio equipo de desarrollo, a pesar de la capacidad de *Rational Rose* y *Enterprise Architectect* para satisfacer las características requeridas por el equipo de desarrollo, el *Visual Paradigm* representa con mejor rendimiento y potencial, por lo que se decidió utilizar ***Visual Paradigm*** como herramienta para el modelado.

1.8.2 Entorno Integrado de Desarrollo (IDE por sus siglas en inglés)

Para la selección del IDE, se analizaron una serie de aspectos, siendo estos: multiplataforma, de código abierto y que se comercialicen bajo una licencia de software libre, además se tuvieron en cuenta otros indicadores como experiencia y las habilidades y capacidades del equipo de desarrollo en esta herramienta. Se analizaron los IDE *NetBeans, Eclipse, BlueJ, JBuilder* por ser muy utilizado en el desarrollo de aplicaciones.

El IDE seleccionado fue **NetBeans v8.0** por ser multiplataforma, de código abierto y que se comercializa bajo una licencia de software libre, cumpliendo esto con las políticas que se rige la Universidad, además el equipo de trabajo posee amplios conocimientos, experiencias y habilidades y capacidades en esta herramienta. Otro factor fundamental para la decisión es el tiempo limitado que se dispone para el desarrollo del sistema lo que dificulta la posibilidad de seleccionar un IDE al cual se desconoce por completo.

1.8.3 Sistema gestor de bases de datos (SGBD)

Un sistema gestor de bases de datos o SGBD es el software que permite a los usuarios procesar, describir, administrar y recuperar los datos almacenados en una base de datos (Cabello, 2010).

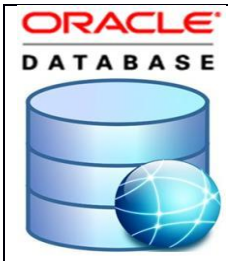
Existen SGBD como son *PostgreSQL, MySql y Oracle*, los dos primeros, los más utilizados en la universidad de acuerdo a los estándares de organización. La selección del SGBD se centró en la escalabilidad, identidad referencial y transacciones que estos poseen. El resultado de estos indicadores, así como los indicadores generales, arrojaron la necesidad de utilizar **PostgreSQL**.

PostgreSQL según (López, 2013) es un sistema de gestión de bases de datos objeto-relacional (ORDBMS) basado en el proyecto POSTGRES, de la universidad de Berkeley. PostgreSQL es un sistema objeto - relacional, ya que incluye características de la orientación a objetos, como puede ser la herencia y los tipos de datos, además de características relacionales como: funciones, restricciones, disparadores, reglas e integridad transaccional. Además de todas estas características técnicas, el equipo de desarrollo se destaca en el conocimiento y la experiencia en esta herramienta.
Para una mejor comprensión de la decisión tomada por el equipo de desarrollo, a continuación se describe en la siguiente tabla las características, ventajas, desventajas y requerimientos de algunos SGBD (Lozano, 2016):

Tabla 2. Descripción de las características de algunos SGBD. Fuente: Elaboración propia.

SGBD	Características	Ventajas	Desventajas	Requerimientos

PostgreSQL	✓ Sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. ✓ Sistema de gestión de bases de datos de código abierto más potente del mercado.	✓ Funciona con grandes cantidades de datos. ✓ Alta concurrencia con varios usuarios accediendo al mismo tiempo. ✓ Ahorro de costos de operación. ✓ Su sintaxis SQL es estándar y fácil de aprender.	✓ Es fácil de vulnerar sin protección adecuada. ✓ El motor *MyISAM* es instalado por defecto y carece de capacidades de integridad relacional. ✓ Reducida cantidad de tipos de datos.	✓ Sistema Operativo: Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64, Windows). ✓ Requerimientos de sistema: Memoria: 256 MB Espacio disponible en disco: 250 MB (mínimo).
SQL Server MySQL	✓ Sistema de gestión de bases de datos relacionales (SGBD) basado en el lenguaje *Transact-SQL*, y específicamente en *Sybase* IQ, fabricado por Microsoft capaz de poner a disposición de muchos usuarios grandes cantidades de datos de manera simultánea.	✓ Soporte de transacciones. ✓ Incluye también un potente entorno gráfico de administración, que permite el uso de comandos DDL y DML gráficamente. ✓ Permite trabajar en modo cliente-servidor, donde la información y datos se alojan en el servidor y las terminales o clientes de la red sólo acceden a la información.	✓ Microsoft SQL *SERVER* es la enorme cantidad de memoria RAM que utiliza para la instalación y utilización del software. ✓ Solo permite alojar un máximo de 64 GB. ✓ Requiere de un sistema operativo de Windows.	✓ El programa de instalación de SQL Server bloqueará las instalaciones en unidades de disco de solo lectura, asignadas o comprimidas. ✓ Se necesita 512 MB en RAM y 525 Mb en Disco Duro, además de tener instalada la versión 2.0 del Framework de .NET
MySQL	✓ Sistema de gestión de bases de datos relacional. ✓ Su diseño multi-hilos le permite soportar una gran carga de forma muy eficiente. ✓ También tiene un Amplio subconjunto del lenguaje SQL.	✓ Conectividad segura. ✓ Disponibilidad en gran cantidad de plataformas y sistemas. ✓ Probado con un amplio rango de compiladores diferentes. ✓ Puede mezclar tablas de distintas bases de datos en la misma consulta.	✓ La principal desventaja de SQL es la gran cantidad de memoria RAM que utiliza para la instalación. ✓ En gran porcentaje de las utilidades de MySQL no están documentadas.	✓ En principio, para un uso normal, como mínimo un AMD a 400 MHz, y mucha memoria RAM, por ejemplo 512 Mb para empezar.

	✓ Sistema gestor de datos relacional de última generación, lo cual quiere decir que está orientado al acceso remoto y redes (internet). ✓ Hoy por hoy Oracle se puede implementar en diferentes plataformas. ✓ Es una herramienta de administración grafica que es mucho más intuitiva y cómoda de maneja.	✓ Es Multiplataforma Soporta bases de datos de todos los tamaños, desde severas cantidades de bytes y gigabytes en tamaño. ✓ Soporta Cliente servidor. ✓ Integración perfecta con PHP.	✓ El mayor inconveniente de Oracle es quizás su precio. ✓ Incluso las licencias de Personal Oracle son Excesivamente caras. ✓ Otro problema es la necesidad de ajustes.	✓ PROCESADOR: Intel (x86), AMD64, e Intel EM64T Oracle proporciona versiones tanto para 32bits como para 64bits. ✓ RAM: 1 GB Ram (Recomendado 2GB en Windows 7 y Windows 2008 Server). ✓ DISCO DURO: 5.39 GB para la instalación típica y 5.89GB para la instalación avanzada.

1.8.4 Herramienta para la gestión de SGBD

PgAdmin es una aplicación gráfica para gestionar el gestor de bases de datos PostgreSQL, seleccionado por el equipo de desarrollo, siendo la más completa y popular con licencia *Open Source*. Está escrita en C++ usando la librería gráfica multiplataforma *wxWidgets,* lo que permite que se pueda usar en Linux, *FreeBSD*, Solaris, Mac OS X y *Windows*, por lo que la dependencia de plataforma no sería inconveniente para el desarrollo de la solución propuesta. Es capaz de gestionar versiones a partir de PostgreSQL 7.3 ejecutándose en cualquier plataforma, así como versiones comerciales de PostgreSQL como *Pervasive Postgres, EnterpriseDB, Mammoth Replicator* y *SRA PowerGres*.

Está diseñado para responder a las necesidades de todos los usuarios, desde escribir consultas SQL simples hasta desarrollar bases de datos complejas. El interfaz gráfico soporta todas las características de PostgreSQL y facilita enormemente la administración.

1.8.5 Herramienta de pruebas

Teniendo en cuenta realizar las pruebas de rendimiento al sistema, el equipo de desarrollo decide utilizar la herramienta **JMeter** (versión 2.13), la misma es una aplicación Java diseñada para cargar cliente de prueba / software de servidor (por ejemplo, una aplicación web). Se puede utilizar para probar el rendimiento tanto en los recursos estáticos y dinámicos como los archivos estáticos, *servlets Java*, *ASP.NET, PHP, scripts CGI*, objetos Java, bases de datos, servidores FTP y mucho más. *JMeter* puede ser utilizado para simular una carga pesada en el servidor, red u objeto, poner a prueba su fuerza o

para analizar el rendimiento general bajo diferentes tipos de carga. Además, puede ayudar a la regresión probar la aplicación, ya que permite crear scripts de prueba con afirmaciones para validar que su aplicación está devolviendo los resultados que espera. Para una máxima flexibilidad, JMeter le permite crear estas afirmaciones utilizando expresiones regulares (JMeter, 2013).

1.8.6 Herramientas de Mapeo Objeto-Relacional (ORM)

El mapeo de objeto-relacional (ORM), consiste en la generación de forma automática de las consultas a la base de datos para convertir los registros en objetos y viceversa. Permite a los programadores avanzar con mayor rapidez debido a que se concentran en codificar la lógica del negocio y no en hacer consultas (Rosales Morales, y otros, 2013).

Las herramientas de mapeo objeto relacional se encargan no solo de manejar y plantear una solución a las diferencias expuestas, sino que además buscan reducir susceptiblemente el código necesario para llevar a cabo las operaciones de persistencia y recuperación de objetos. Proporcionan además interfaces más simples para el manejo de objetos a través de su propio lenguaje de consulta, y proveen al programador de configuraciones que le permiten optimizar los tiempos de respuesta en sus correspondientes aplicaciones.

Unas de las principales ventajas que ofrece la utilización de herramientas ORM son la facilidad y velocidad de uso, así como la rapidez en el desarrollo. La mayoría de estas herramientas permiten la creación de un modelo libre de errores mediante los esquemas de tablas y relaciones. Si en futuros años se decide cambiar el motor de bases de datos, el modelo ORM garantiza que el cambio será más sencillo y no afectará al sistema

Entre las principales herramientas de ORM para el lenguaje PHP se encuentra *Doctrine, Propel, Hibernate e Ibattis*. A continuación, se detalla *Doctrine* que es el utilizado en la investigación, debido a que el equipo de desarrollo tiene amplio conocimiento de esta herramienta, se integra fácilmente al marco de trabajo *Symfony* y el tiempo de desarrollo es limitado, lo cual impide el estudio de otras herramientas.

Doctrine

Framework ORM para PHP en su versión 5.2, entre sus puntos fuertes se destaca su lenguaje DQL (*Doctrine Query Language*) que brinda la posibilidad de escribir consultas de base de datos. Este framework, para crear el modelo, nos da la alternativa de hacer una clase por tabla donde el programador puede especificar, mediante PHP, el tipo de datos almacenados, así como especificar relaciones y añadir funcionalidades extras a las clases autogeneradas. Una de las principales características es el bajo nivel de configuración que se necesita para empezar un proyecto.

1.9 Metodologías de desarrollo de software

De acuerdo al criterio de (Tinoco Gómez, y otros, 2010), definen una metodología de desarrollo como una colección de procedimientos, técnicas, herramientas y documentos auxiliares que ayudan a los desarrolladores de software en sus esfuerzos por implementar nuevos sistemas de información. En las dos últimas décadas, respecto a estas metodologías de desarrollo de software se ha entablado un intenso debate entre dos grandes corrientes. Por un lado, las denominadas metodologías tradicionales, centradas en el control del proceso, con un riguroso seguimiento de las actividades involucradas en ellas. Por otro lado, las metodologías ágiles, centradas en el factor humano, en la colaboración y participación del cliente en el proceso de desarrollo y a un incesante incremento de software con iteraciones muy cortas.

En la selección del enfoque se identificaron una serie de indicadores, los cuales son: capacidades y habilidades de desarrollo del equipo de trabajo, la comunicación entre los miembros del equipo, el trabajo en equipo, el compromiso del cliente, la disposición del equipo ante los cambios durante el proyecto, el tiempo de desarrollo, el tamaño del equipo y los recursos materiales disponibles. En el presente trabajo se utiliza un enfoque ágil teniendo en cuenta que responde a necesidades y condiciones reales del equipo de desarrollo, entre las que se encuentran: el cliente está altamente comprometido y motivado con el desarrollo del sistema, el equipo de desarrollo es pequeño y el tiempo que se dispone para el desarrollo es limitado.

1.9.1 Programación extrema

La metodología Programación Extrema (por sus siglas en inglés XP) guía el desarrollo de software haciendo énfasis en las relaciones interpersonales, fomentando el trabajo en equipo y la estrecha comunicación con el cliente. El producto de software se construye teniendo en cuenta que la solución más simple es la mejor. Está orientada a la adaptación paulatina de los requisitos y de sus cambios en cualquier punto de la vida del proyecto. Define historias de usuario para describir las funciones del sistema, las cuales son escritas por el cliente. El ciclo de desarrollo en XP es iterativo e incremental y la propia metodología está regida por varias fases (Beck, 2005), las cuales se muestran en la siguiente figura.

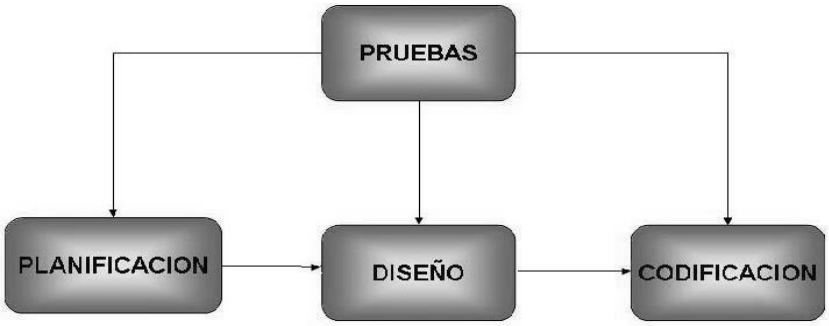


Figura 1. Fases de la metodología XP. Fuente: (Escribano, 2002).

Esta metodología aplica un conjunto de prácticas que hacen la entrega del componente menos complicada y más satisfactoria tanto para los clientes como para el equipo de entrega. Entre las prácticas más importantes se pueden mencionar la codificación por parejas posibilitando que el código sea discutido y revisado mientras se escribe. Pequeñas entregas del sistema en forma de versiones operativas, aunque no incluyan todas las funcionalidades. Diseño simple y dirección de las pruebas unitarias en la codificación. Además, permite la refactorización del código dirigida a simplificarlo para facilitar sus cambios en el futuro, presencia continua del cliente en la producción y utilización de estándares de programación, entre otras.

1.9.2 Scrum

Scrum es una metodología de desarrollo de software iterativa e incremental. Define un marco para la gestión de proyectos orientado a qué construir y en qué orden hacerlo; debido a ello se utiliza en otras prácticas de ingeniería de software tales como Proceso Racional Unificado (por sus siglas en inglés RUP) o XP. Es muy sencilla y responde a los principios de inspección constante y continua, adaptación del producto a las necesidades del cliente en tiempo real e innovación (Eugenia Bahit, 2011). Tiene como meta fomentar la efectividad y productividad de los equipos. Entre sus principales características destacan los ciclos de desarrollo llamados *sprints*[1] con una duración de 30 días, cuyo resultado es un incremento funcional del producto.

1.9.3 OpenUP

OpenUP es un proceso de desarrollo de software mínimamente suficiente, esto quiere decir que incluye solo el contenido fundamental, esto es que no provee orientación sobre temas en los que el proyecto tiene que lidiar, como son: el tamaño del equipo, el cumplimiento, seguridad, orientación tecnológica entre otras. Sin embargo, es completa en el sentido de que manifiesta por completo el proceso de construir un sistema. Para atender las necesidades que no están cubiertas en su contenido, OpenUp

[1] Término utilizado por la metodología para referirse a una iteración del proceso de desarrollo de software.

es extensible a ser utilizado como base sobre la cual se pueden añadir o adaptarse a contenido de otro proceso que sea necesario. Esta metodología permite el desarrollo incremental, el uso de casos de uso y escenarios, el manejo de riesgos y su diseño está basado en la arquitectura (Juarez, 2013).

Selección de la metodología de desarrollo de software

Dentro de las metodologías ágiles se analizaron XP (*eXtreme Programming*), *Scrum* y OpenUP (*Open Unified Process*), por su gran aceptación en el desarrollo del proyecto de software. Para ello se tomaron en cuenta una serie de indicadores: participación del cliente, tamaño del equipo de desarrollo, nivel de documentación que generan y nivel de flexibilidad para el equipo de trabajo. Teniendo en cuenta los factores antes referenciados y la necesidad de agilidad en los procesos para establecer una guía para el desarrollo del software que satisfaga dichas características, se decide seleccionar la metodología de desarrollo de software XP. Además, se presentan otras características las cuales son significativas en el análisis de la metodología para el desarrollo de la propuesta de solución:

- **Versiones pequeñas:** Las mini-versiones deben ser lo suficientemente pequeñas como para poder hacer una cada pocas semanas. Deben ser versiones que ofrezcan algo útil al usuario final y no trozos de código que no puedan ver funcionando.
- **Desarrollo guiado por las pruebas automáticas:** Se deben realizar programas de prueba automática y deben ejecutarse con mucha frecuencia. Cuantas más pruebas se hagan, más efectivo.
- **Integración continua:** Debe tenerse siempre un ejecutable del proyecto que funcione y en cuanto se tenga una nueva pequeña funcionalidad, debe recompilarse y probarse. Es un error mantener una versión congelada dos meses mientras se hacen mejoras y luego integrarlas todas de una vez. Cuando exista alguna falla, no podría establecerse dónde ha sido, sino seguimos la buena práctica de hacerlo en su momento.
- **El código es de todos:** Cualquiera puede y debe interactuar y conocer cualquier parte del código. Para eso se hacen las pruebas automáticas.
- **Normas de codificación:** Debe haber un estilo común de codificación (no importa cuál), de forma que parezca que ha sido realizado por una única persona.

Además, XP se centra también en la generación de una documentación, aunque no exhaustiva, suficiente para soportar la solución, teniendo en cuenta la necesidad de resultados tangibles a corto plazo.

1.10 Conclusiones parciales

- El análisis de los sistemas existentes basados en la automatización de la estrategia de superación pedagógica, evidenció que carecen de mecanismos para darle solución al problema de la presente investigación, pues en su mayoría se limitan al dominio específico en el que se desenvuelve, carecen de escalabilidad y no brindan una Estrategia de Superación Pedagógica al profesor.
- Las características del sistema a construir en la investigación, determinaron el uso de las siguientes herramientas y tecnologías: PHP como lenguaje de programación; Symfony como marco de trabajo; PostgreSQL como gestor de base de datos, NetBeans como entorno desarrollo y JMeter para realizar las pruebas de carga y estrés. Además, UML como lenguaje de modelado; Visual Paradigm como herramienta CASE y XP como metodología de desarrollo.

CAPÍTULO 2 CARACTERÍSTICAS DE LA PROPUESTA DE SOLUCIÓN

2. 1 Introducción

En el presente capítulo se describe la propuesta de solución trazada y se exponen los artefactos generados propios de la metodología de desarrollo utilizada para la implementación de la solución que se propone tales como: las funcionalidades con las que debe contar la aplicación, historias de usuario, tareas de ingeniería y tarjetas CRC (Contenido, Responsabilidad y Colaboración). Además, también se describe la arquitectura del sistema, los patrones de diseño que fueron utilizados y el modelo de datos que fue generado para la propuesta de solución.

2. 2 Caracterización del sistema actual

Para la autora (Radrigan, 2005), el método empírico es un modelo de investigación científica, que se basa en la experimentación y la lógica empírica, que, junto a la observación de fenómenos y su análisis estadístico, es el más usado en el campo de las ciencias sociales y en las ciencias naturales. Su aporte al proceso de investigación es resultado fundamentalmente de la experiencia. Estos métodos posibilitan revelar las relaciones esenciales y las características fundamentales del objeto de estudio.

Para lograr una caracterización de forma directa de la situación problemática y el problema científico, se efectuó el análisis documental y la observación, para obtener la información objetiva del proceso de gestión de estrategias de superación pedagógica en el Centro de Innovación y Calidad de la Educación. Además, se realizaron entrevistas a los profesores, conformada por 9 preguntas, para comprobar el conocimiento previo que se tiene del proceso (Ver Anexo 2).

La entrevista arrojó lo siguiente:

- El SASPED está compuesta por 3 módulos, el **Módulo de Administración** que se encarga de implementar medidas de seguridad para la base de datos a través de contraseñas y niveles de acceso para distintos grupos de usuario distinguiendo los roles por cada uno de ellos. El **Módulo de Gestión de información** que se encarga del control de los datos de los profesores (datos personales e información sobre su currículo) y de las actividades docentes que realizan vinculadas al desempeño de su trabajo y a la estrategia de superación pedagógica y el **Módulo de Adquisición del conocimiento** que se encarga de recoger toda la información relacionada a la estrategia de superación trazada a un profesor relacionada con la evaluación profesoral, los controles a clase y su autoevaluación. Esta estructura se puede reutilizar, ya que centraliza de manera adecuada los procesos de la gestión de información para realizar la caracterización de la superación pedagógica individual. Dentro de este módulo se encuentra el sistema experto, el cual cuenta con una base de conocimiento. Esta base de conocimiento está conformada por

un conjunto de casos validados por expertos, estos casos permiten según las características de un profesor proponerle una estrategia de superación pedagógica.

- El Sistema Automatizado para la Superación Pedagógica (SASPED), tiene como objetivo proponer una estrategia de superación pedagógica individual a cada docente. La estrategia se brinda a partir de 34 características de cada docente, las cuales se agrupan en 5 categorías. De las cuales, a partir de esta propone 4 tipos de programas (Programa acelerado, Programa promedio, Programa especial, Programa completo), donde cada uno está asociado a un conjunto de cursos, entrenamientos y actividades de superación.
- El SASPED se encarga de gestionar las 5 categorías antes mencionadas, además de gestionar los grupos de trabajo de formación pedagógica propuesto por la Estrategia de Superación de la UCI.
- El sistema fue desarrollado con el *framework* Seam 2.1 el cual se encuentra obsoleto, ya que actualmente su evolución ha llegado a la versión 3.1. La empresa que soportaba este *framework* ha decidido detener su evolución y es muy costoso actualmente migrar un sistema desarrollado en versiones antiguas a versiones más actualizadas.
- El desarrollo de nuevas funcionalidades en el SASPED está limitado por la tecnología obsoleta, y el proceso de gestión no cumple con las exigencias del proceso de superación pedagógica en la UCI. Por lo anterior se constata la existencia del problema científico y se corrobora el objetivo propuesto.

Teniendo como premisa los resultados obtenidos de la aplicación de los métodos empíricos y para llevar a cabo la estrategia de superación de cada docente, se ha concebido un Sistema para la Automatización de los Procesos de la Estrategia de Superación Pedagógica, con el objetivo de, según (Bravo, 2013), «perfeccionar el proceso de superación pedagógica del claustro en la UCI e integrarlo con el resto de las formas existentes actualmente de superación profesional y formación académica en las Ciencias Pedagógicas y de la Educación, para contribuir a la elevación de la cultura profesional pedagógica de los actores que intervienen en la formación del Ingeniero en Ciencias Informáticas». Además de cumplir las exigencias de la superación pedagógica en la UCI y elevar la escalabilidad del sistema propuesto.

2. 3 Propuesta de solución

El Modelo de dominio es un artefacto que incorpora comportamientos y datos. Este modelo es la representación visual de las clases conceptuales u objetos del mundo real que reportan interés para la investigación o proyecto. Puede ser empleado en casos donde el negocio no comprende actividades

demasiado cambiantes (Larman, 2004). En su construcción se modela un diagrama de clases compuesto por objetos del dominio o clases conceptuales, relaciones entre estas y los atributos que las componen.

En la presente investigación, el modelo de dominio expresa los conceptos fundamentales del dominio de la investigación y la relación que existe entre cada uno de estos, de forma tal que facilite el entendimiento y contribuya a la solución del problema, el cual va encaminado a trazar estrategias de superación profesoral a cada profesor del claustro de la UCI y que ha sido caracterizado en el epígrafe anterior. Para un mayor conocimiento quedan plasmados los conceptos fundamentales y una representación de sus relaciones en la Figura 2. (Ver Anexo 3)

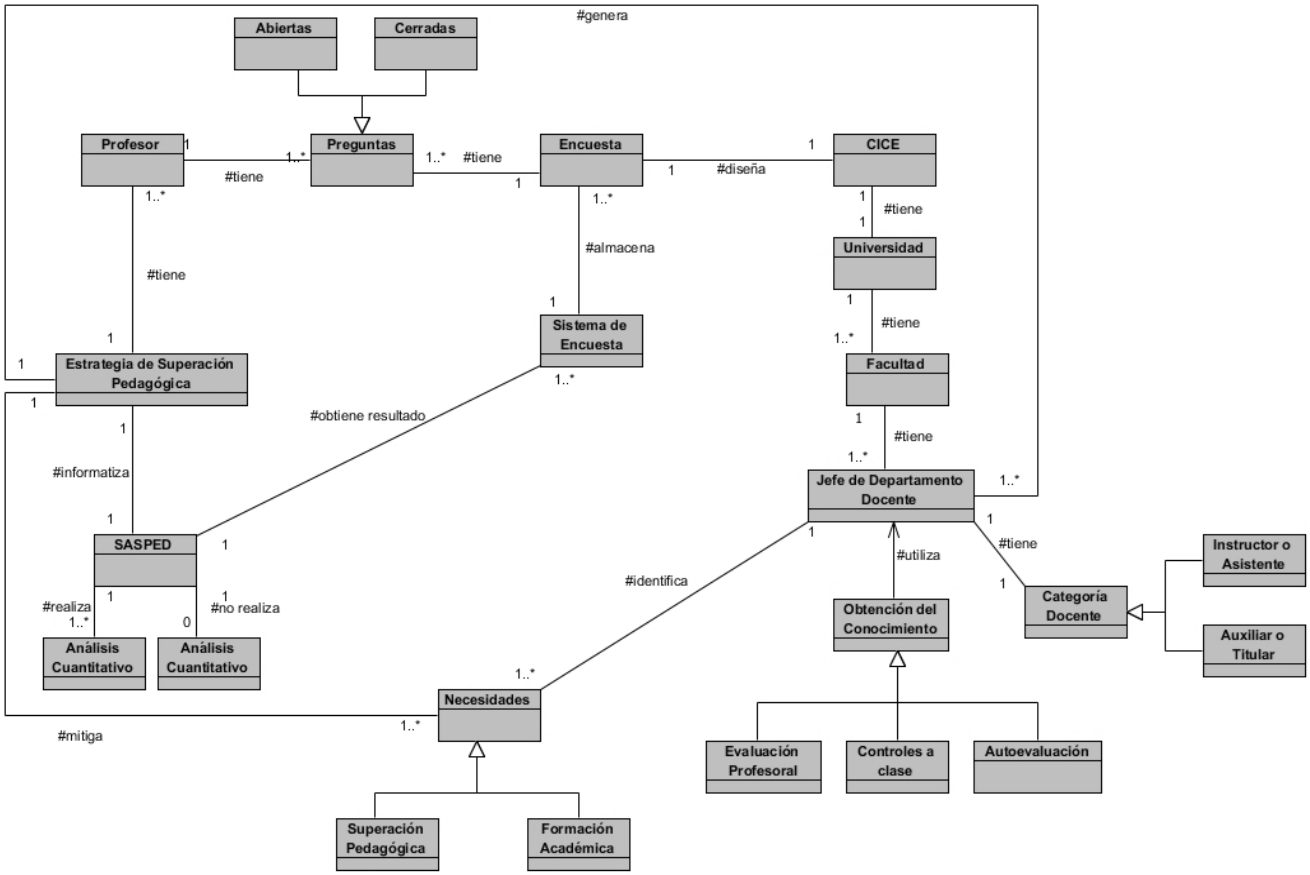


Figura 2. Modelo de dominio. Fuente: Elaboración propia.

2. 4 Funcionalidades del sistema

Las funcionalidades de un sistema definen una función del sistema de software o sus componentes. Una función es descrita como un conjunto de entradas, comportamientos y salidas. Estas funcionalidades pueden ser: cálculos, detalles técnicos, manipulación de datos y otras funcionalidades específicas que se supone, un sistema debe cumplir. Constituyen declaraciones funcionales de los

servicios que proveerá el sistema, de la manera en que éste reaccionará a entradas particulares (Sommerville, 2005). A continuación, se enuncian las funcionalidades que fueron definidas para llevar a cabo el desarrollo de la solución propuesta en aras de detallar aún más la documentación de la investigación.

Módulo administración

1. Gestionar categoría docente.
2. Gestionar categoría científica.
3. Gestionar áreas.
4. Gestionar cargos.
5. Gestionar tipos de clase.
6. Gestionar período.
7. Gestionar grupo.
8. Gestionar profesores.
9. Gestionar organizaciones.
10. Configurar sistema.
11. Respaldar sistema.
12. Cerrar sesión por inactividad.
13. Consumir estructura.
14. Autenticar usuario.
15. Filtrar profesores.
16. Exportar historial del profesor.
17. Asignar rol.

Módulo gestión de información

18. Crear estrategia de SP
19. Gestionar cursos de superación.
20. Gestionar grupo de trabajo de formación pedagógica.
21. Gestionar control.
22. Gestionar evaluación profesoral.
23. Exportar registro de control a clase.
24. Exportar registro de evaluación profesoral.
25. Exportar información del grupo de trabajo de formación pedagógica.
26. Exportar registro de los cursos de superación.

Módulo adquisición del conocimiento

27. Gestionar casos.
28. Atender peticiones
29. Gestionar programas.
30. Gestionar categorías.
31. Gestionar características.
32. Exportar información del programa.

.

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. En muchos casos, estos requerimientos son fundamentales en el éxito del producto y generalmente están vinculados a requerimientos funcionales, además corresponden a aspectos tales como la disponibilidad, soporte, seguridad, apariencia e interfaz externa. A continuación, se presenta el conjunto de requisitos no funcionales del sistema.

Usabilidad

- El sistema debe garantizar un acceso rápido y podrá ser utilizado por usuarios con conocimientos informáticos básicos.
- EL sistema debe garantizar el acceso a los usuarios definidos.
- Los usuarios definidos deben tener acceso solamente a las funcionalidades definidas para ellos.

Rendimiento

- El sistema deberá ser rápido ante las solicitudes de los usuarios en el procesamiento de la información, el tiempo de respuesta no debe ser superior a los cinco segundos.
- El sistema debe ser estable y confiable.

Soporte

- Una vez terminado el sistema se realizarán procesos de despliegue y capacitación.

Portabilidad

- El sistema debe ser multiplataforma.

Diseño e implementación

- Lenguaje de programación: PHP 5.5
- IDE: NetBeans 8.0
- Para el Modelado de UML se utiliza: Visual Paradigm 8.0
- Servidor web: Apache 2.4
- Gestor de base de datos: PostgreSQL 9.1
- Herramienta de pruebas: Apache JMeter 2.13

Software

- El sistema debe ejecutarse en el sistema operativo GNU/Linux preferentemente Nova o Windows XP o superior.
- El sistema gestor de bases de datos será PostgreSQL 9.4.

- Navegador Web: Mozilla Firefox 28.0 o versiones superiores, Chrome 33.0 o versiones superiores.

Hardware

- Las estaciones de trabajo cliente deben tener como mínimo 128 Mb de RAM y procesador Pentium IV o superior.
- El servidor para la instalación del Sistema para la automatización de los procesos de la estrategia de superación pedagógica debe tener como mínimo 1 Gb de RAM, 80 GB de disco duro disponible para el almacenamiento de la Base de datos y procesador superior 2.40 GHZ.

Seguridad

- **Confidencialidad:**
  - El sistema debe garantizar que la información solo podrá ser accedida por los usuarios definidos teniendo en cuenta los privilegios del rol que desempeñen.
- **Integridad:**
  - La información podrá ser modificada solo por el personal autorizado.
  - La información manejada por el sistema será objeto de cuidadosa protección contra la corrupción y estados inconsistentes, mediante la validación de los datos de entrada.
  - Se harán validaciones de la información tanto en el cliente como en el servidor contra ataques de inyección HTML o SQL.
- **Disponibilidad:**
  - El subsistema deberá estar disponible para los usuarios que acceden al mismo aproximadamente un 98% del tiempo, acotado a un año. El resto del tiempo se puede realizar el mantenimiento de hardware o software, siempre en el horario no laborable.

2. 5 Fase 1: Exploración.

En esta fase, se fundamentan las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo se prueba la tecnología propuesta y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. El ciclo de vida de XP enfatiza en el carácter interactivo e incremental del desarrollo. Una iteración de desarrollo es un período de tiempo en el que se realiza un conjunto de funcionalidades determinadas que, en el caso de XP, se corresponden con un conjunto de historias de usuarios.

Historias de Usuario

Las historias de usuario (HU) constituyen la técnica utilizada en XP para especificar los requisitos del

software. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las historias de usuario es muy dinámico y flexible, en cualquier momento historias de usuario pueden romperse, reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas (Letelier, y otros, 2006).

Las HU conducen al proceso de creación de las pruebas de aceptación, las cuales servirán para verificar que estas historias se han implementado correctamente. Otra de las características es que solamente proporcionan los detalles sobre la estimación del riesgo y cuánto tiempo conllevará su implementación. Como resultado del análisis realizado se identificaron un total de 87 historias de usuario (Ver Anexo 4), a continuación, se presentan algunas de las mismas.

Tabla 3. HU29: Registrar profesor. Fuente: Elaboración propia

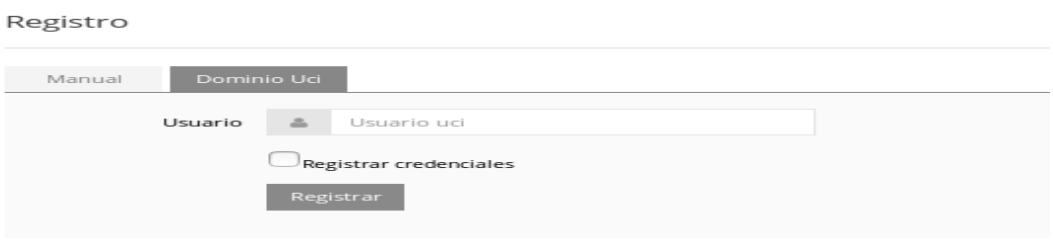
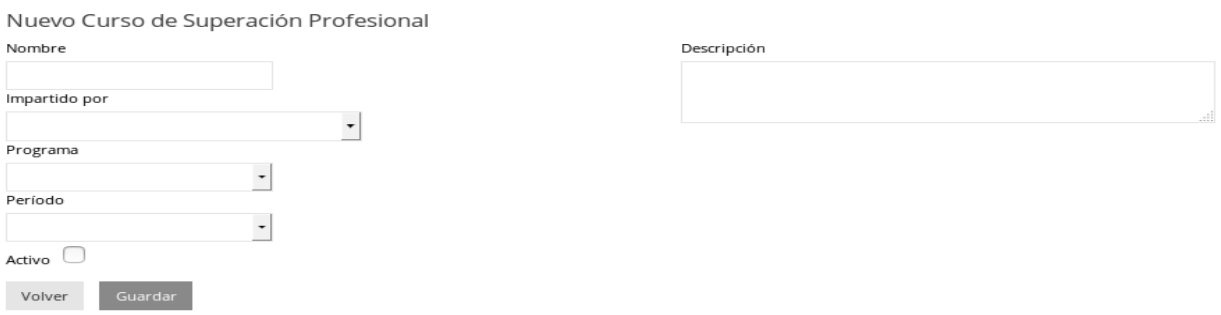
Historias de Usuario	
Número: HU 29	**Nombre del requisito:** Registrar profesor
Programador: Jorge Javier Tamayo Muñiz	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 2 días.
Riesgo en Desarrollo: Alto	**Tiempo Real:** 48 horas.
Descripción: El sistema debe permitir al usuario, en este caso al Administrador, registrar un nuevo profesor dependiendo si es un usuario del dominio UCI o no. En caso que este sea un usuario que no se encuentra en el dominio los datos deben introducirse manualmente introduciendo de forma correcta la información necesaria sobre el mismo y asignándole el rol correspondiente para acceder a los módulos requeridos. En caso que el usuario sea del dominio, debe introducirse el nombre de usuario del mismo donde se valide en el campo que el usuario introducido es correcto en el sistema. Luego, el sistema debe mostrar un mensaje de confirmación después de haber ejecutado cualquier acción sobre el formulario de registro y notificar en caso de errores a través de un mensaje de error.	
Observaciones: El usuario Administrador deberá estar previamente autenticado.	
Prototipo de interfaz: 	

Tabla 4. HU46: Registrar curso de superación. Fuente: Elaboración propia

Historias de Usuario	
Número: HU 46	**Nombre del requisito:** Registrar curso de superación
Programador: Jorge Javier Tamayo Muñiz	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 3 días.
Riesgo en Desarrollo: Alto	**Tiempo Real:** 72 horas.
Descripción: El sistema debe permitir al jefe de departamento o al usuario con los permisos correspondientes a la funcionalidad, registrar un nuevo curso de superación. Al acceder al módulo Gestión de información y luego al requisito Cursos de superación. Al introducir los datos necesarios en los formularios y dar clic en el botón guardar el sistema debe mostrar un mensaje de confirmación y notificar en caso de errores a través de un mensaje de error.	
Observaciones: El usuario debe estar autenticado previamente en la aplicación.	
Prototipo de interfaz: 	

Definición de los actores

Una de las premisas fundamentales durante el desarrollo de cualquier producto es determinar a quién está dirigido un sistema informático. Para esta aplicación se identificaron los siguientes roles de usuarios: Administrador, Profesor, Jefe de Departamento, Directivo y Experto. A continuación, se muestra una tabla con las descripciones de cada uno de los roles.

Tabla 5. Roles de usuario. Fuente: Elaboración propia.

Rol	Objetivo
Administrador	Encargado de gestionar toda la información de los usuarios registrados en el sistema.
Profesor	Usuario principal que interactúa con el sistema y realiza la encuesta para determinar sus necesidades de superación pedagógica.
Jefe de Departamento	Encargado de llevar a cabo la gestión de la superación pedagógica del profesor.
Directivo	Es la persona encargada de controlar el curso de la superación pedagógica de los profesores. Esta persona puede ser: Vicerrector(a) de Formación, Rector(a), Director(a) de Cuadro, Jefe de

	Departamento Docente Central, Director(a) del Departamento Docente Metodológico y Decano(a).
Experto	Persona encargada de gestionar los casos y atender las peticiones generadas por el sistema y tomar la decisión de Rechazarla o Aceptarla según la estrategia trazada al profesor.

2. 6 Fase 2: Planificación.

El propósito de esta fase es el de llegar a un acuerdo entre los clientes, los gerentes o coordinadores y los programadores en cuáles serán las historias a ser implementadas durante cada iteración. Si se hace una buena preparación durante la fase de exploración esta fase no suele llevar más de un día o dos. Es una fase corta, en la que el cliente, los gerentes y el grupo de desarrolladores acuerdan el orden en que deberán implementarse las historias de usuario y, asociadas a éstas, las entregas. Típicamente esta fase consiste en una o varias reuniones grupales de planificación. El resultado de esta fase es un Plan de Entregas (Valarezo Penadillo, y otros, 2013).

La planificación no debe ser estricta puesto que hay muchas variables en juego, el equipo de desarrollo considera que la misma debe ser flexible para poder adaptarse a los cambios que puedan surgir, donde una buena estrategia es hacer planificaciones detalladas para unas pocas semanas y planificaciones mucho más abiertas para unos pocos meses.

2.6.1 Estimación del esfuerzo por historia de usuario

Durante la fase de planificación se realiza una estimación del esfuerzo que costará implementar cada historia de usuario. Este se expresa utilizando como medida el punto. Un punto se considera como una semana ideal de trabajo donde los miembros de los equipos de desarrollo trabajan el tiempo planeado sin ningún tipo de interrupción (Beck, 2012).

El tiempo de implementación de una historia de usuario generalmente es de uno a tres puntos. Por otra parte, el equipo de desarrollo mantiene un registro de la velocidad de desarrollo, establecida en puntos por iteración, basándose principalmente en la suma de puntos correspondientes a las historias de usuario que fueron terminadas en la última iteración.

Para el desarrollo satisfactorio de la solución propuesta, se realizó una estimación de esfuerzo para cada una de las historias de usuario, arrojando los siguientes resultados:

Tabla 6. Plan de estimación de esfuerzo por historias de usuario. Fuente: Elaboración propia.

No	Historias de usuario	Puntos de estimación
1	Gestionar categoría docente.	**1**

2	Gestionar categoría científica.	**1**
3	Gestionar áreas.	**1**
4	Gestionar cargos.	**1**
5	Gestionar tipos de clase.	**1**
6	Gestionar período.	**1**
7	Gestionar grupo.	**1**
8	Gestionar profesores.	**2**
9	Gestionar organizaciones.	**1**
10	Configurar sistema.	**1**
11	Respaldar sistema.	**1**
12	Cerrar sesión por inactividad.	**1**
13	Consumir estructura.	**2**
14	Autenticar usuario.	**1**
15	Filtrar profesores.	**1**
16	Exportar historial del profesor.	**1**
17	Asignar rol.	**1**
18	Crear estrategia de SP	**1**
19	Gestionar cursos de superación.	**1**
20	Gestionar grupo de trabajo de formación pedagógica.	**1**
21	Gestionar control.	**1**
22	Gestionar evaluación profesoral.	**1**
23	Exportar registro de control a clase.	**1**
24	Exportar registro de evaluación profesoral.	**1**
25	Exportar información del grupo de trabajo de formación pedagógica.	**1**
26	Exportar registro de los cursos de superación.	**1**
27	Gestionar casos.	**2**
28	Atender peticiones	**1**
29	Gestionar programas.	**1**
30	Gestionar categorías.	**1**
31	Gestionar características.	**1**
32	Exportar información del programa.	**1**
Total		**35**

2.6.2 Plan de Entrega

El objetivo de este plan es producir rápidamente versiones de la aplicación que sean operativas, aunque estas no cuenten con toda la funcionalidad pretendida, pero sí que constituyan un resultado de valor para el negocio. El plan de entrega que se muestra a continuación propone el tiempo aproximado de entrega de versiones de la aplicación implementadas durante cada una de las iteraciones para las 88 HU generadas, teniendo en cuenta una duración total del proyecto de 35 semanas según el plan de entrega realizado.

Tabla 7. Plan de entrega. Fuente: Elaboración propia.

Entregable	Iteración	Fin de la iteración
Módulo de Administración.	1	Diciembre de 2015
Módulo de Gestión de información.	2	Marzo de 2016
Módulo de Adquisición del conocimiento.	3	Junio de 2016

2.6.3 Plan de iteraciones

Para confeccionar el plan de iteraciones, se establece cuántas iteraciones serán necesarias realizar sobre el sistema para su consecución. Sin embargo, es preciso establecer el contenido de trabajo para todas y cada una de ellas y es aquí donde se establece dicho plan, regulando la cantidad de historias de usuario a implementar dentro del rango establecido por la estimación efectuada. Teniendo en cuenta como referencia los aspectos antes tratados, el sistema que se pretende construir se desarrollará en 3 iteraciones, las mismas serán explicadas detalladamente a continuación:

Iteración 1

La iteración tiene como finalidad implementar las historias de usuario que se consideraron más necesarias atendiendo a su relevancia e impacto para el negocio. Se da respuesta a las funcionalidades pertenecientes al módulo Administración.

Iteración 2

En esta iteración se realizan todas las historias de usuarios relacionadas con el módulo Gestión de información. Además, se corregirán errores o inconformidades con las HU implementadas en la iteración anterior.

Iteración 3

En esta iteración se realizan las historias de usuario relacionadas con la Gestión de la información que se genera en el sistema. De esta manera se obtiene la versión 1.0 del producto final.

Plan de duración de las iteraciones

Para una mayor organización del trabajo como lo plantea el ciclo de vida de XP se crea un plan de duración de las iteraciones, en este caso se realizaría un solo plan ya que existe un único equipo de desarrolladores. Este plan se realiza con el objetivo de reflejar cuáles serán las historias de usuario que se implementarán en cada una de las iteraciones, así como el tiempo destinado a cada una de ellas y el orden en que se implementarán, lo que ayuda a obtener una idea general del tiempo que durará la confección total del sistema.

Tabla 8. Plan de duración de iteraciones. Fuente: Elaboración propia.

Iteraciones	No.HU	Historia de usuario	Duración
Iteración 1	HU_1	Registrar categoría docente	19 semanas
	HU_2	Listar categorías docentes	
	HU_3	Editar categoría docente	
	HU_4	Eliminar categoría docente	
	HU_5	Registrar categoría científica	
	HU_6	Listar categorías científicas	
	HU_7	Editar categoría científica	
	HU_8	Eliminar categoría científica	
	HU_9	Registrar área	
	HU_10	Listar áreas	
	HU_11	Editar área	
	HU_12	Eliminar área	
	HU_13	Registrar cargo	
	HU_14	Listar cargos	
	HU_15	Editar cargo	
	HU_16	Eliminar cargo	
	HU_17	Registrar tipo de clase	
	HU_18	Listar tipos de clase	
	HU_19	Editar tipo de clase	
	HU_20	Eliminar tipo de clase	
	HU_21	Registrar período	
	HU_22	Listar períodos	
	HU_23	Editar período	
	HU_24	Eliminar período	
	HU_25	Registrar grupo	
	HU_26	Listar grupos	
	HU_27	Editar grupo	
	HU_28	Eliminar grupo	
	HU_29	Registrar profesor	
	HU_30	Listar profesores	
	HU_31	Editar profesor	
	HU_32	Eliminar profesor	
	HU_33	Registrar organización	
	HU_34	Listar organizaciones	

	HU_35	Editar organización	
	HU_36	Eliminar organización	
	HU_37	Configurar sistema	
	HU_38	Respaldar sistema	
	HU_39	Cerrar sesión por inactividad	
	HU_40	Consumir estructura	
	HU_41	Autenticar usuario	
	HU_42	Filtrar profesores	
	HU_43	Exportar historial del profesor	
	HU_44	Asignar rol	
Iteración 2	HU_45	Crear estrategia de superación pedagógica	8 semanas
	HU_46	Registrar curso de superación	
	HU_47	Listar cursos de superación	
	HU_48	Editar curso de superación	
	HU_49	Eliminar curso de superación	
	HU_50	Mostrar curso de superación	
	HU_51	Crear grupo de trabajo de formación pedagógica	
	HU_52	Listar grupos de trabajo de formación pedagógica	
	HU_53	Editar grupo de trabajo de formación pedagógica	
	HU_54	Eliminar grupo de trabajo de formación pedagógica	
	HU_55	Registrar control	
	HU_56	Listar controles	
	HU_57	Editar control	
	HU_58	Eliminar control	
	HU_59	Mostrar control	
	HU_60	Registrar evaluación profesoral	
	HU_61	Listar evaluaciones profesorales	
	HU_62	Editar evaluación profesoral	
	HU_63	Eliminar evaluación profesoral	
	HU_64	Mostrar evaluación profesoral	
	HU_65	Exportar registro de control a clase	
	HU_66	Exportar registro de evaluación profesoral	
	HU_67	Exportar información del GTFP	
Iteración 3	HU_68	Registrar caso	8 semanas
	HU_69	Listar casos	
	HU_70	Editar caso	
	HU_71	Eliminar caso	
	HU_72	Mostrar caso	
	HU_73	Atender peticiones	
	HU_74	Registrar categoría	
	HU_75	Listar categorías	
	HU_76	Editar categoría	

	HU_77	Eliminar categoría	
	HU_78	Registrar programa	
	HU_79	Listar programas	
	HU_80	Editar programa	
	HU_81	Eliminar programa	
	HU_82	Exportar información del programa	
	HU_83	Registrar característica	
	HU_84	Listar características	
	HU_85	Editar característica	
	HU_86	Eliminar característica	
	HU_87	Mostrar característica	

2. 7 Fase 3: Diseño

El diseño es fundamental, a diferencia de otras metodologías se realiza durante todo el tiempo de vida del proyecto por lo que se establecen los mecanismos, para que este sea revisado y mejorado continuamente según se van añadiendo funcionalidades al mismo.

La metodología XP establece prácticas especializadas que inciden directamente en la realización del diseño para lograr un sistema robusto y reutilizable tratando de mantener su simplicidad, es decir, crear un diseño evolutivo que se va mejorando incrementalmente y que permite hacer entregas pequeñas y frecuentes de valor para el cliente. A la hora de darle cumplimiento a la actividad de diseñar, XP no especifica ninguna técnica de modelado. Pueden utilizarse indistintamente sencillos esquemas en una pizarra, diagramas de clases utilizando UML o tarjetas CRC (Clase, Responsabilidad y Colaboración) siempre que sean útiles, tributen a la comprensión y no requieran mucho tiempo en su creación.

2.7.1 Tarjetas CRC

Teniendo en cuenta el análisis para el diseño de las aplicaciones, la metodología XP no requiere la presentación del sistema mediante diagramas de clases utilizando notación UML, en su lugar se usan otras técnicas como las tarjetas CRC (Contenido, Responsabilidad y Colaboración). Por tanto, el paso siguiente en el análisis de requerimientos de una aplicación es la creación del modelo CRC.

Las tarjetas CRC identifican y organizan las clases bajo el paradigma orientado a objetos (lo que incluye asignación de responsabilidades), cada tarjeta contiene el nombre de la clase (que representa una o más historias de usuario), una descripción de las responsabilidades o métodos asociados con la clase, así como la lista de las clases con que se relaciona o que colaboran con ella. Las tarjetas CRC son el único trabajo de diseño que se genera como parte del proceso de XP (Holmes, y otros, 2009).

Clases: tienen diferentes categorías:

- Clases de entidad: llamadas clases de modelo o negocios, se extraen de manera directa del enunciado del problema.
- Clases de frontera: se utilizan para crear la interfaz que el usuario ve y con la cual interactúa cuando se utiliza el software.
- Clases de controlador: manejan una “unidad de trabajo” desde el inicio hasta el final.
- Responsabilidad: son los atributos y las operaciones relevantes para la clase.
- Colaboradores: son aquellas clases que se requieren para que una clase reciba la información necesaria para completar una responsabilidad.

A continuación, se muestran las tarjetas CRC de las clases de mayor prioridad para el cliente (Ver Anexo 5).

Tabla 9. Tarjeta CRC_19. Clase Persona. Fuente: Elaboración propia.

Clase Persona	
Responsabilidades	**Colaboradores**
Permite devolver el valor de cualquier atributo de una persona. **Permite cambiarle el valor inicial de uno o varios atributos.** **Permite devolver el nombre completo de la persona.**	

Tabla 10. Tarjeta CRC_20. Clase PersonaController. Fuente: Elaboración propia.

Clase PersonaController	
Responsabilidades	**Colaboradores**
Adicionar una persona. **Eliminar una persona.** **Modificar los datos de una persona.** **Asignar rol a la persona.**	**Persona**

2.7.2 Diseño de base de datos

El diseño de la base de datos es la herramienta conceptual para la descripción de los datos, relaciones entre ellos, semántica y restricciones de consistencia. Tal representación permite contar con una vista del modelo a implementar a través de la perspectiva de entidades.

Modelo de datos

Un modelo de datos es la representación abstracta de los datos en un sistema gestor de base de datos.

Básicamente el modelo de datos está formado por tres elementos fundamentales que son:

- ✓ Objetos (entidades que existen y se manipulan).
- ✓ Atributos (Características básicas de estos objetos).
- ✓ Relaciones (forma en que se enlazan los distintos objetos entre sí).

El modelo entidad-relación (E-R) es un modelo de datos de alto nivel. Está basado en una percepción del mundo real que consiste en una colección de objetos básicos, denominados entidades, y de relaciones entre estos objetos. En la siguiente figura se ilustra el modelo de datos del Sistema para la automatización de los procesos de la estrategia de superación pedagógica de la UCI, en el Anexo 6 está disponible el MER que complementa la representación abstracta del modelo de datos mostrado en la Figura 3.

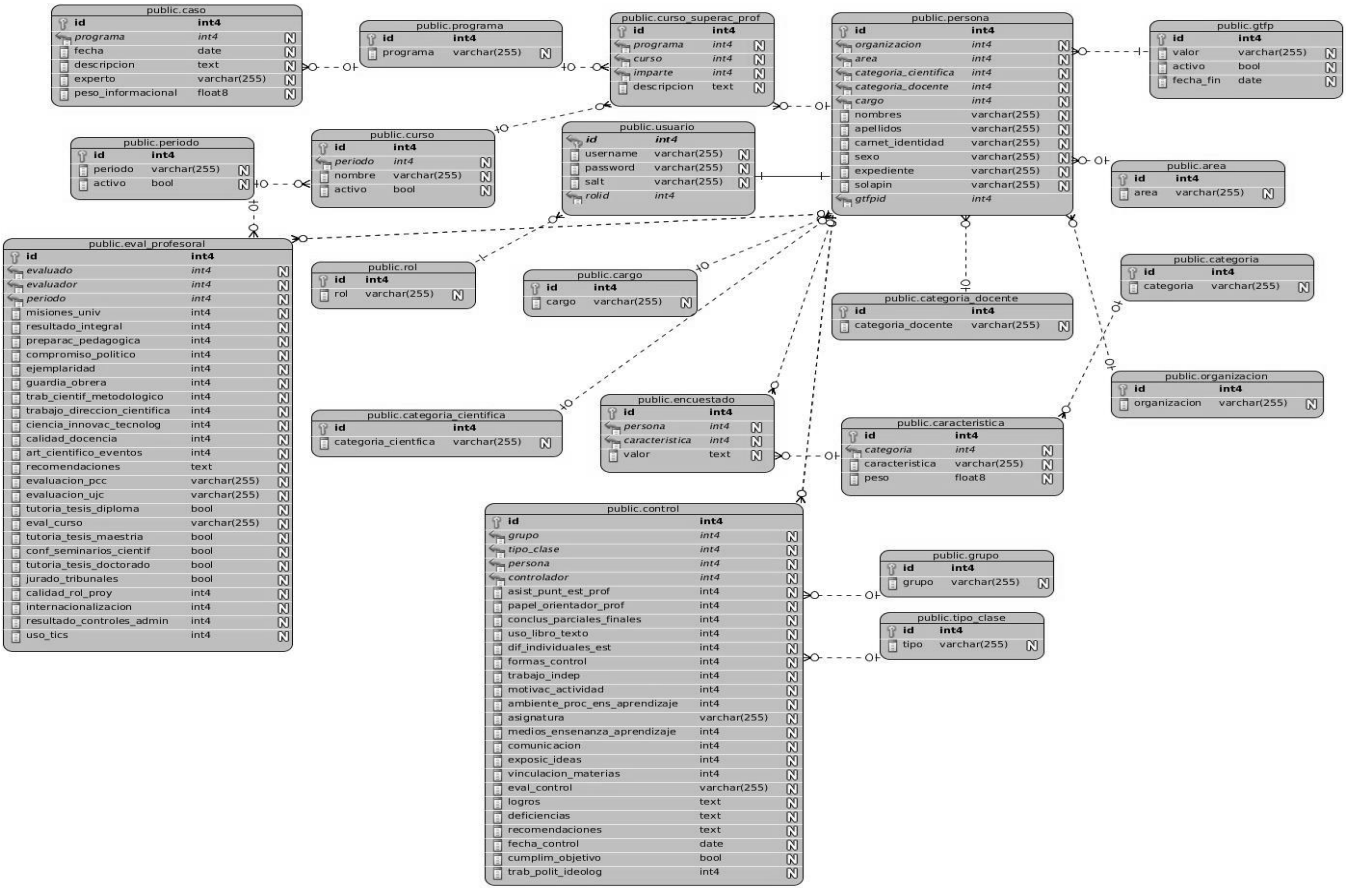


Figura 3. Modelo de datos. Fuente: Elaboración propia.

2.8 Arquitectura del sistema

A grandes rasgos, la arquitectura de software es una vista del sistema que incluye los principales componentes del mismo, la conducta de esos componentes según se percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema. Establece los fundamentos para que analistas, diseñadores, programadores y otros roles, trabajen en una línea común que permita alcanzar los objetivos del sistema, cubriendo todas las necesidades. La vista arquitectónica es una vista abstracta, aportando el más alto nivel de comprensión y la supresión o diferimiento del detalle inherente a la mayor parte de las abstracciones (Campos Días, 2008).

Para facilitar la tarea de desarrollo de la propuesta de solución se utilizó la propia arquitectura de *Symfony* modelo-vista-controlador (MVC), ya que separa la lógica de negocio (el modelo) y la presentación (la vista) por lo que se consigue un mantenimiento más sencillo de las aplicaciones además viene integrado al Symfony2 en las capas de presentación y negocio. Cuando el usuario solicita alguna acción en el sistema, el sistema de enrutamiento determina qué controlador está asociado con la página solicitada por el usuario, Symfony2 ejecuta el controlador asociado a dicha página que el

mismo solicita al modelo los datos correspondientes a la página solicitada. Con los datos devueltos por el modelo, el controlador solicita a la vista que cree una página mediante una plantilla y que inserte los datos del modelo. El controlador entrega al servidor la página creada por la vista y se le devuelve al usuario la página correspondiente a la acción seleccionada.

En el desarrollo de la propuesta de solución la arquitectura está estructurada de la siguiente manera: la vista está formada por las páginas HTML desarrolladas con el motor de plantillas de Symfony *Twig*, Metronic como *framework* ccs basado en *Boostrap* y el *plugins* de *JavaScript JQuery* para las peticiones Ajax, en el modelo se usa Doctrine como herramienta de mapeo objeto relacional y en el controlador se encuentran los controladores que heredan de la clase *controler* de Symfony. A continuación, en la Figura 4 se muestra su funcionamiento:

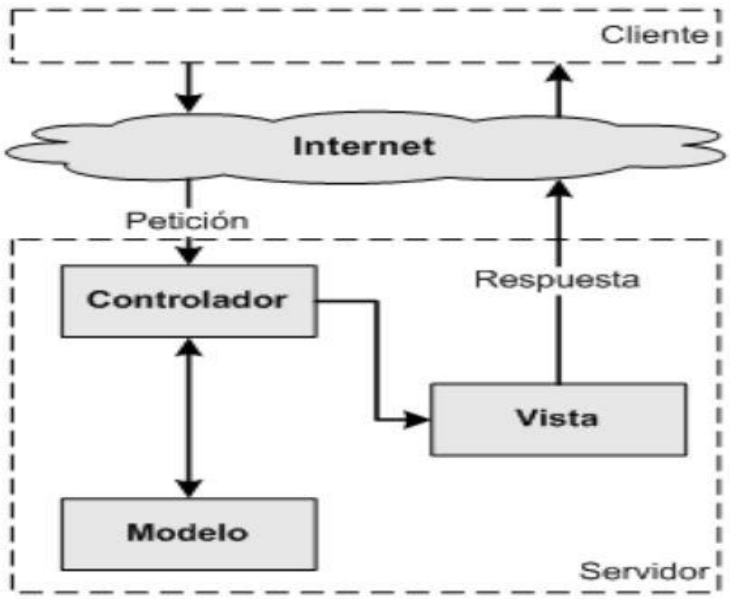


Figura 4. Patrón arquitectónico MVC (Modelo Vista Controlador). Fuente: (Fabien Potencier, 2013)

2.9 Patrones de diseño

Son aquellos que expresan esquemas para definir estructuras de diseño o las relaciones que existen entre ellos a partir de las cuales se pueden construir sistemas de software. Proponen una forma de reutilizar experiencias de los desarrolladores, para ello clasifican y describen formas de solucionar problemas frecuentes durante el desarrollo de software. Estos se aplican a un elemento específico como un agregado de componentes para resolver un determinado problema de diseño, relaciones entre los componentes o los mecanismos para efectuar la comunicación componente a componente (Pressman, 2005).

Dentro de los patrones de diseño se encuentran dos grupos fundamentales conocidos por Patrones Generales de Asignación de Responsabilidades de Software (GRASP por sus siglas en inglés) y Banda de los Cuatro (GOF por sus siglas en inglés).

Los Patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. De los patrones GRASP en la propuesta de solución se utilizaron los siguientes:

2.9.1 Patrón Experto

Permite asignar una responsabilidad al experto en información o sea a la clase que cuenta con la información necesaria para cumplir una determinada responsabilidad. Symfony2 utiliza este patrón con la inclusión de Doctrine para el mapeo de bases de datos. Se utiliza específicamente para crear una capa de abstracción en el modelo, encapsular toda la lógica de los datos y generar las clases entidades con todas las funcionalidades comunes (GET, SET y el constructor de la entidad); las clases de abstracción de datos poseen un grupo de funcionalidades que están relacionadas directamente con la entidad que representan y contienen la información necesaria de la tabla asociada.

Beneficios del patrón Experto:

- Conserva el encapsulamiento, ya que los objetos se valen de su propia información para realizar todas las peticiones. Esto posibilita tener sistemas de fácil mantenimiento.
- El comportamiento se distribuye entre las clases que cuentan con la información requerida para cumplir con la responsabilidad asignada.

Un ejemplo de este patrón puede visualizarse en la clase *Área*. En la siguiente figura se muestra un ejemplo del mismo:

```php
namespace uci\SaspedBundle\Entity;

use Doctrine\ORM\Mapping as ORM;
use Symfony\Component\Validator\Constraints as Assert;
use Symfony\Bridge\Doctrine\Validator\Constraints\UniqueEntity;

/**
 * Area
 *
 * @ORM\Table(name="area")
 * @ORM\Entity
 * @UniqueEntity("area")
 */
class Area
{
    /**
     * @var integer
     *
     * @ORM\Column(name="id", type="integer", nullable=false)
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="SEQUENCE")
     * @ORM\SequenceGenerator(sequenceName="area_id_seq", allocationSize=1, initialValue=1)
     */
```

Figura 5. Ejemplo de patrón experto. Fuente: Elaboración propia.

2.9.2 Patrón Creador

Una vez implementado, permite identificar quién debe ser el responsable de la creación o instanciación de nuevos objetos o clases. Plantea que se debe asignar a una clase A la responsabilidad de crear una instancia de una clase B. Como la creación de objetos es una de las actividades más frecuentes en un sistema orientado a objetos es importante el uso de este patrón para guiar la asignación de responsabilidades relacionadas con la creación de objetos. Este patrón es utilizado en la implementación de las clases controladoras donde se encuentran las acciones definidas para el sistema. En dichas acciones se crean los objetos de las clases que representan las entidades.

Beneficios del patrón Creador

- Se crean menos dependencias y existe mayor posibilidad de reutilización de código.

En la siguiente figura se muestra un ejemplo del mismo:

```
public function createAction(Request $request) {
    $entity = new Area();
    $form = $this->createCreateForm($entity);
    $form->handleRequest($request);

    if ($form->isValid()) {
        $em = $this->getDoctrine()->getManager();
        $em->persist($entity);
        $em->flush();

        return $this->redirect($this->generateUrl('area_show', array('id' => $entity->getId())));
    }

    return $this->render('SaspedBundle:Area:new.html.twig', array(
                'entity' => $entity,
                'form' => $form->createView(),
    ));
}
```

Figura 6. Ejemplo de patrón creador. Fuente: Elaboración propia.

2.9.3 Patrón Controlador

Este patrón sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método encargado de la ejecución. Se manifiesta en todo el sistema debido a que cada uno de los eventos generados por el usuario son atendidos por el archivo *routing.yml* que es el encargado de redirigir la petición al método de una clase controladora que realice las operaciones solicitadas, pero siempre manteniendo las clases controladoras sin sobrecarga, es decir siempre manteniendo la alta cohesión.

La siguiente figura muestra un ejemplo de este patrón:

```
area:
    path:       /
    defaults:  { _controller: "SaspedBundle:Area:index" }

#area_show:
#    path:       /{id}/show
#    defaults:  { _controller: "SaspedBundle:Area:show" }
#
#area_new:
#    path:       /new
#    defaults:  { _controller: "SaspedBundle:Area:new" }
#
#area_create:
#    path:       /create
#    defaults:  { _controller: "SaspedBundle:Area:create" }
#    methods:   POST

area_edit:
    path:       /{id}/edit
    defaults:  { _controller: "SaspedBundle:Area:edit" }

area_update:
    path:       /{id}/update
    defaults:  { _controller: "SaspedBundle:Area:update" }
    methods:   [POST, PUT]

area_delete:
    path:       /{id}/delete
    defaults:  { _controller: "SaspedBundle:Area:delete" }
```

Figura 7. Ejemplo de patrón controlador. Fuente: Elaboración propia.

2.9.4 Patrón Alta cohesión

La alta cohesión hace referencia a cuanta responsabilidad tiene una determinada clase controladora, o sea una clase debe tener responsabilidades moderadas. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas, que no realicen un trabajo enorme. Symfony2 permite la organización del trabajo en cuanto a la estructura del proyecto y la asignación de responsabilidades con una alta cohesión. Se puede observar en el sistema, ya que cada clase controladora maneja solamente las responsabilidades correspondientes a las entidades con las que se relaciona, además para cada vista existe una página controladora encargada de manejar sus solicitudes.

Beneficios del patrón Alta cohesión:

- Mejoran la claridad y la facilidad con que se entiende el diseño.
- Se simplifican el mantenimiento y las mejoras en funcionalidad.

La siguiente figura muestra un ejemplo de su uso:

```
public function createAction(Request $request) {
    $entity = new Area();
    $form = $this->createCreateForm($entity);
    $form->handleRequest($request);

    if ($form->isValid()) {
        $em = $this->getDoctrine()->getManager();
        $em->persist($entity);
        $em->flush();

        return $this->redirect($this->generateUrl('area_show', array('id' => $entity->getId())));
    }

    return $this->render('SaspedBundle:Area:new.html.twig', array(
                'entity' => $entity,
                'form' => $form->createView(),
    ));
}
```

Figura 8. Ejemplo de patrón de alta cohesión. Fuente: Elaboración propia.

2.9.5 Patrón Bajo acoplamiento

El patrón bajo acoplamiento plantea que se debe asignar las responsabilidades de forma tal que las clases dependan del menor número de clases que sea posible. Este patrón resuelve el problema de cómo dar soporte a una dependencia escasa y a un aumento de la reutilización. El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas.

En Symfony2 las clases controladoras heredan de la clase *Controller* alcanzando de esta manera un bajo acoplamiento de clases. Las clases que implementan la lógica del negocio y de acceso a datos se encuentran en el modelo, y no tienen asociaciones con las de la vista o el controlador, lo que proporciona bajas dependencias.

Beneficios del patrón Bajo acoplamiento:

- Fáciles de entender por separado.
- Fáciles de reutilizar.

La siguiente figura muestra un ejemplo de su uso:

```
class AreaController extends Controller {

    /**
     * Lists all Area entities.
     *
     */
    public function indexAction(Request $request) {
        $em = $this->getDoctrine()->getManager();

        $em=$this->getDoctrine()->

        $entities = $em->getRepository('SaspedBundle:Area')->findAll();
        $entity = new Area();
        $form = $this->createCreateForm($entity);
        $form->handleRequest($request);

        if ($form->isValid()) {
            $em->persist($entity);
            $em->flush();
      $request->getSession()->getFlashBag()->add('success', 'El área fue creada exitosamente');
            return $this->redirect($this->generateUrl('area'));
        }
        return $this->render('SaspedBundle:Area:index.html.twig', array(
                    'entities' => $entities,
```

Figura 9. Ejemplo de patrón de bajo acoplamiento. Fuente: Elaboración propia.

Los patrones GOF (del inglés Gang-of-Four) más conocidos como “Pandilla de los Cuatro” son utilizados básicamente para ocultarles a los usuarios la complejidad de un sistema, mostrándole solamente lo que necesita ver (Gerrero, y otros, 2013). Del patrón GOF se utilizó el siguiente:

2.9.6 Patrón Decorador

Este patrón añade funcionalidad a una clase dinámicamente, proporcionando una alternativa flexible a

la especialización mediante herencia, cuando se trata de añadir funcionalidades. Tiene como ventaja que aporta una mayor flexibilidad que la herencia estática, permitiendo, entre otras cosas, añadir una funcionalidad dos o más veces, evita concentrar en lo alto de la jerarquía clases “guiadas por las responsabilidades”, es decir, que pretenden satisfacer todas las posibilidades y de esta forma las nuevas funcionalidades se componen de piezas simples que se crean y se combinan con facilidad, independientemente de los objetos cuyo comportamiento extienden.

En Symfony2 este patrón es fácilmente visible ya que la vista se separa por niveles, en hasta 3 niveles, una plantilla base y varias plantillas que heredan de esta. Normalmente, la plantilla base es global en toda la aplicación y contiene el código HTML que es común a la mayoría de las páginas. Su uso aporta una mayor flexibilidad que la herencia estática, permitiendo, entre otras cosas, añadir una funcionalidad varias veces.

La siguiente figura muestra un ejemplo de dicho patrón:

Figura 10. Ejemplo del patrón Decorador. Fuente: Elaboración propia.

2.10 Conclusiones parciales

- La caracterización del sistema automatizado para la superación pedagógica refleja insuficiencias en el proceso de gestión y en la tecnología en la cual fue desarrollado, lo que limita la confección de la estrategia individual de Superación Pedagógica a cada docente y la escalabilidad del sistema.
- El análisis del dominio de aplicación junto a las necesidades del cliente, permitió identificar 88 historias de usuario y 38 tarjetas CRC, para garantizar el correcto funcionamiento del sistema para la automatización de los procesos de la estrategia de superación pedagógica.

CAPÍTULO 3 IMPLEMENTACIÓN Y PRUEBA

3.1 Introducción

En este capítulo se incluye la programación llevada a cabo a partir de los requerimientos, así como las pruebas realizadas para su validación. Por otra parte, se establecen los estándares de codificación a tener en cuenta para el desarrollo del sistema.

3.2 Fase de implementación

En la fase de Planificación se detallaron las HU correspondientes a cada una de las iteraciones a desarrollar. Las mismas se descomponen en tareas de programación o ingeniería, que se asignan a un equipo de desarrollo o persona. Estas tareas no tienen que necesariamente ser entendidas por el cliente, pueden ser escritas en lenguaje técnico y son para el uso estricto de los programadores.

3.2.1 Estándar de codificación

Establecer un estándar de codificación que sea aceptado por todo el equipo de desarrollo es muy importante en una metodología como XP que promueve la propiedad colectiva del código y la constante refactorización. El propósito fundamental de los estándares de codificación es que el sistema en cuestión tenga una arquitectura y un estilo consistente, con lo cual resulte fácil de entender y por supuesto fácil de mantener.

Los estándares de codificación son un complemento a la programación por pares, o sea en equipo, y no sólo es importante usar un estándar, sino usar un buen estándar de codificación, de esta forma se deberá promover la intención del código e incorporar las mejores prácticas de la codificación.

Entre otros aspectos se apunta a las variaciones en las convenciones o adiciones necesarias en el código del lenguaje PHP especialmente dirigido al trabajo con el marco de trabajo Symfony2. Los principales aspectos incluidos en dicho estándar se relacionan a continuación:

- ✓ Agregar un único espacio alrededor de operadores (==, &&).
- ✓ Agregar un único espacio antes de los paréntesis de apertura de una palabra clave de control (*if, else, for, while*).
- ✓ Agregar una línea en blanco antes de la sentencia *return*.
- ✓ Utiliza constantes de tipo PHP nativas en minúsculas*: false, true* y *null*. Lo mismo aplica para array ().
- ✓ Utilizar llaves para indicar el cuerpo de las estructuras de control sin importar el número de sentencias que éstas contengan.

- ✓ Colocar las llaves en sus propias líneas para clases, métodos y declaración de funciones.
- ✓ Definir una clase por archivo.
- ✓ Declarar las propiedades de las clases antes de los métodos.
- ✓ Declarar los métodos públicos primero, luego los protegidos y finalmente los privados.

Convención de Nombres

- ✓ Utilizar el estilo *camelCase* y no guiones bajos para las variables, funciones y nombres de métodos.
- ✓ Utilizar el estilo *StudlyCaps* para los nombres de clases.
- ✓ Utilizar guiones bajos para definir opciones, argumentos y nombres de parámetros.
- ✓ Utilizar los *namespace* para todas las clases.
- ✓ Utilizar Symfony como el *namespace* de primer nivel.

3.3 Diagrama de despliegue

El diagrama de despliegue describe la distribución física de un software en un ambiente de producción o prueba. En él se grafican los nodos y las conexiones necesarias para el funcionamiento exitoso del producto software. Muestra también, dónde se localizan los componentes del software, o sea, en qué servidores, ordenadores o dispositivo hardware.

La siguiente figura muestra el diagrama de despliegue correspondiente al sistema para la implementación de la estrategia de superación pedagógica de la UCI. El mismo está compuesto por dos servidores, el primero será el Servidor de Aplicaciones y contendrá el Sistema para la implementación de la estrategia de superación pedagógica de cada usuario de la UCI. El segundo servidor contendrá la base de datos del sistema y deberá poseer *PostgresSQL*. La comunicación entre dichos servidores será a través de la clase *pgp_conection*. El diagrama representa además una estación cliente la cual permitirá la interacción con los dispositivos. Esta estación de trabajo utiliza el protocolo SOAP 1 para la comunicación con el servidor de aplicaciones.

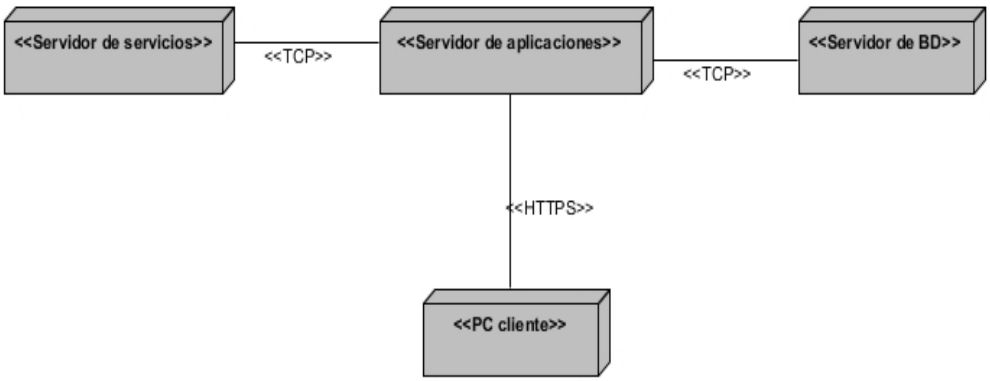


Figura 11. Diagrama de despliegue. Fuente: Elaboración propia.

3.4 Estrategia de pruebas

Roger S. Pressman plantea que una estrategia de prueba del *software* integra los métodos de diseño de caso de pruebas en una serie bien planeada de pasos que desembocará en la eficaz construcción de *software*. Una estrategia de prueba debe ser lo suficientemente flexible como para promover un enfoque personalizado. Al mismo tiempo, debe ser lo adecuadamente rígido como para promover una planeación razonable y un seguimiento administrativo del avance del producto (Pressman, 2014).

Pressman propone las siguientes estrategias de pruebas:

- ✓ Pruebas de unidad.
- ✓ Pruebas de validación.
- ✓ Pruebas de sistema.

Teniendo en cuenta estos aspectos, la solución propuesta constará de las pruebas de unidad, de validación y del sistema.

3.4.1 Pruebas unitarias

En programación, una prueba unitaria es una forma de probar el correcto funcionamiento de un módulo de código. Esto sirve para asegurar que cada una de las partes que integran nuestra aplicación funcionen correctamente por separado (Benítez, 2011).

Por tanto, las pruebas unitarias se realizan para controlar el funcionamiento de pequeñas porciones de código como son subprogramas o métodos (en la programación orientada a objeto POO). Generalmente son realizadas por los mismos programadores puesto que al conocer con mayor detalle el código, se les simplifica la tarea de elaborar conjuntos de datos de prueba para probarlo. Los métodos de pruebas unitarias residen en clases *Test*, que se almacenan en los archivos de código fuente. Por tanto, este tipo de prueba siempre está orientada a caja blanca.

Las pruebas de caja blanca según (Yague, y otros, 2009) están basadas en estudiar el código fuente y se utilizan, principalmente, desde una perspectiva interna en el desarrollo de software, logrando como resultado que disminuya en un gran porciento el número de errores existentes en los sistemas y por ende una mayor calidad y confiabilidad. Mediante la prueba de caja blanca el ingeniero del software puede obtener casos de prueba que:

- ✓ Garanticen que se ejecute por lo menos una vez todos los caminos independientes de cada módulo, programa o método.
- ✓ Ejerciten todas las decisiones lógicas en las vertientes verdadera y falsa.
- ✓ Ejecuten todos los bucles en sus límites operacionales.
- ✓ Ejerciten las estructuras internas de datos para asegurar su validez.

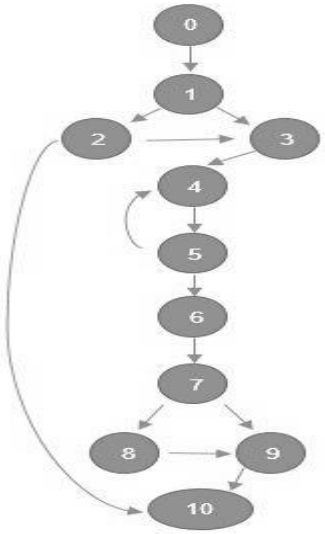
Para el desarrollo del proceso, se aplicó la técnica de camino básico a diferentes módulos del sistema, la misma permite obtener una medida de la complejidad lógica de un diseño y usarla como guía para la definición de un conjunto básico. La idea es derivar casos de prueba a partir de un conjunto dado de caminos independientes por los cuales puede circular el flujo de control. Para obtener dicho conjunto de caminos independientes se construye el grafo de flujo asociado y se calcula su complejidad ciclomática.

Se realizaron pruebas de caja blanca a las funcionalidades *personalAction* y *createAction*.

1.Funcionalidad: personalAction

La funcionalidad implementada para el método personalAction se encuentra disponible en el Anexo 7.

Grafo del flujo



Complejidad ciclomática

A= 14 (aristas)

N= 11 (nodos)

V(G) = **A-N+2**

V(G) = 14-11+2

V(G) = 5

P = 4 (nodos predicados) **R** = 5 (regiones del grafo)

V(G) = P+1 V(G) = R

V(G) = 4+1 V(G) = 5

V(G) = 5

Figura 12. Grafo de flujo y Complejidad ciclomática correspondiente al método personalAction.
Fuente: Elaboración propia

Caminos linealmente independientes

Camino 1. (0-1-2-10)

Camino 2. (0-1-3-4-5-6-7-9-10)

Camino 3. (0-1-3-4-5-7-8-10)

A continuación, se diseñan los casos de prueba que cubren los caminos independientes presentados.

Caso de prueba del camino 1:

Entrada: recibe como parámetro el id de la estructura.

Resultado esperado: -

Objetivo: se garantiza que se cumpla la condición de que no hay conexión con el servidor.

Caso de prueba del camino 2:

Entrada: recibe como parámetro el id de la estructura.

Resultado esperado: se obtienen todas las personas que posean el id del área pasado por parámetro.

Objetivo: se garantiza que al menos exista una persona que posea el id del área pasado por parámetro.

Caso de prueba del camino 3:

Entrada: recibe como parámetro el id de la estructura.

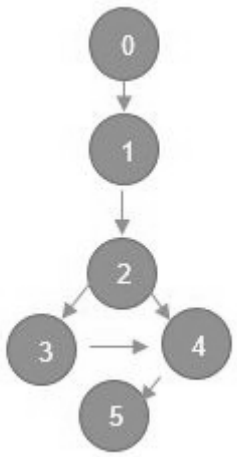
Resultado esperado: el sistema lanza una excepción no visible para el usuario conociendo que la búsqueda no arrojó resultados.

Objetivo: se garantiza que no existan personas dentro del área con id igual al pasado por parámetro.

2. Funcionalidad: createAction

La funcionalidad implementada para el método createAction se encuentra disponible en el Anexo 7.

Grafo de flujo



Complejidad ciclomática

A=6 **V(G)=A-N+2**

N=6 V(G)=6-6+2 V(G)=2

P=1 (nodos predicados)

V(G)=P+1 V(G)=1+1 V(G)=2

R=2 V(G)=R V(G)=2

Figura 13. Grafo de flujo y Complejidad ciclomática correspondiente al método createAction.
Fuente: Elaboración propia

Caminos linealmente independientes

Camino 1. (1-2-3-4-5-6)

Camino 2. (1-2-3-5-6)

Caso de prueba del camino 1:

Entrada: recibe como parámetro un formulario que contiene una característica.

Resultado esperado: muestra un mensaje en la vista indicando que la característica fue creada exitosamente.

Objetivo: se garantiza que los datos introducidos en el formulario sean correctos.

Caso de prueba del camino 2:

Entrada: recibe como parámetro un formulario que contiene una característica.

Resultado esperado: la característica no es creada y se da tratamiento a los errores cometidos a la hora de llenar el formulario.

Objetivo: se garantiza que no se introduzcan datos erróneos al llenar el formulario.

3.4.2 Pruebas de validación

En las pruebas de validación desaparece la distinción entre *software* convencional y orientado a objetos, estas se concentran en las acciones visibles para el usuario y en la salida del sistema que este puede reconocer. La validación del software se logra mediante una serie de pruebas que demuestran que se cumplen los requisitos funcionales. Se realiza un plan de pruebas para asegurar que se satisfagan todos los requisitos y que alcance todas las características de comportamiento (Pressman, 2005).

Para efectuar las pruebas de validación se realizaron **pruebas de caja negra**, estas se centran principalmente en los requisitos funcionales del *software* y se llevan a cabo sobre la interfaz del *software*, por lo que los casos de prueba pretenden demostrar que las funciones del *software* son operativas. Estas pruebas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. En ellas se ignora la estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándolos (Torres, 2012).

A continuación se muestra el caso de prueba de aceptación correspondiente a la historia de usuario **Registrar profesor y Registrar curso de superación**. Los casos de prueba de aceptación definidos por el cliente para el resto de las historias de usuario se especifican en el Anexo 8.

Tabla 11. Caso de prueba: Registrar profesor. Fuente: Elaboración propia.

Caso de prueba	
Código de CP: CP_HU29	**Nombre HU:** Registrar profesor
Nombre de la persona que realiza la prueba: Jorge Javier Tamayo Muñiz	
Descripción de la prueba: El Administrador accede al módulo de Administración y luego a la funcionalidad Gestionar profesores, donde el sistema debe mostrar una vista con la lista de profesores existentes en la base de datos y la opción Registrar un nuevo profesor. El sistema ofrece dos formas de registrar un profesor, una manual y la otra utilizando el servicio ldap de la UCI. Si se selecciona la opción manual, el sistema debe permitir introducir la información del profesor manualmente; sino al seleccionar la opción Dominio UCI, el administrador deberá introducir el usuario del profesor y el sistema debe verificar que las credenciales son válidas y permitir registrar el usuario en la base de datos.	
Condiciones de ejecución: El usuario Administrador debe estar autenticado previamente en el sistema.	
Entrada/Pasos de ejecución: 1. Si el registro es manual introducir los datos del profesor, sino introducir credencial del profesor. 2. Validar si los datos introducidos son correctos. 3. Insertar datos del profesor en la base de datos.	
Resultado esperado: Usuario registrado en el sistema.	
Evaluación: Satisfactoria.	
Prototipo: 	

Tabla 12. Caso de prueba: Registrar curso de superación. Fuente: Elaboración propia.

Caso de prueba	
Código de CP: CP_HU46	**Nombre HU:** Registrar curso de superación.
Nombre de la persona que realiza la prueba: Jorge Javier Tamayo Muñiz	
Descripción de la prueba: El Administrador o el rol que posea el privilegio de insertar nuevos cursos de superación, accede al módulo de Gestión de información y luego a la funcionalidad Cursos de superación, donde el sistema debe mostrar una vista con la lista de cursos existentes en la base de datos y la opción Registrar nuevo curso de superación. Luego, se introducen los datos asociados al profesor y el sistema debe verificar que los datos sean válidos y permitir registrar el curso en la base de datos.	
Condiciones de ejecución: El rol con los privilegios sobre el módulo de Gestión de información debe estar autenticado previamente en el sistema.	
Entrada/Pasos de ejecución: 4. Introducir los datos asociados al curso. 5. Validar si los datos introducidos son correctos. 6. Insertar datos del curso en la base de datos.	
Resultado esperado: El curso de superación fue registrado satisfactoriamente.	
Evaluación: Satisfactoria.	
Prototipo:	

3.4.3 Pruebas del sistema

Las pruebas del sistema abarcan una serie de pruebas diferentes cuyo propósito principal es ejercitar profundamente el sistema de cómputo. Aunque cada prueba tiene un objetivo distinto, todas trabajan para verificar que se hayan integrado adecuadamente todos los elementos del sistema y que realizan las funciones apropiadas (Pressman, 2005). La prueba del sistema realizada a la solución fue la prueba de rendimiento.

Las pruebas de rendimiento tienen que diseñarse para asegurar que el sistema pueda procesar su carga esperada. Esto normalmente implica planificar una serie de pruebas en las que la carga se va incrementando regularmente hasta que el rendimiento del sistema se hace inaceptable. Las pruebas de rendimiento se ocupan tanto de demostrar que el sistema satisface sus requerimientos como de describir problemas y defectos en el sistema. Estas pruebas implican estresar el sistema realizando demandas que están fuera de los límites del diseño del software, acercándose a la máxima carga del diseño del sistema hasta que el sistema falla.

Para medir el rendimiento correspondiente al SAPESP se elige el módulo de Administración, el mismo fue sometido a pruebas de carga y estrés para medir su funcionamiento. Estas pruebas estuvieron enfocadas al análisis del comportamiento de los tiempos de respuesta del servicio de recepción de las órdenes en dependencia de condiciones variables como el crecimiento de las conexiones concurrentes de los sistemas externos al SAPESP. Esta prueba se realizó primeramente para un total de veinticinco muestras, luego para cincuenta y finalmente para setenta y cinco. Dicho proceso se llevó a cabo en una computadora con un procesador *Intel core* i3 y 8 gigabyte de RAM. Para realizar las pruebas de rendimiento se utilizó la herramienta Apache JMeter en su versión 2.13.

3.5 Resultados de las pruebas

Después de haber realizado las pruebas unitarias a los módulos desarrollados se detectaron varias no conformidades, por lo que se ejecutaron tres iteraciones de revisión. En una primera iteración se encontraron seis funcionalidades con errores y posteriormente en una segunda iteración se detectaron

solamente dos, las cuales fueron resueltas y para una tercera iteración se detectaron cero no conformidades (Ver figura 14).

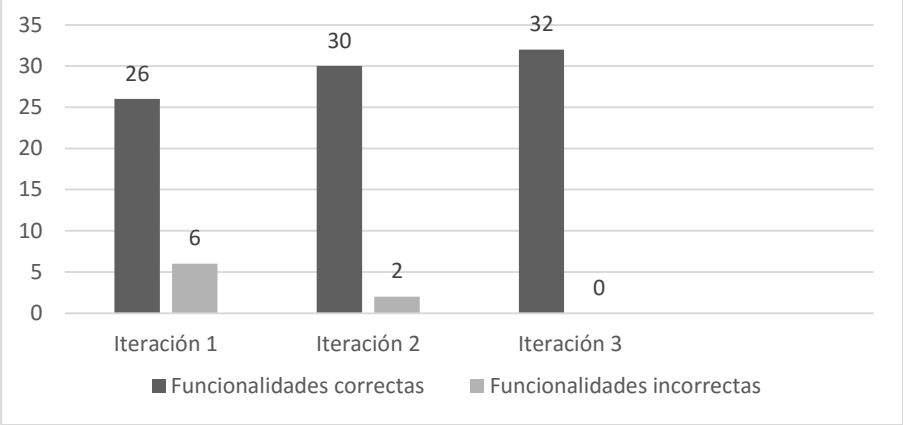


Figura 14. Resultados de las pruebas unitarias. Fuente: Elaboración propia.

Para validar las pruebas de integración se tuvieron en cuenta las pruebas unitarias realizadas anteriormente, dado que las funcionalidades analizadas a las que se le hicieron las pruebas son procesos fundamentales para elaborar la evaluación profesoral de cada uno de los docentes del claustro de la UCI. A continuación, se muestra una vista de la evaluación profesoral realizada a un profesor asociada a cada una de las características que serán evaluadas en un criterio de 2 a 5.

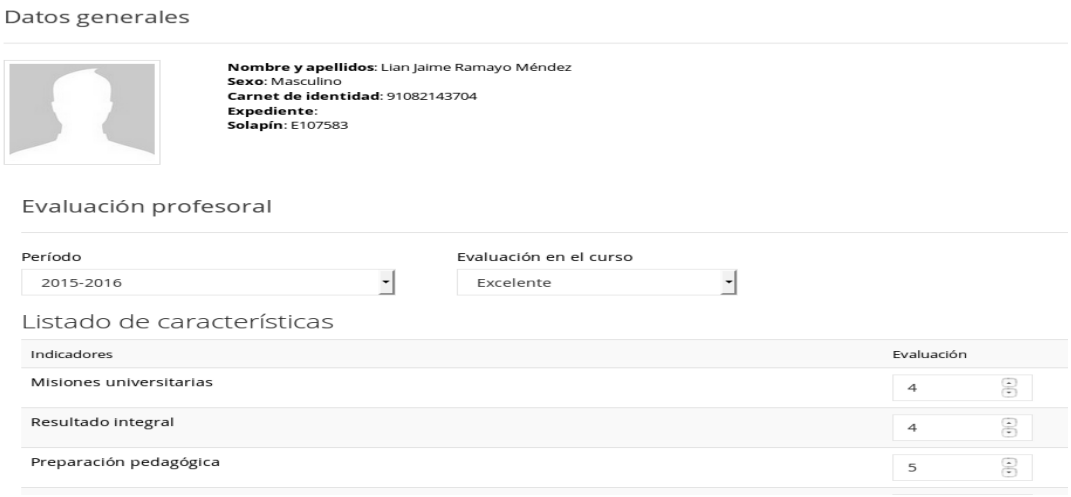


Figura 15. Evaluación profesoral. Fuente: Elaboración propia.

La realización de las pruebas de validación se ejecutó en tres iteraciones obteniéndose en cada una 8, 3 y 0 no conformidades respectivamente. Estas dificultades fueron solucionadas en un corto plazo, antes de pasar a la iteración posterior. (Ver figura 16).

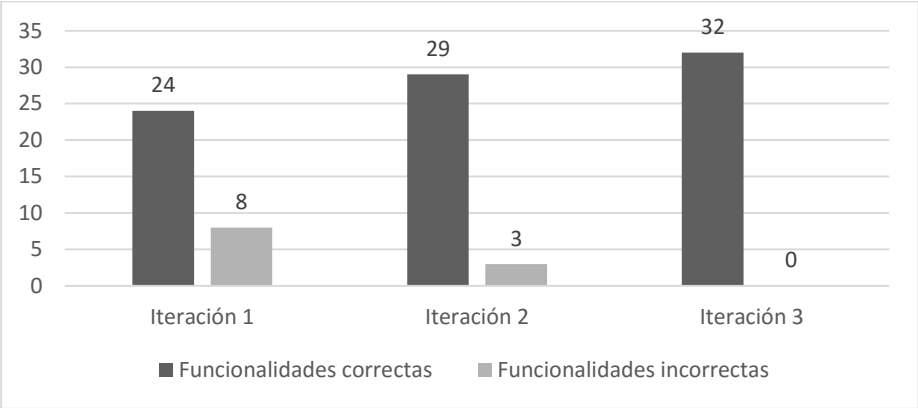


Figura 16. Diagrama de las pruebas de validación. Fuente: Elaboración propia.

La realización de las pruebas de rendimiento demostró que el sistema soporta el procesamiento concurrente de varias órdenes de impresión. A continuación se muestra el resultado de la prueba realizada en las siguientes figuras (Ver Figura 17, 18 y 19) para tres iteraciones, con valores de 25, 50 y 75 muestras para la funcionalidad Gestionar profesores perteneciente al módulo de Administración. Se describe en las mismas el tiempo alcanzado en la ejecución de las peticiones, los tiempos mínimos y máximos para cada solicitud y el rendimiento final alcanzado en cada iteración, comprobando el éxito o no de las peticiones procesadas en porciento de errores en el tiempo, en este caso con resultados satisfactorios para el análisis de los procesos del sistema.

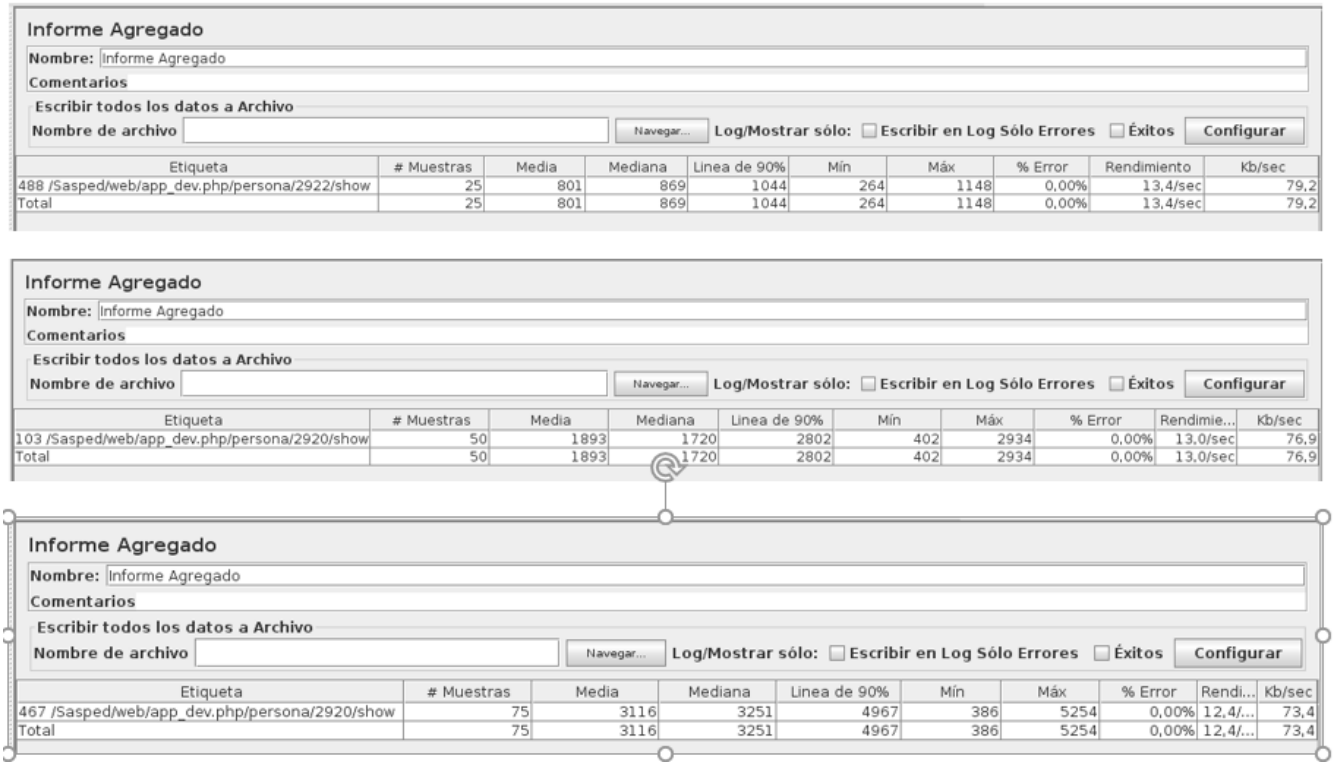


Figura 17. Resultados de las pruebas del sistema. Mostrar profesor. Fuente: Elaboración propia.

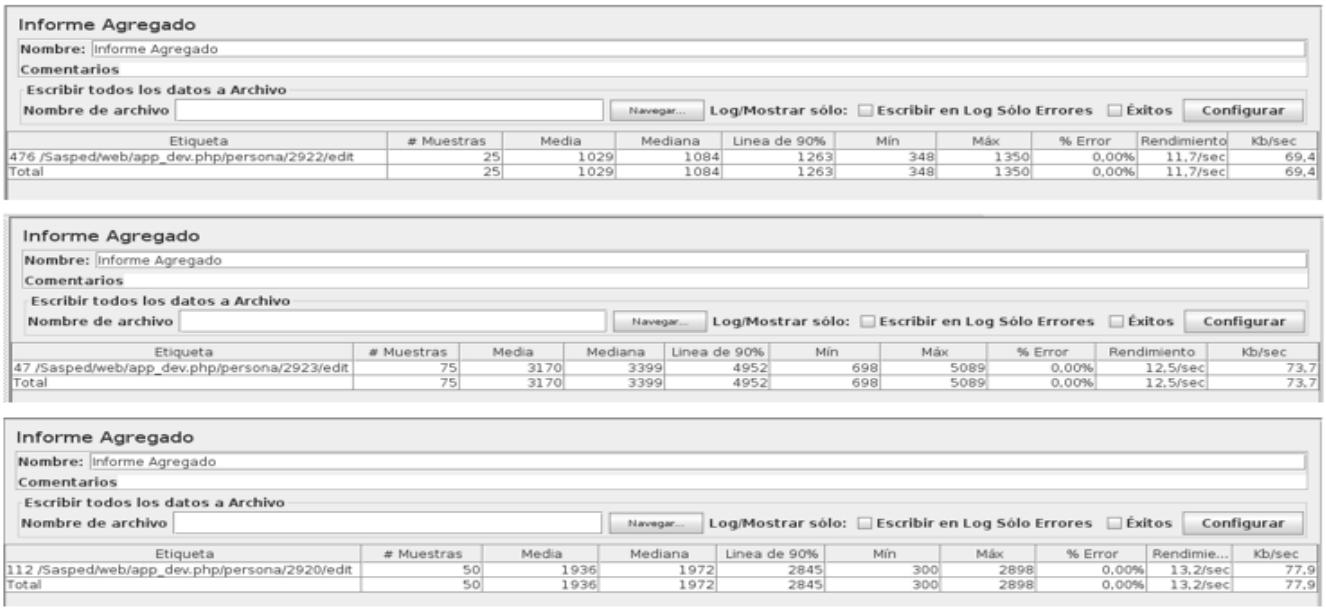


Figura 18. Resultados de las pruebas del sistema. Editar profesor. Fuente: Elaboración propia.

Figura 19. Resultados de las pruebas del sistema. Eliminar profesor. Fuente: Elaboración propia.

3.6 Conclusiones parciales

En este capítulo se describieron las fases de diseño, implementación y prueba.

- Se definió el estilo de codificación utilizado en la implementación del sistema, lo que permitió mantener una uniformidad en la codificación.
- El diagrama de despliegue definido mostró con mayor claridad la distribución física del sistema sobre una arquitectura de hardware.
- La aplicación de la técnica camino básico y el uso de la herramienta JMeter, permitió corroborar los resultados respecto a los casos de pruebas, evidenciando la escalabilidad del sistema.
- La realización de pruebas unitarias, demostró el funcionamiento de procedimientos internos del software y se encontraron un conjunto de no conformidades, las cuales fueron la base fundamental para que el sistema mejorara su rendimiento y funcionamiento.

CONCLUSIONES GENERALES

- La descripción y fundamentación de los sistemas informáticos que automatizan los procesos de superación pedagógica, corroboró que no ofrecen una estrategia de superación pedagógica individual y permitió la reutilización de funcionalidades para la propuesta de solución.

- En la caracterización del proceso de superación pedagógica generado por el SASPED en el CICE, se constata las insuficiencias en el framework de desarrollo y en los procesos de gestión de la información, de modo que se evidencia la necesidad de contribuir a los niveles de escalabilidad del software.

- A partir de la identificación de las funcionalidades del Sistema para la Automatización de los Procesos de la Estrategia de Superación Pedagógica y con los artefactos que genera la metodología XP, se realizó la modelación y el desarrollo de la propuesta de solución.

- Teniendo en cuenta las pruebas de unidad, de validación y de rendimiento realizadas al Sistema para la Automatización de los Procesos de la Estrategia de Superación Pedagógica, se demostró que la propuesta constituye una herramienta práctica y escalable para la gestión de dichos procesos.

RECOMENDACIONES

- Incrementar el número de funcionalidades del sistema en correspondencia con las necesidades del CICE, permitiendo así una mayor productividad del mismo.
- Integrar el sistema con gestión universitaria, para que el sistema sea más robusto y se puedan brindar otras funcionalidades.

BIBLIOGRAFÍA

A. Guerrero, Carlos, y otros. 2014. *Estudio comparativo de marcos de trabajo para el desarrollo de software orientado a aspectos.* 2014.

Addine Fernández, Fátima y González Soca, Ana María. 2007. *Principios para la dirección del proceso pedagógico.* La Habana:Encimed : En: Compendio de pedagogía., 2007.

Alegsa. 2014. Alegsa. [En línea] 5 de Marzo de 2014. [Citado el: 16 de Mayo de 2016.] http://www.alegsa.com.ar/Dic/sistema.php.

Alvarez, Miguel Angel. 2014. Desarrolloweb.com. [En línea] 2014. [Citado el: 9 de Diciembre de 2015.] http://www.desarrolloweb.com/articulos/composer-gestor-dependencias-para-php.html.

Atehortúa Hurtado , Federico. 2005. Gestión y auditoría de la calidad para organizaciones públicas. [En línea] 2005. https://books.google.com.cu.

Beck, Kent. 2012. Extreme Programming Explained. [En línea] Enero de 2012. [Citado el: 29 de Mayo de 2016.] https://www.amazon.com/Extreme-Programming-Explained-Embrace-Change/dp/0321278658/188-6447600-0144053?ie=UTF8&*Version*=1&*entries*=0#reader_0321278658.

—. 2005. *Extreme Programming Explained: Embrace Change.* 2005.

Benítez, Carlos. 2011. QUnit, tesyeando nuestras aplicaciones Javascript. [En línea] 1 de Febrero de 2011. [Citado el: 7 de Abril de 2016.] http://www.etnassoft.com/2011/02/01/qunit-testeando-nuestras-aplicaciones-javascript/.

Blanco, Antonio Pérez , Martínez, Marta Llantada y Castellamos, Doris Simons. 2002. *Pedagogía para Educadores (versión digital).* La Habana : Instituto Superior Pedagógica "Enrique José Varona", 2002. pág. 142.

Bravo, Tito Díaz. 2013. *Estrategia para la superación pedagógica del Claustro de la Universidad de las Ciencias Informáticas (UCI).* La Habana, Cuba : s.n., 2013.

Cabello, Victoria Nevado. 2010. *Introducción a Las Bases de Datos Relacionales.* s.l. : Visión Libros, 2010.

Campos Días, Yoandy. 2008. Arquitectura de Software. [En línea] 21 de Junio de 2008. [Citado el: 24 de Marzo de 2016.] http://arquitecturasoftware.blogspot.com/.

Canto, Carlos. 2010. Automatización: Conceptos Generales. [En línea] 2010.

Carlderón Ariosa, Regla Margarita y Castro Lamas, Julio. 2002. *Formación Pedagógica para profesores de las sedes universitarias.* s.l. : Revista pedagógica Universitaria, 2002. págs. 54-61.

CCM. 2016. Lenguajes de programación. [En línea] Marzo de 2016. [Citado el: 25 de Marzo de 2016.] http://es.ccm.net/contents/304-lenguajes-de-programacion.

Cordanov, Anjes. 2011. Scribd. [En línea] 18 de Julio de 2011. [Citado el: 20 de Febrero de 2016.] http://es.scribd.com.

Cruz, Daisy Estupiñán y Linares, Manuel Cordero. 2013. Estrategia de superación pedagógica para docentes de la carrera de Medicina. [En línea] Septiembre de 2013. [Citado el: 17 de Febrero de 2016.]

DRAE. 2016. DRAE. [En línea] 2016. [Citado el: 15 de Mayo de 2016.] http://www.rae.es/.

Eguiluz, Javier. 2012. *Desarrollo web ágil coon Symfony2.* 2012. pág. 613.

Escribano, Gerardo Fernández. 2002. Extreme Programming. [En línea] 2002. [Citado el: 19 de Mayo de 2016.]

Eugenia Bahit. 2011. Desarrolloweb.com. [En línea] 2011. [Citado el: 19 de Mayo de 2016.] http://www.desarrolloweb.com/articulos/desarrollo-agil-scrum.html.

Fabien Potencier, Ryan Weaver. 2013. *Synfony 2.4.* Segunda. 2013.

Friedman, J. 2007. *Planificación.* s.l. : Ministerio de Administraciones Públicas, 2007.

Gerrero, Carlos y Suarez, Johana. 2013. scielo. [En línea] 2013. [Citado el: 6 de Abril de 2016.] http://www.scielo.cl/scielo.php?.

Giménez, Jorge Vilar. 2003. Revista Énfasis Logística. [En línea] 1 de Junio de 2003. [Citado el: 16 de Mayo de 2016.] http://www.logisticamx.enfasis.com/notas/3671-sistemas-automatizados-vida-las-empresas.

Henao, Víctor Mejía. 2009. *La informática y su contribución a la automatización de procesos.* 2009.

Holmes, Harry y Joice, Daniel. 2009. *Object-Oriented Programming With Java.* 2nd edition. 2009.

JMeter. 2013. JMeter. [En línea] 2013. [Citado el: 18 de Mayo de 2016.] http://jmeter.apache.org/usermanual/intro.html.

Juarez, Efrain Alberto Olguin. 2013. Open UP. [En línea] Septiembre de 2013. [Citado el: 14 de Diciembre de 2015.] http://openupeaojmp.blogspot.com/2013/09/metodologia-open-up.html.

Larman, Craig. 2004. *UML y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado.* [ed.] Feliz Varela. Segunda edición. La Habana : s.n., 2004. pág. 624.

López, Ingrid Analy Rojas. 2013. Sistema de Gestión e Inventario para equipos de Cómputo del H. Ayuntamiento de Orizaba. [En línea] 2013. http://s3.amazonaws.com/academia.edu.documents/34465231/reporte_residencias_aliz.pdf?AWSAccessKeyId=AKIAJ56TQJRTWSMTNPEA&Expires=1465479976&Signature=XqmJ116CCk8PIhlGMeAenIPm%2FZ8%3D&response-content-disposition=inline%3B%20filename%3DInstituto_Tecnologic.

Lozano, Carlos. 2016. ACADEMIA. [En línea] 2016. [Citado el: 22 de Febrero de 2016.] http://www.academia.edu/8199329/SGBD_CARACTERISTICAS_VENTAJAS_DESVENTAJAS_REQUERIMIENTOS.

Magaña, Sara. 2009. Estudio comparativo de Lenguajes de Modelado de Procesos de Negocio para su integración en Procesos de Desarrollo de Software dirigido por modelos. Universidad Carlos III. [En línea] 2009. [Citado el: 29 de Mayo de 2016.] http://e-archivo.uc3m.es/handle/10016/9077.

Mira, J.J, y otros. 2006. *La Gestión por Procesos.* Alicante : s.n., 2006.

Ortiz Sánchez, Yurisnel , y otros. 2014. Comunidades Virtuales, Ministerio de la Educación Superior. [En línea] 24 de Abril de 2014. [Citado el: 16 de Mayo de 2016.] http://cvi.mes.edu.cu/peduniv/index.php/peduniv/article/viewFile/615/pdf_36.

Otto, Mark y Thorton, Jacob. 2012. librosweb. [En línea] 2012. [Citado el: 25 de Marzo de 2016.] http://librosweb.es/libro/bootstrap_3/.

Pressman, Roger. 2005. *Ingeniería de Software. Un enfoque práctico.* 2005.

Pressman, Roger S. 2014. *Prueba de aplicaciones web. INGENIERIA DE SOFTWARE. Un enfoque práctico.* 2014.

Radrigan, Marisa. 2005. *Metodología de la Investigación.* 2005.

Rosales Morales, Yanet, Marrero Clark, Michael Eduardo y Trujillo Oliva, Adrian. 2013. Ingenieria y gestión de software. [En línea] 2013. http://publicaciones.uci.cu/index.php/SC .

Sommerville. 2005. *Ingeniería del software.* Séptima edición. Madrid : s.n., 2005. pág. 712.

Tinoco Gómez, Oscar , Rosales López, Pedro Pablo y Salas Bacalla, Julio. 2010. Criterios de selección de metodologías de desarrollo de software. [En línea] 2010. http://revistasinvestigacion.unmsm.edu.pe/index.php/idata/article/view/6191.

Torres, Manolo. 2012. indalog. *Técnicas de prueba.* [En línea] 2012. [Citado el: 7 de Abril de 2016.] http://indalog.ual.es/mtorres/LP/Prueba.pdf.

Valdés, Damián Pérez. 2007. MaestrosDel Web. [En línea] 2 de Noviembre de 2007. [Citado el: 22 de Febrero de 2016.] http://www.maestrosdelweb.com/los-diferentes-lenguajes-de-programacion-para-la-web/.

Yague, Agustín y Garbajosa, Juan. 2009. Comparativa práctica de las pruebas en entornos. [En línea] 2009. http://www.redalyc.org/articulo.oa?id=92217159004.