Universidad de las Ciencias Informáticas

Facultad 4

Sistema de gestión de información para los resultados de las pruebas de eficiencia física en la Universidad de las Ciencias Informáticas.

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

Autores:

Victor Angel Fong Rios

Roger Armando González Rodríguez

Tutor:

Ing. Luis Felipe Díaz Barrios

La Habana, junio 2013

Año 55 del Triunfo de la Revolución

Declaración de Autoría

Por este medio declaramos que somos los únicos autores de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas (UCI) para que hagan el uso que estimen pertinente con este trabajo.

Para que así conste firmamos la presente a los ____ días del mes de junio del 2013.

_____	_____
Victor Angel Fong Rios	Roger Armando González Rodríguez
Firma del Autor	Firma del Autor

_____

Ing. Luis Felipe Diaz Barrios

Firma del Tutor

"La disciplina es la parte más importante del éxito".

Truman Capote

Agradecimientos

De Victor

A Dios por su amor incondicional, por darme fuerzas, salud y sabiduría durante estos año y porque todo lo que soy se lo debo a ÉL.

A mi mamá y mi papá por todo el sacrificio, amor, entrega, dedicación y confianza que siempre me han brindado y por esperar siempre lo mejor de mí.

A mis pastores Raúl y Martha por recibirme como su hijo, por su apoyo incondicional en todo momento, por corregirme y enseñarme a ser una mejor persona cada día.

A mis hermanos en Cristo por apoyarme en oración, por cada una de las palabras de aliento que me dieron, por compartir momentos de alegría y de tristeza.

A toda mi familia por su ayuda en todo momento y en especial a mi abuela por preocuparse por mí.

A mis compañeros de apartamento, en especial a Juan Carlos, a Alexey, a Yadrián, a los Ramones, a Ivis, por compartir junto a ustedes muchas horas de estudio.

A mi compañero de tesis Roger, sin el cual no hubiera podido llegar a la meta y por soportarme durante todo este tiempo.

De Roger

En primer lugar les agradezco a mis padres, a mi mamá por ser mi guía en todo momento, por siempre luchar conmigo para que fuera siempre por el camino correcto, repitiéndome los consejos una y otra vez, a mi papá por creer siempre en mi y darme fuerza, apoyándome en todo momento.

A mi abuela y mi hermana por estar siempre preocupándose de mi situación y brindarme siempre su apoyo en todo momento.

A mi tío Pepe por estar pendiente de lo que me podía faltar, por sus consejos siempre que los necesité y por ser un segundo padre para mí.

A Saly por haber compartido durante estos 5 años de universidad conmigo, por ser la persona que más confió en mí y por toda su ayuda incondicional.

Agradecimientos

A todos los socios míos que venimos de primero juntos a Eric, Sol, el chino, a Yasel aunque ya no se encuentra con nosotros y al Wilfre.

A mi mejor amiga Dailena por toda su ayuda durante estos años en la universidad, siempre batallando conmigo, atenta a todos mis problemas, siendo en muchas ocasiones mi paño de lágrima.

A Virgilio, Rodiel, Rafael Tamayo, Dannier, Angel, el Rafa, Yandry, Amado y el Yonki.

Agradezco a mi compañero de tesis Victor por haber compartido conmigo la realización de este trabajo.

De Victor

A mi papá Silbiades Fong, que aunque ya no se encuentra presente, me inspiró y me dio fuerzas para seguir adelante.

A mi mamá Mirtha Rios Martínez, por ser la persona más especial de mi vida y por esperar siempre lo mejor de mí.

A mi abuela Manuela Martínez, por su apoyo incondicional y por darme ánimo en todo momento.

De Roger

Le dedico el trabajo de diploma a mi familia especialmente a mi mamá, mi papá, mi hermana, mi abuela y mi Tío, por haber pasado tantos ratos amargos y sufrir al lado mío siempre que los resultados no fueron los mejores.

El presente Trabajo de Diploma tiene como objetivo elaborar un Sistema de gestión de información para los resultados de las pruebas de eficiencia física en la Universidad de las Ciencias Informáticas.

Para ello se realizó un estudio de los sistemas similares, se seleccionó como metodología de desarrollo XP, como *framework Symfony2,* utilizando a PHP 5.3 como lenguaje de programación. Se generaron los artefactos correspondientes a las fases de la metodología utilizada como: historias de usuario, plan de iteraciones, plan de duración de iteraciones, plan de entregas y tarjetas CRC.

Se obtuvo como resultado un sistema que además de permitir la gestión de las pruebas, ejercicios, grupos, profesores, usuarios y estudiantes, asigna ejercicios a los grupos de acuerdo a los niveles alcanzados por los alumnos en cada una de los tipos de pruebas (planchas, abdominales, carrera de velocidad, carrera de resistencia y salto de longitud). Mediante las pruebas se validó el correcto funcionamiento del sistema.

Palabras Claves: ejercicios físicos, pruebas de eficiencia física, sistema de gestión de información.

Contenido

Índice de tablas

Introducción

El deporte y la educación física juegan un papel importante a nivel nacional e internacional, la práctica de estos tiene un impacto positivo sobre la salud y desarrollo de las personas, además de establecer lazos de amistad entre las naciones.

En Cuba la eficiencia física o rendimiento motor, como también se conoce, constituye la expresión del desarrollo de las capacidades físicas alcanzadas por el hombre como consecuencia del fenómeno educativo y formativo en la aplicación de los planes y programas que en materia de cultura física, deportes y recreación, lleva a efecto el INDER en su tarea de masificar la actividad física sistemática en nuestro país.

La práctica masiva de deportes y las actividades físicas disminuyen el riesgo de mortalidad por enfermedades cardiovasculares, mejora la digestión, contribuye a mantener en buen estado los sistemas respiratorio, osteomuscular y circulatorio, ayuda a conciliar y a mejorar el sueño, entre otros muchos de los beneficios que pueden disfrutar las personas.

En la Universidad de las Ciencias Informáticas (UCI), las actividades físicas tienen gran relevancia dadas las características de internado del ciento por ciento de su matrícula, por lo que no es de extrañar la afluencia de una parte de estos a los campos deportivos y ciclo vías que las circundan. Como en todas las universidades de Cuba, la Educación Física (EF) forma parte del currículo de asignaturas que se imparten durante los primeros años de la carrera. Al comienzo de cada semestre, se aplican las pruebas de eficiencia física iniciales (diagnóstico), con el objetivo de evaluar los niveles físicos que presentan los estudiantes, estas pruebas de eficiencia física son:

- Planchas
- Abdominales
- Salto vertical
- Carrera de resistencia
- Velocidad

Los datos recolectados en el diagnóstico inicial son utilizados para planificar la actividad física de cada grupo durante el resto del semestre.

Debido al gran volumen de información que generan los resultados de la aplicación de estas pruebas y que se encuentran solo en formato físico, en forma de planillas, el procesamiento de la información se hace difícil y por ende la decisión sobre cuál es la mejor alternativa para desarrollar las aptitudes físicas, motrices básicas y deportivas de cada estudiante. El modo de almacenamiento de esta información puede producir la pérdida o deterioro de la misma, además de esta manera se torna engorroso llevar el control de cada una de las pruebas que se realizan a cada grupo de estudiantes, lo que dificulta la planificación y el seguimiento de las actividades físicas.

Teniendo en cuenta la situación problémica anteriormente descrita, se identifica el siguiente **problema de investigación:** ¿Cómo contribuir a mejorar el proceso de gestión de resultados de las pruebas de eficiencia física en la Universidad de las Ciencias Informáticas?

Con el fin de darle solución al problema se define como **objeto de estudio:** Sistemas de gestión de información (SGI).

Se concibió como **campo de acción**: Sistemas de gestión de información en el área del deporte.

El **objetivo general** de la investigación es: Elaborar un Sistema de gestión de información para los resultados de las pruebas de eficiencia física en la Universidad de las Ciencias Informáticas.

Se define como **idea a defender:** La elaboración de un Sistema de gestión de información contribuirá a mejorar el proceso de gestión de resultados de las pruebas de eficiencia física en la Universidad de las Ciencias Informáticas.

Los **objetivos específicos** de la investigación son los siguientes:

- Elaborar el marco teórico de la investigación sobre el SGI para los resultados de las pruebas de eficiencia física en la UCI.
- Realizar el análisis y diseño del SGI para los resultados de las pruebas de eficiencia física en la UCI.
- Implementar el SGI para los resultados de las pruebas de eficiencia física en la UCI.
- Validar la solución propuesta mediante las pruebas de *software*.

Las **tareas de la investigación** son:

- Revisión bibliográfica de los diferentes Sistemas de gestión de información (SGIs).

- Valoración de las nuevas tendencias relacionadas con los SGIs.
- Diagnóstico a los profesores de la asignatura de EF para la descripción del proceso de recopilación de información de las pruebas de eficiencia física en la UCI.
- Elaboración del estado del arte de los SGIs los principales conceptos y elementos teóricos del tema a tratar.
- Identificación de la metodología para la elaboración del SGI para los resultados de las pruebas de eficiencia física en la UCI.
- Identificación de las tecnologías y herramientas para la elaboración del SGI para los resultados de las pruebas de eficiencia física en la UCI.
- Descripción de los artefactos generados durante el desarrollo de la solución propuesta.
- Implementación de las funcionalidades del SGI.
- Realización de las pruebas al *software*.

Para la realización de la investigación se emplearon los siguientes métodos:

Métodos Teóricos:

- Analítico-Sintético: se empleó para realizar un análisis y síntesis de la teoría de los Sistemas de gestión de información.
- Análisis histórico-lógico: se utiliza durante todo el desarrollo de la investigación, incluye el estudio de los temas relacionados con los SGIs y la exploración de las posibles herramientas que se emplearán para la solución.

Métodos Empíricos:

- Entrevista[1]: se empleó para realizar las entrevistas a los profesores de EF acerca de las pruebas de eficiencia física en la UCI.

Resultados Esperados:

Como aporte del trabajo de diploma se espera obtener un SGI capaz de gestionar la información resultante de las pruebas de eficiencia física que se les aplican a los estudiantes en la asignatura EF.

Para lograr el cumplimiento de los objetivos propuestos, el documento de tesis se estructura en:

Capítulo 1: "Fundamentación teórica"

[1] Ver Anexo 2 Entrevista al profesor de Educación Física acerca de las pruebas de eficiencia física en la UCI.

En este capítulo se abordan los aspectos teóricos que dan sustento a la investigación. Se concreta un estudio del estado del arte de los SGIs existentes. Se realiza además un análisis detallado de las metodologías de desarrollo, herramientas y tecnologías a utilizar en el proceso de desarrollo del sistema.

Capítulo 2: "Propuesta de solución"

En este capítulo se definen las principales características que debe cumplir el sistema a desarrollar, es decir, se precisan las funcionalidades y características que debe cumplir, así como el diseño y la estructuración del mismo. Se generan además los artefactos que propone la metodología escogida.

Capítulo 3: "Implementación y Prueba"

Se definen los patrones utilizados, además del modelo de datos del SGI. Se implementan las funcionalidades identificadas a través de las tareas de la ingeniería describiéndose cada una de ellas. Se detallan las pruebas realizadas al *software* con el objetivo de asegurar la calidad y eficiencia de la solución.

Bibliografía

Está formada por documentos y artículos que se emplearon para el desarrollo de la investigación.

Glosario de términos

Muestra un catálogo de palabras con definiciones o explicaciones de cada una de ellas.

Anexos

Se muestran algunos artefactos generados en el transcurso del desarrollo de la solución.

Capítulo 1: Fundamentación teórica

1.1 Introducción

En el presente capítulo se investigan definiciones relacionadas con los Sistemas de gestión de información (SGIs), así como sus características, ventajas y desventajas. Además se hace un estudio de estos sistemas a nivel internacional y nacional. Se analizan las herramientas, tecnologías, metodologías de desarrollo y los lenguajes de programación a utilizar para la realización de la aplicación, que pudieran darle cumplimiento al objetivo general de la investigación.

1.2 Sistema de gestión de información

La gestión de la información es un proceso realizado por un conjunto de actividades que permiten la obtención de información, lo más pertinente, relevante y económica posible, para ser usada en el desarrollo y el éxito de una organización (1).

Los SGIs son un conjunto de elementos interrelacionados con el fin de apoyar las actividades de una organización o empresa. Surgen con el objetivo de brindar un conjunto de funcionalidades que permiten la obtención de la información, se caracterizan por ser:

- Fiable: sin errores.
- Relevante: al nivel de detalle que se precise, ni más ni menos.
- Oportuno: en el momento en que se necesita.
- Selectivo: sólo la información necesaria, obviando el resto.
- Flexible: fácilmente adaptable a los cambios que se requieran.

Estos sistemas realizan cuatro actividades básicas: entrada, almacenamiento, procesamiento y salida de información (2):

- **Entrada de información**: es el proceso mediante el cual toma los datos que requiere para procesar la información. Las entradas pueden ser manuales o automáticas. Las manuales son aquellas que se proporcionan en forma directa por el usuario, mientras que las automáticas son datos o información que provienen o son tomados de otros sistemas o módulos.
- **Almacenamiento de información**: el almacenamiento es una de las actividades o capacidades más importantes que tiene una computadora, ya que a través de esta

propiedad el sistema puede recordar la información guardada en la sección o proceso anterior.

- **Procesamiento de información:** es la capacidad del sistema para efectuar cálculos de acuerdo con una secuencia de operaciones. Estos cálculos pueden efectuarse con datos introducidos recientemente en el sistema o con datos que están almacenados. Esta característica de los sistemas permite la transformación de datos fuente en información que puede ser utilizada para la toma de decisiones, lo que hace posible, entre otros aspectos, que un tomador de decisiones genere una proyección financiera a partir de los datos que contiene un estado de resultados.
- **Salida de información:** es la capacidad de extraer la información procesada. La salida puede constituir la entrada a otro SGI o módulo. En este caso, también existe una interfaz automática de salida.

Los SGIs constituyen, no sólo soportes de los negocios, sino, un instrumento de ventajas competitivas sostenibles al permitir gestionar los activos tangibles e intangibles y convertirse en una herramienta integral de control (2).

Ventajas

Los sistemas de gestión ayudan a lograr las metas y objetivos de una organización mediante una serie de estrategias, que incluyen la optimización de procesos, el enfoque centrado en la gestión y el pensamiento disciplinado (3), por lo que la implementación de estos sistemas trae como ventajas:

- Reducir los costos de la entidad.
- Aumentar la satisfacción de clientes.
- Lograr mejoras continuas en cuanto a los procesos de la empresa.
- Gestionar los riesgos financieros.
- Facilitar la administración de procesos orientados a los objetivos.
- Reducir costos por documentación y actividades.
- Optimizar tiempos de procesos.
- Mejorar comunicación interna.
- Dar mayor coherencia a la organización.

Desventajas

- **Tiempo de inactividad**: es la posibilidad de que el sistema no funcione correctamente. Si el sistema no funciona por la razón que sea, no se puede acceder a la información almacenada en esos equipos. Sin los medios necesarios para acceder a la información del cliente, una empresa puede ser incapaz de cumplir con los pedidos, responder a las preguntas de los clientes u otra manera de hacer negocios.
- **Datos accesibles**: todos los datos se almacenan en una ubicación central y es fácilmente accesible. Este método es una ventaja cuando la información se consulta por la razón correcta, pero también hace que sea más fácil para los piratas informáticos robar la identidad de los consumidores y usarla como propia.
- **Requerimientos de entrenamiento y conocimientos**: toma tiempo y esfuerzo crear y tener a todos los usuarios completamente capacitados, con conocimientos sobre cómo utilizar correctamente el sistema.
- **El ingreso de datos exactos**: la información contenida en un sistema informático se debe introducir a la computadora por una persona. Por lo tanto, la exactitud de esta entrada es esencial para asegurar la información correcta a los clientes correctos o pacientes. Una entrada incorrecta podría significar la posible confusión en una fecha posterior para la facturación u otro tipo de actividad que significan a menudo emplear mucho tiempo para buscar, evaluar y corregir (4).

1.3 Tendencias actuales de los Sistemas de gestión de información

Debido al desarrollo impetuoso de las TIC (Tecnologías de la Información y la Comunicación), los SGIs juegan un papel importante en las organizaciones o empresas tanto a nivel nacional como internacional ya que están vinculadas en todas las esferas de la sociedad (salud, educación, deporte, entre otras). Esto ha conllevado a que los SGIs muestren una serie de tendencias imparables que a continuación se explican:

Tendencia Nro. 1: Hacia la gestión de contenidos: En los SGIs la gestión de contenidos constituye un aspecto importante debido a que es una necesidad de tratar de manera global y sistemática distintos tipos de información (5):

- La información interna que se produce.
- La información que proviene de fuentes externas.
- La información pública que la organización quiere trasmitir a su entorno.

Tendencia Nro. 2: Hacia la necesidad de proceso de información: El mantenimiento y explotación de los SGIs constituyen uno de los pilares de la gestión de la información debido a están volcados hacia la construcción de grandes bases de datos corporativas en las que se registran toda la información de las organizaciones: contabilidad, facturación, recursos humanos, producción y/o clientes (6).

Tendencia Nro. 3: Hacia el reconocimiento de la tecnología como herramienta: Esta tendencia es doble, ya que por un lado, los propios desarrolladores de herramientas cuentan cada vez más con expertos que les ayuden a poner en marcha nuevas funcionalidades o adaptaciones. Por otra parte, las empresas comienzan a establecer nuevas funciones que no están en manos de los informáticos (7).

Tendencia Nro. 4: Hacia la máxima importancia de la accesibilidad: En los SGIs cada vez tienen menos importancia la gestión de los soportes o los medios en los que se recoge la información, pasando a primer plano la accesibilidad de la misma. No importa donde esté físicamente la información, lo que se requiere es que sea accesible en el momento que se necesita (7).

1.4 Estudio de sistemas similares vinculados al campo de acción

A partir del desarrollo de las tecnologías de la informática, muchas esferas como la medicina y el deporte se han beneficiado en la informatización de sus procesos. En este epígrafe se realiza un estudio de los SGIs vinculados al campo de la cultura física y el deporte.

1.4.1 Internacionales

En el ámbito internacional en la bibliografía consultada se mostró la presencia de los siguientes sistemas relacionados con el deporte:

Sistema Informático para Profesores de Educación Física (SIPREF): es un *software* argentino diseñado a la medida para Profesores en Educación Física para administrar la información de los alumnos y obtener resultados de forma automática. Calcula totales, porcentajes y promedios correspondientes a las Evaluaciones, Asistencia, Test de Aptitud Física, Índice de Masa Corporal (IMC). Estos resultados constituyen indicadores de la gestión que sirven para la toma de decisiones correctivas o de proyección hacia objetivos futuros.

SIPREF es una herramienta que cuenta con gráficos, listados y cuadros de evolución comparativos entre los distintos grupos para todas las opciones de gestión que administra (8).

Características del producto:

- Cálculo automático del IMC.
- Seguimiento de la evolución del IMC por alumno y por grupos.
- Gestión de Test de Aptitud Física.
- Cálculos de calificaciones, totales y promedios automáticos.
- Control de Asistencias.

***Woplanner*:** es un producto argentino, para todo el segmento relacionado con el deporte. Este *software* permite evaluar, planificar, generar reportes y organizar las sesiones de entrenamiento aplicables a todo tipo de deporte, sea individual o de conjunto. El sistema es altamente flexible, permitiendo realizar cambios en el funcionamiento interno, logrando una personalización para cada usuario. Con esta herramienta se obtiene enormes beneficios en el plano laboral, ya que ayuda a llevar el control de atletas de alto rendimiento y a desarrollar las capacidades físicas de cada uno de ellos (9).

1.4.2 Nacionales

En el ámbito nacional y en el contexto de la UCI se encontraron los siguientes sistemas de gestión relacionados con el deporte:

Marabana: este SGI está diseñado para llevar a cabo todo el proceso de inscripción a la carrera de maratón Marabana y gestionar los resultados del mismo. También los concursantes inscriptos tienen la posibilidad de realizar la certificación de participación. Este sistema no brinda ningún aporte a la investigación.

Sistema para la Gestión de préstamos de implementos deportivos en la Universidad de las Ciencias Informáticas: este sistema garantiza el control de forma óptima de las prestaciones de los implementos deportivos a los estudiantes en la UCI. Algunas de las funcionalidades que brinda son: gestionar tipo de implemento, gestionar el estado del implemento y gestionar préstamo de implemento deportivo.

Sistema de Gestión Deportivo para la Universidad de las Ciencias Informáticas: mejora la gestión de la información deportiva que se lleva a cabo en la universidad de forma manual, este sistema permite: llevar el control de las estadísticas de los juegos inter-facultades en la

universidad, agiliza el proceso de búsqueda de los resultados de los juegos y brinda una actualización de cómo se van desarrollando los juegos deportivos.

Sistema informático para la gestión de información del área terapéutica de la cultura física en la UCI: es un sistema informático basado en tecnologías *web* que facilita la gestión de información en el área terapéutica de la cultura física, provee la búsqueda y recuperación de información de la misma. Se mantiene un estricto control de los pacientes y terapeutas que pertenecen al área, así como de los documentos que se gestionan dentro de la misma.

El estudio de los sistemas similares al SGI para los resultados de las pruebas eficiencia física en la UCI, aportó conocimientos sobre lo importante que puede resultar el uso de estos sistemas en el área del deporte. No es posible la utilización de los SGIs estudiados pertenecientes al ámbito internacional, puesto que son *software* privativos, lo que significa que requieren de pago. Además se consultaron SGIs que presentan funcionalidades semejantes a la aplicación a desarrollar, como por ejemplo en SIPREF la funcionalidad Test de Aptitud Física. Se le pueden incorporar además distintas funcionalidades al SGI a realizar, como el control de asistencias a clases, promedio de evaluaciones y exportar a pdf los ejercicios asignados.

1.5 Metodologías de desarrollo de software

Para desarrollar el SGI se debe seleccionar una metodología de desarrollo de *software* para guiar todas las fases de desarrollo del producto, donde el impacto de elegir la metodología más adecuada para el equipo de trabajo es trascendental para el éxito del *software*.

Las metodologías de desarrollo se clasifican en tradicionales y ágiles (Ver Anexo 1 Comparación entre metodologías ágiles y tradicionales). Entre las principales metodologías tradicionales están RUP y MSF, estas se recomiendan en proyectos complejos y de larga duración, obteniendo resultados satisfactorios cuando el equipo de trabajo tiene experiencia en su aplicación. La corrección de errores es muy costosa por ser metodologías poco flexibles a los cambios, debido a que el cliente y el equipo de desarrollo no mantienen una relación directa, no siendo el caso de la presente investigación.

Con la utilización de estas metodologías se genera una documentación amplia generando artefactos necesarios e innecesarios en algunos casos. Estas características hacen engorroso el desarrollo de proyectos pequeños, ralentizando su implementación y generando

documentación que no será utilizada. Las metodologías robustas se guían por una fuerte planificación, centrando toda la atención en la documentación exhaustiva de los procesos llevados a cabo en el desarrollo, así como en las herramientas y notaciones que se utilizarán y en cumplir el plan del proyecto (10).

Para el desarrollo del SGI quedó descartada la selección de una metodología pesada, dado que sus características no se adaptan a los requerimientos de la investigación. Entre las metodologías ágiles se destacan SCRUM, Crystal y XP. Estas hacen menos énfasis en la documentación, puesto que el funcionamiento y el desarrollo del *software* son más importantes que la misma. La regla a seguir es no realizar documentos a menos que sean necesarios de forma inmediata, el cliente es muy importante el cual pasa a formar parte del equipo de desarrollo y también son muy flexibles ante los cambios y no siguen estrictamente un plan. En cuanto a metas a seguir las metodologías ágiles presentan los siguientes principios (11):

- La prioridad es satisfacer al cliente mediante tempranas y continuas entregas de *software* que le aporte un valor.
- Dar la bienvenida a los cambios. Se capturan los cambios para que el cliente tenga una ventaja competitiva.
- Entregar frecuentemente *software* que funcione desde un par de semanas a un par de meses, con el menor intervalo de tiempo posible entre entregas.
- Los clientes del negocio y los desarrolladores deben trabajar juntos a lo largo del proyecto.
- Construir el proyecto en torno a individuos motivados. Darles el entorno y el apoyo que necesitan y confiar en ellos para conseguir finalizar el trabajo.
- El diálogo cara a cara es el método más eficiente y efectivo para comunicar información dentro de un equipo de desarrollo.
- El *software* que funciona es la medida principal de progreso.
- Los procesos ágiles promueven un desarrollo sostenible. Los promotores, desarrolladores y usuarios deberían ser capaces de mantener una paz constante.
- La atención continua a la calidad técnica y al buen diseño mejora la agilidad.
- La simplicidad es esencial.
- Las mejores arquitecturas, requisitos y diseños surgen de los equipos organizados por sí mismos.

1.5.1 SCRUM

La Metodología SCRUM define un marco para la gestión de proyectos, que se ha utilizado con éxito durante los últimos 10 años. Es una metodología que no se fundamenta en el seguimiento de un plan, sino en la adaptación continua de las circunstancias de la evolución del proyecto. Su modo de desarrollo es adaptable antes de predictivo, orientado a las personas más que a los procesos y emplea un modelo de desarrollo incremental basado en iteraciones y revisiones, cada iteración es denominada *sprint* (12).

Está especialmente indicada para proyectos con un rápido cambio de requisitos. Sus principales características se pueden resumir en dos, el ciclo de vida está dividido en tres fases: planificación del *sprint*, seguimiento del *sprint* y revisión del *sprint*. El resultado de cada *sprint* es un incremento ejecutable que se muestra al cliente. La segunda característica importante se refiere a las reuniones a lo largo del proyecto, entre ellas destaca la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración (11).

1.5.2 Crystal

Se trata de un conjunto de metodologías para el desarrollo de *software* caracterizadas por estar centradas en las personas que componen el equipo y la reducción al máximo del número de artefactos producidos. Han sido desarrolladas por *Alistair Cockburn*. El desarrollo de *software* se considera un juego cooperativo de invención y comunicación, limitado por los recursos a utilizar. El equipo de desarrollo es un factor clave, por lo que se deben invertir esfuerzos en mejorar sus habilidades y destrezas, así como tener políticas de trabajo en equipo definidas. Estas políticas dependerán del tamaño del equipo, estableciéndose una clasificación por colores, por ejemplo *Crystal Clear* (3 a 8 miembros) y *Crystal Orange* (25 a 50 miembros) (11).

Ventajas

- Son apropiadas para entornos ligeros.
- Al estar diseñada para el cambio experimenta reducción de costo.
- Presenta una planificación más transparente para los clientes.
- Se definen en cada iteración cuáles son los objetivos de la siguiente.
- Permite tener una muy útil realimentación de los usuarios.

Desventajas

- Delimita el alcance del proyecto con el cliente.

1.5.3 XP

La programación extrema es una metodología de desarrollo ligero (o ágil) basada en una serie de valores y de prácticas de buenas maneras que persigue el objetivo de aumentar la productividad a la hora de desarrollar programas.

Este modelo de programación se basa en una serie de metodologías de desarrollo de *software* en la que se da prioridad a los trabajos que dan un resultado directo y que reducen la burocracia que hay alrededor de la programación.

Una de las características principales de este método de programación, es que sus ingredientes son conocidos desde el principio de la informática. Los autores de XP han seleccionado aquellos que han considerado mejores y han profundizado en sus relaciones y en cómo se refuerzan los unos con los otros. El resultado de esta selección ha sido esta metodología única y compacta. Por esto, aunque no está basada en principios nuevos, sí que el resultado es una nueva manera de ver el desarrollo de *software* (13).

A continuación se presentan algunas características de XP que se adaptan a las necesidades de un proyecto, así como a las condiciones de trabajo:

- Desarrollo iterativo e incremental: pequeñas mejoras, unas tras otras.
- Pruebas unitarias continuas, frecuentemente repetidas y automatizadas, incluyendo pruebas de regresión. Se aconseja escribir el código de la prueba antes de la codificación.
- Programación por parejas: se recomienda que las tareas de desarrollo se lleven a cabo por dos personas en un mismo puesto. Se supone que la mayor calidad del código escrito de esta manera es revisado y discutido mientras se escribe.
- Frecuente interacción del equipo de programación con el cliente o usuario. Se recomienda que un representante del cliente trabaje junto al equipo de desarrollo.
- Corrección de todos los errores antes de añadir nueva funcionalidad. Hacer entregas frecuentes.
- Refactorización del código, es decir, reescribir ciertas partes del código para aumentar su legibilidad y mantenibilidad pero sin modificar su comportamiento. Las pruebas han de garantizar que en la refactorización no se ha introducido ningún fallo.

- Propiedad del código compartida: en vez de dividir la responsabilidad en el desarrollo de cada módulo en grupos de trabajo distintos, este método promueve el que todo el personal pueda corregir y extender cualquier parte del proyecto. Las frecuentes pruebas de regresión garantizan que los posibles errores sean detectados.
- Simplicidad en el código: la programación extrema apuesta que es más sencillo hacer un código simple y tener un poco de trabajo extra para cambiarlo si se requiere, que realizar funciones complicadas y quizás nunca utilizarlas.

1.5.4 Fundamentación de la elección

Luego de un análisis de algunas de las metodologías ágiles se decide utilizar XP ya que propone una estructura de roles adaptada al proyecto, la realimentación continua entre el cliente y el equipo de desarrollo constituye un factor muy importante ya que conduce el desarrollo del trabajo y planifica el tiempo de entrega de cada iteración, el ambiente de desarrollo basado en un único local y un ordenador por parte del equipo de trabajo se ajustan perfectamente a las características de XP, además el *software* será desarrollado por 2 programadores lo cual coincide con la característica de la metodología XP programación por pareja, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. La metodología XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, donde existe un alto riesgo técnico. XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo del *software*. Promueve el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, propiciando un buen clima de trabajo (11).

1.6 Tipos de Aplicación

Para el desarrollo de una aplicación se debe estudiar qué tipo de aplicación se debe realizar, las aplicaciones pueden ser de dos formas: de escritorio o *web*. Para seleccionar qué tipo de aplicación se debe desarrollar, se tiene que analizar una serie de parámetros en correspondencia con las peticiones del cliente, estos parámetros se pueden traducir en requisitos funcionales y requisitos no funcionales del *software* a desarrollar. A continuación se muestra un estudio de los dos tipos de aplicaciones.

1.6.1 Aplicación de Escritorio

Una aplicación de Escritorio (también llamada *Desktop*) es aquella que está instalada y configurada en el ordenador de un usuario y es ejecutada directamente por su sistema operativo.

Las aplicaciones de escritorio no se conectan a un servidor *web*, se pueden adaptar a las necesidades de una empresa y su rendimiento depende de diversas configuraciones de hardware como memoria RAM o capacidad del disco duro.

Ventajas:

- Su ejecución no necesita comunicarse con un servidor *web*, sino que se realiza de forma local. Esto repercute en mayor velocidad de procesamiento y por tanto en mayores capacidades a la hora de programar herramientas más complicadas o funcionales.
- El tiempo de respuesta es rápido.
- Suelen ser muy seguras.

Desventaja:

- Son dependientes del sistema operativo en que están instaladas.
- Las instalaciones y actualizaciones se realizan de forma personalizada.

1.6.2 Aplicación Web

Se denomina aplicación *web* a aquellas aplicaciones que los usuarios utilizan accediendo a un servidor *web* a través de Internet o de una Intranet mediante un navegador *web* (14).

Las aplicaciones *web* se caracterizan por dar soluciones puntuales expresadas en un *software* navegable por lo que los usuarios acceden a la aplicación sin la necesidad de su instalación.

Ventajas:

- Multiplataforma: el sistema puede funcionar en múltiples plataformas como Windows o Linux.
- Acceso inmediato y desde cualquier lugar: no necesitan ser descargadas, ni instaladas. Pueden ser accedidas desde cualquier ordenador conectado a la red.
- Son fáciles de actualizar y mantener.

- Menos requerimientos de hardware para su funcionamiento.
- Seguridad en la información que manejan.

Desventaja:

- Es necesaria la conexión a una red, la comunicación constante con el servidor que ejecuta la aplicación establece una dependencia a una buena conexión. Además, el servidor debe tener las prestaciones necesarias para ejecutar la aplicación de manera fluida, no solo para un usuario sino para todos los que la utilicen de forma concurrente.

1.6.3 Fundamentación de la elección

Para el desarrollo del SGI se escogió el uso de una aplicación *web* debido a que la misma permitirá que se acceda desde cualquier sistema operativo. Los profesores y el administrador de la aplicación podrán conectarse desde un ordenador que tenga conexión a la red de la universidad y un navegador *web* instalado, donde podrán gestionar los resultados de las pruebas de eficiencia física en cualquier momento, por lo que el sistema brinda la posibilidad de ser usado desde cualquier sitio.

1.7 Lenguajes de programación

Un lenguaje de programación dentro de la informática es simplemente la forma que posee de comunicarse un programador con los dispositivos *hardware* y *software* existentes en un ordenador, que en su conjunto le permite crear programas.

1.7.1 Lenguajes del lado del servidor

El uso de los lenguajes de programación depende de los intereses del cliente y el tipo del producto a desarrollar, ya que cada uno brinda ventajas y funcionalidades diferentes. Su selección está en dependencia de las necesidades del producto. En la actualidad se puede encontrar la existencia de varios lenguajes de programación que nos permitan el desarrollo de una aplicación. Los lenguajes de programación más utilizados en el desarrollo de aplicaciones *web* son Java y PHP.

Java

Java surgió en 1991 cuando un grupo de ingenieros de *Sun Microsystems* trató de diseñar un nuevo lenguaje de programación destinado a correr sobre cualquier plataforma, desde

computadoras hasta efectos electrodomésticos. Con este fin fueron de los primeros en introducir el concepto de máquina virtual con el objetivo de que sus aplicaciones fueran totalmente independientes de la Unidad Central de Procesamiento (CPU, por sus siglas en inglés) que las procesará. *Sun Microsystems* describe a Java como simple, orientado a objetos, distribuido, interpretado, robusto, seguro, de arquitectura neutra, portable, de altas prestaciones, multitarea y dinámico (15).

Java permite agrupar en estructuras encapsuladas tanto sus datos como los métodos o funciones que manipulan estos. Fue diseñado para crear *software* altamente fiable, lo que proporciona a los programadores la disminución de errores y la liberación explícita de memoria. La seguridad radica en las barreras impuestas en el lenguaje y en el sistema de ejecución en tiempo real. Proporciona una colección de clases para su uso en aplicaciones de red, que permiten establecer y aceptar conexiones con servidores o clientes remotos, para así facilitar la creación de aplicaciones distribuidas.

Ventajas

Dentro de las ventajas que presenta este lenguaje, es que es multiplataforma, lo que se traduce que se ejecuta en la mayoría de los sistemas operativos, inclusive en los móviles. Otra ventaja es que Java es un *software* de distribución libre, puesto que no es necesario pagar su licencia para su utilización. Este lenguaje es muy completo y poderoso, ya que posee una librería y utilidades muy completas que facilitan la programación (16).

Desventajas

Una de las desventajas que presenta Java es que puede ser un lenguaje de ejecución lento, debido al uso de la máquina virtual de Java, a diferencia de otros lenguajes de programación de más bajo nivel como lo es “C”, su velocidad de ejecución disminuye drásticamente al compararse con este lenguaje. Otra desventaja es que Java es considerado un lenguaje difícil de aprender, esto se debe a su compleja sintaxis.

ASP

El lenguaje ASP (*Active Server Pages*), es un lenguaje de programación de servidores para generar páginas *web* dinámicamente. Se conocen cuatro versiones de este lenguaje las 1.0, 2.0, 3.0 y la ASP.NET que se la conoce como la ASP Clásica.

El lenguaje de programación ASP nace aproximadamente en el año 1996, lo que ofrecía de nuevo este lenguaje era que se podía crear una página *web* en la que se pudiese programar para que nos ofreciera unos determinados datos. Esto era una gran ventaja porque en aquella época solo se podía dibujar una tabla e incluir unos pocos datos.

Posteriormente se crea el lenguaje ASP.Net que es un lenguaje mucho más complejo que el original ASP. Este lenguaje permite separar en las páginas *web* la parte de diseño que contiene la página, no interviniendo para nada el código HTML. Así el trabajo de los diseñadores y programadores es mucho más sencillo. Cada cual se ocupa de su parte del trabajo dentro de la página *web* sin interferir en la parte de otro.

El ASP es un lenguaje de programación para servidores, es adecuado para acceso a bases de datos, lectura de ficheros. Se vale de dos lenguajes de Script, como son el VBScript y el *JavaScript* para que lo que programemos con el ASP sea visible (17).

Ventajas

El lenguaje de programación ASP, nos ofrece las siguientes ventajas: separar el código *html* del ASP, mayor facilidad para realizar cambios, fácil instalación y funcionamiento y mayor protección del código (17).

Desventajas

Las principales desventajas que presenta este lenguaje son:

- Código desorganizado.
- Se necesita escribir mucho código para realizar funciones sencillas.
- Tecnología propietaria.
- Hospedaje de sitios *web* costosos.
- Es muy bueno para páginas que manejan gran cantidad de datos, no para una página que vaya a utilizarse para uso personal.

PHP

Es un lenguaje de programación utilizado para la creación de sitio *web*. PHP es un acrónimo recursivo que significa “PHP *Hypertext Pre-processor*”, (inicialmente se *llamó Personal Home Page*). Surgió en 1995, desarrollado por *PHP Group.*

PHP es un lenguaje de script interpretado en el lado del servidor utilizado para la generación de páginas *web* dinámicas, embebidas en páginas *HTML* y ejecutadas en el servidor. PHP no

necesita ser compilado para ejecutarse. Para su funcionamiento necesita tener instalado Apache o IIS con las librerías de PHP. La mayor parte de su sintaxis ha sido tomada de C, Java y Perl con algunas características específicas. Los archivos cuentan con la extensión .php.

Ventajas

- Muy fácil de aprender.
- Se caracteriza por ser un lenguaje muy rápido.
- Soporta en cierta medida la orientación a objetos, clases y herencia.
- Es un lenguaje multiplataforma: Linux, Windows, entre otros.
- Capacidad de conexión con la mayoría de los manejadores de base de datos: MySQL, PostgreSQL, Oracle, MS SQL Server, entre otras.
- Capacidad de expandir su potencial utilizando módulos.
- Posee documentación en su página oficial, donde incluye descripción y ejemplos de cada una de sus funciones.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Incluye gran cantidad de funciones.
- No requiere definición de tipos de variables ni manejo detallado del bajo nivel.

Desventajas

- Se necesita instalar un servidor *web*.
- Todo el trabajo lo realiza el servidor y no delega al cliente. Por tanto puede ser más ineficiente a medida que las solicitudes aumenten.
- La legibilidad del código puede verse afectada al mezclar sentencias HTML y PHP.
- La programación orientada a objetos es aún muy deficiente para aplicaciones grandes.
- Dificulta la organización por capas de la aplicación.

Fundamentación de la elección

Para la realización del SGI se seleccionó el lenguaje PHP ya que es un lenguaje de código abierto que está diseñado esencialmente para el desarrollo web y que puede ser incrustado en HTML. Es un lenguaje multiplataforma, fácil de aprender y muy rápido por lo que consume pocos recursos, contiene una amplia documentación. Tiene la capacidad de conectarse fácilmente con el SGBD *PostgreSQL* e incluye gran cantidad de funciones, por ejemplo, para la validación de los campos.

1.7.2 Lenguajes de marcado

Un lenguaje de marcado es totalmente independiente del servidor, lo cual permite que la página pueda ser albergada en cualquier sitio. Dentro de estos se encuentran HTML 5(*Hyper Text Markup Language*), *JavaScript* y CSS (*Cascading Style Sheets*).

1.7.2.1 Hojas de estilo en cascada (CSS 3)

CSS es un lenguaje de hojas de estilos creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y XHTML. CSS es la mejor forma de separar los contenidos y su presentación y es imprescindible para crear páginas *web* complejas.

Separar la definición de los contenidos y la definición de su aspecto presenta numerosas ventajas, ya que obliga a crear documentos HTML/XHTML bien definidos y con significado completo (también llamados *"*documentos semánticos*"*). Además, mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite visualizar el mismo documento en infinidad de dispositivos diferentes.

El lenguaje CSS se utiliza para definir el aspecto de cada elemento: color, tamaño y tipo de letra del texto, separación horizontal y vertical entre elementos, posición de cada elemento dentro de la página, entre otros (18).

1.7.2.2 Javascript 1.5

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas *web* dinámicas. Técnicamente, *JavaScript* es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con *JavaScript* se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios.

Además posibilita la creación de efectos llamativos en las páginas *web* e interacción con el usuario. El navegador (*browser*) del cliente es el encargado de interpretar las instrucciones *Javascript* y ejecutarlas para realizar estas acciones, de modo que el mayor recurso con que cuenta este lenguaje es el navegador. Desde su aparición, *JavaScript* siempre fue utilizado de forma masiva por la mayoría de sitios de Internet. La aparición de *Flash* disminuyó su popularidad, ya que *Flash* permitía realizar algunas acciones imposibles de llevar a cabo mediante *JavaScript*. Sin embargo, la aparición de las aplicaciones *AJAX* programadas con

JavaScript le ha devuelto una popularidad sin igual dentro de los lenguajes de programación *web* (19).

1.7.2.3 HTML5

El HTML5 es la quinta revisión del lenguaje de programación básico de la *World Wide Web*, el HTML. Esta nueva versión pretende remplazar al actual XHTML, corrigiendo problemas con los que los desarrolladores *web* se encuentran, así como rediseñar el código actualizándolo a nuevas necesidades que demanda la *web* de hoy en día.

A diferencia de otras versiones de HTML, los cambios en HTML5 comienzan añadiendo semántica y accesibilidad implícitas, especificando cada detalle y borrando cualquier ambigüedad (20).

HTML 5 incluye novedades significativas en diversos ámbitos. No sólo se trata de incorporar nuevas etiquetas o eliminar otras, sino que supone mejoras en áreas que hasta ahora quedaban fuera del lenguaje y para las que se necesitaba utilizar otras tecnologías (21), a continuación se pueden apreciar algunas de las ventajas de HTML 5 utilizadas:

- **Estructura del cuerpo**: La mayoría de las *web*s tienen un formato común, formado por elementos como cabecera, pie y navegadores. HTML 5 permite agrupar todas estas partes de una *web* en nuevas etiquetas que representarán cada uno de las partes típicas de una página.
- **Bases de datos locales**: El navegador permite el uso de una base de datos local, con la que se podrá trabajar en una página *web* por medio del cliente y a través de una API. Son utilizadas para almacenar grandes cantidades de información, lo que permitirá la creación de aplicaciones *web* que funcionen sin necesidad de estar conectados a Internet.
- **Fin de las etiquetas de presentación**: Todas las etiquetas que tienen que ver con la presentación del documento, es decir, que modifican estilos de la página, serán eliminadas. La responsabilidad de definir el aspecto de una *web* correrá a cargo únicamente de CSS.

1.8 Herramientas

En la elaboración de un SGI se deben utilizar diferentes herramientas y es de mucha importancia la sabia elección de cada una de ellas, para garantizar el mejor funcionamiento posible.

1.8.1 Servidores Web

1.8.1.1 Internet Information Services

Internet *Information Services* o IIS es un servidor *web* y un conjunto de servicios para el sistema operativo *Microsoft Windows*. Internet *Information Server* tiene servicios de *software* que admiten la creación, configuración y administración de sitios *web*. Los servicios de *Microsoft* Internet *Information Server* incluyen el protocolo de transferencia de noticias a través de la red (NNTP), el protocolo de transferencia de archivos (FTP) y el protocolo simple de transferencia de correo (SMTP).

IIS permite autenticación robusta y segura de los usuarios, así como comunicaciones seguras vía SSL; además podemos crear contenido dinámico utilizando los componentes y secuencias de comandos del servidor para crear contenido dinámico independiente del explorador mediante páginas Active Server (ASP) (22).

Ventajas IIS (23)

- Fácil de usar.
- ASP preparado en la instalación por defecto.
- Configuración gráfica y en línea de comandos.

Desventajas IIS (23)

- Multitud de fallos de seguridad, como por ejemplo parseo de los nombres de archivo con la extensión punto y coma, es decir ";.jpg" a un archivo .asp, por ejemplo, los sistemas que simplemente analizan la ejecutabilidad del código basándose en su terminación pueden ser engañados; un archivo llamado "malicioso.asp;jpg" podría ser ejecutado como un archivo .asp.
- La mayoría de funcionalidad extra debe ser comprada separadamente.

Características de IIS (23)

- IIS tiene la forma de asegurar los datos es mediante SSL (Secure Sockets Layer). Esto proporciona un método para transferir datos entre el cliente y el servidor de forma segura, permitiendo también que el servidor pueda comprobar al cliente antes de que inicie una sesión de usuario.
- La autenticación implícita que permite a los administradores autenticar a los usuarios de forma segura a través de servidores de seguridad y proxy.
- IIS también es capaz de impedir que aquellos usuarios con direcciones IP conocidas obtengan acceso no autorizado al servidor, permitiendo especificar la información apropiada en una lista de restricciones.

1.8.1.2 Cherokee

Cherokee es un servidor *web* multiplataforma, de licencia GNU, cuyo principal objetivo es ser rápido y funcional, sin dejar de ser fácil su instalación y su posterior administración. Este servidor *web* implementa una librería para dotar a toda clase de aplicaciones de servicios *web* de una forma fácil y rápida. En su desarrollo se realiza un esfuerzo especial en mantener un núcleo reducido de forma tal que pueda utilizarse en sistemas empotrados y efectuar todas las funcionalidades como módulos cargables en tiempo de ejecución (24).

Características de *Cherokee*

- La alta eficiencia y una arquitectura lo suficientemente flexible como para poder escalar a servidores SMP son características de *Cherokee* que pueden suponer un paso adelante respecto a los servidores *web* libres existentes.
- Una de sus principales características es que tiene la librería: *libcherokee*, la cual implementa las características básicas de un servidor *web*, facilitando cargar desde módulos muchas otras. Esta arquitectura modular se seleccionó para admitir cargar y ejecutar solamente las partes y funcionalidades que son necesarias en cada caso específico. De esta forma, se ahorran recursos, se aumenta la seguridad (menos código en ejecución implica menos posibilidad de existir un bug en él) y se disminuye ligeramente la carga del servidor *web*.
- Existen tres grandes grupos de módulos cargables: *handlers, encoders, validators*.

Desventajas

- Una de las desventajas que tiene es que si lo quieres configurar desde un entorno textual, tienes que tener amplios conocimientos, ya que todo se encuentra en un único fichero de configuración, con una configuración específica de *Cherokee*.
- Otra desventaja es la poca documentación que existe. *Cherokee* puede ser usado como servidor *web* en aquellos lugares donde su administrador tenga amplios conocimientos de la materia (25).

1.8.1.3 Apache 2.2.22

Apache es un servidor *web* HTTP de código abierto, para plataformas Unix (BSD, GNU/Linux), *Microsoft Windows, Macintosh* y otras, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual. Apache es usado principalmente para enviar páginas *web* estáticas y dinámicas en la *World Wide Web*. Muchas aplicaciones *web* están diseñadas asumiendo como ambiente de implantación a Apache (26).

Apache es una muestra, al igual que el sistema operativo Linux (un Unix desarrollado inicialmente para PC), de que el trabajo voluntario y cooperativo dentro de Internet es capaz de producir aplicaciones de calidad profesional difíciles de igualar.

La licencia Apache es una descendiente de la licencias BSD, no es GPL. Esta licencia permite modificar o reescribir el código fuente (incluso forks y productos propietarios) siempre que se les reconozca su trabajo (27).

Algunas ventajas que brinda del servidor *web* Apache (27):

- Funciona en una multitud de Sistemas Operativos, lo que lo hace prácticamente universal.
- Apache es una tecnología gratuita, de código fuente abierto. El hecho de ser gratuita es importante pero no tanto como que se trate de código fuente abierto. Esto le da una transparencia al *software* de manera, que permite ver qué es lo que se está instalando como servidor.
- Apache es un servidor altamente configurable de diseño modular. Es muy sencillo ampliar las capacidades de este servidor. Actualmente existen muchos módulos para Apache que son adaptables a este, para que se instalen cuando se necesiten.
- Apache trabaja con gran cantidad de Perl, PHP y otros lenguajes de *script*. Perl destaca en el mundo del *script* y Apache utiliza su parte del pastel de Perl tanto con soporte CGI

como con soporte *modperl*. También trabaja con Java y páginas jsp.; teniendo todo el soporte que se necesita para tener páginas dinámicas.

- Apache permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. Es posible configurar Apache para que ejecute un determinado *script* cuando ocurra un error en concreto.

1.8.1.4 Fundamentación de Elección

Después del estudio realizado se seleccionó Apache como servidor *web* porque soporta el lenguaje PHP al igual que IIS, pero debido a que IIS es *software* propietario y que solamente puede ser utilizado en Windows. Además la licencia de IIS requiere de pago, todo lo contrario a Apache que es un *software* multiplataforma y puede ser utilizado en diferentes sistemas operativos, como por ejemplo Linux que es el sistema usado en el desarrollo de la investigación. Además su licencia permite todo tipo de acceso al código fuente. Apache presenta mucha más documentación que *Cherokee* dado que esta es unas de las desventajas que muestra este servidor. También hay que tener conocimientos avanzados para utilizarlo, la mayoría de las configuraciones se realiza desde una interfaz *web*, siendo poco probable en un entorno de producción y a la vez siendo otra desventaja.

1.8.2 Sistema gestor de base de datos

Un Sistema Gestor de Bases de Datos (SGBD) es un sistema de *software* que permite la definición de bases de datos; así como la elección de las estructuras de datos necesarios para el almacenamiento y búsqueda de los datos, ya sea de forma interactiva o a través de un lenguaje de programación (28). Para el desarrollo de la aplicación se realizó un estudio sobre los sistemas gestores de base de datos *MySQL*, *PostgreSQL* y *Oracle*. A continuación se muestra el estudio realizado.

1.8.2.1 MySQL 5.5.8

MySQL es un SGBD Relacional, licenciado bajo la GPL de la GNU. Su diseño multihilos le permite soportar una gran carga de forma muy eficiente. Actualmente existen infinidad de librerías y otras herramientas que posibilitan su uso a través de gran cantidad de lenguajes de programación, además de su fácil instalación y configuración. Las principales características de este gestor de bases de datos son las siguientes (29):

- Soporta gran cantidad de tipos de datos para las columnas.

- Gran portabilidad entre sistemas.
- Soporta hasta 32 índices por tabla.
- Gestión de usuarios y contraseñas, manteniendo un buen nivel de seguridad en los datos.
- Es versátil ya que trabaja tanto con sistemas operativos basados en Unix como con el sistema operativo de *Microsoft*.

1.8.2.2 Oracle

Es una aplicación de *software* de bases de datos proporcionada por *Oracle* para su uso en los negocios. *Oracle* puede ejecutarse en todas las plataformas. Soporta todas las funciones que se esperan de un servidor. Presenta un lenguaje de diseño de bases de datos muy completo (PL/SQL), que permite implementar diseños activos, con *triggers* y procedimientos almacenados (30).

Ventajas

- Puede ejecutarse en todas las plataformas, desde una PC hasta un súper computador.
- Permite el uso de particiones para la mejora de la eficiencia, de replicación e incluso ciertas versiones admiten la administración de bases de datos distribuidas.
- Hace que sea posible encontrar las relaciones en los datos, si estás trabajando con grandes cantidades de datos regularmente, esto puede ahorrarte tiempo y encontrar las relaciones que necesitas en los datos.
- También tiene la habilidad de descubrir patrones en los datos a través del tiempo.

Desventajas

- Un inconveniente que presenta es que aún no aparecen buenos libros que asesoren a los usuarios en cuanto a la instalación y administración.
- Otro de los inconvenientes es su precio, sus licencias son excesivamente caras.

1.8.2.3 PostgreSQL 9.1

PostgreSQL es un SGBD objeto-relacional, utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Desde su creación, hace más de 16 años la estabilidad, potencia, robustez, facilidad de administración e implementación de estándares han sido las características que más se han tenido en cuenta. *PostgreSQL* funciona muy bien con grandes cantidades de datos y una alta concurrencia de

usuarios accediendo a la vez al sistema. Es de código abierto, al igual que todo el *software* libre. Cuenta con dos ventajas claras: un código fuente optimizado que puede ser modificado y adaptado, y una baja inversión por implementación, ya que no existen costos por licencia (31).

Entre sus principales características están (31):

- Incluye herencia entre tablas (aunque no entre objetos, ya que no existen), por lo que a este gestor de base de datos se le incluye entre los gestores objeto-relacionales.
- Permite la gestión de diferentes usuarios, como también los permisos asignados a cada uno de ellos.
- Posee características significativas del motor de datos, entre las que se pueden incluir las subconsultas y los valores por defecto.
- Incorpora una estructura de datos *array*.
- Permite la declaración de funciones propias, así como la definición de disparadores.
- Soporta el uso de índices, reglas y vistas.

1.8.2.4 Fundamentación de la elección

Después del estudio realizado de los SGBD anteriormente mencionados se seleccionó *PostgreSQL* para llevar a cabo el desarrollo de la aplicación *web* ya que es un *software* de código abierto, posee una gran escalabilidad, es capaz de ajustarse al número de unidades de procesamiento y a la cantidad de memoria que posee el sistema de forma óptima. Es multiplataforma y soporta el lenguaje de programación PHP. Además otra de las ventajas de *PostgreSQL* es que es libre, lo que no requiere pago de licencia, mientras que *MySQL* es una herramienta de carácter propietario en sus últimas versiones, lo cual imposibilita la obtención de su licencia. También queda descartado el uso de *Oracle* ya que sus licencias son caras y su curva de aprendizaje es baja.

1.8.3 NetBeans 7.3

NetBeans IDE es un entorno de desarrollo, una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java, pero puede servir para cualquier otro lenguaje de programación, como es el caso del lenguaje PHP, seleccionado para la solución. Existe además un número importante de módulos para extender el *NetBeans* IDE. *NetBeans* IDE es un producto libre y gratuito sin restricciones de uso.

El código fuente está disponible para su reutilización de acuerdo con la *Common Development and Distribution License* (CDDL) v1.0 *and the GNU General Public License* (GPL) v2 (32).

1.9 Herramienta para el desarrollo de la aplicación web

Para la selección de la herramienta a utilizar en el desarrollo de la aplicación *web* se realizó un estudio de los CMS y *frameworks*. A continuación se muestra la investigación realizada.

1.9.1 CMS

CMS (*Content Management System*) es un programa que permite crear una estructura de soporte para la creación y administración de contenidos, principalmente en páginas *web*, por parte de los administradores, editores y demás roles, sin necesidad de grandes conocimientos informáticos (33).

Consiste en una interfaz que controla una o varias bases de datos donde se aloja el contenido del sitio. El sistema permite manejar de manera independiente el contenido y el diseño. Así, es posible manejar el contenido y darle en cualquier momento un diseño distinto al sitio sin tener que darle formato al contenido de nuevo, además de permitir la fácil y controlada publicación en el sitio a varios editores proporcionando un entorno que posibilita la actualización, mantenimiento y ampliación del portal *web* con la colaboración de múltiples usuarios (34).

La utilización de un CMS para el desarrollo *web* brinda numerosas funcionalidades y facilidades a los desarrolladores, destacándose las que a continuación se muestran: (35)

Inclusión de nuevas funcionalidades en la *web*: Esto puede ser tan simple como incluir un módulo realizado por terceros sin que eso suponga muchos cambios en la *web*.

Mantenimiento de gran cantidad de páginas: En una *web* con muchas páginas hace falta un sistema para distribuir los trabajos de creación, edición y mantenimiento con permisos de acceso a las diferentes áreas. También se tienen que gestionar los metadatos de cada documento, las versiones, la publicación y caducidad de páginas y los enlaces rotos, entre otros aspectos.

Reutilización de objetos o componentes: Un CMS permite la recuperación y reutilización de páginas, documentos y en general de cualquier objeto publicado o almacenado.

Páginas interactivas: Las páginas estáticas llegan al usuario exactamente como están almacenadas en el servidor *web*. En cambio, las páginas dinámicas no existen en el servidor tal

como se reciben en los navegadores, sino que se generan según las peticiones de los usuarios.

Cambios del aspecto de la *web*: Los CMS facilitan cambios en el diseño con la utilización del estándar Hojas de Estilo en Cascada (*Cascading Style Sheets*: CSS) con lo que se consigue la independencia entre presentación y contenido.

Consistencia de la *web*: La consistencia en una *web* no quiere decir que todas las páginas sean iguales, sino que hay un orden (visual) en vez de caos. Los CMS pueden aplicar un mismo estilo en todas las páginas con el mencionado CSS y aplicar una misma estructura mediante patrones de páginas.

Control de acceso: Controlar el acceso a un sitio *web* significa gestionar los diferentes permisos a cada área de la *web*, aplicados a grupos (roles) o individuos (usuarios).

1.9.2 Framework

Un *framework* es una estructura de soporte definida a partir de la cual un proyecto de *software* puede ser organizado y/o desarrollado. Típicamente, un *framework* incluye soporte de programas, bibliotecas y un lenguaje interpretado para ayudar a desarrollar y unir los diferentes componentes de un proyecto (36).

Un *framework* simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. Además proporciona estructura al código fuente, forzando al desarrollador a crear código más legible y más fácil de mantener. También facilita la programación de aplicaciones, ya que encapsula operaciones complejas en instrucciones sencillas (37).

Entre las ventajas de usar un *framework*, se pueden contar (38):

- Facilita integrar a otras personas a los proyectos pues comparten convenciones de desarrollo comunes.
- No hay que preocuparse por mantener actualizadas las distintas partes, generalmente los *frameworks* son abiertos y soportados por una comunidad que actualizan la funcionalidad y corrigen los *bugs* de manera sostenida.
- No reinventar la rueda, puesto que se aprovechan los componentes existentes aumentando la velocidad de desarrollo.
- Reducción en el tiempo de desarrollo de nuevas aplicaciones.

- Reducción del costo de mantenimiento.
- Mayor nivel de confiabilidad (comparado con escribir código nuevo), en la medida que hay reuso y el *framework* se estabiliza.
- Abstracción de URLs (Localizador de Recursos Uniforme) y sesiones, porque no es necesario manipular directamente las URLs ni las sesiones, el *framework* ya se encarga de hacerlo.
- Fácil acceso a datos. Incluyen las herramientas e interfaces necesarias para integrarse con herramientas de acceso a datos.
- Autenticación y control de acceso, pues incluyen mecanismos para la identificación de usuarios mediante *login* y *password* que permiten restringir el acceso a determinadas páginas a determinados usuarios.

Como desventajas se pueden nombrar:

- Agrega código adicional que no es elaborado por el programador de la aplicación.
- Hay que invertir tiempo en aprender a usarlos.
- En algunos casos una aplicación desarrollada con un *framework* puede ser más lenta (en cuanto a rendimiento) que una diseñada y desarrollada desde cero.

1.9.3 Fundamentación de la elección

Después del estudio realizado, se seleccionó para el desarrollo de la aplicación un *framework*, debido a que estos brindan un mayor número de funcionalidades y mejores prácticas que los CMS, para el desarrollo de aplicaciones que llevan un gran volumen de código. Como por ejemplo los frameworks ofrecen un conjunto de librerías, estas proporcionan diversas funcionalidades tales como: sistemas de templates (plantillas), manejo de sesiones de usuario, interfaces comunes, entre otras. Estos también brindan la posibilidad al programador de reutilizar código y a la vez fuerzan al desarrollador a crear código más legible, ya que proporciona una estructura de código fuente. Mientras que los CMS están más enfocados al trabajo de administración de contenidos en los sitios *web*.

1.10 Selección del Framework de desarrollo

En la selección del *framework* de desarrollo se realizó un estudio de distintos *framework* como son el *CodeIgniter*, *Zend Framework,* Cakephp y *Symfony*. A continuación se puede apreciar dicho estudio.

1.10.1 CodeIgniter

CodeIgniter contiene una serie de librerías que sirven para el desarrollo de aplicaciones web y además propone una manera de desarrollarlas que se debe seguir para obtener provecho de la aplicación. Este marca una manera específica de codificar las páginas *web* y clasificar sus diferentes *scripts*, que sirve para que el código esté organizado y sea más fácil de crear y mantener. *CodeIgniter* implementa el proceso de desarrollo llamado Modelo Vista Controlador (MVC), que es un estándar de programación de aplicaciones, utilizado tanto para hacer sitios *web* como programas tradicionales (39).

Ventajas:

- Compatibilidad: es compatible con la versión PHP 4, lo que hace que se pueda utilizar en cualquier servidor, incluso en algunos antiguos, también en PHP 5.
- Facilidad de instalación: solo es necesaria una cuenta de FTP para subir *CodeIgniter* al servidor y su configuración se realiza con la edición de un archivo.
- Flexibilidad: define una manera de trabajar específica, pero en muchos de los casos se puede seguir o no y sus reglas de codificación muchas veces se pueden saltar para trabajar más a gusto. Algunos módulos como el uso de plantillas son totalmente opcionales.
- Ligereza: el núcleo de *CodeIgniter* es bastante ligero, lo que permite que el servidor no se sobrecargue interpretando o ejecutando grandes porciones de código.
- Documentación tutorializada: la documentación de este es fácil de seguir y de asimilar, porque está escrita en modo de tutorial, aunque no facilita mucho la referencia rápida, cuando ya se sabe acerca del *framework* y se quiere consultar sobre una función o un método en concreto.

Desventajas (40)

- Curva de aprendizaje: necesidad de aprender nuevas funciones, estructuras y métodos de programación.
- Dificultad para adaptar el código escrito en PHP tradicional.

1.10.2 Zend Frameworks[2]

Es un *framework* de código abierto para desarrollar aplicaciones y servicios *web* con PHP 5 ZF es una implementación que usa código ciento por ciento orientado a objetos. La estructura de

[2] Más información http://*framework*.zend.com/

los componentes de ZF es única; cada componente está construido con una baja dependencia de otros componentes. *Zend Framework* es un *framework* para PHP, desarrollado por *Zend*, empresa encargada de la mayor parte de las mejoras hechas a PHP, por lo que se podría decir que es el *framework* "oficial" (41).

Ventajas

- Facilita integrar a otras personas a sus proyectos ya que se comparten convenciones de desarrollo comunes.
- Gran cantidad de componentes, incluidos algunos de *Microsoft*, *Google* y *Adobe*.
- Es un *software* libre.
- Capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad, destaca su conectividad con *MySQL* y *PostgreSQL*.
- Es desarrollado por *Zend* Technologies que es la empresa que respalda comercialmente a PHP.
- Trabaja con MVC.
- El Marco de Zend también incluye objetos de las diferentes bases de datos, por lo que es extremadamente simple para consultar su base de datos, sin tener que escribir ninguna consulta SQL.
- Una solución para el acceso a base de datos que balancea el ORM con eficiencia y simplicidad.
- Completa documentación y *tests* de alta calidad.
- Robustas clases para autenticación y filtrado de entrada.
- Clientes para servicios *web*, incluidos Google Data APIs y *StrikeIron*.
- Muchas otras clases útiles para hacerlo tan productivo como sea posible.
- Componentes que implementan formas de representación de formularios HTML.
- Validación y filtrado de datos.

Desventajas

- Ninguna protección por definición de la licencia BSD.
- No tiene mucha Independencia estratégica.

1.10.3 Cakephp

Cakephp provee una base robusta para las aplicaciones *web*. Puede manejar cualquier aspecto, desde la solicitud inicial del usuario hasta el renderizado final de la página *web*.

Además, como este sigue los principios Modelo Vista Controlador (en adelante MVC), permite fácilmente personalizar y extender muchos aspectos de esta. El *framework* también suministra una estructura de organización básica, desde los nombres de los archivos hasta los de las tablas de la base de datos, manteniendo toda la aplicación consistente y lógica (42).

Está caracterizado por:

- Compatible con PHP4 y PHP5.
- CRUD de la base de datos integrado.
- URLs amigables.
- Sistema de plantillas rápido y flexible.
- Ayudas para AJAX, *Java script*, HTML, *forms*.
- Trabaja en cualquier subdirectorio del sitio.
- *Scaffolding* (andamiaje) de las aplicaciones.
- Listas de Control de Acceso.
- Componentes de seguridad y sesión.

Ventajas (43)**:**

- Licencia flexible, *Cakephp* está distribuido bajo la MIT *License.*
- Desarrollo rápido.
- Buenas prácticas, *Cake* es muy fácil de entender y cumple los estándares en seguridad y autentificación, manejo de sesiones y muchas otras características.

Desventajas

- Ninguna protección por definición de la licencia MIT.
- No ofrecen más apoyo que foros, listas de correo y *Google* Grupos.

1.10.4 Symfony 2.0.22

Symfony es un completo *framework* diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones *web*. Separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación *web*. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación *web* compleja. Además, automatiza las tareas más comunes, permitiendo 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*.

Symfony está desarrollado completamente con PHP 5. Ha sido probado en numerosos proyectos reales y se utiliza en sitios *web* de comercio electrónico de primer nivel. *Symfony* es compatible con la mayoría de gestores de bases de datos, como *MySQL*, *PostgreSQL*, *Oracle* y *SQL Server* de *Microsoft*. Se puede ejecutar tanto en plataformas *nix (Unix y Linux) como en plataformas Windows (37).

Características de *Symfony*

Symfony está diseñado para que se ajuste a los siguientes requisitos (37):

- Fácil de instalar y configurar en la mayoría de plataformas.
- Independiente del sistema gestor de bases de datos.
- Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.
- Basado en la premisa de *"convenir en vez de configurar"*, en la que el desarrollador solo debe configurar aquello que no es convencional.
- Sigue la mayoría de mejores prácticas y patrones de diseño para la *web*.
- Preparado para aplicaciones empresariales y adaptables a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.
- Código fácil de leer que incluye comentarios de *phpDocumentor* y que permite un mantenimiento muy sencillo.
- Fácil de extender, lo que permite su integración con librerías desarrolladas por terceros.

1.10.5 Fundamentación de la elección

Después de haber estudiado diferentes *frameworks* de desarrollo se decide utilizar *Symfony2* para el desarrollo de la aplicación porque agrupa las mejores características de los *frameworks* enunciados anteriormente. Incluye las funciones para administrar bases de datos, crear formularios, validar diferentes tipos de datos, administrar usuarios. Los archivos de configuración se pueden escribir en PHP, XML, YAML o INI por lo que internamente *Symfony* 2 siempre utiliza XML, para permitir el autocompletado en los IDE y para poder validar los archivos. *Symfony* 2 utiliza *Twig* para crear todas las plantillas de la aplicación ya que es un motor y lenguaje de plantillas para PHP muy rápido y eficiente por lo que resultan ser mucho más limpias y concisas, las plantillas de *Twig* son seguras por defecto. Posee licencia de

código abierto, es independiente del gestor de bases de datos y es muy adecuado para metodologías ágiles de desarrollo como XP.

La arquitectura interna de *Symfony* 2 está completamente desacoplada, lo que permite reemplazar o eliminar fácilmente aquellas partes que no encajan en el proyecto. En *Symfony*, el modelo contiene la información necesaria para poder generar de forma automática el código de las operaciones de insertar, obtener, modificar y borrar (CRUD), de forma que se simplifica el desarrollo inicial de la parte de administración de las aplicaciones.

1.11 Conclusiones

Después del estudio y análisis realizado del objeto de investigación, apoyado en los métodos de la investigación científicos definidos, se pudo construir el marco teórico-conceptual que soporta la investigación. El estudio del marco teórico de los SGI aportaron elementos sólidos para la realización del SGI para los resultados de las prueba de eficiencia fisca en la UCI, se adquirieron los conocimientos necesarios sobre las herramientas, metodología y lenguajes utilizados en el desarrollo de soluciones informáticas, seleccionándose los más adecuados para el cumplimiento del objetivo propuesto.

Como metodología de desarrollo de *software* se escoge XP, el lenguaje de programación seleccionado fue PHP 5.3 y como lenguaje de marcado *Javascript* 1.5, CSS 3 y HTML 5. Como *framework* de desarrollo se selecciona Symfony 2.0.22 para la elaboración de la aplicación, el IDE de desarrollo es *Netbeans* en su versión 7.3, como servidor *web* Apache 2.2.22 y como gestor de base de datos *PostgreSQL* 9.1.

Capítulo 2 Propuesta de solución

2.1 Introducción

En este capítulo se describen las fases principales de la metodología XP: Planificación y Diseño para la solución propuesta. Se desarrollarán los artefactos importantes de estas fases como la Metáfora, Definición de la audiencia, Historia de Usuarios, Plan de Iteraciones, Plan de Duración de Iteraciones, Plan de Entregas y Tarjetas CRC (Clases, Responsabilidad y Colaboración).

2.2 Metáfora

La definición de la metáfora del sistema constituye uno de los pasos fundamentales en la metodología XP. Una metáfora es una historia compartida que describe cómo debería funcionar el sistema. Martin Fowler explica que la práctica de la metáfora consiste en formar un conjunto de nombres que actúen como vocabulario para hablar sobre el dominio del problema. Este conjunto de nombres ayuda a la nomenclatura de clases y métodos del sistema (11).

La metáfora que se propone es la siguiente:

“Un espacio donde se ‘facilite’ la información de las pruebas de eficiencia física de los estudiantes”.

2.3 Definición de la audiencia

La audiencia constituye uno de los elementos más importantes de esta fase de desarrollo del sistema, ya que consiste en definir al público a quien va dirigida la solución del SGI, en este caso la solución va dirigida a los profesores de la asignatura de Educación Física de la UCI.

2.4 Usuarios relacionados con el sistema

La aplicación *web* que se desarrolla muestra un grupo de funcionalidades y servicios para cumplir con los objetivos trazados. En la implementación se establecen roles para asignar los diferentes permisos para su acceso. Los roles y permisos son utilizados para restringir el nivel de acceso a las funciones del sistema de cada usuario que interactúe con el sistema, es decir que en dependencia de los roles de cada usuario y los permisos que tengan cada uno de los roles será su acceso a las funciones del sistema. Se denomina usuario a cualquier persona

relacionada con el sistema, ya sea vinculada al desarrollo del mismo o que de una forma u otra interactúa con la aplicación, incluyendo a los que mantendrán el sistema funcionando y actualizado (44).

Tabla 1. Usuarios relacionados con el sistema

Usuarios	Descripción
Profesor	Es el encargado de realizar la gestión de las pruebas de eficiencia física y la planificación de los ejercicios.
Administrador	Son los usuarios que tienen permiso para administrar todas las funcionalidades del sistema.

2.5 Historias de Usuarios

La metodología XP utiliza la técnica de las Historias de Usuario (HU) para sustituir a los documentos de especificación funcional y a los casos de uso. Estas HU son escritas por el cliente en su propio lenguaje como descripciones cortas de lo que el sistema debe realizar. El tratamiento de las HU es muy dinámico y flexible, permite que en cualquier momento se puedan romper, reemplazar por otras más específicas o generales, añadirse nuevas o ser modificadas. Para ser implementadas las HU, el cliente y los desarrolladores se reúnen para detallar las funcionalidades de cada una. El tiempo de desarrollo ideal para una HU varía entre 1 y 3 semanas (45).

Según Kent Beck cada HU recoge al menos los siguientes aspectos (46):

- **Número:** Posee el número asignado a la HU.
- **Nombre de HU:** Atributo que contiene el nombre de la HU.
- **Usuario:** El usuario del sistema que utiliza o protagoniza la HU.
- **Prioridad en el negocio:** Evidencia el nivel de prioridad de la HU en el negocio.
- **Riesgo de desarrollo:** Evidencia el nivel de riesgo en caso de no realizarse la HU.
- **Puntos estimados:** Este atributo no es más que una estimación hecha por el equipo de desarrollo del tiempo de duración de la HU. Cuando el valor es 1 equivale a una semana ideal de trabajo. En la metodología XP está definida una semana ideal como 5 días hábiles trabajando 40 horas, es decir, 8 horas diarias. Por lo que cuando el valor de

dicho atributo es 0.5 equivale a 2 días y medio de trabajo, lo que se traduce en 20 horas.

- **Puntos reales:** Igual que el parámetro anterior, pero en este caso será el tiempo real en el que se realizó la HU.
- **Descripción:** Posee una breve descripción de lo que realizará la HU.

Para las HU que describen las características del SGI, se decidió excluir los siguientes elementos: usuario, prioridad en el negocio, puntos estimados, puntos reales y riesgo de desarrollo y agregarle un campo con el nombre **Observación** que brindará información extra para que la HU sea más comprensible.

A continuación se muestran las HU de las funcionalidades del sistema.

Tabla 2. HU Gestionar Usuario

Historia de Usuario	
Número: 1	**Nombre de la HU:** Gestionar Usuario.
Usuario: Administrador	
Prioridad en el negocio: Baja	**Puntos estimados:** 0.2
Riesgo de Desarrollo: Alto	**Puntos reales:** 0.2
Descripción: Permite adicionar, modificar, eliminar y listar los datos de los usuarios que interactúan con el sistema.	

Tabla 3. HU Gestionar Grupo

Historia de Usuario	
Número: 2	**Nombre de la HU:** Gestionar Grupo
Usuario: Administrador	
Prioridad en el negocio: Media	**Puntos estimados:** 0.2
Riesgo de Desarrollo: Alto	**Puntos reales:** 0.2
Descripción: Permite adicionar, modificar, eliminar y listar los datos de los grupos.	

Tabla 4. HU Gestionar Estudiante

Historia de Usuario	
Número: 3	**Nombre de la HU:** Gestionar Estudiante
Usuario: Profesor	

Prioridad en el negocio: Media	**Puntos estimados:** 0.2
Riesgo de Desarrollo: Alto	**Puntos reales:** 0.2
Descripción: Permite adicionar, modificar, eliminar y listar los datos de los estudiantes.	

Tabla 5. HU Gestionar Profesor

Historia de Usuario	
Número: 4	**Nombre de la HU:** Gestionar Profesor
Usuario: Administrador	
Prioridad en el negocio: Baja	**Puntos estimados:** 0.2
Riesgo de Desarrollo: Baja	**Puntos reales:** 0.2
Descripción: Permite adicionar, modificar, eliminar y listar los datos de los profesores.	

Tabla 6. HU Gestionar Prueba

Historia de Usuario	
Número: 5	**Nombre de la HU:** Gestionar Prueba
Usuario: Profesor	
Prioridad de en el negocio: Alta	**Puntos estimados:** 0.2
Riesgo de Desarrollo: Alto	**Puntos reales:** 0.2
Descripción: Permite adicionar, modificar, eliminar y listar los datos de las pruebas de eficiencia física.	

Tabla 7. HU Gestionar Tipo Prueba

Historia de Usuario	
Número: 6	**Nombre de la HU:** Gestionar Tipo Prueba
Usuario: Administrador	
Prioridad en el negocio: Alta	**Puntos estimados:** 0.2
Riesgo de Desarrollo: Alto	**Puntos reales:** 0.2
Descripción: Permite adicionar, modificar, eliminar y listar el tipo de prueba.	

Tabla 8. HU Gestionar Ejercicio

Historia de Usuario	
Número: 7	**Nombre de la HU:** Gestionar Ejercicio
Usuario: Administrador	

Prioridad en el negocio: Media	**Puntos estimados:** 0.2
Riesgo de Desarrollo: Medio	**Puntos reales:** 0.2
Descripción: Permite adicionar, modificar, eliminar y listar el plan de ejercicios físicos.	

Tabla 9. HU Asignar Ejercicios

Historia de Usuario	
Número: 8	**Nombre de la HU:** Asignar Ejercicios
Usuario: Profesor	
Prioridad en el negocio: Alta	**Puntos estimados:** 2
Riesgo de Desarrollo: Medio	**Puntos reales:** 2
Descripción: El sistema asigna ejercicios a cada grupo de estudiantes a partir de los resultados obtenidos en las pruebas de eficiencia física. Cada grupo de estudiantes tiene asociados un nivel (del 1 al 4) en cada una de las pruebas y de acuerdo al valor de los niveles alcanzados, se asignarán ejercicios para mejorar las condiciones físicas de cada grupo.	

Tabla 10. Seguridad

Historia de Usuario	
Número: 9	**Nombre de la HU:** Seguridad
Descripción: La autenticación en el sistema está definida para cada uno de los usuarios mediante sus roles y permisos.	
Observación: Para la utilización del sistema, la aplicación sólo podrá ser accedida por medio de un usuario y un *password*. El usuario registrado podrá realizar las operaciones correspondientes de acuerdo a su rol definido en el sistema.	

Tabla 11. Usabilidad

Historia de Usuario	
Número: 10	**Nombre de la HU:** Usabilidad
Descripción: El SGI podrá ser empleado por personas con pocos, medios o avanzados conocimientos de informática.	
Observación: El sistema muestra una interfaz sencilla y amigable. Brinda los botones con un tamaño adecuado y con nombres claros que permiten a los usuarios realizar las operaciones que deseen de forma sencilla.	

Tabla 12. Condiciones tecnológicas

Historia de Usuario	
Número: 11	**Nombre de la HU:** Condiciones tecnológicas
Descripción: Asegurar que el acceso al sistema se realice desde ordenadores con buenas prestaciones.	
Observación: Las máquinas donde se van a utilizar la aplicación deben de tener como mínimo 250 MG de memoria RAM, tener instalado Sistema Operativo *Linux* o *Windows,* un navegador *web* y debe de estar conectada a la red de la UCI.	

2.6 Estimación de esfuerzos por HU

Las HU deben tener el detalle mínimo como para que los programadores puedan realizar una estimación poco riesgosa del tiempo que llevará su desarrollo. Cuando llegue el momento de la implementación, los desarrolladores dialogan directamente con el cliente para obtener todos los detalles necesarios (45). Inicialmente, el equipo de desarrolladores estima el esfuerzo necesario para implementar las HU y los clientes aprueban los objetivos y tiempos de entrega, la estimación temporal se basa en un cálculo estimado por parte de los desarrolladores de cada una de las HU (47).

Las estimaciones realizadas en esta fase son primarias y podrían variar cuando se analicen más en detalle en cada iteración (45).

Para la realización del SGI se utilizarán las semanas como medida para la estimación de esfuerzos establecidas por los programadores. A continuación se muestra la tabla Estimación de esfuerzos:

Tabla 13. Estimación de esfuerzos

No	Historia de Usuario	Estimación (por semanas)
1	Gestionar Usuario	0.2
2	Gestionar Grupo	0.2
3	Gestionar Estudiante	0.2
4	Gestionar Profesor	0.2
5	Gestionar Prueba	0.2
6	Gestionar Tipo de Prueba	0.2

7	Gestionar Ejercicio	0.2
8	Asignar Ejercicios	2

2.7 Plan de iteraciones

Todo proyecto que emplea metodología XP debe dividirse en iteraciones. Las iteraciones son fases o etapas de la implementación donde se obtienen resultados en un tiempo estimado. En el plan de iteraciones se especifican detalladamente el orden de desarrollo de las HU dentro de cada iteración conjuntamente con la duración de las mismas.

Iteración 1: En esta iteración se implementan las HU que tienen prioridad alta en el negocio, las principales son las HU 1, 2, y 3 las cuales son de vital importancia para el SGI ya que conforman la base de la estructura del sistema. Con esta iteración se obtiene la primera versión de la aplicación la cual se utilizará para mostrar al cliente y permitirá al grupo de trabajo tener una retroalimentación.

Iteración 2: En esta iteración se realiza la implementación de las HU 4, 5 y 6. Con esta iteración se corrigen errores o inconformidades del cliente con las HU implementadas en la iteración anterior. De esta forma se obtiene la segunda versión del sistema. Esta segunda iteración es también mostrada al cliente con el objetivo de evaluar las aceptaciones por parte del cliente con el *software*.

Iteración 3: En esta iteración se realiza la implementación de la HU 7 y 8 corrigiéndose lo errores de las iteraciones anteriores.

2.8 Plan de duración de iteraciones

El plan de duración de las iteraciones se realiza luego de tener el estimado en días que demora implementar cada historia de usuario (45). El objetivo del plan de duración de iteraciones es especificar detalladamente el orden de desarrollo de las HU dentro de cada iteración y la duración de las iteraciones.

Tabla 14. Plan de duración de iteraciones

Iteraciones	Orden de las HU a implementar	Duración (semanas)
Iteración 1	Gestionar Usuario Gestionar Grupo	0.6

	Gestionar Estudiante	
Iteración 2	Gestionar Profesor Gestionar Pruebas Gestionar Tipo Pruebas	0.6
Iteración 3	Gestionar Ejercicio Asignar Ejercicios	2.2

2.9 Plan de entregas

En el plan de entregas se establecen qué HU son agrupadas para conformar una entrega y el orden de las mismas.

Tabla 15. Plan de entregas

Historia de Usuario	Primera Iteración	Segunda Iteración	Tercera Iteración
Gestionar Usuario	V 1.0	Finalizado	-
Gestionar Grupo	V 1.0	Finalizado	-
Gestionar Estudiante	V 1.0	Finalizado	-
Gestionar Profesor	-	V 1.0	Finalizado
Gestionar Prueba	-	V 1.0	Finalizado
Gestionar Tipo Prueba	-	V 1.0	Finalizado
Gestionar Ejercicio	-	-	Finalizado
Asignar Ejercicios	-	-	Finalizado

2.10 Prototipo de interfaz de usuarios

El prototipo de interfaz de usuario son elementos de diseño visual que permiten al usuario tener una idea de las interfaces que mostrará el sistema para obtener una retroalimentación sobre los requerimientos del mismo (48).

A continuación se muestra el prototipo de interfaz de usuario del SGI:

Figura 1: Prototipo de Interfaz de usuario del SGI.

2.11 Tarjetas CRC

Las tarjetas CRC son en la práctica pequeñas tarjetas de cartón que se elaboran para ser mostradas al cliente, de manera que se pueda llegar a un acuerdo sobre la validez de las abstracciones propuestas, lo que ayuda al equipo durante el diseño e implementación del sistema. Estas constituyen documentación adicional que será adjuntada a las HU (49).

Las tarjetas CRC trabajan con la técnica de modelado basada en objetos, representando cada tarjeta CRC a un objeto, identificando las clases y sus responsabilidades. Las tarjetas están compuestas por el nombre de la clase colocado como título, en la parte izquierda se colocan las responsabilidades (funcionalidades) y en la parte derecha las clases que se implican en cada funcionalidad (45).

Tabla 16. Tarjeta CRC Gestionar Usuario

Clase: Usuario

Responsabilidad	Colaboración
Gestionar Usuario	

Tabla 17. Tarjeta CRC Gestionar Estudiante

Clase: Estudiante	
Responsabilidad	**Colaboración**
Gestionar Estudiante	Tb[3] Grupo

Tabla 18. Tarjeta CRC Gestionar Prueba

Clase: Prueba	
Responsabilidad	**Colaboración**
Gestionar Prueba	Tb Estudiante Tb Tipo Prueba

Tabla 19. Tarjeta CRC Gestionar Profesor

Clase: Profesor	
Responsabilidad	**Colaboración**
Gestionar Profesor	Tb Grupo

Tabla 20. Tarjeta CRC Gestionar Grupo

Clase: Grupo	
Responsabilidad	**Colaboración**
Gestionar Grupo	

Tabla 21. Tarjeta CRC Gestionar Tipo Prueba

Clase: Tipo Prueba	
Responsabilidad	**Colaboración**
Gestionar Tipo Prueba	Tb Prueba

Tabla 22. Tarjeta CRC Gestionar Ejercicios

Clase: Ejercicio

[3] Tabla

Responsabilidad	Colaboración
Gestionar Ejercicio	

Tabla 23. Tarjeta CRC Asignar Ejercicios

Método: Asignar Ejercicios	
Responsabilidad	**Colaboración**
Asignar Ejercicios	Tb Estudiante Tb Prueba Tb Grupo Tb Ejercicio

2.12 Conclusiones

La metodología seleccionada para la elaboración del SGI para los resultados de las pruebas de eficiencia física, permite realizar el análisis y diseño de la aplicación. El estudio de la audiencia determinó que existen 2 tipos de usuarios que interactúan con el sistema. Se definieron un total de 11 HU que describen los aspectos principales a tener en cuenta para el desarrollo de la solución. Se precisó la prioridad de cada una de las HU que describen las funcionales del sistema, puntualizando el orden de su implementación y las iteraciones en que serán implementadas. A partir de las HU se construyó el plan de entregas y 8 tarjetas CRC que traducen los requerimientos a funcionalidades a implementar.

Capítulo 3 Implementación y Prueba

3.1 Introducción

Según Joskowicz (2008) la fase de iteraciones es la fase principal en el ciclo de desarrollo de XP. Las funcionalidades son desarrolladas en esta fase, generando al final de cada una un entregable funcional que implementa las historias de usuario asignadas a la iteración.

En el presente capítulo se detallan las iteraciones llevadas a cabo durante la elaboración del sistema, además de exponer las pruebas realizadas para probar el *software*, los patrones de diseños utilizados en el desarrollo de la aplicación y el modelo de base de dato.

3.2 Patrones de Arquitectura

Un patrón es un modelo a seguir, surgen de la experiencia de seres humanos al tratar de lograr ciertos objetivos, capturan la experiencia existente y probada para promover buenas prácticas (50).

Los patrones arquitectónicos son los que definen la estructura de un sistema de *software*, los cuales a su vez se componen de subsistemas con sus responsabilidades, también tienen una serie de directivas para organizar los componentes del mismo sistema, con el objetivo de facilitar la tarea del diseño de tal sistema (51).

Los patrones arquitectónicos son patrones de alto nivel que fijan la arquitectura global de una aplicación.

3.2.1 Patrón Modelo Vista Controlador

El patrón Modelo Vista Controlador (MVC) es un patrón de arquitectura de *software* que separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos. Este patrón se ve frecuentemente en aplicaciones *web*, donde la vista es la página HTML y el código que provee de datos dinámicos a la página, el modelo es el Sistema de Gestión de Base de Datos y el controlador representa la lógica de negocio (52).

Symfony 2 basa su funcionamiento interno en la arquitectura MVC utilizada por la mayoría de *frameworks web*. En el SGI para los resultados de las pruebas de eficiencia, los principios de la arquitectura MVC se aplican de la siguiente manera:

1. Cuando el usuario envía una ruta, el sistema de enrutamiento determina qué Controlador está asociado a la ruta.
2. *Symfony* 2 ejecuta el Controlador correspondiente a la ruta. En el SGI, los Controladores asociados a las rutas se encuentran en la carpeta Controller dentro del *bundle backenBundle.*
3. El Controlador solicita al Modelo los datos de la clase.
4. 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.
5. El Controlador entrega al servidor la página creada por la Vista.

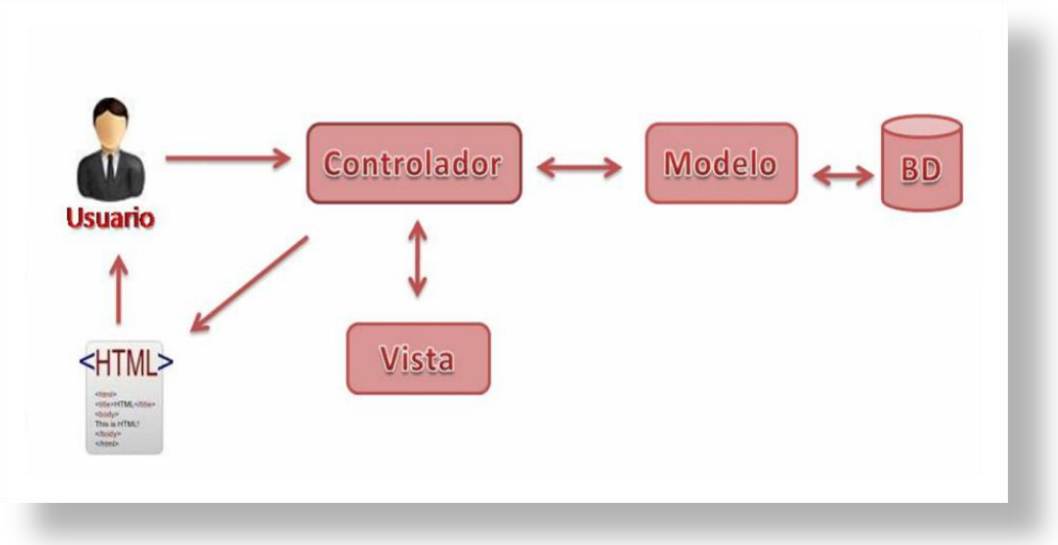

Figura 2: Arquitectura MVC en el SGI.

3.3 Patrones de Diseño

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular. Estos patrones identifican Clases, Instancias, Roles, Colaboraciones y la distribución de responsabilidades (50).

Los patrones de diseño se dividen en dos grandes grupos los GRASP (del inglés *General Responsibility Asignment Software Patterns)* (patrones generales de *software* para asignar responsabilidades) y los GOF (del inglés *Gand of Four*). A continuación se realiza un estudio de cómo fueron utilizados en el SGI.

3.3.1 Patrones GRASP

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones (21).

En el diseño del sistema se destacan el uso de 4 patrones principales que son:

- **Alta Cohesión**: en la perspectiva del diseño orientado a objetos, la cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. Como ejemplo en el SGI, a la clase *PruebaManager.php* se le delegan responsabilidades relacionadas con las pruebas y a la clase *NivelManager.php* se le asignan responsabilidades relacionadas con la descripción y determinación de los niveles de las pruebas.
- **Bajo Acoplamiento**: el acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases y con que recurre a ellas. Una clase con bajo o débil acoplamiento no depende de muchas otras. Ejemplo: la clase Prueba *Manager* hace uso de la clase Nivel *Manager* solo cuando necesita de Nivel *Manager*. Si se realizan cambios en la clase Nivel *Manager*, no tienen que realizarse cambios en la clase Prueba Manager y viceversa.
- **Experto**: asignar una responsabilidad al experto en información (la clase que tiene la información necesaria para realizar la responsabilidad). Este patrón se puede observar en el propio *framework* quien se encarga de implementarlo a través del ORM *Doctrine* 2. Genera las clases de acceso a datos asignándoles las responsabilidades de ejecutar todas las funcionalidades comunes de las entidades que representan y de la cual poseen información.
- **Controlador:** lo utiliza la clase controladora como manipuladora de las peticiones o eventos del sistema.

3.3.2 Patrones GOF

Los patrones de diseño Gof se clasifican según el libro GOF en tres categorías: de creación, estructurales y de comportamiento. Los patrones de creación abstraen el proceso de creación de instancias, los estructurales se ocupan de cómo clases y objetos son utilizados para componer estructuras de mayor tamaño y los de comportamiento atañen a los algoritmos y a la asignación de responsabilidades entre objetos.

Algunos de estos patrones utilizados en el SGI son:

- **Abstract Factory:** pertenece a la familia de los patrones creacionales, el cual se refiere al proceso de creación de objetos. Este patrón permite que el sistema sea independiente a cómo se creen sus objetos y refleja cómo crear familias de objetos relacionados con productos sin instanciar clases directamente. Además, se ve reflejado en las clases controladoras al crear los formularios. Ejemplo:

```
public function newAction($idGrupo)
    {$entity = new Estudiante();
    $form = $this->createForm(new EstudianteType(), $entity);
    ...}
```

- **Factory Method**: este patrón define una interfaz para la creación de un objeto, pero permitiendo a las subclases decidir de qué clase instanciarlo. Permite, por tanto, que una clase difiera la instanciación en favor de sus subclases (53). Un ejemplo de la utilización de este patrón se pone en práctica en las clases entidades que son las encargadas de crear objetos de un tipo determinado.

3.5 Diseño de la base de datos

Una de las tareas más importantes para la elaboración de un sistema *web* es la construcción de la base de datos. A continuación se muestra el modelo de datos:

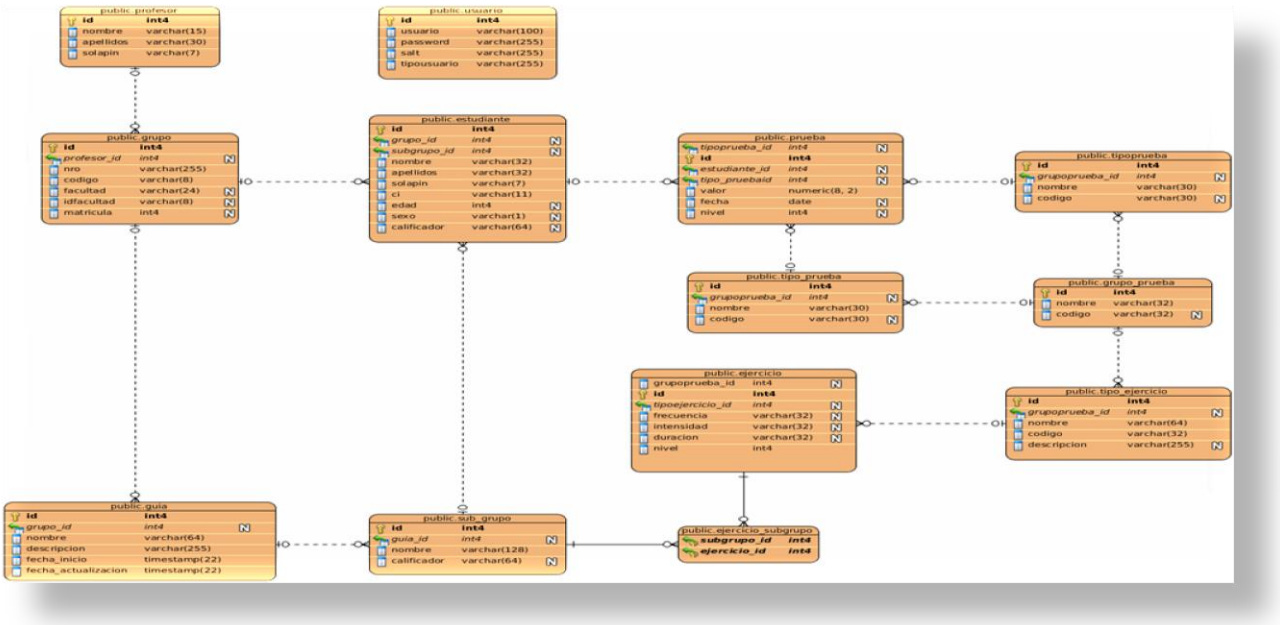

Figura 3: Modelo de datos del SGI.

3.6 Tareas de la Ingeniería

Las tareas de la ingeniería son las distintas funcionalidades operativas que conforman una historia de usuario y que permiten testear si se está trabajando bien. Este trabajo conjunto constituye un paso decisivo para comenzar la implementación del *software*, pues permite organizar el trabajo en pasos lógicos, de acuerdo a la planificación correspondiente a esa historia de usuario (54).

Las tareas de la ingeniería se realizan con el objetivo de resolver las HU que pueden tener una o más tareas de ingeniería en dependencia de la complejidad de la funcionalidad a desarrollar.

En el sistema propuesto se identificaron las siguientes tareas de la ingeniería:

Tabla 24. Tareas de la Ingeniería

Nro. HU	Nombre HU	Nro. TI	Tarea de la Ingeniería
1	Gestionar Usuario	1	Gestionar Usuario
2	Gestionar Estudiante	1	Adicionar Estudiante
		2	Modificar Estudiante
		3	Eliminar Estudiante
		4	Listar Estudiante
3	Gestionar Grupo	1	Adicionar Grupo
		2	Modificar Grupo
		3	Eliminar Grupo
		4	Listar Grupo
4	Gestionar Profesor	1	Adicionar Profesor
		2	Modificar Profesor
		3	Eliminar Profesor
		4	Listar Profesor
5	Gestionar Prueba	1	Adicionar Prueba
		2	Modificar Prueba
		3	Eliminar Prueba
		4	Listar Prueba

6	Gestionar Tipo Prueba	1	Adicionar Tipo Prueba
		2	Modificar Tipo Prueba
		3	Eliminar Tipo Prueba
		4	Listar Tipo Prueba
7	Gestionar Ejercicio	1	Adicionar Ejercicio
		2	Modificar Ejercicio
		3	Eliminar Ejercicio
		4	Listar Ejercicio
8	Asignar Ejercicios	1	Asignar Ejercicios

3.6.1 Desarrollo de las Tareas de la ingeniería

A continuación se detallan el desarrollo de las tareas de la ingeniería divididas en las iteraciones correspondientes a cada historia de usuario.

3.6.1.1 Iteración 1

Tabla 25. Descripción de la Tarea de la ingeniería 1 Gestionar Usuario

Número de Tarea: 1	**HU (Nro. 1:)**Gestionar Usuario
Nombre de la Tarea: Autenticar Usuario	
Tipo Tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha Inicio:6 de febrero 2013	**Fecha Fin:** 8 de febrero 2013
Programador Responsable: Roger A. González Rodríguez y Victor A. Fong Rios	
Descripción: Se muestra una interfaz donde el usuario debe introducir un usuario y una contraseña, el sistema verifica que los datos estén correctos y que permita el acceso al sistema según los permisos definidos para ese usuario.	

Tabla 26. Descripción de la Tarea de la Ingeniería 1 Adicionar Estudiante

Número de Tarea: 1	**HU (Nro. 2:)** Gestionar Estudiante
Nombre de la Tarea: Adicionar Estudiante	
Tipo Tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha Inicio:9 de febrero 2013	**Fecha Fin:**10 de febrero 2013
Programador Responsable: Roger A. González Rodríguez y Victor A. Fong Rios	
Descripción: Se muestra una interfaz donde se adiciona un estudiante con sus datos a la	

base de datos.

Tabla 27. Descripción de la Tarea de la Ingeniería 2 Modificar Estudiante

Número de Tarea: 2	**HU (Nro. 2:)** Gestionar Estudiante
Nombre de la Tarea: Modificar Estudiante	
Tipo Tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha Inicio: 9 de febrero 2013	**Fecha Fin:** 10 de febrero 2013
Programador Responsable: Roger A. González Rodríguez y Victor A. Fong Rios	
Descripción: Se muestra una interfaz donde se modifican los datos de un estudiante.	

En los Anexos se encuentran las demás Tareas de la ingeniería que pertenecen a esta iteración.

3.6.1.2 Iteración 2

Tabla 28. Descripción de la Tarea de la Ingeniería 1 Adicionar Prueba

Número de Tarea: 1	**HU (Nro. 7:)** Gestionar Prueba
Nombre de la Tarea: Adicionar Prueba	
Tipo Tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha Inicio: 15 de febrero 2013	**Fecha Fin:** 16 de febrero 2013
Programador Responsable: Roger A. González Rodríguez y Victor A. Fong Rios	
Descripción: Se muestra una interfaz donde se adiciona una prueba con sus datos a la base de datos.	

Tabla 29. Descripción de la Tarea de la Ingeniería 2 Modificar Prueba

Número de Tarea: 2	**HU (Nro. 7:)** Gestionar Prueba
Nombre de la Tarea: Modificar Prueba	
Tipo Tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha Inicio: 15 de febrero 2013	**Fecha Fin:** 16 de febrero 2013
Programador Responsable: Roger A. González Rodríguez y Victor A. Fong Rios	
Descripción: Se muestra una interfaz donde se modifica una prueba de la base de datos.	

Tabla 30. Descripción de la Tarea de la Ingeniería 3 Eliminar Prueba

Número de Tarea: 3	**HU (Nro. 7:)** Gestionar Prueba
Nombre de la Tarea: Eliminar Prueba	
Tipo Tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha Inicio: 15 de febrero 2013	**Fecha Fin:** 16 de febrero 2013
Programador Responsable: Roger A. González Rodríguez y Victor A. Fong Rios	
Descripción: Se muestra una interfaz donde se elimina una prueba de la base de datos.	

Tabla 31. Descripción de la Tarea de la Ingeniería 4 Listar Prueba

Número de Tarea: 4	**HU (Nro. 7:)** Gestionar Prueba
Nombre de la Tarea: Listar Prueba	
Tipo Tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha Inicio: 15 de febrero 2013	**Fecha Fin:** 16 de febrero 2013
Programador Responsable: Roger A. González Rodríguez y Victor A. Fong Rios	
Descripción: Se muestra una interfaz donde se listan las pruebas de la base de datos.	

En los Anexos se encuentran las demás Tareas de la ingeniería que pertenecen a esta iteración.

3.6.1.3 Iteración 3

Tabla 32. Descripción de la Tarea de la Ingeniería 1 Asignar Ejercicios

Número de Tarea: 1	**HU (Nro. 12:)** Asignar Ejercicios
Nombre de la Tarea: Asignar Ejercicios	
Tipo Tarea: Desarrollo	**Puntos estimados:** 3
Fecha Inicio: 11 de marzo 2013	**Fecha Fin:** 1 de abril 2013
Programador Responsable: Roger A. González Rodríguez y Victor A. Fong Rios	
Descripción: Se muestra una interfaz donde se asignan los ejercicios para cada grupo de estudiantes.	

3.7 Pruebas

Las pruebas constituyen un aspecto importante en el proceso de elaboración del sistema ya que permiten medir el éxito de las funcionalidades del sistema. La metodología XP divide las pruebas en pruebas unitarias y pruebas de aceptación.

3.7.1 Pruebas de Unitarias

Las pruebas unitarias son una de las piedras angulares de XP. Todos los módulos deben de pasar las pruebas unitarias antes de ser liberados o publicados. Por otra parte, revisar el código, que todo código liberado pase correctamente las pruebas unitarias es lo que habilita que funcione la propiedad colectiva del código. En este sentido, el sistema y el conjunto de pruebas deben ser guardados junto con el código, para que pueda ser utilizado por otros desarrolladores, en caso de tener que corregir, cambiar o recodificar parte del mismo (45).

Las pruebas unitarias aplicadas a la aplicación fueron los test unitarios que propone Symfony2. Estas pruebas fueron llevadas a cabo por los programadores, encargadas de verificar el código de forma automática, se aplicaron a todas las funcionalidades del sistema comprobando el correcto funcionamiento de cada una de estas y obteniendo resultados satisfactorios.

3.7.2 Pruebas de Aceptación

Las pruebas de aceptación son creadas en base a las HU, en cada ciclo de la iteración del desarrollo. El cliente debe especificar uno o diversos escenarios para comprobar que una historia de usuario ha sido correctamente implementada. Las pruebas de aceptación son consideradas como "pruebas de caja negra" ("*Black box systemtests*"). Los clientes son responsables de verificar que los resultados de estas pruebas sean correctos. Así mismo, en caso de que fallen varias pruebas, deben indicar el orden de prioridad de resolución.

Una HU no se puede considerar terminada hasta tanto pase correctamente todas las pruebas de aceptación. Dado que la responsabilidad es grupal, es recomendable publicar los resultados de las pruebas de aceptación, de manera que todo el equipo esté al tanto de esta información (45).

A continuación se exponen las pruebas de aceptación utilizadas para probar el sistema divididas en las iteraciones correspondientes a cada historia de usuario:

3.7.2.1 Iteración 1

Tabla 33. Descripción de la Prueba de Aceptación 1 HU Autenticar Usuario

Caso de Prueba de Aceptación	
Código: HU1_p1	**Historia de Usuario (No.1):** Autenticar Usuario.

Nombre: Entrada al sistema con datos erróneos.
Descripción: Se desea probar que el sistema no de acceso a un usuario con datos erróneos.
Condiciones de ejecución: Juego de datos incorrectos o incompletos.
Entrada/ Pasos de ejecución: Datos incorrectos. El sistema comprueba que los datos estén correctos para dar acceso al usuario.
Resultado esperado: Mostrar un mensaje de error: Nombre de usuario o Contraseña inválido
Evaluación de la prueba: Satisfactoria.

Tabla 34. Descripción de la Prueba de Aceptación 2 HU Autenticar Usuario

Caso de Prueba de Aceptación	
Código: HU1_p2	**Historia de Usuario (No.1):** Autenticar Usuario.
Nombre: Entrada al sistema con datos válidos.	
Descripción: Se desea probar que el sistema de acceso a un usuario.	
Condiciones de ejecución: Juego de datos válidos de autenticación de usuario.	
Entrada/ Pasos de ejecución: El sistema comprueba que los datos estén correctos para dar acceso al usuario.	
Resultado esperado: Entrada al sistema del usuario esperado con el rol que le pertenece.	
Evaluación de la prueba: Satisfactoria.	

Tabla 35. Descripción de la Prueba de Aceptación 1 HU Gestionar Estudiante

Caso de Prueba de Aceptación	
Código: HU2_p1	**Historia de Usuario (No.2):** Gestionar Estudiante.
Nombre: Adicionar un estudiante con juego de datos incompletos.	
Descripción: Probar que no se puede adicionar un estudiante con datos incompletos o incorrectos.	
Condiciones de ejecución: Deben existir errores en el juego de datos del nuevo estudiante.	
Entrada/ Pasos de ejecución: Juego de datos con casillas en blanco y datos con errores, se intenta que el sistema adicione al estudiante.	
Resultado esperado: El sistema debe mostrar un mensaje de error.	
Evaluación de la prueba: Satisfactoria.	

En los Anexos se encuentran las demás descripciones las Pruebas de Aceptación que pertenecen a esta iteración.

3.7.2.2 Iteración 2

Tabla 36. Descripción de la Prueba de Aceptación 1 HU Gestionar Tipo prueba

Caso de Prueba de Aceptación	
Código: HU8_p1	**Historia de Usuario (No.8):** Gestionar Tipo prueba
Nombre: Adicionar un tipo de prueba con juego de datos incompletos.	
Descripción: Probar que no se puede adicionar un tipo de prueba con datos incompletos o incorrectos.	
Condiciones de ejecución: Deben existir errores en el juego de datos del nuevo tipo de prueba.	
Entrada/ Pasos de ejecución: Juego de datos con casillas en blanco y datos con errores, se intenta que el sistema adicione un tipo de prueba.	
Resultado esperado: El sistema debe mostrar un mensaje de error.	
Evaluación de la prueba: Satisfactoria.	

Tabla 37. Descripción de la Prueba de Aceptación 3 HU Gestionar Tipo prueba

Caso de Prueba de Aceptación	
Código: HU8_p3	**Historia de Usuario (No.2):** Gestionar Tipo prueba
Nombre: Modificar datos de un tipo de prueba con juego de datos correctos.	
Descripción: Se desea probar que el sistema modifique los datos del tipo de prueba correctamente.	
Condiciones de ejecución: Se cambian los datos del tipo de prueba en el sistema.	
Entrada/ Pasos de ejecución: Juego de datos correctos que modifiquen un tipo de prueba existente.	
Resultado esperado: Se actualice el listado de los tipos de pruebas con los cambios realizados.	
Evaluación de la prueba: Satisfactoria.	

Tabla 38. Descripción de la Prueba de Aceptación 4 HU Gestionar Prueba

Caso de Prueba de Aceptación	
Código: HU7_p4	**Historia de Usuario (No.7):** Gestionar Prueba
Nombre: Eliminar Prueba.	
Descripción: Se desea probar que el sistema elimine satisfactoriamente una prueba.	
Condiciones de ejecución: Se accede a la opción de eliminar prueba.	
Entrada/ Pasos de ejecución: Seleccionar la prueba a eliminar.	
Resultado esperado: Se actualice el listado de las pruebas.	

Evaluación de la prueba: Satisfactoria.

En los Anexos se encuentran las demás descripciones las Pruebas de Aceptación que pertenecen a esta iteración.

3.7.2.3 Iteración 3

Tabla 39. Descripción de la Prueba de Aceptación 1 HU Asignar Ejercicios

Caso de Prueba de Aceptación	
Código: HU9_p1	**Historia de Usuario (No.9):** Asignar Ejercicios
Nombre: Asignar Ejercicios.	
Descripción: Se desea probar que el sistema asigne ejercicios al grupo.	
Condiciones de ejecución: Se accede a la opción de Asignar Ejercicios.	
Entrada/ Pasos de ejecución: Seleccionar el grupo al que se le va a asignar ejercicios.	
Resultado esperado: Se muestra una guía de ejercicios a asignar para el grupo.	
Evaluación de la prueba: Satisfactoria.	

3.8 Resultados de las pruebas

Partiendo de las pruebas de aceptación, se realizaron 3 iteraciones para probar el correcto funcionamiento del sistema. Se detectaron un total de 9 no conformidades significativas, 6 no significativas y 6 recomendaciones.

Las pruebas aplicadas contribuyeron a mejorar la calidad y las funcionalidades del sistema donde se arrojaron resultados visibles. A continuación se muestras estos resultados:

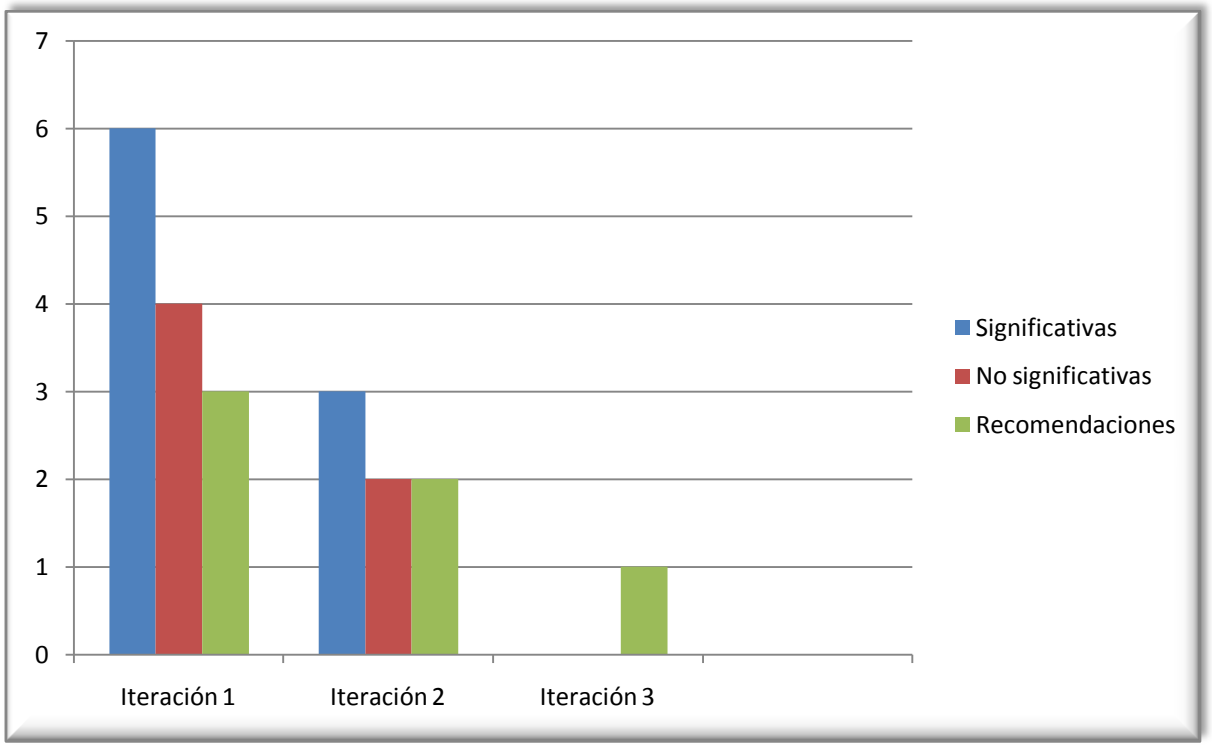

Figura 4: Gráfico de las no conformidades.

Una vez concluida cada iteración, se corrigieron cada una de las no conformidades encontradas por lo que luego de la tercera iteración no se encontraron no conformidades culminándose así las pruebas al sistema.

3.9 Conclusiones

En el presente capítulo se planteó el modelo de datos del sistema, se describieron las tareas de la ingeniería para darle solución a las HU además de las pruebas de aceptación que se llevaron a cabo con el objetivo de brindar un producto que cumpla con las necesidades del cliente. Con la conclusión de este capítulo se considera terminada la propuesta de solución del sistema.

Conclusiones generales

Después de elaborar el SGI para los resultados de las pruebas de eficiencia física en la UCI, se llegó a las siguientes conclusiones:

- Los métodos científicos utilizados permitieron desarrollar los conceptos y teorías que sustenta la investigación para el desarrollo de la solución.
- La selección de la metodología de desarrollo, lenguaje de programación, tecnologías y herramientas posibilitaron al equipo de desarrollo cumplir con el objetivo general de la investigación.
- La implementación del SGI posibilitó obtener un producto funcional capaz de satisfacer las necesidades del cliente.
- La validación de la solución demostró el correcto funcionamiento del sistema.

Recomendaciones

Como parte del proceso de desarrollo de la investigación se recomiendan los siguientes aspectos:

- A los administradores de la aplicación, mantener actualizados los servicios UCI que se brindan para los estudiantes, profesores y grupos.
- Adicionarle nuevas funcionalidades para un mejor empleo del Sistema. Ejemplo de estas funcionalidades son exportar a pdf los ejercicios asignados, control de asistencias a clases y promedio de evaluaciones.

Bibliografía

1. **Ecured.** Conocimiento con todos y para todos. [En línea] [Citado el: 13 de junio de 2013.] http://www.ecured.cu/index.php/Gesti%C3%B3n_de_la_Informaci%C3%B3n.

2. **Universidad del Cauca.** Conceptos básicos de los sistemas de información. [En línea] [Citado el: 13 de junio de 2013.] http://fccea.unicauca.edu.co/old/siconceptosbasicos.htm.

3. ¿Qué son los sistemas de gestión? - Bsi. [En línea] [Citado el: 13 de junio de 2013.] http://www.bsigroup.com.mx/es-mx/Auditoria-y-Certificacion/Sistemas-de-Gestion/De-un-vistazo/Que-son-los-sistemas-de-gestion/.

4. **Writing, Alexis.** Desventajas de los sistemas de información computarizada - eHow en Español. [En línea] [Citado el: 13 de junio de 2013.] http://www.ehowenespanol.com/desventajas-sistemas-informacion-computarizada-info_91378/.

5. **Vega, Lic. Brian Hernández.** *Diseño de prototipo de un sistema de gestión de la información para el grupo de trabajo de alimentos y bebidas del hotel "Brisas Santa Lucia".* Universidad de Camagüey: Centro de Estudios Multidisciplinarios del Turismo, 2007.

6. **Ramakrishnan, Raghu y Gehrke, Johannes.** *Database Management Systems. Second Edition.* s.l.: Editorial Mac-Graw Hill Higher Education. ISBN: 0-07-246535-2.

7. **Ruesta, Bustelo, García, C. Elisa y Huidobro, Morales.** Tendencias en la Gestión de la información, la documentación y el Conocimiento en las organizaciones. *El Profesional de la Información.* [En línea] diciembre de 2001. [Citado el: 21 de junio de 2013.] http://www.inforarea.es. vol. 10, n. 12.

8. **Sipref.** Sistema Informático para profesores de educación física. [En línea] [Citado el: 13 de junio de 2013.] http://www.softwhere.com.ar/.

9. **Woplanner.** Software de Entrenamiento Deportivo. [En línea] 2 de mayo de 2007. [Citado el: 13 de junio de 2013.] http://woplanner.fullblog.com.ar/.

10. **Pérez González, Rodrigo, Carrillo Pérez, Isaías y Rodríguez Martín, Aureliano David.** *Metodología de Desarrollo del Software.* 2008.

11. **Patricio Letelier, Mª Carmen Penadés José H. Canó.** *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP).* Universidad Politécnica de Valencia. : s.n.

12. **Palacios, Juan.** *ScrumManager: Gestión de proyectos.* Septiembre - 2008.

13. Procesos de Software - Metodología Extreme Programming(XP). [En línea] [Citado el: 20 de marzo de 2013.] http://procesosdesoftware.wikispaces.com/METODOLOGIA+XP.

14. **Thiele Diseño.** Aplicaciones Web vs Aplicaciones de Escritorio. [En línea] 24 de octubre de 2011. [Citado el: 14 de junio de 2013.] http://loretothiele.blogspot.com/2011/10/aplicaciones-web-vs-aplicaciones-de.html.

15. **García de Jalón, Javier.** *Aprenda Java como si estuviera en primero.* San Sebastián : s.n., 2000.

16. **Villalobos, Jorge.** Introducción a Java - codigoprogramacion.com. [En línea] 5 de septiembre de 2010. [Citado el: 14 de junio de 2013.] http://codigoprogramacion.com/cursos/java/47-introjava.html.

17. **La Revista Informática.com.** Lenguaje de Programación ASP. [En línea] [Citado el: 12 de junio de 2013.] http://www.larevistainformatica.com/ASP.htm.

18. **Pérez, Javier Eguíluz.** *Introducción a CSS.* 2009.

19. —. *Introducción a JavaScript.* 2009.

20. **Castillo Cantón, Alejandro.** *Manual de HTML5 en español.*

21. **Alvarez, Miguel Angel.** Novedades de HTML 5 - ¿ Qué es HTML 5? [En línea] 14 de octubre de 2009. [Citado el: 2 de mayo de 2013.] http://www.desarrolloweb.com/articulos/que-es-html5.html.

22. Definición de Servidor IIS. [En línea] [Citado el: 13 de junio de 2013.] http://sauce.pntic.mec.es/crer0052/iis/definici.htm.

23. Características de IIS. [En línea] [Citado el: 28 de mayo de 2013.] http://es.scribd.com/doc/27519905/8/Caracteristicas-de-IIS.

24. **López, Álvaro.** *Cherokee Web server.* 2003.

25. **Bueno, Leonardo Bernal y Madrena Lucenilla, Antonio.** Servidor web Cherokee - slideshare. [En línea] http://www.slideshare.net/mcdrena/cherokee-11271119.

26. Servidor HTTP Apache - Ecured. [En línea] [Citado el: 14 de enero de 2013.] http://www.ecured.cu/index.php/Servidor_HTTP_Apache.

27. Ciberaula. [En línea] [Citado el: 23 de enero de 2013.] http://linux.ciberaula.com/articulo/linux_apache_intro.

28. **BERTINO, E. A. y MARTINO, L. A.** *Sistemas de bases de datos orientadas a objetos.* s.l. : Ediciones Díaz de Santos, 1995.

29. Ecured. Sistema Gestor de Base de Datos. [En línea] [Citado el: 9 de abril de 2013.] http://www.ecured.cu/index.php/Sistema_Gestor_de_Base_de_Datos.

30. **Todo expertos.** Ventajas y desventajas de Oracle. [En línea] [Citado el: 13 de junio de 2013.] http://www.todoexpertos.com/categorias/tecnologia-e-internet/bases-de-datos/oracle/respuestas/14706/vetajas-y-desventajas..

31. Postgresql-es. [En línea] 2 de octubre de 2010. [Citado el: 12 de enero de 2013.] http://www.postgresql.org.es/sobre_postgresql.
32. Bienvenido a NetBeans y www.netbeans.org, Portal del IDE Java de Código Abierto. [En línea] [Citado el: 11 de enero de 2013.] http://netbeans.org/index_es.html.
33. scribd. [En línea] [Citado el: 5 de mayo de 2013.] http://es.scribd.com/doc/8716954/e-Learning..
34. **Alvarez, Miguel Angel.** desarrolloweb.com- ¿qué es un CMS? [En línea] 11 de noviembre de 2008. [Citado el: 24 de febrero de 2013.] http://www.desarrolloweb.com/articulos/que-es-un-cms.html.
35. **Alfonso, X. C. G. Y. J. M. Mosaic.** Introducción a los Sistemas de Gestión de Contenido. . [En línea] [Citado el: 5 de mayo de 2013.] http://mosaic.uoc.edu/articulos/cms1204.html.
36. **Cisneros, Cirley y Tupe, Juan.** *Evaluación y Selección de Framework de Desarrollo PHP:Symfony,Kumbia,CakePHP y Zend.*
37. **Fabien Potencier, Francois Zaninotto.** *Symfony la guía definitiva.*
38. KICKBILL. *Kickbill.Tus primeros pasos con Zend Framework: Parte 1.* [En línea] 2009. [Citado el: 24 de marzo de 2013.] http://www.kickbill.com/?p=1232.
39. **Álvarez, Miguel Angel.** Desarrolloweb - CodeIgniter. [En línea] 2009. [Citado el: 25 de marzo de 2013.] http://www.desarrolloweb.com/articulos/codeigniter.html.
40. **Argulo, Iván y Campos, Emilio.** Comunidad CodeIgniter - Usando CodeIgniter en un proyecto real. [En línea] 13 de febrero de 2009. [Citado el: 10 de abril de 2013.] http://comunidadcodeigniter.wordpress.com/.
41. Comunidad de software libre Paraguay - Framework de Desarrollo . [En línea] [Citado el: 24 de marzo de 2013.] http://www.pti.org.py/csl/index.php.
42. CooKbook. [En línea] 2009. [Citado el: 25 de marzo de 2013.] http://book.cakephp.org/es/.
43. **Chávez, Mallelin Bolufe.** Monografias.com - Frameworks para el desarrollo de aplicaciones con PHP. [En línea] mayo de 2009. http://www.monografias.com/trabajos70/frameworks-desarrollo-aplicaciones-php/frameworks-desarrollo-aplicaciones-php.shtml.
44. Ecured - Usuario (Informática). [En línea] [Citado el: 20 de marzo de 2013.] http://www.ecured.cu/index.php/Usuario_%28Inform%C3%A1tica%29.
45. **Joskowicz, José.** *Reglas y Prácticas en eXtremeProgramming.* España : s.n., 2008.
46. **Beck, Kent.** *Extreme Programming Explained.* s.l. : 1 edición s.l. : Addison-Wesley Pub Co, 1999.

47. **Carvajal Riola, José Carlos.** *Metodologías ágiles: Herramientas y modelo de desarrollo para aplicaciones Java EE como métodología empresarial.* 2008.

48. Prototipo de la Interfaz de Usuario - MeRinde. [En línea] [Citado el: 15 de junio de 2013.] http://merinde.net/index.php?option=com_content&task=view&id=490&Itemid=291.

49. **Jeffries, Ron, Anderson, Ann y Hendrickson, Chet.** *Extreme Programming Installed.* s.l. : Addison Wesley, 2000.

50. **Larman, C.** *UML y patrones.* Tomo I Capítulos 18, Páginas 185-215.

51. **PBworks.** Patrones arquitectónicos. [En línea] [Citado el: 14 de junio de 2013.] http://isg3.pbworks.com/w/page/7624479/Patrones%20Arquitect%C3%B3nicos.

52. **Henney, Kevlin.** *¿What is Software Architecture?* 2007.

53. **Unidad Docente de Ingeniería del Software. Facultad de informática .** *Patrones del "Gang of Four".* Madrid : s.n.

54. **Arlee Enedina Paneque García, Noel González Fernández.** *Sistema de Gestión de Auditoría de la Dirección de Supervisión y Control de la Universidad de las Ciencias Informáticas.* Ciudad de la Habana : s.n., 2010.

Glosario de términos

AJAX (*Asynchronous JavaScript And XML)*: (*JavaScript* asíncrono y XML), es una técnica de desarrollo *web* para crear aplicaciones interactivas.

CRUD: Es el acrónimo de Crear, Obtener, Actualizar y Borrar (*Create, Retrieve, Update* y *Delete* en inglés).

FTP (*File Transfer Protocol*): (Protocolo de Transferencia de Archivos) es un protocolo de red para la transferencia de archivos entre sistemas conectados a una red TCP (*Transmission Control Protocol*), basada en la arquitectura cliente-servidor.

MVC (Modelo-Vista-Controlador): es un estilo de arquitectura de *software* que separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos.

ORM (mapeo objeto-relacional): (*Object-Relationalmapping*, o sus siglas O/RM, ORM y O/R *mapping*) es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional**.**

PHP*:* *Hypertext* Pre-*processor*. Es un ambiente script del lado del servidor que permite crear y ejecutar aplicaciones *web* dinámicas e interactivas.

PostgreSQL: Sistema de gestión de base de datos relacional orientada a objetos de *software* libre, publicado bajo la licencia BSD.

SQL (*Structured Query Language*): (lenguaje de consulta estructurado) es un lenguaje declarativo de acceso a bases de datos relacionales.

Anexos

Anexo 1. Diferencias entre las Metodologías ágiles y tradicionales.

Metodologías Ágiles	Metodologías Tradicionales
Basadas en heurísticas provenientes de prácticas de producción de código	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo
Especialmente preparados para cambios durante el proyecto	Cierta resistencia a los cambios
Impuestas internamente (por el equipo)	Impuestas externamente
Proceso menos controlado, con pocos principios	Proceso mucho más controlado, con numerosas políticas/normas
No existe contrato tradicional o al menos es bastante flexible	Existe un contrato prefijado
El cliente es parte del equipo de desarrollo	El cliente interactúa con el equipo de desarrollo mediante reuniones
Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio	Grupos grandes y posiblemente distribuidos
Pocos artefactos	Más artefactos
Pocos roles	Más roles
Menos énfasis en la arquitectura del *software*	La arquitectura del *software* es esencial y se expresa mediante modelos

Anexo 2. Entrevista al profesor de Educación Física acerca de las pruebas de eficiencia física en la UCI.

1. **Según el plan de clases de la UCI, cómo se realizan las pruebas de eficiencia física.**

Las pruebas de eficiencia física la realizan los estudiantes de 1ro y 2do año de la carrera. Se tratará en todos los casos de realizar las pruebas en ropa deportiva con zapatos tenis para poder obtener el máximo de confiabilidad en las mismas. Estas pruebas consistes en planchas (30 s), abdominales (30 s), velocidad (50 m), resistencia (1000 m) y salto de longitud.

2. **¿Cómo se evalúan los datos obtenidos en las pruebas de eficiencia física para asignar ejercicios?**

El profesor determinará los niveles por pruebas y general de sus alumnos por los resultados en cada prueba por edad y sexo de acuerdo al Plan de Normativas de Eficiencia Física INDER - 2000.

3. **¿Cuáles son los ejercicios físicos que se asignarían para mejorar los resultados alcanzados por los estudiantes en las pruebas de eficiencia física?**

Los ejercicios a asignar para mejorar los resultados alcanzados en las pruebas de eficiencia física son:

- Caminar en carretilla
- Planchas
- Cangrejo
- Cuclillas
- Salto de rana
- Tramos de velocidad
- Carrera continua de corta duración hasta 8 min
- Carrera continúa de larga duración

Tareas de la Ingeniería

Tabla 40. Descripción de la Tarea de la Ingeniería 3 Eliminar Estudiante

Número de Tarea: 3	**HU (Nro. 2:)**Gestionar Estudiante
Nombre de la Tarea: Eliminar Estudiante	
Tipo Tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha Inicio: 9 de febrero 2013	**Fecha Fin:** 10 de febrero 2013
Programador Responsable: Roger A. González Rodríguez y Victor A. Fong Rios	
Descripción: Se muestra una interfaz donde se elimina un estudiante de la base de datos.	

Tabla 41. Descripción de la Tarea de la Ingeniería 4 Listar Estudiante

Número de Tarea: 4	**HU (Nro. 2:)**Gestionar Estudiante
Nombre de la Tarea: Listar Estudiante	
Tipo Tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha Inicio: 9 de febrero 2013	**Fecha Fin:**10 de febrero 2013
Programador Responsable: Roger A. González Rodríguez y Victor A. Fong Rios	
Descripción: Se muestra una interfaz donde se listan los estudiantes.	

Tabla 42. Descripción de la Tarea de la Ingeniería 1 Adicionar Grupo

Número de Tarea: 1	**HU (Nro. 3:)** Gestionar Grupo

Nombre de la Tarea: Adicionar Grupo	
Tipo Tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha Inicio: 11 de febrero 2013	**Fecha Fin:** 12 de febrero 2013
Programador Responsable: Roger A. González Rodríguez y Victor A. Fong Rios	
Descripción: Se muestra una interfaz donde se adiciona un grupo con sus datos a la base de datos.	

Tabla 43. Descripción de la Tarea de la Ingeniería 2 Modificar Grupo

Número de Tarea: 2	**HU (Nro. 3:)** Gestionar Grupo
Nombre de la Tarea: Modificar Grupo	
Tipo Tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha Inicio: 11 de febrero 2013	**Fecha Fin:** 12 de febrero 2013
Programador Responsable: Roger A. González Rodríguez y Victor A. Fong Rios	
Descripción: Se muestra una interfaz donde se modifica un grupo de la base de datos.	

Tabla 44. Descripción de la Tarea de la Ingeniería 3 Eliminar Grupo

Número de Tarea: 3	**HU (Nro. 3:)** Gestionar Grupo
Nombre de la Tarea: Eliminar Grupo	
Tipo Tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha Inicio: 11 de febrero 2013	**Fecha Fin:** 12 de febrero 2013
Programador Responsable: Roger A. González Rodríguez y Victor A. Fong Rios	
Descripción: Se muestra una interfaz donde se elimina un grupo de la base de datos.	

Tabla 45. Descripción de la Tarea de la Ingeniería 4 Listar Grupo

Número de Tarea: 4	**HU (Nro. 3:)** Gestionar Grupo
Nombre de la Tarea: Listar Grupo	
Tipo Tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha Inicio: 11 de febrero 2013	**Fecha Fin:** 12 de febrero 2013
Programador Responsable: Roger A. González Rodríguez y Victor A. Fong Rios	
Descripción: Se muestra una interfaz donde se listan los grupos de la base de datos.	

Tabla 46. Descripción de la Tarea de la Ingeniería 1 Adicionar Profesor

Número de Tarea: 1	**HU (Nro. 4:)** Gestionar Profesor

Nombre de la Tarea: Adicionar Profesor	
Tipo Tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha Inicio: 13 de febrero 2013	**Fecha Fin:** 14 de febrero 2013
Programador Responsable: Roger A. González Rodríguez y Victor A. Fong Rios	
Descripción: Se muestra una interfaz donde se adiciona un profesor con sus datos a la base de datos.	

Tabla 47. Descripción de la Tarea de la Ingeniería 2 Modificar Profesor

Número de Tarea: 2	**HU (Nro. 4:)** Gestionar Profesor
Nombre de la Tarea: Modificar Profesor	
Tipo Tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha Inicio: 13 de febrero 2013	**Fecha Fin:** 14 de febrero 2013
Programador Responsable: Roger A. González Rodríguez y Victor A. Fong Rios	
Descripción: Se muestra una interfaz donde se modifica un profesor de la base de datos.	

Tabla 48. Descripción de la Tarea de la Ingeniería 3 Eliminar Profesor

Número de Tarea: 3	**HU (Nro. 4:)** Gestionar Profesor
Nombre de la Tarea: Eliminar Profesor	
Tipo Tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha Inicio: 13 de febrero 2013	**Fecha Fin:** 14 de febrero 2013
Programador Responsable: Roger A. González Rodríguez y Victor A. Fong Rios	
Descripción: Se muestra una interfaz donde se elimina un profesor de la base de datos.	

Tabla 49. Descripción de la Tarea de la Ingeniería 4 Listar Profesor

Número de Tarea: 4	**HU (Nro. 4:)** Gestionar Profesor
Nombre de la Tarea: Eliminar Profesor	
Tipo Tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha Inicio:13 de febrero 2013	**Fecha Fin:** 14 de febrero 2013
Programador Responsable: Roger A. González Rodríguez y Victor A. Fong Rios	
Descripción: Se muestra una interfaz donde se listan los profesores de la base de datos.	

Tabla 50. Descripción de la Tarea de la Ingeniería 1 Adicionar Tipo Prueba

Número de Tarea: 1	**HU (Nro. 8:)** Gestionar Tipo prueba

Nombre de la Tarea: Adicionar Tipo prueba	
Tipo Tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha Inicio: 17 de febrero 2013	**Fecha Fin:** 18 de febrero 2013
Programador Responsable: Roger A. González Rodríguez y Victor A. Fong Rios	
Descripción: Se muestra una interfaz donde se adiciona un tipo de prueba con sus datos a la base de datos.	

Tabla 51. Descripción de la Tarea de la Ingeniería 2 Modificar Tipo Prueba

Número de Tarea: 2	**HU (Nro. 8:)** Gestionar Tipo prueba
Nombre de la Tarea: Modificar Tipo prueba	
Tipo Tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha Inicio: 17 de febrero 2013	**Fecha Fin:** 18 de febrero 2013
Programador Responsable: Roger A. González Rodríguez y Victor A. Fong Rios	
Descripción: Se muestra una interfaz donde se modifica un tipo de prueba de la base de datos.	

Tabla 52. Descripción de la Tarea de la Ingeniería 3 Eliminar Tipo Prueba

Número de Tarea: 3	**HU (Nro. 8:)** Gestionar Tipo prueba
Nombre de la Tarea: Eliminar Tipo prueba	
Tipo Tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha Inicio: 17 de febrero 2013	**Fecha Fin:** 18 de febrero 2013
Programador Responsable: Roger A. González Rodríguez y Victor A. Fong Rios	
Descripción: Se muestra una interfaz donde se elimina un tipo de prueba de la base de datos.	

Tabla 53. Descripción de la Tarea de la Ingeniería 3 Listar Tipo Prueba

Número de Tarea: 4	**HU (Nro. 8:)** Gestionar Tipo prueba
Nombre de la Tarea: Listar Tipo prueba	
Tipo Tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha Inicio: 17 de febrero 2013	**Fecha Fin:** 18 de febrero 2013
Programador Responsable: Roger A. González Rodríguez y Victor A. Fong Rios	
Descripción: Se muestra una interfaz donde se listan los tipos de pruebas de la base de datos.	

Tabla 54. Descripción de la Tarea de la Ingeniería 1 Adicionar Ejercicio

Número de Tarea: 1	**HU (Nro. 6:)** Gestionar Ejercicio
Nombre de la Tarea: Adicionar Ejercicio	
Tipo Tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha Inicio: 21 de febrero 2013	**Fecha Fin:** 22 de febrero 2013
Programador Responsable: Roger A. González Rodríguez y Victor A. Fong Rios	
Descripción: Se muestra una interfaz donde se adiciona un ejercicio con sus datos a la base de datos.	

Tabla 55. Descripción de la Tarea de la Ingeniería 2 Modificar Ejercicio

Número de Tarea: 2	**HU (Nro. 6:)** Gestionar Ejercicio
Nombre de la Tarea: Modificar Ejercicio	
Tipo Tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha Inicio: 21 de febrero 2013	**Fecha Fin:** 22 de febrero 2013
Programador Responsable: Roger A. González Rodríguez y Victor A. Fong Rios	
Descripción: Se muestra una interfaz donde se modifica un ejercicio de la base de datos.	

Tabla 56. Descripción de la Tarea de la Ingeniería 3 Eliminar Ejercicio

Número de Tarea: 3	**HU (Nro. 6:)** Gestionar Ejercicio
Nombre de la Tarea: Eliminar Ejercicio	
Tipo Tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha Inicio: 21 de febrero 2013	**Fecha Fin:** 22 de febrero 2013
Programador Responsable: Roger A. González Rodríguez y Victor A. Fong Rios	
Descripción: Se muestra una interfaz donde se elimina un ejercicio de la base de datos.	

Tabla 57. Descripción de la Tarea de la Ingeniería 4 Listar Ejercicio

Número de Tarea: 4	**HU (Nro. 6:)** Gestionar Ejercicio
Nombre de la Tarea: Listar Ejercicio	
Tipo Tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha Inicio: 21 de febrero 2013	**Fecha Fin:** 22 de febrero 2013
Programador Responsable: Roger A. González Rodríguez y Victor A. Fong Rios	
Descripción: Se muestra una interfaz donde se listan los ejercicios de la base de datos.	

Pruebas de Aceptación

Iteración 1

Tabla 58. Descripción de la Prueba de Aceptación 3 HU Gestionar Estudiante

Caso de Prueba de Aceptación	
Código: HU2_p3	**Historia de Usuario (No.2):** Gestionar Estudiante.
Nombre: Modificar datos de un estudiante con juego de datos correctos.	
Descripción: Se desea probar que el sistema modifica los datos del estudiante correctamente.	
Condiciones de ejecución: Se cambian los datos de los estudiantes en el sistema.	
Entrada/ Pasos de ejecución: Juego de datos correctos que modifiquen un estudiante existente.	
Resultado esperado: Se actualice el listado de estudiantes con los cambios realizados.	
Evaluación de la prueba: Satisfactoria.	

Tabla 59. Descripción de la Prueba de Aceptación 4 HU Gestionar Estudiante

Caso de Prueba de Aceptación	
Código: HU2_p4	**Historia de Usuario (No.2):** Gestionar Estudiante.
Nombre: Eliminar Estudiante.	
Descripción: Se desea probar que el sistema elimine satisfactoriamente un estudiante.	
Condiciones de ejecución: Se accede a la opción de eliminar estudiante.	
Entrada/ Pasos de ejecución: Seleccionar el usuario a eliminar.	
Resultado esperado: Se actualice el listado de estudiantes.	
Evaluación de la prueba: Satisfactoria.	

Tabla 60. Descripción de la Prueba de Aceptación 3 HU Gestionar Grupo

Caso de Prueba de Aceptación	
Código: HU3_p3	**Historia de Usuario (No.3):** Gestionar Grupo
Nombre: Modificar datos de un grupo con juego de datos correctos.	
Descripción: Se desea probar que el sistema modifica los datos del grupo correctamente.	
Condiciones de ejecución: Se cambian los datos del grupo en el sistema.	
Entrada/ Pasos de ejecución: Juego de datos correctos que modifiquen un grupo existente.	
Resultado esperado: Se actualice el listado de grupos con los cambios realizados.	

Evaluación de la prueba: Satisfactoria.

Tabla 61. Descripción de la Prueba de Aceptación 4 HU Gestionar Grupo

Caso de Prueba de Aceptación	
Código: HU3_p4	**Historia de Usuario (No.3):** Gestionar Grupo
Nombre: Eliminar Grupo.	
Descripción: Se desea probar que el sistema elimine satisfactoriamente un grupo.	
Condiciones de ejecución: Se accede a la opción de eliminar grupo.	
Entrada/ Pasos de ejecución: Seleccionar el grupo o los grupos a eliminar.	
Resultado esperado: Se actualice el listado de estudiantes.	
Evaluación de la prueba: Satisfactoria	

Tabla 62. Descripción de la Prueba de Aceptación 1 HU Gestionar Profesor

Caso de Prueba de Aceptación	
Código: HU4_p1	**Historia de Usuario (No.4):** Gestionar Profesor
Nombre: Adicionar un profesor con juego de datos incompletos.	
Descripción: Probar que no se puede adicionar un profesor con datos incompletos o incorrectos.	
Condiciones de ejecución: Deben existir errores en el juego de datos del nuevo profesor.	
Entrada/ Pasos de ejecución: Juego de datos con casillas en blanco y datos con errores, se intenta que el sistema adicione al profesor.	
Resultado esperado: El sistema debe mostrar un mensaje de error.	
Evaluación de la prueba: Satisfactoria	

Tabla 63. Descripción de la Prueba de Aceptación 3 HU Gestionar Profesor

Caso de Prueba de Aceptación	
Código: HU4_p3	**Historia de Usuario (No.4):** Gestionar Profesor
Nombre: Modificar datos de un profesor con juego de datos correctos.	
Descripción: Se desea probar que el sistema modifica los datos del profesor correctamente.	
Condiciones de ejecución: Se cambian los datos de los profesores en el sistema**.**	
Entrada/ Pasos de ejecución: Juego de datos correctos que modifiquen un profesor existente.	
Resultado esperado: Se actualice el listado de profesores con los cambios realizados.	
Evaluación de la prueba: Satisfactoria	

Tabla 64. Descripción de la Prueba de Aceptación 4 HU Gestionar Profesor

Caso de Prueba de Aceptación	
Código: HU4_p4	**Historia de Usuario (No.2):** Gestionar Profesor
Nombre: Eliminar Profesor.	
Descripción: Se desea probar que el sistema elimine satisfactoriamente un profesor.	
Condiciones de ejecución: Se accede a la opción de eliminar profesor.	
Entrada/ Pasos de ejecución: Seleccionar el profesor o los profesores a eliminar.	
Resultado esperado: Se actualice el listado de profesores.	
Evaluación de la prueba: Satisfactoria	

Tabla 65. Descripción de la Prueba de Aceptación 1 HU Gestionar Ejercicio

Caso de Prueba de Aceptación	
Código: HU6_p1	**Historia de Usuario (No.6):** Gestionar Ejercicio
Nombre: Adicionar un ejercicio con juego de datos incompletos.	
Descripción: Probar que no se puede adicionar un ejercicio con datos incompletos o incorrectos.	
Condiciones de ejecución: Deben existir errores en el juego de datos del nuevo ejercicio.	
Entrada/ Pasos de ejecución: Juego de datos con casillas en blanco y datos con errores, se intenta que el sistema adicione al ejercicio.	
Resultado esperado: El sistema debe mostrar un mensaje de error.	
Evaluación de la prueba: Satisfactoria.	

Tabla 66. Descripción de la Prueba de Aceptación 2 HU Gestionar Ejercicio

Caso de Prueba de Aceptación	
Código: HU6_p2	**Historia de Usuario (No.6):** Gestionar Ejercicio
Nombre: Adicionar un ejercicio repetido.	
Descripción: Se desea probar que no se puede adicionar un ejercicio repetido.	
Condiciones de ejecución: Insertar juego de datos de un ejercicio que ya existe y se intenta que el sistema cree un nuevo estudiante con este juego de datos.	
Entrada/ Pasos de ejecución: Juego de datos de un ejercicio ya existente y que esté correcto.	
Resultado esperado: El sistema debe mostrar un mensaje de error.	
Evaluación de la prueba: Satisfactoria.	

Tabla 67. Descripción de la Prueba de Aceptación 3 HU Gestionar Ejercicio

Caso de Prueba de Aceptación	
Código: HU6_p3	**Historia de Usuario (No.6):** Gestionar Ejercicio
Nombre: Modificar datos de un ejercicio con juego de datos correctos.	
Descripción: Se desea probar que el sistema modifica los datos del ejercicio correctamente.	
Condiciones de ejecución: Se cambian los datos del ejercicio en el sistema.	
Entrada/ Pasos de ejecución: Juego de datos correctos que modifiquen un ejercicio existente.	
Resultado esperado: Se actualice el listado de ejercicios con los cambios realizados.	
Evaluación de la prueba: Satisfactoria.	

Tabla 68. Descripción de la Prueba de Aceptación 4 HU Gestionar Ejercicio

Caso de Prueba de Aceptación	
Código: HU6_p4	**Historia de Usuario (No.6):** Gestionar Ejercicio
Nombre: Eliminar Ejercicio.	
Descripción: Se desea probar que el sistema elimine satisfactoriamente un ejercicio.	
Condiciones de ejecución: Se accede a la opción de eliminar ejercicio.	
Entrada/ Pasos de ejecución: Seleccionar el ejercicio a eliminar.	
Resultado esperado: Se actualice el listado de ejercicios.	
Evaluación de la prueba: Satisfactoria.	

Tabla 69. Descripción de la Prueba de Aceptación 1 HU Gestionar Prueba

Caso de Prueba de Aceptación	
Código: HU7_p1	**Historia de Usuario (No.7):** Gestionar Prueba
Nombre: Adicionar una prueba con juego de datos incompletos.	
Descripción: Probar que no se puede adicionar una prueba con datos incompletos o incorrectos.	
Condiciones de ejecución: Deben existir errores en el juego de datos de la nueva prueba.	
Entrada/ Pasos de ejecución: Juego de datos con casillas en blanco y datos con errores, se intenta que el sistema adicione la prueba.	
Resultado esperado: El sistema debe mostrar un mensaje de error.	
Evaluación de la prueba: Satisfactoria.	

Tabla 70. Descripción de la Prueba de Aceptación 2 HU Gestionar Prueba

Caso de Prueba de Aceptación	
Código: HU7_p2	**Historia de Usuario (No.2):** Gestionar Prueba
Nombre: Adicionar una prueba repetida.	
Descripción: Se desea probar que no se puede adicionar una prueba repetida.	
Condiciones de ejecución: Insertar juego de datos de una prueba y se intenta que el sistema cree una nueva prueba con este juego de datos.	
Entrada/ Pasos de ejecución: Juego de datos de una prueba ya existente y que esté correcto.	
Resultado esperado: El sistema debe mostrar un mensaje de error.	
Evaluación de la prueba: Satisfactoria.	

Tabla 71. Descripción de la Prueba de Aceptación 3 HU Gestionar Prueba

Caso de Prueba de Aceptación	
Código: HU7_p3	**Historia de Usuario (No.7):** Gestionar Prueba
Nombre: Modificar datos de una prueba con juego de datos correctos.	
Descripción: Se desea probar que el sistema modifica los datos de la prueba correctamente.	
Condiciones de ejecución: Se cambian los datos de la prueba en el sistema.	
Entrada/ Pasos de ejecución: Juego de datos correctos que modifiquen una prueba existente.	
Resultado esperado: Se actualice el listado de las pruebas con los cambios realizados.	
Evaluación de la prueba: Satisfactoria.	

Tabla 72. Descripción de la Prueba de Aceptación 4 HU Gestionar Tipo prueba

Caso de Prueba de Aceptación	
Código: HU8_p4	**Historia de Usuario (No.8):** Gestionar Tipo prueba
Nombre: Eliminar Tipo prueba.	
Descripción: Se desea probar que el sistema elimine satisfactoriamente un tipo de prueba.	
Condiciones de ejecución: Se accede a la opción de eliminar tipo de prueba.	
Entrada/ Pasos de ejecución: Seleccionar el tipo de prueba a eliminar.	
Resultado esperado: Se actualice el listado de los tipos de pruebas.	
Evaluación de la prueba: Satisfactoria.	

.