UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

SISTEMA INFORMÁTICO PARA ELABORAR Y EVALUAR TEST PEDAGÓGICOS DE LA EDUCACIÓN FÍSICA Y EL DEPORTE EN LA UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

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

Autores:

Mario Alejandro Remón Pi

Leonel Camaraza Rivas

Tutores:

MsC. Vladir A. Parrado Cruz

Ing. Yunier Velázquez Batista

Dr. Armando Pérez Fuentes

Ciudad de La Habana, junio 2010

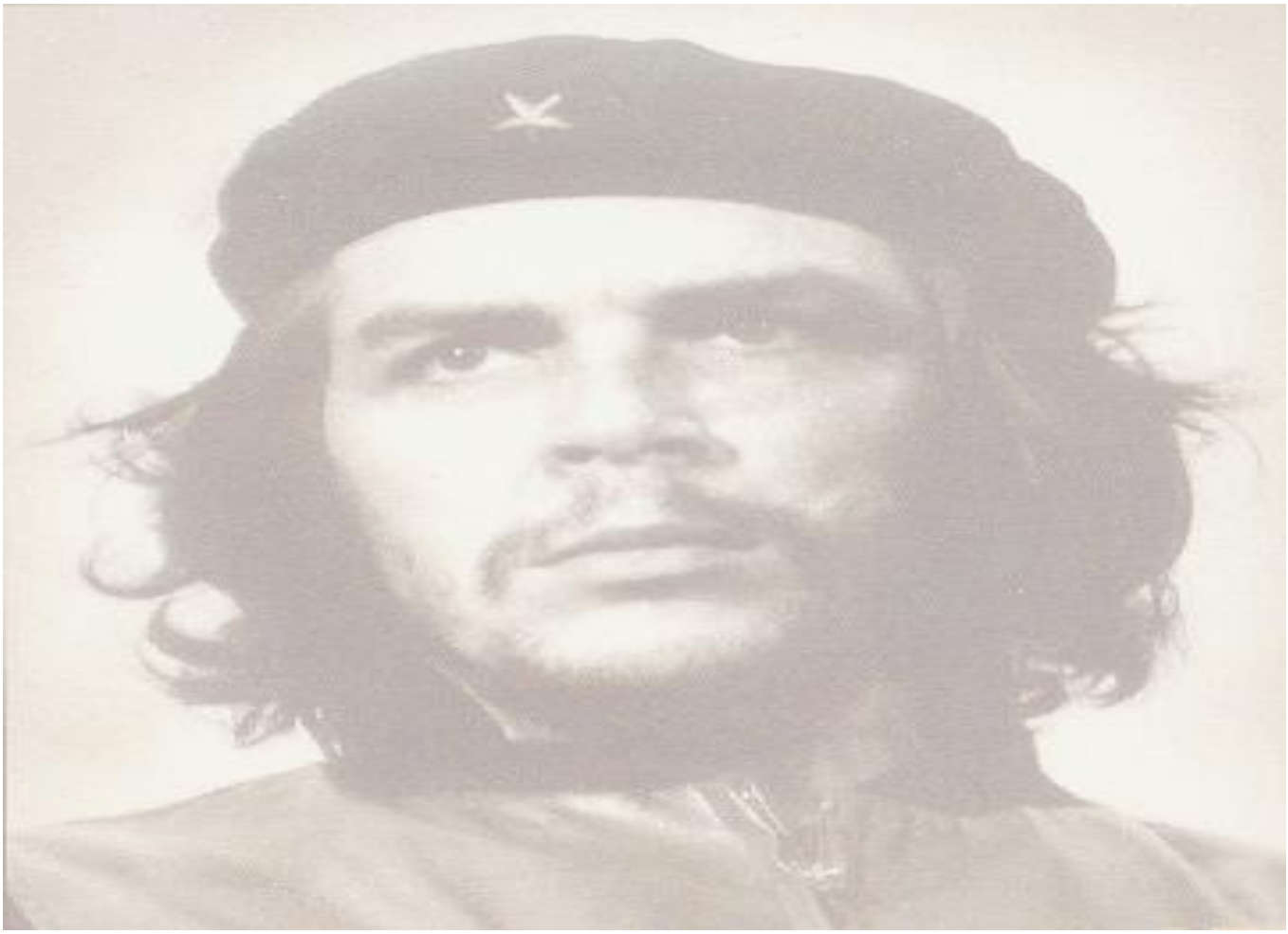
"Año 52 de la Revolución"

Agradecimientos

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Autores: Mario Alejandro Remón Pí Leonel Camaraza Rivas

_____ _____



"...aquí está una de las tareas de la juventud: empujar, dirigir con el ejemplo la producción del hombre del mañana. Y en esta producción, en esta dirección, está comprendida la producción de sí mismos..."

Agradecimientos

De Leonel

Termina una de las etapas más importantes de mi vida. He llegado al final del derrotero que inicié al comenzar mis estudios en la Universidad de las Ciencias Informáticas. En este empeño se agrupan un sinfín de personas que han sido el cimiento de mi carrera, los que me guiaron y me fortalecieron cuando flaquee en algunas ocasiones, a ellos van dirigidos estos agradecimientos. A mi madre por ser el manantial de mi inspiración, la que ha sido mi hálito en los momentos de desespero, siendo el refugio de mis penas cuando más necesite desahogarme, en resumidas cuentas es la persona más importante en mi vida. A mi padre, por enseñarme lo importante que es la familia, lo esencial que es no perderse un solo instante de los momentos de tus seres más allegados. A Pedro por ser un magnífico amigo, un padre en todos los sentidos. A mis abuelos tanto maternos como paternos, los cuales no les perdono que me hayan mimado tanto, por tal motivo les tengo reservado una buena porción de mi corazón, en especial a mi abuelo Rodrigo que ya no le tengo conmigo pero sé que me acompaña en toda ocasión. A mi tía Rosy por ser una de las personas dentro de mi vida que más ha aportado en mi formación como ser humano, como mejor hombre, porque me cuida, me aconseja, soporta todas mis malacrianzas, por lo que siempre la tendré como una madre. A mis tías Rafaela y Graciela por su infinito amor. A toda mi familia en general por haber confiado en mí, porque en ellos estriba este importante momento, sin su ayuda no hubiese sido posible alcanzar este sueño. A mis amigos Alejandro, Susel, Yuri, Luis Jiménez por todos los momentos que pasamos juntos, porque aprecio mucho contar con su sincera amistad. A todos los amigos que me han acompañado a lo largo de mi carrera. A mis tutores Vladir, Yunier y Armando por tener una paciencia infinita conmigo, pero sobre todo por haberme guiado en todo momento con sus consejos. A mi compañero de tesis.

Agradecimientos

De Mario:

Concluye el ciclo de mi vida como estudiante. Durante todos estos años como alumno he acumulado conocimientos y experiencias que me han formado como ser humano. La perseverancia ha dado como fruto que haya logrado ser ingeniero de las ciencias informáticas. En esta labor tan ardua, existen una infinidad de personas que a lo largo de mi vida como alumno han contribuido a mi formación. Primeramente tengo que agradecer a mi madre, quien es la persona que me motivó en todo momento a sobreponerme ante las dificultades que han ido apareciendo a lo largo de toda mi vida. A mi papá por enseñarme a valorar las cosas buenas de la vida, por ser un ejemplo de superación. A mis abuelos tanto maternos, Mima y Papiquiqui que me guiaron en todo momento por el buen camino, me ofrecieron su cariño inquebrantable cuando más lo necesité y sobre todo porque los considero como mis padres, como a los paternos, Olga y Delio. A Tania y Kiki, por servirme de base en mi afán de ser un profesional. A mis cuatro hermanos, Ailín, Rauly, Mayitín y Marian. A mis tías, Mary, Yaya, Olguita, Mily y Maira, que son como si fueran cinco madres para mí. A mis tíos, Dely, Héctor, Joaquín y a Domingo. A mis primos y hermanos de crianza, Tito, Eve, Amy, Ángel y Adrián. Agradezco a los amigos que estuvieron conmigo en las buenas y en las malas, tanto a los viejos de la infancia como a los actuales, en especial a Juan, Arturo, Castillo, Arnel, Paqui, Julito, Viño y kelly. A los profesores que me transmitieron sus conocimientos. A mis tutores Vladir, Armando y Yunier por confiar en mí. A mi compañero de tesis. A todos muchas gracias.

Dedicatoria

A nuestros padres por ser los principales protagonistas a lo largo de todos estos años, por su infinito apoyo en todo momento. A todas aquellas personas que nos brindaron su sincera amistad y nos ofrecieron su ayuda cuando más la necesitamos, a la Revolución por esta oportunidad, a todos aquellos que aunque no estuvieron presente forman parte de de este inolvidable momento.

RESUMEN

Los estudios demuestran la importancia de los test pedagógicos aplicados a la Educación Física y el Entrenamiento Deportivo como elemento esencial para el diagnóstico, control y evaluación del proceso de las clases o del entrenamiento.

No obstante, con frecuencia, las normativas establecidas no se ajustan a las poblaciones estudiadas porque el nivel de exigencia de las mismas se queda por debajo o por encima de las posibilidades de los evaluados. Además los profesores realizan este trabajo de forma manual, ocasionando lentitud y errores en el mismo.

En este trabajo se presenta una herramienta para elaborar normativas por el propio profesor o entrenador y que las mismas cumplan con el requisito de que se ajusten a las características de sus alumnos o atletas. Además resuelve el problema de almacenamiento y centralización de la información que se deriva de los procesos de la Educación Física y el entrenamiento deportivo. Por otra parte se elabora un sistema de ranking con los resultados, que permite motivar a los estudiantes en su proceso de aprendizaje. Para la creación y almacenamiento de las normativas, baterías de pruebas y ranking, se implementó una aplicación web utilizando herramientas de software libre.

ÍNDICE

INTRODUCIÓN

Con el desarrollo de la ciencia y la técnica ha aumentado el conocimiento, apareciendo nuevos conceptos y términos en las ramas del saber. Una de estas ramas es la Informática, que cada día, con el desarrollo tecnológico (Internet, comunicaciones móviles, banda ancha, satélites, microondas, etc.), avanza extraordinariamente.

Surge aquí un concepto importante, el de las Tecnologías de la Información y las Comunicaciones (TIC). Las mismas no significan realizar las cosas más rápidas o fáciles, sino que implican nuevas y distintas formas de vincular las tecnologías, la información y las personas. Son potencialidades significativas para el desarrollo personal y colectivo, con posibilidades y limitaciones siempre dependientes de las intencionalidades y de las condiciones de uso. Las TICs no son sólo aparatos o soportes físicos más o menos sofisticados, sino que constituyen poderosos sistemas que implican además, las formas de hacer, de producir, de reproducir y de transmitir información(1).

En la Universidad de las Ciencias Informáticas están creadas las condiciones tecnológicas necesarias para el desarrollo de las TICs en el área que se desee. La infraestructura universitaria cuenta con una gran cantidad de laboratorios, y computadoras distribuidas en todas las edificaciones del centro. Estas máquinas se encuentran conectadas en una red local de una velocidad de 100Mb\s. Una solución informática que aproveche e involucre todos estos recursos en resolver un problema, es una buena respuesta.

Una de las áreas que pudiera contar con este tipo de soluciones es el campo de la Educación Física y el Entrenamiento Deportivo. La misma es dirigida por el Departamento de Deportes. Actualmente en este Departamento los profesores realizan pruebas de eficiencia física, y diagnósticos de forma sistemática. Esta información se recopila individualmente por cada profesor y la almacenan en hojas de papel, documentos de texto o en Excel, pudiéndose extraviar la información.

Es necesario que estos datos se mantengan consistentes por largos períodos de tiempo, debido a que son necesarios para realizar análisis posteriores y hacer comparaciones que permitan mejorar la educación física y el entrenamiento.

En estos momentos el almacenamiento digital centralizado y seguro de los datos de las pruebas de eficiencia física permitan mejorar la educación física y el entrenamiento. En estos momentos el almacenamiento digital centralizado y seguro de los datos de las pruebas de eficiencia física, es un problema a resolver.

Por otra parte el INDER tiene estipulado normativas que por lo general no se adecuan a las características de un determinado centro de estudio. En este sentido en la UCI los profesores muy a menudo se dan cuenta que las normativas establecidas se quedan por encima o por debajo de las expectativas de la población a evaluar. Por lo que con frecuencia acuden a otras bibliografías que contemplan variadas pruebas.

Estas pruebas son obtenidas en el estudio de la literatura científica actual. No obstante los profesores se encuentran con la dificultad que las normativas de esas pruebas no están contextualizadas y no se ajustan a su población o simplemente no existen normativas para medirlas o evaluarlas. Como no existen los medios en la Universidad para que los profesores puedan crear sus propias normativas, se crea el inconveniente de no aprovechar al máximo las capacidades del estudiantado.

En el mundo existen aplicaciones que abordan soluciones que contemplan las dificultades que se presentan en el área de Educación Física. Estas herramientas son dinámicas, confiables y sobre todo posibilitan el trabajo del profesor. Pero cuenta con inconvenientes que frenan de cierta manera su utilización en la UCI. Por lo general son propietarios o están implementadas en aplicaciones propietarias. Lo cual trae como consecuencia que no sea posible su uso, sin comprar la licencia.

Por ello surge el siguiente **problema científico**: ¿Cómo contribuir al mejoramiento de la evaluación y el desempeño de las pruebas en la Educación Física y el Entrenamiento Deportivo en la Universidad de las Ciencias Informáticas? De aquí que el **objetivo general** es “desarrollar un sistema informático para el mejoramiento de la evaluación y el desempeño de las pruebas en la Educación Física y el Entrenamiento Deportivo en la Universidad de las Ciencias Informáticas”. Todo ello enmarcando el **objeto de estudio** en

el empleo de las Tecnologías de la Información y las Comunicaciones". El **campo de acción** abarca "las herramientas informáticas para el tratamiento de la información en la

Educación Física y el Entrenamiento Deportivo"

Los **objetivos específicos** que se persiguen son**:**

1. Estudiar posibles metodologías de desarrollo de software y herramientas a utilizar.
2. Realizar análisis, diseño, implementación y prueba de un sistema informático que permita elaborar pruebas y baterías de pruebas.
3. Conformar toda la documentación con vistas al desarrollo de nuevas versiones.

Para cumplir con los objetivos planteados se ha decido realizar las siguientes **tareas de investigación**:

1. Búsqueda de la documentación existente relacionada con el trabajo de los profesores de educación física.
2. Estudio de las tecnologías y herramientas a utilizar.
3. Estudio de las metodologías de desarrollo de software.
4. Entrevistas a los profesores de educación física y entrenadores de la Universidad de las Ciencias Informáticas.
5. Diseño e Implementación del sistema Informático Norma-Test.
6. Realización de pruebas al sistema con el objetivo del mejoramiento de la calidad del producto.
7. Elaboración de documentos relacionados con el producto a desarrollar.

Una vez realizada la aplicación se espera obtener el siguiente **resultado:**

Contar con una aplicación que les brinde a los profesores y entrenadores de Educación Física, la posibilidad de realizar sus propias baterías de prueba, y de esta forma garantizar

que el trabajo de los mismos sea más flexible, cómodo y eficiente.

Métodos Científicos

Para la realización de las tareas se emplearán métodos científicos. En este caso se refiere a los **Métodos Teóricos** y **Empíricos** de la investigación científica.

El **método teórico** que se va a utilizar es:

Método histórico – lógico.
Se utilizará este método pues se requiere un análisis de la trayectoria completa del fenómeno, para conocer cuál era la situación de la Educación Física antes de aplicarse el método de las normativas y qué ventajas podría traer el empleo de las TICs en la Asignatura.

El **método empírico** que se empleara es el siguiente:

La entrevista individual.
La entrevista será uno de los métodos fundamentales a utilizar pues los conocimientos que puedan brindar las personas calificadas en el tema, serán imprescindibles para cumplir con los objetivos del trabajo, que no es más que mejorar el proceso pedagógico de la Educación Física.

Capítulo 1 Fundamentación Teórica

1.1 Introducción

En el presente capítulo se abordarán las técnicas, tecnologías, metodologías y herramientas en las cuales el equipo de desarrollo se apoyará para el desarrollo del sistema informático *Norma-Test*. Para determinar cuál de ellas se usarán, se hará una comparación entre las mismas exponiendo sus ventajas y desventajas.

1.2 Conceptos Fundamentales

Percentiles: En una población de datos estadísticos valor por debajo del cual se encuentra un porcentaje determinado del total de los datos (2).

Según los percentiles que se escojan, se crean cotas. Para un caso hipotético, en el percentil ochenta se alcanza la cota superior, para este caso se tiene el veinte por ciento de los estudiantes que alcancen la máxima puntuación. Para la cota intermedia será el percentil cincuenta, donde se agruparán los estudiantes que alcancen cuatro puntos y por último el percentil veinte recogerá el criterio para saber los que no llegaron a aprobar el examen. A continuación se muestra una tabla donde se recoge el criterio de percentiles para una escala de evaluación.

Tabla 1 Criterio de percentiles para la escala de evaluación.

Percentil	Evaluación
80 ó más	5
79 al 50	4
49 al 20	3
20 ó menos	2

Normativas: Son reglas que permiten seleccionar percentiles dependiendo de las características del grupo que se desea evaluar. Por lo general estas reglas son establecidas por el INDER, teniendo en cuenta la cantidad de estudiantes o si la institución a la que se le quiere realizar la prueba es de carácter deportivo o no (3).

Test Pedagógicos: En el deporte, los *Test Pedagógicos* son pruebas que se realizan para determinar las capacidades físicas de los evaluados (3).

Batería de Pruebas: Conjunto de pruebas.

Ranking: Es una relación entre un conjunto de elementos tales que, para uno o varios criterios, el primero de ellos presenta un valor superior al segundo, este a su vez mayor que el tercero y así sucesivamente, permitiéndose que dos o más elementos diferentes puedan tener la misma posición (4).

1.3 Estado del Arte

1.3.1 Nacionales

Aplicación para la planificación del entrenamiento deportivo

En la facultad de Cultura Física de la provincia de Villa Clara se ha implementado un sistema de aplicaciones que permite automatizar la planificación del entrenamiento deportivo en distintos deportes como: Atletismo, Pesas y Esgrima. Este sistema comprende varias funcionalidades que están especializadas en un deporte en particular. Estas aplicaciones utilizan Microsoft Access para crear una base de datos donde se registran las estadísticas de los estudiantes alcanzadas en el entrenamiento y brinda la posibilidad de crear un plan de entrenamiento por el cual se deben regir los estudiantes. Un inconveniente de este sistema de aplicaciones es que al estar implementado en Microsoft Access, no es posible contar con una base de datos centralizada y segura de los datos, debido a que Microsoft Access no es una base de datos de servidor, es decir no acepta conexiones de red para acceder a su contenido, además de no ser multiplataforma, pues solo está disponible para Microsoft. Su uso es inadecuado para grandes proyectos de software que requieran muchos datos y su capacidad para ambientes de alta concurrencia es muy limitada (5; 6).

1.3.2 Internacionales

Informática y Deportes

Software argentino centrado en el trabajo de los entrenadores deportivos, es uno de los más usado en América Latina. Brinda la posibilidad de planificar, controlar y evaluar el entrenamiento deportivo de los atletas en todos los deportes. Agrupa los deportes en dos categorías: deportes de grupo y deportes individuales. Para el caso de los deportes de grupo ofrece una aplicación que permite evaluar, controlar y planificar el entrenamiento, pero desde un punto de vista táctico-técnico. En el caso de los deportes individuales, este control, evaluación y planificación se realiza teniendo en cuenta las estadísticas de los deportistas durante el entrenamiento, teniendo cierta similitud con el sistema que se desea implementar. Una característica de este software que no se debe pasar por alto es su condición de propietario, por lo que para utilizarlo, es necesario comprarlo. Otro de los inconvenientes que podríamos mencionar es que los datos no podrán ser consultados por otros entrenadores; esto está dado porque el software está diseñado para que sea usado por un entrenador individual y no para un grupo de entrenadores o profesores de Educación Física que trabajan en conjunto o que forman parte de una institución (7).

Woplanner

Dentro de la amplia gama de aplicaciones relacionadas con el tema de los deportes Woplanner es uno de los más íntegros. Este software es ágil, dinámico y eficaz a la hora de evaluar, planificar, generar reportes y organizar las sesiones de entrenamiento aplicables a todo tipo de deporte, sea individual o de conjunto, con la versatilidad que no existía hasta el momento. El sistema es altamente flexible, permitiendo realizar cambios en el funcionamiento interno, logrando para el usuario una buena personalización. Con esta tecnología, se obtienen enormes beneficios en el plano laboral, complementando su accionar profesional en todo momento. El principal inconveniente de este software es que es propietario, lo cual interfiere con el derrotero previsto por la Universidad de las Ciencias Informáticas en el sentido de dar prioridad a la creación de software libre (8).

1.4 Metodología

En un proyecto de desarrollo de software la metodología define quién debe hacer qué, cuándo y cómo debe hacerlo. Una metodología es un proceso. No existe una metodología de software universal. Las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigen que el proceso sea configurable (9).

En la actualidad las metodologías de desarrollo de software se bifurcan en tradicionales y ágiles. Las metodologías tradicionales hacen más énfasis en la planificación y el control del proyecto, dentro de esta gama hayamos MSF (Microsoft Solution Framework) y RUP (Rational Unified Process). La contraparte a las metodologías tradicionales son las llamadas metodologías ágiles, las cuales tienen la capacidad de proveer respuestas rápidas y ser adaptables al cambio. Dentro de esta rama existen innumerables metodologías como son SCRUM, DSDM (Dynamic Systems Developmemt Method), XP (Extreme Programming) y otras.

Ya planteado esto, se hará un estudio de algunas de las diferentes metodologías señaladas anteriormente.

1.4.1 Programación Extrema (XP)

Es una de las metodologías de desarrollo de software más exitosas en la actualidad, utilizada para proyectos de corto plazo. La metodología consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto. La Programación Extrema es una metodología de desarrollo de software que se basa en la simplicidad, comunicación, y la retroalimentación (Feedback). La comunicación interpersonal es la mejor forma de lograr un buen entendimiento entre los desarrolladores y el cliente. Gracias a este método el equipo puede evacuar de manera desahogada los nuevos cambios que vayan surgiendo durante el ciclo de vida del software. La metodología XP aboga por la programación en pares, que consiste en dos programadores trabajando en una misma estación de trabajo, esto permite que todos tengan autoridad para hacer cambios a cualquier código (10). Una característica fundamental de esta metodología es que el cliente forma parte del equipo de

desarrollo en todo momento, realizando pruebas constantemente al software y elaborando nuevos requerimientos. Esto favorece que el equipo haga más hincapié en obtener un software listo para cuando el cliente lo necesite.

1.4.2 Proceso Unificado de Desarrollo de Software (RUP)

Es una de las metodologías más generales y usadas que existen en la actualidad, pues está pensada para adaptarse a cualquier proyecto. Constituye además, una propuesta de proceso para el desarrollo de software orientado a objeto, utilizando UML, como lenguaje para visualizar, especificar, construir, y documentar algunos artefactos que se crean durante el proceso.

En RUP se han agrupado las actividades en grupos lógicos definiéndose nueve flujos de trabajo principales. Los seis primeros son conocidos como flujos de ingeniería y los tres últimos como de apoyo. El ciclo de vida de RUP se caracteriza por ser *dirigidos por casos de uso.* En los casos de uso se reflejan las necesidades de los usuarios finales, las cuales se captan cuando se modela el negocio. RUP se caracteriza por tener una robusta arquitectura, la cual muestra la visión común del sistema completo.

Un proyecto es un raudal de procesos, por lo tanto para organizarlo es importante dividirlo en mini-proyectos. RUP propone que cada fase se divida en iteraciones, en las cuales se agrupan todas las actividades de los flujos de trabajo, luego al finalizar cada iteración se obtendrá un incremento del producto (11).

1.4.3 Método de Desarrollo de Sistema Dinámico (DSDM)

El DSDM fue desarrollado en el Reino Unido en los años 90 por un consorcio de proveedores y de expertos en la materia del desarrollo de sistemas de información. Esta metodología se basa en programación rápida de aplicaciones (RAD). El método empieza con un estudio de viabilidad y negocio. El estudio de viabilidad considera si DSDM es apropiado para el proyecto. El estudio de negocio es una serie corta de talleres para entender el área de negocio dónde tiene lugar el desarrollo. También propone arquitecturas de esbozos del sistema y un plan de proyecto.

El resto del proceso forma tres ciclos entretejidos: el ciclo del modelo funcional produce documentación de análisis y prototipos, el ciclo de diseño del modelo diseña el sistema para uso operacional, y el ciclo de implantación se ocupa del despliegue al uso operacional. DSDM tiene principios subyacentes que incluyen una interacción activa del usuario, entregas frecuentes, equipos autorizados, pruebas a lo largo del ciclo. Como otros métodos ágiles usan ciclos de plazos cortos de entre dos y seis semanas. Por último en esta metodología hay un énfasis en la alta calidad y adaptabilidad hacia requisitos cambiantes (12).

1.4.4 Fundamentación de la metodología

Una vez concluido el estudio realizado sobre las posibles metodologías de desarrollo de software que podrían haber guiado el proceso actual, XP resultó ser la metodología más indicada en este caso, debido a las múltiples ventajas que ofrece. Como metodología ágil de desarrollo de software se ajusta al proceso actual, en primer lugar porque una de las características de esta metodología es que el cliente forme parte del equipo de desarrollo, requisito que se cumple debido a la cercanía constante del cliente; por todo ello, los requerimientos del usuario son más fáciles de modificar gracias al “feedback”. Por otra parte RUP a pesar de utilizar los Casos de Uso para guiar el proceso, y definir ciclos de trabajo, presta especial atención al establecimiento temprano de una arquitectura fuerte, pero que puede ser afectada por los cambios que surgen en la etapa de construcción y mantenimiento del software. La comunicación que propone RUP, entre los desarrolladores y el cliente, al ser mediante un intermediario, genera gran cantidad de documentación que es muy útil para el control, la organización y la detección de errores pero que no es factible para proyectos pequeños como el que se tiene en cuestión, más que ayudar con estas características, tiende a atrasar el proceso de desarrollo de un proyecto de tan poca duración. En cuanto a DSDM, su reputación en el Reino Unido hace eco en todo el mundo, siendo respaldada por una organización que la apoya sin cesar. Pero, cuenta con una etiqueta de precio, lo cual limita su utilización. También hay que señalar que es una metodología más orientada a la gestión de proyecto, a diferencia de XP que se centra más en el proceso de ingeniería.

1.5 Lenguaje Unificado de Modelado (UML)

UML (Unified Modeling Language) es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software y está compuesto por diversos elementos gráficos que se combinan para conformar diagramas. Este lenguaje que permite la modelación de sistemas con tecnología orientada a objetos y describe lo que supuestamente hará un sistema, pero no dice cómo implementar dicho sistema

El UML permite a los creadores de sistemas generar diseños que capturen sus ideas en una forma convencional y fácil de comprender para comunicarlas a otras personas que estén involucradas en el proceso de desarrollo de los sistemas, esto se lleva a cabo mediante un conjunto de símbolos y diagramas(13).

1.6 Herramientas CASE

En las últimas décadas se ha trabajado en el área de desarrollo de sistemas para encontrar técnicas que permitan incrementar la productividad y el control de calidad en cualquier proceso de elaboración de software, y hoy en día la tecnología CASE (Computer Aided Software Engineering) reemplaza al papel y al lápiz por el ordenador para transformar la actividad de desarrollar software en un proceso automatizado.

La tecnología CASE garantiza la automatización del desarrollo del software, contribuyendo así a elevar la productividad y la calidad en el desarrollo de los sistemas de información de forma análoga a lo que suponen las técnicas CAD/CAM en el área de fabricación (14).

1.6.1 Rational Rose Enterprise (Rational Software Corporation, 2002)

Esta herramienta CASE creada por los ingenieros Booch, Rumbaugh y Jacobson es de fácil utilización. Permite una modelación absoluta de los procesos del negocio y del sistema, además resulta de gran utilidad para los desarrolladores de proyectos, pues ella cubre todo el ciclo de vida del proyecto desde su fase inicial hasta su culminación.

Rational Rose posibilita establecer una trazabilidad entre los modelos (análisis y diseño). Cada rol tiene su propia vista de arquitectura (vista de Casos de Uso, vista Lógica, vista de Componentes y vista de Despliegue), pero utilizan un lenguaje común para comprender y comunicar la estructura y funcionalidad del sistema en construcción.

Cada analista, desarrollador o diseñador puede usar Rational Rose para definir y comunicar el negocio, el diseño y la arquitectura de la aplicación que se esté desarrollando. Es una completa solución para mostrar de forma gráfica el análisis de los procesos del negocio y los requerimientos del sistema (15).

1.6.2 Visual Paradigm

Visual Paradigm es una herramienta muy potente y fácil de utilizar para crear los artefactos necesarios en la confección de un software. Está diseñada para varios tipos de usuarios, incluyendo Ingenieros de Software, Analistas de Sistemas, Analistas de Negocio y Arquitectos de Sistemas. Es una suite completa de herramientas CASE que da soporte al modelado visual con UML 2.0 ofreciendo distintas perspectivas del sistema.

Resulta ser independiente de la plataforma y dotada de una buena cantidad de productos o módulos para facilitar el trabajo durante la confección de un software así como garantizar la calidad del producto final. Es posible generar código desde Visual Paradigm para plataformas como .NET y lenguajes de programación como Java y PHP, así como obtener diagramas de clases a partir del código, siendo de gran utilidad pues ahorra tiempo a los desarrolladores y reduce las posibilidades de cometer errores(16).

Brinda la posibilidad de obtener una Base de Datos relacional y el código necesario para acceder a esta a partir de un diagrama Entidad–Relación; así como conectarse fácilmente a varios servidores de Base de Datos e integrarse con varios ambientes de desarrollo integrados (IDE) lo cual permite pasar del código al modelado y viceversa. Establece interoperabilidad con otras aplicaciones como el Rational Rose y documenta todo el trabajo y especificaciones de Casos de Uso sin necesidad de utilizar herramientas externas, por ejemplo editores de texto, utilizando plantillas que se encuentran o que pueden ser creadas por los usuarios. Además se encuentra disponible en múltiples

lenguajes y plataformas: Microsoft Windows (98, 2000, XP, o Vista) Linux, Mac OS X, Solaris o Java.

1.6.3 Fundamentación de la herramienta seleccionada

Luego de un análisis de las herramientas que existen para modelar los sistemas, se determinó que la herramienta que más se ajusta al sistema que se desea realizar es Visual Paradigm, ya que ofrece un entorno de creación de diagramas para UML 2.0, el mismo tiene la disponibilidad de trabajar en múltiples plataformas. Permite generar código de varios lenguajes de programación como Java, C++, PHP, CORBA IDL, Esquema de XML, Ada y Python, además de que es muy potente y fácil de utilizar.

1.7 Framework

Un framework provee una infraestructura de programación para el desarrollo de aplicaciones de software, principalmente para centrar el trabajo y agilizar el proceso de desarrollo. Además de incluir soportes de programas, bibliotecas y un lenguaje interpretado entre otros software que ayudan a desarrollar y unir los diferentes componentes de un proyecto.

Un framework representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. También provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio (17).

Framework de desarrollo Web

En la actualidad las aplicaciones web están respaldas por variedades de frameworks. Dentro de los más usados destacados en el mundo encontramos:

1. ASP .NET
2. Spring
3. Symfony
4. CakePHP
5. Django

A continuación se analizarán varios de estos frameworks.

1.7.1 Symfony

Symfony es un framework diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones web. 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 los gestores de bases de datos, como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft. Se puede ejecutar tanto en la plataforma de Linux. como en plataformas Windows. Sin embargo su Mapeo de Objeto Relacional, de sus siglas en ingles ORM (Object Relational Mapping), para conectarse a la base de datos es propenso a lentitud pues conlleva a gastos innecesarios en la generación/configuración de código cuando se están creando y administrando cientos de tablas. El aspecto de autenticación utiliza medidas de seguridad muy débiles, los roles en Symfony son llamados credenciales. Un usuario puede mantener varias credenciales, en ocasiones más de las que necesita. Esto no es una medida de seguridad por roles puesto que para saber a qué usuario se le han otorgado credenciales, antes tiene que haber visto toda la lista de roles. Además como Symfony está programado parcialmente en PHP 5 y está enfocado al desarrollo de aplicaciones web en el mismo lenguaje de programación, es obligatorio disponer de conocimientos avanzados en este lenguaje para sacar el máximo partido al framework (18; 19).

1.7.2 Spring

Spring es un framework de desarrollo de aplicaciones web para plataforma Java. Tiene la capacidad de ser modular, lo que permite usar sólo las partes de él que son necesarias, sin tener que involucrar el resto. Spring ha sido diseñado para ser no invasivo, lo cual significa que las dependencias dentro del framework mismo son nulas (o mínimas dependiendo del área de uso). El framework Spring contiene una gran cantidad de características las cuales están organizadas en alrededor de veinte módulos. Dentro de los más fundamentales encontramos el módulo Core y el módulo ORM. El primero provee el contenedor IoC (Inversion of Control), que es un proceso en el cual los objetos definen sus dependencias con otros objetos, a través de argumentos del constructor o

propiedades que son configuradas en las instancias de los objetos una vez que son creados. En el caso del módulo de ORM, soporta integración con Hibernate, JPA (*Java Persistence API,* API de Persistencia para Java) y JDO (*Java Data Objects*, Objetos de Datos Java). Por otra parte provee interfaces para que el desarrollador pueda utilizarlas, promoviendo así la reutilización de código y un estándar del paradigma orientado a objetos. Sin embargo cuenta con el inconveniente de que al llevar a cabo el desarrollo de una aplicación es necesario escribir mucho código dentro de los archivos JSP (*Java Server Page*, Página del Servidor de Java). También muchas veces es necesario utilizar demasiados archivos XML, lo cual en ocasiones es inevitable para configurar ciertos aspectos del framework (20).

1.7.3 Django

Django es un framework de alto nivel basado en *Python* que facilita el desarrollo de aplicaciones web dinámicas. Es un framework que abstrae a los programadores de los problemas comunes del desarrollo web y acelera las tareas más frecuentes en la programación. Proporciona un método de mapear las URLs ejecutando un código en especial para cada una. Permite mostrar y validar formularios de manera muy simple, manipulando el código del formulario y adaptándolo a las necesidades de la aplicación. El mismo convierte los datos enviados por los usuarios, en estructuras de datos que pueden ser manipuladas fácilmente. A través de plantillas ayuda a separar el contenido de la presentación evitando tener que manipular la lógica de negocio cuando se necesite realizar cambios de apariencia en la página. Permite lidiar con trescientas peticiones web por segundo. Su ORM es muy poderoso, permitiendo definir los modelos de datos enteramente en Python. Además provee un grandioso soporte para el uso de plantillas, filtros, etiquetas y formularios (21).

1.7.4 ASP.NET

ASP (*Active Server Pages*) es una tecnología de Microsoft del tipo "lado del servidor" para páginas web generadas dinámicamente. ASP.NET es la versión mejorada de ASP. Esta tecnología se aprovecha de toda la capacidad del CLR (Common Languaje Runtime) que es el motor en tiempo de ejecución del framework y para realizar las aplicaciones se

tienen nuevas características tales como los controles de usuario y modelos de programación de más alto nivel. En ASP.NET el código se separa totalmente de la lógica de la aplicación o sea se escribe en archivos independientes, de esta forma la lógica puede realizarse con cualquiera de los lenguajes soportados por .NET. Igualmente brinda la posibilidad de escoger el lenguaje que desee el programador, por defecto lleva integrado C# y VB.NET. Por otra parte ASP.NET tiene la opción de crear campos ocultos o sea es permitido almacenar información específica de la página como forma de conservar el estado de la misma. Sin embargo esta opción crea posibles riesgos de seguridad, ya que se pude ver la información de un campo oculto si se obtiene acceso al código fuente de la página directamente. Además ASP.NET por las múltiples funciones que tiene es más lento que la mayoría de los frameworks de desarrollo web. Pero sin lugar a dudas el mayor inconveniente que presenta, es su condición de propietario (22).

1.7.5 Fundamentación del framework seleccionado

El framework seleccionado es Django, debido a la calidad de su código fuente y a la amplia documentación con que cuenta. Posee una robusta Arquitectura basada en el Modelo Vista Controlador (MVC), que junto con el patrón bajo acoplamiento permite que se puedan hacer cambios en el código sin que esto afecte a la interfaz del sistema. Su ORM ofrece los modelos de datos completamente en Python y obtiene los datos mediante una potente API de acceso dinámico a la base de datos. Symfony es un magnífico framework, pero su ORM ralentiza la aplicación, ya que genera gastos innecesarios en la configuración del código en el momento de crear y administrar cientos de tablas. En el caso de ASP.NET, no se determinó su elección por ser un software propietario. Respecto a Spring, se obvió su uso, puesto que el equipo no tiene conocimiento en el lenguaje Java y comenzar un estudio del mismo conllevaría retraso en el desarrollo de la aplicación.

1.8 Lenguaje de programación

El lenguaje de programación a usar es Python, ya que se eligió como framework Django que está implementado completamente en este lenguaje. A diferencia de PHP, Python no es tan difundido en la Universidad de las Ciencias Informáticas, pero es un lenguaje que

todos deberían conocer porque es sencillo y fácil de usar, además posee una gran cantidad de librerías, tipos de datos y funciones incorporadas en el propio lenguaje, que ayudan a realizar muchas tareas habituales sin necesidad de tener que programarlas de nuevo. Otra característica que impulsó al equipo de desarrollo a usar este lenguaje es que exige indentar el código, lo cual es muy importante para la creación de nuevas versiones ya que es muy sencillo seguir el flujo del programa. También se puede usar en cualquier plataforma como: Unix, Windows, Mac, Linux, entre otras. Por último nos brinda una gran ventaja al ser gratuito incluso para fines empresariales (23).

1.9 Sistema Gestor de Base de Datos

Se puede definir como todo el conjunto de herramientas que acompañan al motor de base de datos y permiten a los diferentes tipos de usuarios interactuar con la BD. Recuerde que el motor de BD es la arquitectura lógica o mejor dicho el algoritmo que permite acceder a los datos. Los Sistemas Gestores de Bases de Datos (SGBD) también se conocen como sistemas administradores de BD (24).

1.9.1 PostgreSQL

PostgreSQL es el gestor de bases de datos objeto-relacional de código abierto más avanzado hoy en día, ofreciendo control de concurrencia multi-versión (permite a los accesos de sólo lectura continuar leyendo datos consistentes durante la actualización de registros), soportando casi toda la sintaxis SQL (incluyendo sub-consultas, transacciones, y tipos y funciones definidas por el usuario), contando también con un amplio conjunto de enlaces con lenguajes de programación (incluyendo C, C++, Java, Perl y Python), además de proveer una arquitectura que ha ganado una fuerte reputación por su fiabilidad e integridad de sus datos. Las transacciones de este gestor permiten el paso entre dos estados consistentes manteniendo la integridad de los datos. También permite bloqueos de tabla y filas para controlar el acceso concurrente a los datos en tablas. Algunos de estos modos de bloqueo los adquiere PostgreSQL automáticamente antes de la ejecución de una declaración, mientras que otros son proporcionados para ser usados por las aplicaciones. Brinda soporte de tipos y funciones de usuario. Incorpora una estructura de

datos Array: Conectividad TCP/IP, JDBC y ODBC. Interfaz con diversos lenguajes C, C++, Java, Delphi, Python, Perl, PHP, Bash. Por último señalar que es multiplataforma, estando disponible en casi cualquier Unix (34 plataformas en la última versión estable), y ahora en versión nativa para Windows (25).

1.9.2 MySQL

MySQL es un sistema de gestión de bases de datos relacional. Su diseño multihilo le permite soportar una gran carga de forma muy eficiente. MySQL fue creada por la empresa sueca MySQL AB, que mantiene el copyright del código fuente del servidor SQL, así como también de la marca. Aunque MySQL es software libre, MySQL AB distribuye una versión comercial de MySQL, que no se diferencia de la versión libre más que en el soporte técnico que se ofrece, y la posibilidad de integrar este gestor en un software propietario, ya que de no ser así, se vulneraría la licencia GPL.

Este gestor de bases de datos es, probablemente, el gestor más usado en el mundo del software libre, debido a su gran rapidez y facilidad de uso. Esta gran aceptación es debida, en parte, a que existen infinidad de librerías y otras herramientas que permiten su uso a través de gran cantidad de lenguajes de programación, además de su fácil instalación y configuración.

Soporta gran cantidad de tipos de datos para las columnas. Dispone de APIs en gran cantidad de lenguajes (C, C++, Java, PHP, entre otros). Su bajo consumo lo hace apto para ser ejecutado en una máquina con escasos recursos sin ningún problema. Carece de soporte para transacciones, rollback's y sub-consultas. No es viable su uso con grandes bases de datos, a las que se acceda continuamente, ya que no implementa una buena escalabilidad (24).

1.9.3 Oracle

Oracle es el motor de base de datos relacional más usado a nivel mundial. Puede ejecutarse en todas las plataformas. Oracle 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. 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. El software del servidor puede ejecutarse en multitud de sistemas operativos. La manera en que Oracle maneja los objetos ha comenzado a evolucionar, añadiendo tipos de clases, referencias, tablas anidadas, matrices y otras estructuras de datos complejas. Desafortunadamente la implementación actual de las mismas no ofrece una ventaja clara en eficiencia, y sí provocan la incompatibilidad de los diseños que aprovechan las nuevas características con otras bases de datos. Por otra parte el mayor inconveniente de Oracle es quizás su precio, sus licencias son excesivamente caras. Otro gran inconveniente es que aún no aparecen buenos libros que asesoren a los usuarios en cuanto a la instalación y administración de Oracle (26).

1.9.4 Fundamentación del Gestor de base de datos

El gestor de base de datos seleccionado es *PostgreSQL,* principalmente por ser multiplataforma lo cual permitirá utilizarlo tanto en *Windows* como en *Linux*, dependiendo de los intereses del equipo. Otro aspecto fundamental a tener en cuenta es que es libre, parámetro en el cual lleva cierta ventaja sobre *MySQL* debido a que el mismo está patrocinado por una compañía capitalista, lo cual conlleva a necesitar una licencia para su uso. En cuanto a Oracle no cabe duda que es muy potente, pero como se señaló anteriormente su licencia es demasiado cara.

1.10 IDE para el desarrollo del sistema.

Un IDE es un conjunto de programas que se ejecuta a partir de una única interfaz de usuario (27). Python es un lenguaje que no cuenta con muchos IDEs. Se pudo constatar luego de una profunda investigación, que este lenguaje de programación cuenta con apenas dos IDEs para su desarrollo sin que aparezcan problemas técnicos a lo largo del ciclo de desarrollo del software. Por tal motivo el estudio se centró en Eclipse, que integra a Python y por último Wing IDE que es el IDE por excelencia de este lenguaje.

1.10.1 Eclipse

La Plataforma Eclipse es una herramienta para todo y para nada en particular. A pesar de que la funcionalidad de Eclipse es mucha, gran parte de la funcionalidad es muy genérica. Permite que nuevos componentes puedan utilizar distintos tipos de contenido, para realizar determinadas tareas con contenidos existentes. La Plataforma Eclipse permite descubrir, e invocar funcionalidad implementada en componentes llamados plug-ins.

La Plataforma Eclipse está diseñada para soportar la construcción de gran variedad de herramientas de desarrollo. Soporta las herramientas proporcionadas por diferentes fabricantes de software independientes (ISV's). Contiene herramientas que permiten manipular diferentes contenidos (por ejemplo HTML, Java, C, JSP, EJB, XML, y GIF). Facilita una integración transparente entre todas las herramientas y tipos de contenidos sin tener en cuenta al proveedor. Proporciona entornos de desarrollo gráfico (GUI) o no gráficos. Permite ejecutarse en una gran variedad de sistemas operativos, incluyendo Windows y Linux.

El principal objetivo de la plataforma Eclipse es proporcionar mecanismos, reglas que puedan ser seguidas por los fabricantes para integrar de manera transparente sus herramientas. Mediante APIs interfaces, clases y métodos se exponen estos mecanismos. La Plataforma también nos posibilita la construcción de nuevas herramientas que extenderán la funcionalidad de la misma (28).

1.10.2 Wing IDE

Es un potente IDE para el lenguaje de programación Python. Elementos como *Código Inteligente* le permite al desarrollador auto-completamiento de código, un asistente fuente que posibilita fuentes de documentación, indicadores de errores y otras ventajas. También soporta la mayoría de las versiones de CPython e integra el Shell de Python con auto-realización. Como parte de un apoyo sostenido a la comunidad de desarrolladores de este lenguaje, Wing IDE permite depurar versiones del mismo. Presenta una interfaz de usuario personalizable facilitando que se puedan crear instancias de múltiples herramientas. Brinda una navegación rápida con el teclado a archivos y a símbolos.

Existen muchas más características que hacen que este IDE sea muy atractivo, pero cuenta con una desventaja muy grande, es un software propietario (29).

1.10 Fundamentación del IDE seleccionado

Luego de discernir acerca de las ventajas de ambos IDEs se optó por Eclipse. Este IDE es mucho más robusto y eficiente que Wing IDE. Eclipse cuenta con una comunidad de usuarios muy amplia que ayudan a resolver disímiles contratiempos que se sucedan a lo largo del desarrollo de la aplicación. Por otra parte Eclipse es un IDE de código abierto a diferencia de Wing IDE que es propietario.

1.11 Conclusiones

En el presente capítulo se definieron las herramientas y tecnologías a utilizar, quedando entre las seleccionadas, la metodología de desarrollo de software Extreme Programming y para realizar el modelado del sistema la herramienta case Visual Paradigm, apoyándose en el Lenguaje Unificado de Modelado (UML) . El IDE seleccionado fue Eclipse junto con el lenguaje de programación interpretado y orientado a objeto Python y como framework Django; finalmente para el tratamiento de los datos se escogió el gestor de base de datos PostgreSQL.

Capítulo 2 Descripción del sistema. Planificación y Diseño

2.1 Introducción

Para poder entender el mecanismo de un proyecto de software es necesario mostrar el flujo de sus procesos. Es fundamental que queden claro cuáles serán los involucrados que estarán presente durante todo su ciclo de vida. También es clave durante el desarrollo del software que se logre una buena planificación o sea llevar a cabo un análisis exhaustivo antes de entrar en el diseño del mismo, características estas de los procesos tradicionales. La metodología XP toma estos procesos tradicionales y los voltea en un ángulo de noventa grados, adaptándolos a su favor. En el presente capítulo mediante las dos primeras etapas de la Metodología XP, se abordará todo lo antes planteado.

2.2 Flujo actual de los procesos

El actual desarrollo de la gestión de la información en las clases de Educación Física fluye de la siguiente manera:

En primer lugar los profesores preparan pruebas de eficiencia física mediante las normativas indicadas por el INDER. Luego de cada sesión de clases los profesores registran el resultado de las actividades consumadas por sus estudiantes en planillas, documentos de Word o Excel. Al finalizar cada semestre se hacen pruebas generales que son archivadas de la misma forma, pues no existe un medio de almacenamiento digital que lleve la consistencia de estos datos. En el departamento de la asignatura solo se les exige que entreguen las notas finales de sus estudiantes, extraviándose de esta forma posibles estudios que realicen los profesores con sus estudiantes acerca del desempeño de los mismos en las clases.

2.3 Propuesta del sistema

Se implementará un sistema capaz de gestionar toda la información recogida en las distintas pruebas llevadas a cabo por los profesores. Además, se obtendrán mejores resultados con las nuevas características que requiere la clase de Educación Física, las cuales se define a continuación:

1. Trabajar tanto con las normativas establecidas por el INDER como las creadas por el profesor
2. Posibilitar manejar test pedagógicos
3. Hacer baterías de pruebas con los test pedagógicos
4. Mostrar un ranking con los resultados de las pruebas

El primer aspecto brindará la posibilidad al profesor no solo de trabajar con las normativas establecidas por el INDER sino que además pueda crear sus propias normativas atendiendo al procedimiento estadístico de los percentiles.

En el segundo punto el profesor contará con la posibilidad de crear test pedagógicos y estos a su vez contribuirán a la formación de las baterías de pruebas que miden las variadas capacidades físicas que se necesitan desarrollar.

Las baterías posteriormente arrojaran los resultados de los estudiantes, mostrando en un ranking el nivel de esfuerzo de cada alumno, esta escala se visualizará al estudiante tanto a nivel de grupo como de año. El sistema provee no solo estas funcionalidades sino que también permitirá que el estudiante dada una batería de prueba pueda contar con disímiles combinaciones de los posibles resultados que llegaría a alcanzar.

Mediante el framework Django se logrará que todo esté bien organizado a través de su sistema de URL, vistas y plantillas. Igualmente cuenta con una interfaz de administrador bien robusta que permite otorgar los permisos pertinentes a los usuarios del sistema.

2.3.1 Personal relacionado con el sistema

Se entiende como personal relacionado con el sistema todo aquel que interactúe con la aplicación y que obtiene un resultado de valor de uno o varios procesos que se ejecutan en la misma.

Tabla 2 Personal relacionado con el sistema

Personal relacionado con el sistema	Descripción
Administrador	Es el encargado de cerciorar que se gestionen correctamente los datos, además otorga los permisos correspondientes a los usuarios que interactúen con el sistema
Estudiante	Usuario que podrá interactuar con el sistema, bajo ciertos permisos.
Profesor	Usuario que tendrá una interacción con el sistema más profunda.
Jefe de año	Encargado de otorgarle los permisos a los profesores.

2.4 Planificación

En esta etapa se especifican los requisitos del cliente, redactados en las historias de usuario. Las historias de usuario ayudan al equipo a estimar el tiempo y los recursos necesarios para construir la aplicación. Una vez realizadas las estimaciones se comienza a realizar un cronograma o plan de entrega, en el cual estará fijado el alcance del trabajo. En el momento en el que se fija el plan de entrega, se inicia una fase de iteraciones, en las cuales se desarrollarán las historias de usuario.

2.4.1 Historias de Usuario

Las historias de usuarios reflejan los requisitos del cliente. Tienen la misma finalidad que los casos de uso, pero con la diferencia de que constan de tres o cuatro líneas escritas por el cliente en un lenguaje sencillo. Su función también es estimar el tiempo de desarrollo de la aplicación. Además se utilizan en la sección de las pruebas, para verificar si el programa cumple con lo que especifica la historia de usuario.

Tabla 3 HU Gestionar usuario

Historia de Usuario	
Número: 1	**Nombre**: Gestionar usuario
Usuario: Administrador	
Prioridad en Negocio: Alta	**Riesgo en desarrollo**: Bajo
Iteración Asignada: 1	
Descripción: Se realiza la acción de crear, modificar y eliminar un usuario. El sistema debe dar la posibilidad de añadir un usuario que no se encuentre registrado. Además de permitir tanto modificar como eliminar un usuario.	
Observaciones:	

Tabla 4 HU Gestionar prueba

Historia de Usuario	
Número: 2	**Nombre**: Gestionar prueba
Usuario: Profesor	
Prioridad en Negocio: Alta	**Riesgo en desarrollo**: Alto
Iteración Asignada: 1	
Descripción: Se realiza la acción de crear, modificar y eliminar una prueba. El sistema debe permitir añadir una prueba así como de modificar o eliminar dicha prueba.	
Observaciones:	

Tabla 5 HU Gestionar Batería de Prueba

Historia de Usuario	
Número: 3	**Nombre**: Gestionar Batería de Prueba
Usuario: Profesor	
Prioridad en Negocio: Alta	**Riesgo en desarrollo**: Alto
Iteración Asignada: 1	
Descripción: Se realiza la acción de crear, modificar y eliminar una batería de prueba.	
Observaciones:	

Tabla 6 HU Evaluar datos de una Batería de Prueba

Historia de Usuario	
Número: 4	**Nombre**: Evaluar datos de una Batería de Prueba
Usuario: Profesor	
Prioridad en Negocio: Alta	**Riesgo en desarrollo**: Bajo
Iteración Asignada: 2	
Descripción: El sistema debe brindar la posibilidad de evaluar la batería de prueba. Se obtienen las evaluaciones por cada prueba, se le calcula el promedio y se establece una evaluación de acuerdo a una escala establecida.	
Observaciones:	

Tabla 7 HU Gestionar normativas

Historia de Usuario	
Número: 5	**Nombre**: Gestionar normativas
Usuario: Profesor	
Prioridad en Negocio: Alta	**Riesgo en desarrollo**: Alto
Iteración Asignada: 1	
Descripción: Se realizan las acciones de crear, modificar y eliminar las normativas. El sistema debe brindar la posibilidad de elaborar una normativa, además tanto de modificarla si mejora el rendimiento del grupo que trate el profesor como de eliminarla si no cumple sus expectativas.	
Observaciones:	

Tabla 8 HU Ranking por grupo

Historia de Usuario	
Número: 6	**Nombre**: Ranking por grupo
Usuario: Profesor y estudiante	
Prioridad en Negocio: Media	**Riesgo en desarrollo**: Bajo
Iteración Asignada:3	
Descripción: Inicia en el momento en que el profesor o el estudiante escoge la opción de conformar un ranking. El sistema debe brindar la posibilidad de conformar un ranking por pruebas, por grupos de pruebas y general por cada grupo.	
Observaciones:	

Tabla 9 HU Autenticar

Historia de Usuario	
Número: 7	**Nombre:** Autenticar
Usuario: Profesor, jefe de año y estudiante	
Prioridad en Negocio: Media	**Riesgo en desarrollo:** Bajo
Iteración Asignada: 3	
Descripción: El sistema debe permitir que cualquier usuario que acceda al sistema se le brinde la opción de introducir sus datos (usuario y contraseña) para de esta manera verificar y entonces asignarle los permisos que le competen en la aplicación.	
Observaciones:	

Tabla 10 HU Posibles evaluaciones

Historia de Usuario	
Número: 8	**Nombre:** Posibles evaluaciones
Usuario: Profesor, jefe de año y estudiante	
Prioridad en Negocio: Media	**Riesgo en desarrollo:** Bajo
Iteración Asignada: 4	
Descripción: El sistema brinda la posibilidad que dada una batería de pruebas el estudiante pueda tener combinaciones de posibles evaluaciones.	
Observaciones:	

Tabla 11 HU Ranking por año

Historia de Usuario	
Número: 9	**Nombre:** Ranking por año
Usuario: Estudiante, Profesor	
Prioridad en Negocio: Media	**Riesgo en desarrollo:** Bajo
Iteración Asignada: 4	
Descripción: El sistema debe brindar la posibilidad de mostrar una lista con los resultados ordenados ascendente o descendentemente según el caso a nivel de año en el caso del estudiante.	
Observaciones:	

Tabla 12 HU Evaluar datos de una prueba

Historia de Usuario	
Número: 10	**Nombre:** Evaluar datos de una prueba
Usuario: Estudiante, Profesor	
Prioridad en Negocio: Media	**Riesgo en desarrollo:** Alto
Iteración Asignada: 2	
Descripción: El sistema muestra una lista con todos los estudiantes de un grupo determinado, sus resultados obtenidos en una prueba específica y la nota calculada a partir de dichos resultados.	
Observaciones:	

Tabla 13 HU Ranking por prueba

Historia de Usuario	
Número: 11	**Nombre:** Ranking por prueba
Usuario: Estudiante, Profesor	

Prioridad en Negocio: Media	**Riesgo en desarrollo:** Alto
Iteración Asignada: 3	
Descripción: El sistema muestra una lista con todos los estudiantes de un grupo determinado, sus resultados obtenidos en una prueba específica y la posición ordenada a partir de dichos resultados.	
Observaciones:	

2.4.2 Plan de entrega

En este apartado las historias de usuario juegan un papel importante. Estas breves descripciones elaboradas en un lenguaje sencillo permiten que se hagan estimaciones, previendo los riesgos que conllevaría realizarlas.

Este plan recoge las historias que serán agrupadas para conformar una entrega, y el orden de las mismas. El plan se realiza en base de las estimaciones de esfuerzos de desarrollo realizadas por los programadores. Las estimaciones de esfuerzos asociadas a la implementación de las historias tendrán como medida el punto. Un punto, se corresponde a una semana de programación ideal. Las historias deben poder ser programadas en un tiempo entre una y tres semanas.

La planificación se puede realizar basándose en el tiempo o en el alcance. La velocidad del proyecto es aprovechada para establecer cuantas historias de usuario se pueden hacer antes de una fecha determinada o cuánto tiempo tomará implementar un conjunto de historias. Para contar con una mayor organización se crearon paquetes en los cuales estarán acopladas las historias de usuario, en dichos paquetes se agrupan las historias que tengan cierta similitud, esta manera de ordenar las historias de usuario es meramente organizativa, no guarda ninguna relación con la implementación de la aplicación.

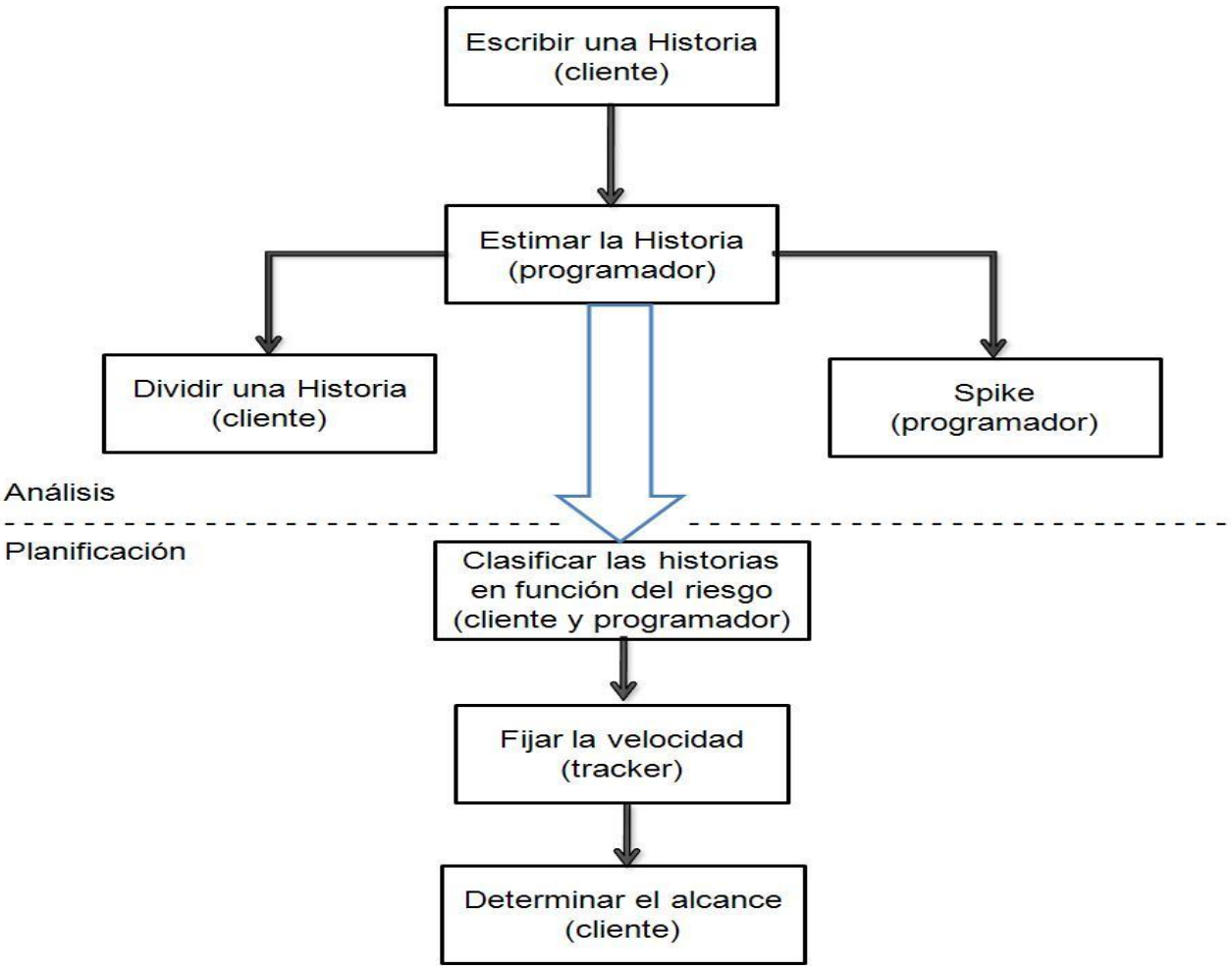


Figura 1 Plan de entrega

Tabla 14 Plan de estimación de esfuerzo e iteración por Historia de Usuario

No.	Nombre	Prioridad	Riesgo	Esfuerzo	Iteración
1	Gestionar usuario	Alta	Bajo	1	1
2	Gestionar prueba	Alta	Alto	1	1

3	Gestionar una batería de prueba	Alta	Alto	1	1
4	Evaluar datos de una batería de prueba	Alta	Bajo	1	2
5	Gestionar normativas	Alta	Alto	1	1
6	Ranking por grupo	Media	Bajo	1	3
7	Autenticar	Media	Bajo	1	3
8	Posibles evaluaciones	Media	Bajo	1	4
9	Ranking por año	Media	Bajo	1	4
10	Evaluar datos de una prueba	Media	Alto	1	2
11	Ranking por prueba	Media	Alto	1	3

2.4.3 Plan de la distribución por paquetes

Para que sea viable la elaboración del plan de entrega se aunaron las funcionalidades referentes a un mismo tema en paquetes, quedando éstos de la siguiente manera:

Tabla 15 Plan de la distribución por paquetes

Paquetes	Historias de usuario que abarcan
Normativas	5-Gestionar normativas

Pruebas	2-Gestionar prueba 3-Gestionar Batería de Prueba
Evaluaciones	4-Evaluar datos de una Batería de Prueba 8-Posibles evaluaciones 10-Evaluar datos de una prueba
Ranking	6-Ranking por grupo 9-Ranking por año 11-Ranking por prueba
Usuario	1-Gestionar usuario 7-Autenticar

A continuación se presenta el plan de entrega elaborado para la fase de implementación. Como producto del plan de entregas se harán raleases del sistema en las fechas que se indican a continuación:

Tabla 16 Plan de duración de entrega

Paquetes	**Final 1ra iteración 4ta semana de febrero**	**Final 2da iteración 3ra semana de marzo**	**Final 3ra iteración 3ra semana de abril**	**Final 4ta iteración 2da semana de mayo**
NORMATIVAS	1.0	Finalizado	Finalizado	Finalizado
PRUEBAS	0.1	0.2	1.0	Finalizado
EVALUACIONES		0.1	0.2	Finalizado

RANKING		1.0	Finalizado	Finalizado
USUARIO	0.1	0.2	Finalizado	Finalizado

2.4.3 Plan de iteraciones

Luego de estimar el esfuerzo con el cual serán desarrolladas las historias de usuario y contar con un plan de entrega, se da paso a la planificación de la etapa de implementación del sistema. En este plan se recogen las historias de usuario que serán implementadas en cada iteración, de acuerdo al orden preestablecido. De acuerdo con lo antes mencionado se decide realizar el sistema en cuatro iteraciones, las cuales se especifican a continuación:

Iteración primera:

Esta primera iteración tiene como meta implementar las historias de usuarios 1, 2, 3 y 5, las cuales son: gestionar usuario, gestionar prueba, gestionar batería de prueba y gestionar normativas. Las mencionadas historias de usuario son de vital importancia para la aplicación pues constituyen la estructura básica del sistema, además, a partir de ellas se derivan las demás funcionalidades. Con la finalización de esta iteración se obtendrá la primera entrega del sistema con el propósito de mostrar al cliente lo realizado y recibir retroalimentación del mismo.

Iteración segunda:

En esta iteración se implementarán las historias de usuario que tienen prioridad media para el cliente. Al concluir dicha iteración se contará con las funcionalidades descritas en las historias de usuario 4 y 10, las cuales son: evaluar datos de una batería de pruebas y evaluar datos de una prueba. La versión de prueba referente a esta iteración junto con las implementaciones anteriores, serán mostradas al cliente con el objetivo de realizar cambios en base a la opinión del mismo.

Iteración tercera:

El objetivo de esta iteración es la implementación de funcionalidades que tienen prioridad media para el cliente y cuentan con un riesgo de desarrollo alto. Con la finalización de la misma se tendrán implementadas las peticiones del cliente descritas en las historias de usuario 6, 7 y 11, las cuales son: ranking por grupo, autenticar y ranking por prueba. Como resultado de esta iteración se tendrá una parte fundamental de la aplicación.

Iteración cuarta:

En esta iteración se implementarán las restantes funcionalidades que tienen prioridad baja para el cliente con un riesgo de desarrollo alto para el equipo. Con la culminación de la misma se obtendrán las historias de usuario 8 y 9, las cuales son: posibles evaluaciones y ranking por año. Al finalizar esta iteración, luego de que haya pasado satisfactoriamente por la etapa de pruebas se dispondrá de la aplicación con todas las funcionalidades descritas por el cliente.

2.4.5 Plan de duración de iteraciones

Como parte del ciclo de vida de un proyecto utilizando la metodología XP se crea el plan de duración de cada una de las iteraciones. Mediante este plan se obtiene una medida de cuánto tiempo conllevará realizar las historias de usuario y el orden de implementación de las mismas.

Tabla 17 Plan de duración de iteraciones

Iteraciones	Historias de usuario a implementar	Duración total de las iteraciones
Primera	1-Gestionar usuario 2-Gestionar prueba 3-Gestionar batería de prueba 5-Gestionar normativas	3(semanas)
Segunda	4-Evaluar datos de una batería de pruebas 10-Evaluar datos de una prueba	3(semanas)
Tercera	6-Ranking por grupo 7-Autenticar 11-Ranking por prueba	3(semanas)
Cuarta	8-Posibles evaluaciones 9-Ranking por año	3(semanas)

2.4.6 Historias de usuario dividas en tareas

Las historias de usuario están embebidas dentro de tareas de programación, que serán definidas en cada iteración. Estas tareas son escritas en fichas como las historias de usuario, pero en el lenguaje de los programadores. Cada tarea debería ser estimada en un período de uno a tres días de desarrollo.

2.4.6.1Iteración primera

Tabla 18 Tareas abordadas en la primera iteración

Historia de usuario	Tareas
Gestionar usuario	-Definir las tabla de usuarios
Gestionar prueba	-Definir las tablas de pruebas

Gestionar batería de prueba	-Definir las tablas de la batería de pruebas
Gestionar normativas	-Definir las tablas de las normativas

Tabla 19 Tarea genérica

No. de la Historia de Usuario	Tarea genérica
1,2,3,5	-Sincronizar con la base de datos

Tabla 20 Sincronizar con la base de datos

Tarea	
Número de tarea: 1	**Número de HU:** 1,2,3,5
Nombre de la tarea: Sincronizar con la base de datos.	
Tipo de tarea: Configuración	
Fecha de inicio: 8 de febrero de 2010	**Fecha de fin:** 8 de febrero de 2010
Programador responsable: Mario Alejandro Remón-Leonel Camaraza	
Descripción: Se van a sincronizar los datos con la base de datos.	

Tabla 21 Definir las tablas de usuario

Tarea	
Número de tarea: 2	**Número de HU:** 1
Nombre de la tarea: Definir las tablas de usuario	
Tipo de tarea: Configuración	
Fecha de inicio: 9 de febrero de 2010	**Fecha de fin:** 9 de febrero de 2010
Programador responsable: Mario Alejandro Remón-Leonel Camaraza	
Descripción: Se van a definir las clases de usuario que se corresponden con las entidades de igual nombre en la base de datos.	

Tabla 22 Definir las tablas de prueba

Tarea	
Número de tarea: 3	**Número de HU:** 2
Nombre de la tarea: Definir las tablas de prueba	
Tipo de tarea: Configuración	
Fecha de inicio: 9 de febrero de 2010	**Fecha de fin:** 10 de febrero de 2010
Programador responsable: Mario Alejandro Remón-Leonel Camaraza	
Descripción: Se van a definir las clases de la prueba en el archivo model.py del framework Django, que se corresponden con las entidades de igual nombre en la base de datos.	

Tabla 23 Definir las tablas de la batería de pruebas

Tarea	
Número de tarea: 4	**Número de HU:** 3
Nombre de la tarea: Definir los tablas de la batería de pruebas	
Tipo de tarea: Configuración	
Fecha de inicio: 10 de febrero de 2010	**Fecha de fin:** 10 de febrero de 2010
Programador responsable: Mario Alejandro Remón-Leonel Camaraza	
Descripción: Se van a definir las clases de la batería de pruebas en el archivo model.py del framework Django, que se corresponden con las entidades de igual nombre en la base de datos.	

Tabla 24 Definir las tablas de las normativas

Tarea	
Número de tarea: 5	**Número de HU:** 5
Nombre de la tarea: Definir las tablas de las normativas	
Tipo de tarea: Desarrollo	
Fecha de inicio: 10 de febrero de 2010	**Fecha de fin:** 12 de febrero de 2010
Programador responsable: Mario Alejandro Remón-Leonel Camaraza	
Descripción: Se van a definir las clases de las normativas en el archivo model.py del framework Django, que se corresponden con las entidades de igual nombre en la base de datos.	

2.4.6.2 Iteración segunda

Tabla 25 Tareas abordadas en la segunda iteración

Historia de usuario	**Tareas**
Evaluar datos de una batería de pruebas	-Crear template de evaluar datos de batería -Permitir introducir datos de la batería de pruebas -Calcular evaluaciones -Mostrar resultados
Evaluar datos de una prueba	-Crear template de evaluar datos de una prueba -Mostrar formulario para seleccionar prueba

Tabla 26 Crear template de evaluar datos de batería

Tarea	
Número de tarea: 6	**Número de HU:** 4
Nombre de la tarea: Crear template de evaluar datos de batería	
Tipo de tarea: Desarrollo	
Fecha de inicio: 15 de febrero de 2010	**Fecha de fin:** 17 de febrero de 2010
Programador responsable: Mario Alejandro Remón-Leonel Camaraza	
Descripción: Se crea un archivo con la extensión HTML, el mismo hereda de la plantilla base y define los bloques que se van a implementar para evaluar datos de una batería de prueba.	

Tabla 27 Permitir introducir datos de la batería de pruebas

Tarea	
Número de tarea: 7	**Número de HU:** 4
Nombre de la tarea: Permitir introducir datos de la batería de pruebas	
Tipo de tarea: Desarrollo	
Fecha de inicio: 1 de marzo de 2010	**Fecha de fin:** 2 de marzo de 2010
Programador responsable: Mario Alejandro Remón-Leonel Camaraza	
Descripción: El usuario podrá introducir los resultados recogidos en las pruebas.	

Tabla 28 Calcular evaluaciones

Tarea	
Número de tarea: 8	**Número de HU:** 4
Nombre de la tarea: Calcular evaluación	
Tipo de tarea: Desarrollo	
Fecha de inicio: 3 de marzo de 2010	**Fecha de fin:** 5 de marzo de 2010
Programador responsable: Mario Alejandro Remón-Leonel Camaraza	

Descripción: Dada la batería de pruebas se calculan las posibles evaluaciones que el estudiante podría obtener.

Tabla 29 Mostrar resultados

Tarea	
Número de tarea: 9	**Número de HU:** 4
Nombre de la tarea: Mostrar resultados	
Tipo de tarea: Desarrollo	
Fecha de inicio: 6 de marzo de 2010	**Fecha de fin:** 8 de marzo de 2010
Programador responsable: Mario Alejandro Remón-Leonel Camaraza	
Descripción: Muestra los resultados en una tabla, mostrando los nombres de los estudiantes, las pruebas, los resultados en cada prueba y una evaluación general de las pruebas que sería la evaluación de la batería.	

Tabla 30 Crear template de evaluar datos de una prueba

Tarea	
Número de tarea: 10	**Número de HU:** 11
Nombre de la tarea: Crear template de evaluar datos de una prueba	
Tipo de tarea: Desarrollo	
Fecha de inicio: 8 de marzo de 2010	**Fecha de fin:** 11 de marzo de 2010
Programador responsable: Mario Alejandro Remón-Leonel Camaraza	
Descripción: Se crea un archivo con la extensión HTML, el mismo hereda de la plantilla base y define los bloques que se van a implementar para evaluar datos de una de prueba.	

Tabla 31 Mostrar formulario para seleccionar una prueba

Tarea	
Número de tarea: 11	**Número de HU:** 11
Nombre de la tarea: Mostrar formulario para seleccionar una prueba	
Tipo de tarea: Desarrollo	
Fecha de inicio: 12 de marzo de 2010	**Fecha de fin:** 15 de marzo de 2010
Programador responsable: Mario Alejandro Remón-Leonel Camaraza	
Descripción: Se le muestra un formulario al usuario en el cual podrá escoger tanto el grupo como la prueba que desee y luego se visualizará el resultado, siendo calculado previamente.	

2.4.6.3 Iteración tercera

Tabla 32 Tareas abordadas en la iteración tercera

Historia de usuario	**Tareas**
Ranking por grupo	-Crear template para ranking por grupo -Permitir escoger el grupo -Implementar vista -Mostrar ranking
Autenticar	-Definir los privilegios del usuario -Conectar al servidor LDAP
Ranking por prueba	-Crear template para ranking por prueba -Permitir escoger la prueba -Implementar vista -Mostrar ranking

Tabla 33 Crear template para ranking por grupo

Tarea	
Número de tarea: 12	**Número de HU:** 6
Nombre de la tarea: Crear template para ranking por grupo	
Tipo de tarea: Desarrollo	
Fecha de inicio: 22 de marzo de 2010	**Fecha de fin:** 25 de marzo de 2010
Programador responsable: Mario Alejandro Remón-Leonel Camaraza	
Descripción: Se crea un archivo con la extensión HTML, el mismo hereda de la plantilla base y define los bloques que se van a implementar para mostrar los resultados de un grupo en un ranking.	

Tabla 34 Permitir escoger el grupo

Tarea	
Número de tarea: 13	**Número de HU:** 6
Nombre de la tarea: Permitir escoger el grupo	
Tipo de tarea: Desarrollo	
Fecha de inicio: 25 de marzo de 2010	**Fecha de fin:** 27 de marzo de 2010
Programador responsable: Mario Alejandro Remón-Leonel Camaraza	
Descripción: El sistema brinda la posibilidad de escoger el grupo que el usuario desee.	

Tabla 35 Implementar vista

Tarea	
Número de tarea: 14	**Número de HU:** 6
Nombre de la tarea: Implementar vista	
Tipo de tarea: Desarrollo	
Fecha de inicio: 29 de marzo de 2010	**Fecha de fin:** 31 de marzo de 2010
Programador responsable: Mario Alejandro Remón-Leonel Camaraza	

Descripción: Método para extraer los datos de la base de datos y mostrarlos al usuario.

Tabla 36 Mostrar ranking

Tarea	
Número de tarea: 15	**Número de HU:** 6
Nombre de la tarea: Mostrar ranking	
Tipo de tarea: Desarrollo	
Fecha de inicio: 31 de marzo de 2010	**Fecha de fin:** 31 de marzo de 2010
Programador responsable: Mario Alejandro Remón-Leonel Camaraza	
Descripción: Dados los datos extraídos de la base de datos se muestra una tabla con los mismos.	

Tabla 37 Definir los privilegios del usuario

Tarea	
Número de tarea: 16	**Número de HU:** 7
Nombre de la tarea: Definir los privilegios del usuario	
Tipo de tarea: Desarrollo	
Fecha de inicio: 31 de marzo de 2010	**Fecha de fin:** 31 de marzo de 2010
Programador responsable: Mario Alejandro Remón-Leonel Camaraza	
Descripción: Se van a precisar los privilegios que le permitirán navegar a los usuarios en la aplicación.	

Tabla 38 Conectar al servidor LDAP

Tarea	
Número de tarea: 17	**Número de HU:** 7
Nombre de la tarea: Conectar al servidor LDAP	

Tipo de tarea: Desarrollo	
Fecha de inicio: 1 de abril de 2010	**Fecha de fin:** 2 de abril de 2010
Programador responsable: Mario Alejandro Remón-Leonel Camaraza	
Descripción: Conectarse al servidor LDAP, para que los usuarios puedan autenticarse.	

Tabla 39 Crear template para ranking por prueba

Tarea	
Número de tarea: 18	**Número de HU:** 11
Nombre de la tarea: Crear template para ranking por prueba	
Tipo de tarea: Desarrollo	
Fecha de inicio: 2 de abril de 2010	**Fecha de fin:** 3 de abril de 2010
Programador responsable: Mario Alejandro Remón-Leonel Camaraza	
Descripción: Se crea un archivo con la extensión HTML, el mismo hereda de la plantilla base y define los bloques que se van a implementar para obtener un plan dada un prueba.	

Tabla 40 Permitir escoger la prueba

Tarea	
Número de tarea: 19	**Número de HU:** 11
Nombre de la tarea: Permitir escoger la prueba	
Tipo de tarea: Desarrollo	
Fecha de inicio: 5 de abril de 2010	**Fecha de fin:** 7 de abril de 2010
Programador responsable: Mario Alejandro Remón-Leonel Camaraza	
Descripción: El sistema brinda la posibilidad de escoger la prueba que el usuario desee.	

Tabla 41 Implementar vista

Tarea	
Número de tarea: 20	**Número de HU:** 11
Nombre de la tarea: Implementar vista	
Tipo de tarea: Desarrollo	
Fecha de inicio: 7 de abril de 2010	**Fecha de fin:** 9 de abril de 2010
Programador responsable: Mario Alejandro Remón-Leonel Camaraza	
Descripción: Se obtiene de la base de datos todo lo referente al grupo y la prueba que formarán el ranking.	

Tabla 42 Mostrar ranking

Tarea	
Número de tarea: 21	**Número de HU:** 11
Nombre de la tarea: Mostrar ranking	
Tipo de tarea: Desarrollo	
Fecha de inicio: 12 de abril de 2010	**Fecha de fin:** 15 de abril de 2010
Programador responsable: Mario Alejandro Remón-Leonel Camaraza	
Descripción: Dados los datos extraídos de la base de datos se muestra una tabla con los mismos.	

2.4.6.4 Iteración cuarta

Tabla 43 Tareas abordadas en la iteración cuarta

Historia de usuario	**Tareas**
Posibles evaluaciones	-Crear template para posibles evaluaciones

	-Crear formulario para escoger la batería de prueba -Fijar establecer las evaluaciones que desea obtener el estudiante -Calcular plan de evaluaciones - Mostrar plan de evaluaciones
Ranking por año	- Crear template para ranking por año -Crear Formulario -Implementar vista -Mostrar ranking

Tabla 44 Crear template para posibles evaluaciones

Tarea	
Número de tarea: 27	**Número de HU:** 8
Nombre de la tarea: Crear template para posibles evaluaciones	
Tipo de tarea: Desarrollo	
Fecha de inicio: 19 de abril de 2010	**Fecha de fin:** 20 de abril de 2010
Programador responsable: Mario Alejandro Remón-Leonel Camaraza	
Descripción: Se crea un archivo nombrad con la extensión HTML, el mismo hereda de la plantilla base y define los bloques que se van a implementar para obtener una combinación de posibles evaluaciones.	

Tabla 45 Crear formulario para escoger la batería de prueba

Tarea	
Número de tarea: 28	**Número de HU:** 8
Nombre de la tarea: Crear formulario para escoger la batería de prueba	
Tipo de tarea: Desarrollo	
Fecha de inicio: 22 de abril de 2010	**Fecha de fin:** 24 de abril de 2010
Programador responsable: Mario Alejandro Remón-Leonel Camaraza	

Descripción: El usuario determina que batería de pruebas desea.

Tabla 46 Fijar las evaluaciones que desea obtener el estudiante

Tarea	
Número de tarea: 29	**Número de HU:** 8
Nombre de la tarea: Fijar las evaluaciones que desea obtener el estudiante	
Tipo de tarea: Desarrollo	
Fecha de inicio: 26 de abril de 2010	**Fecha de fin:** 26 de abril de 2010
Programador responsable: Mario Alejandro Remón-Leonel Camaraza	
Descripción: Se muestran las posibles evaluaciones que obtendría un estudiante dado una batería de prueba determinada.	

Tabla 47 Calcular plan de evaluaciones

Tarea	
Número de tarea: 30	**Número de HU:** 8
Nombre de la tarea: Calcular plan de evaluaciones	
Tipo de tarea: Desarrollo	
Fecha de inicio: 26 de abril de 2010	**Fecha de fin:** 28 de abril de 2010
Programador responsable: Mario Alejandro Remón-Leonel Camaraza	
Descripción: Dada la batería de pruebas se calculan las posibles evaluaciones que el estudiante podría obtener.	

Tabla 48 Mostrar plan de evaluaciones

Tarea	
Número de tarea: 31	**Número de HU:** 8
Nombre de la tarea: Mostrar plan de evaluaciones	
Tipo de tarea: Desarrollo	
Fecha de inicio: 28 de abril de 2010	**Fecha de fin:** 30 de abril de 2010

Programador responsable: Mario Alejandro Remón-Leonel Camaraza
Descripción: Muestra una tabla con el plan de evaluaciones a seguir.

Tabla 49 Crear template para ranking por año

Tarea	
Número de tarea: 32	**Número de HU:** 10
Nombre de la tarea: Crear template para ranking por año	
Tipo de tarea: Desarrollo	
Fecha de inicio: 28 de abril de 2010	**Fecha de fin:** 29 de abril de 2010
Programador responsable: Mario Alejandro Remón-Leonel Camaraza	
Descripción: Se crea un archivo nombrad con la extensión HTML, el mismo hereda de la plantilla base y define los bloques que se van a implementar para mostrar el ranking por año.	

Tabla 50 Crear formulario

Tarea	
Número de tarea: 33	**Número de HU:** 10
Nombre de la tarea: Crear formulario	
Tipo de tarea: Desarrollo	
Fecha de inicio: 30 de abril de 2010	**Fecha de fin:** 1 de mayo de 2010
Programador responsable: Mario Alejandro Remón-Leonel Camaraza	
Descripción: El usuario determina el año que será mostrado en el ranking.	

Tabla 51 Implementar vista

Tarea	
Número de tarea: 34	**Número de HU:** 10
Nombre de la tarea: Implementar vista	
Tipo de tarea: Desarrollo	
Fecha de inicio: 30 de abril de 2010	**Fecha de fin:** 2 de mayo de 2010
Programador responsable: Mario Alejandro Remón-Leonel Camaraza	
Descripción: Método para sacar los datos de la base de datos y mostrarlos al usuario**.**	

Tabla 52 Mostrar ranking

Tarea	
Número de tarea: 35	**Número de HU:** 10
Nombre de la tarea: Mostrar ranking	
Tipo de tarea: Desarrollo	
Fecha de inicio: 3 de mayo de 2010	**Fecha de fin:** 5 de mayo de 2010
Programador responsable: Mario Alejandro Remón-Leonel Camaraza	
Descripción: Dados los datos extraídos de la base de datos se muestra una tabla con los mismos.	

2.5 Tarjetas CRC (Cargo o clase, Responsabilidad y Colaboración)

Para diseñar el sistema como un equipo es debe cumplir con tres principios: Cargo o clase, Responsabilidad y Colaboración (CRC). Las tarjetas CRC permiten apartarse del método de trabajo basado en procedimientos y centrarse más en una metodología basada en objetos. Las tarjetas CRC permiten que el equipo entero contribuya al diseño. Una tarjeta CRC es usada para representar un objeto.

Una tarjeta CRC es usada para representar un objeto. El nombre de la clase se escribe en la parte superior de la tarjeta a modo de título, las responsabilidades de esta clase son escritas en el lado izquierdo y la colaboración con otras clases se lista a la derecha de cada responsabilidad.

Las tarjetas CRC determina el comportamiento de cada actividad. En nuestro sistema existen disimiles objetos, por lo que en nuestra aplicación está formada por las siguientes clases:

- Persona
- Estudiante
- Profesor
- Grupo
- Percentil
- Prueba
- Batería de pruebas
- Resultado por prueba
- Normativa

En la siguiente figura se ilustra cómo estará conformada una clase

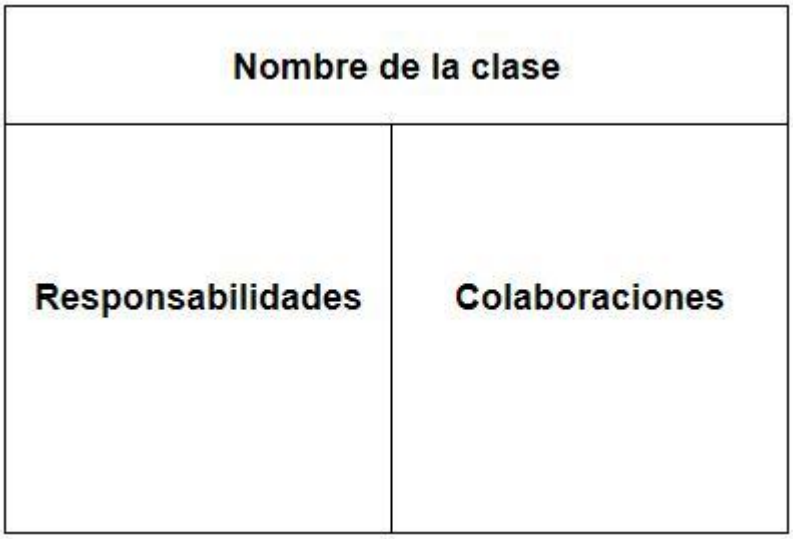


Figura 2 Modelo de una Tarjeta CRC

Tabla 53 Tarjeta CRC clase Persona

Persona	
Define los atributos de los usuarios que interactúan con la aplicación.	

Tabla 54 Tarjeta CRC clase Estudiante

Estudiante	
Hereda de Persona.	Persona

Tabla 55 Tarjeta CRC clase Profesor

Profesor	
Hereda de Persona, redefiniendo nuevos atributos.	Persona

Tabla 56 Tarjeta CRC clase Grupo

Grupo	
Define las propiedades de los grupos de estudiantes.	Estudiante, Profesor

Tabla 57 Tarjeta CRC clase Percentil

Percentil	
Define las propiedades de los percentiles.	Profesor

Tabla 58 Tarjeta CRC clase Prueba

Prueba	
Define las propiedades de las pruebas.	Profesor

Calcular la evaluación de los estudiantes en una prueba.	

Tabla 59 Tarjeta CRC clase Batería de Prueba

Batería de prueba	
Define las propiedades de las baterías de prueba.	Profesor
Calcula las evaluaciones de los estudiantes en una batería de prueba.	Prueba

Tabla 60 Tarjeta CRC clase Batería por prueba

Resultado por prueba	
De cada estudiante se recogen los resultados obtenidos en una prueba determinada	Estudiante
Calcular ranking.	Prueba aplicada

Tabla 61 Tarjeta CRC clase Normativa

Normativa	
Define los datos de las normativas	Profesor
Establecer normativas.	Percentil

2.6 Conclusiones

Con la finalización de este capítulo se describe una propuesta del sistema que se desea implementar obteniéndose el plan de entrega como parte de la fase de planificación, con lo cual se deja plasmado el alcance que tendrá el trabajo. Este plan es flexible puesto que la metodología XP abraza los cambios que puedan surgir durante el transcurso del desarrollo de la aplicación. Igualmente se dejaron claras las iteraciones por las cuales va a transitar la aplicación, señalando las historias de usuario que serán implementadas en cada una de ellas. Como parte del diseño del sistema se describieron las tarjetas CRC, las cuales representan cada clase del sistema, logrando con este método aunar a todo el equipo para conformar el diseño del sistema.

Capítulo 3 Implementación y prueba

3.1 Introducción

Como parte del incremento progresivo en un proyecto de software es importante dejar bien claro cuáles serán los pasos que se seguirán para llevar una implementación diáfana. En este sentido los estándares de codificación son formas que posibilitan la organización de todo el desarrollo que se lleve a cabo durante el progreso del software que se desee obtener. Por otra parte la metodología XP plantea que se pruebe constantemente, recibiendo en todo momento retroalimentación del cliente. De esta manera se consigue manejar los errores que puedan surgir de forma impecable. En el siguiente capítulo se abordará todo lo concerniente a la implementación y a las pruebas hechas al sistema.

3.2 Estándares de codificación

Los estándares de codificación ayudan a garantizar que todos los usuarios tengan acceso a la información que el sitio Web les brinda, igualmente será más fácil para las personas con necesidades específicas utilizar el sistema (28). Puesto que el código es la mejor documentación en el equipo de desarrollo, se acuerda un estándar para el código, dejando de lado estilos de programación particulares. En este sentido el lenguaje Python, utilizado para la implementación de la aplicación cuenta con un estilo que mejora ampliamente la legibilidad del código.

Para un mayor entendimiento, se muestran los pasos que se siguieron para escribir el código.

3.2.1 Definiciones generales

Las definiciones se realizarán de manera descriptiva, evitando las abreviaturas y los nombres cortos.

Clases:

Las clases deben comenzar con mayúscula al inicio de la palabra y en caso de estar conformada por palabras compuestas, la definición debe ser continua y cada palabra debe iniciar con mayúscula siguiendo el estilo determinado.

Ejemplos:

```
class BateriaDePrueba( models.Model ):
…
class RankingxGrupoForm(forms.Form):
…
```

Métodos:

Los métodos deben empezar con minúsculas, escribiéndose de manera seguida en caso de estar conformados por palabras compuestas. Se escoge iniciar las funcionalidades con minúsculas para diferenciarlas de las clases, pero en caso de ser compuestas, se colocan con mayúsculas las palabras que no son las primeras de la definición.

Ejemplos:

```
def establecerNormativas( request ):
…
def evaluacionXBateriaSig(request):
…
```

Variables:

Las variables comenzarán con minúsculas, aquellas que sean compuestas estarán separadas por un guión bajo.

Ejemplos:

```
desclasificados = []
normativas_prueba = []
```

3.2.2 Comentarios

Se utilizan comentarios para especificar clases o funciones. La utilización de los mismos no estará presente en todo el código, solo en aquellos casos que sea necesario.

Ficheros:

Al inicio de cada fichero, especificando las clases o funciones que comprende.

Ejemplo:

```
'''
Archivo que permite la definición de las tablas de la base de datos
'''
```

Clases y funciones:

Aquellas *clases* o *funciones* en las que se utilicen comentarios para especificar su papel en el código, deben tener dicha aclaración seguida de su definición, buscando una mayor interrelación con el código y evitando que se confunda con el código que le antecede.

Ejemplo:

```
def evaluacionXBateriaSig(request):
#Método para Calcular las evaluaciones solo con las normativas de los
#profesores de acuerdo a sus pruebas aplicadas
```

3.3 Diagramas

Los diagramas favorecen un mejor entendimiento de la implementación del software. En este sentido el equipo elaboró dos diagramas, que permitirán comprender con claridad lo que se ha implementado. En los siguientes apartados se abordarán los diagramas dispuestos por el equipo de desarrollo.

3.3.1 Diagrama de Despliegue

Un Diagrama de Despliegue modela la arquitectura en tiempo de ejecución de un sistema. Esto muestra la configuración de los componentes de hardware (nodos) y muestra cómo los elementos y artefactos del software se trazan en esos nodos.

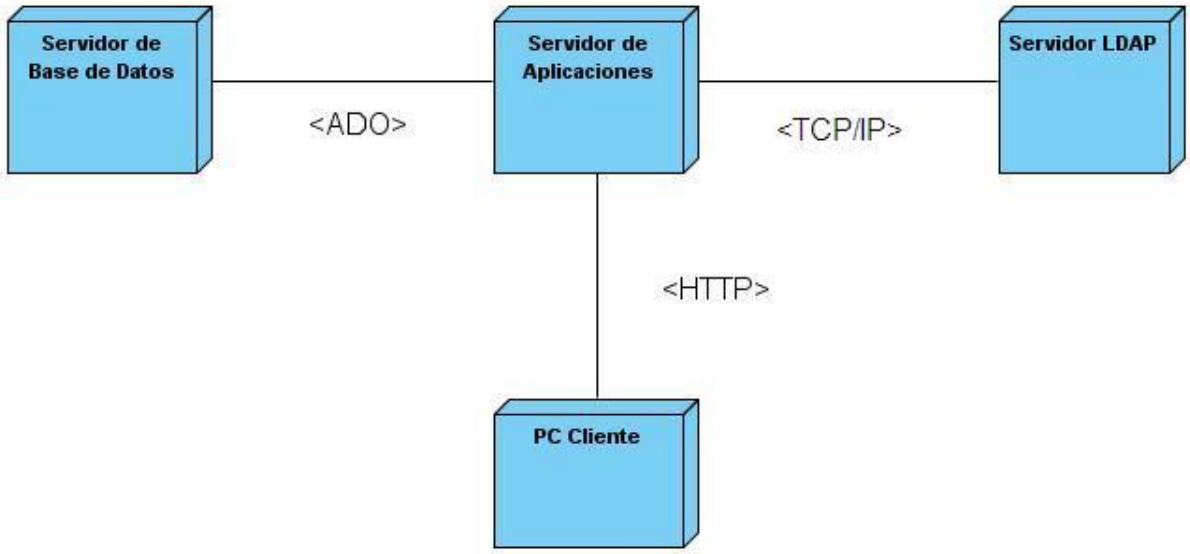


Figura 3 Diagrama de Despliegue

3.3.2 Diagrama Entidad-Relación

Mediante un diagrama Entidad-Relación se modela todo lo referente a la disposición de la base de datos que se ha creado.

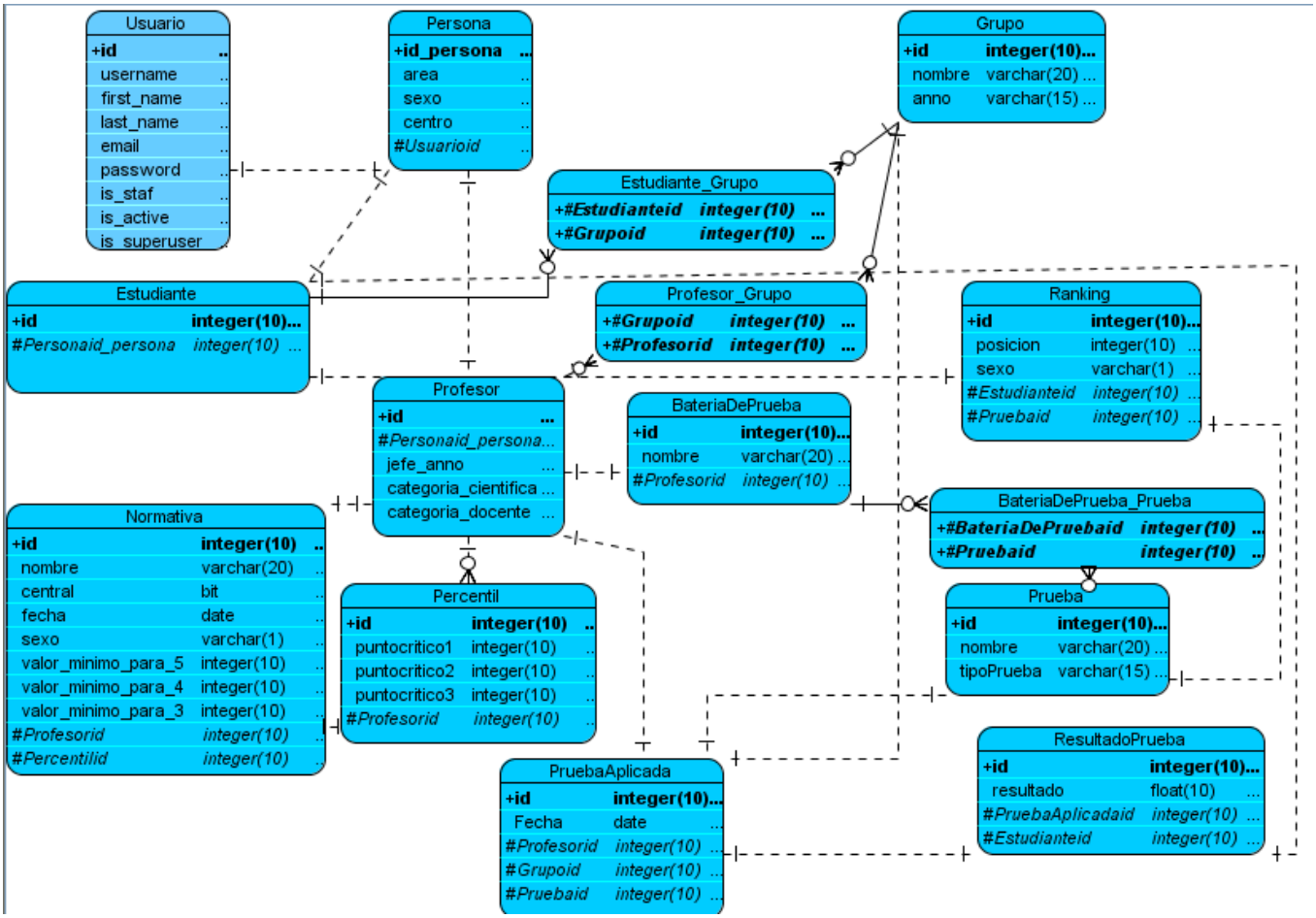


Figura 4 Diagrama Entidad-Relación

3.5 Prueba

Uno de los pilares de la metodología eXtreme Programming es la etapa de pruebas. Es muy importante que se pruebe constantemente el software. Esto permite aumentar la calidad de los sistemas reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección. También incrementa la seguridad de evitar efectos colaterales no deseados a la hora de realizar modificaciones y refactorizaciones

XP divide las pruebas del sistema en dos grupos: pruebas unitarias, encargadas de verificar el código y diseñada por los programadores, y pruebas de aceptación o pruebas

funcionales destinadas a evaluar si al final de una iteración se consiguió la funcionalidad requerida diseñadas por el cliente final.

3.5.1 Pruebas de Aceptación

El objetivo de estas pruebas es verificar los requisitos, por este motivo, los propios requisitos del sistema son la principal fuente de información a la hora de construir las pruebas de aceptación. Las pruebas de aceptación son creadas a partir de las historias de usuario. Durante una iteración la historia de usuario seleccionada en la planificación de iteraciones se convertirá en una prueba de aceptación.

Una historia de usuario puede tener más de una prueba de aceptación, tantas como sean necesarias para garantizar su correcto funcionamiento y no se considera completa hasta que no supera sus pruebas de aceptación. Una prueba de aceptación es como una caja negra. Cada una de ellas representa una salida esperada del sistema. Es responsabilidad del cliente verificar la corrección de las pruebas de aceptación y tomar decisiones a cerca de las mismas.

A continuación se muestran las pruebas de aceptación propuestas a realizarse por iteración para una mayor organización.

3.5.1.1 Iteración primera

Tabla 62 Prueba 1 HU Gestionar Usuario

Caso de Prueba de Aceptación	
Código: HU1_P1	**Historia de Usuario:** 1
Nombre: Adicionar un usuario	
Descripción: Prueba para la funcionalidad de adicionar un usuario	
Condiciones de ejecución: El administrador debe estar autenticado. Se utilizará un usuario con datos válidos.	
Entradas/Pasos de ejecución:	

Para añadir un usuario en el sistema se insertan los datos válidos del mismo y luego se le define el rol que le corresponde.
Resultado esperado: El usuario es añadido sin errores, de lo contrario muestra un mensaje señalando que el usuario ya existe.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 63 Prueba 2 HU Gestionar Usuario

Caso de Prueba de Aceptación	
Código: HU1_P2	**Historia de Usuario:** 1
Nombre: Modificar un usuario	
Descripción: Prueba para la funcionalidad de modificar un usuario	
Condiciones de ejecución: El administrador y el jefe de año deben estar autenticados. Se usará un usuario con datos válidos. El usuario que se quiere modificar debe estar inscrito en la aplicación.	
Entradas/Pasos de ejecución: Para modificar a uno de los usuarios del sistema se dará paso a escoger al usuario previamente inscrito en él, y luego se procederá a modificar dicho usuario.	
Resultado esperado: El usuario es modificado sin generar errores.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 64 Prueba 3 HU Gestionar Usuario

Caso de Prueba de Aceptación	
Código: HU1_P3	**Historia de Usuario:** 1
Nombre: Eliminar un usuario	
Descripción: Prueba para la funcionalidad de eliminar un usuario	
Condiciones de ejecución:	

El administrador y el jefe de año deben estar autenticados. Se usará un usuario con datos válidos. El usuario que se quiere eliminar debe estar inscrito en la aplicación.
Entradas/Pasos de ejecución: Para eliminar a uno de los usuarios del sistema se dará paso a escoger el usuario previamente inscrito en él, y luego se procederá a eliminar dicho usuario.
Resultado esperado: El usuario es eliminado sin generar errores.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 65 Prueba 1 HU Gestionar Prueba

Caso de Prueba de Aceptación	
Código: HU2_P1	**Historia de Usuario:** 2
Nombre: Añadir una prueba.	
Descripción: Prueba para la funcionalidad de añadir una prueba.	
Condiciones de ejecución: El jefe de año o un profesor deben estar autenticados. Se usará un usuario con datos válidos.	
Entradas/Pasos de ejecución: Se añade una prueba, especificando el nombre de la misma. Luego se determina el tipo de prueba o sea si va a ser de tiempo o de marca. Por último se da paso a añadirla	
Resultado esperado: La prueba es añadida sin generar errores.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 66 Prueba 2 HU Gestionar Prueba

Caso de Prueba de Aceptación	
Código: HU2_P3	**Historia de Usuario:** 2
Nombre: Modificar una prueba.	
Descripción: Prueba para la funcionalidad de modificar una prueba.	
Condiciones de ejecución: El jefe de año o un profesor deben estar autenticados. La prueba debe estar insertada. Se usará un usuario con datos válidos.	
Entradas/Pasos de ejecución: Se da paso a escoger una prueba añadida de antemano, luego se elige el tipo de la prueba o sea si es de tiempo o de marca y a continuación se modifica con los datos nuevos.	
Resultado esperado: La prueba es modificada sin generar errores.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 67 Prueba 3 HU Gestionar Prueba

Caso de Prueba de Aceptación	
Código: HU2_P2	**Historia de Usuario:** 2
Nombre: Eliminar una prueba.	
Descripción: Prueba para la funcionalidad de eliminar una prueba.	
Condiciones de ejecución: El jefe de año o un profesor deben estar autenticados. La prueba debe estar insertada. Se usará un usuario con datos válidos.	
Entradas/Pasos de ejecución:	

Se da paso a escoger una prueba añadida de antemano, luego se elige el tipo de la prueba o sea si es de tiempo o de marca y después se elimina. El profesor escoge una prueba que
Resultado esperado: La prueba es eliminada sin generar errores.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 68 Prueba 1 HU Gestionar Batería de Prueba

Caso de Prueba de Aceptación	
Código: HU3_P1	**Historia de Usuario:** 3
Nombre: Añadir una batería de prueba.	
Descripción: Prueba para la funcionalidad de añadir una batería de prueba.	
Condiciones de ejecución: El jefe de año o un profesor deben estar autenticados. Se usará un usuario con datos válidos.	
Entradas/Pasos de ejecución: Se especifica el nombre de la batería de prueba. Luego se escoge el profesor que la crea, acto seguido se seleccionan las pruebas previamente añadidas, que formarán parte de la batería de prueba. Por último se da paso a añadir la batería de prueba.	
Resultado esperado: La batería de prueba es añadida sin generar errores.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 69 Prueba 2 HU Gestionar Batería de Prueba

Caso de Prueba de Aceptación	
Código: HU3_P2	**Historia de Usuario:** 3
Nombre: Modificar una batería de prueba.	
Descripción: Prueba para la funcionalidad de modificar una batería de prueba.	

Condiciones de ejecución: El jefe de año o un profesor deben estar autenticados. Se usará un usuario con datos válidos.
Entradas/Pasos de ejecución: Se da paso a escoger una batería de prueba añadida de antemano y a continuación se modifica con los datos nuevos.
Resultado esperado: La batería de prueba es modificada sin generar errores.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 70 Prueba 3 HU Gestionar Batería de Prueba

Caso de Prueba de Aceptación	
Código: HU3_P3	**Historia de Usuario:** 3
Nombre: Eliminar una batería de prueba.	
Descripción: Prueba para la funcionalidad de eliminar una batería de prueba.	
Condiciones de ejecución: El jefe de año o un profesor deben estar autenticados. Se usará un usuario con datos válidos.	
Entradas/Pasos de ejecución: Se da paso a escoger una batería de prueba añadida antes y después se elimina.	
Resultado esperado: La batería de prueba es eliminada sin generar errores.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 71 Prueba 1 HU Gestionar normativas

Caso de Prueba de Aceptación	
Código: HU5_P1	**Historia de Usuario:** 5
Nombre: Añadir una normativa.	
Descripción: Prueba para la funcionalidad de añadir una normativa.	
Condiciones de ejecución: El jefe de año o un profesor deben estar autenticados. Se usará un usuario con datos válidos.	
Entradas/Pasos de ejecución: Para añadir una normativa a la aplicación se debe escoger el percentil anteriormente introducido y luego se marca la prueba a la cual se le va a aplicar la normativa.	
Resultado esperado: La normativa es añadida sin generar errores, de lo contrario muestra un mensaje indicando que la normativa ya existe.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 72 Prueba 2 HU Gestionar normativas

Caso de Prueba de Aceptación	
Código: HU5_P2	**Historia de Usuario:** 5
Nombre: Modificar una normativa.	
Descripción: Prueba para la funcionalidad de modificar una normativa.	
Condiciones de ejecución: El jefe de año o un profesor deben estar autenticados. Se usará un usuario con datos válidos. La normativa que se seleccione debe estar inscrita en la aplicación.	
Entradas/Pasos de ejecución: Para modificar una normativa se debe escoger la normativa anteriormente añadida y	

luego se da paso a introducir los nuevos datos.
Resultado esperado: La normativa es modificada sin generar errores.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 73 Prueba 3 HU Gestionar normativas

Caso de Prueba de Aceptación	
Código: HU5_P3	**Historia de Usuario:** 5
Nombre: Eliminar una normativa.	
Descripción: Prueba para la funcionalidad de eliminar una normativa.	
Condiciones de ejecución: El jefe de año o un profesor deben estar autenticados. Se usará un usuario con datos válidos. La normativa que se seleccione debe estar inscrita en la aplicación.	
Entradas/Pasos de ejecución: Se escoge la normativa y luego se da paso a su eliminación.	
Resultado esperado: La normativa es eliminada sin generar errores.	
Evaluación de la prueba: Prueba satisfactoria.	

3.5.1.2 Iteración segunda

Tabla 74 Prueba 1 HU Evaluar datos de una batería de prueba

Caso de Prueba de Aceptación	
Código: HU4_P1	**Historia de Usuario:** 4
Nombre: Evaluar datos de una batería de prueba.	
Descripción: Prueba para la funcionalidad de evaluar datos de una batería de prueba.	
Condiciones de ejecución:	

El jefe de año o un profesor deben estar autenticados. Se usará un usuario con datos válidos. El grupo al cual se va a evaluar debe estar inscrito al igual que la batería de prueba.
Entradas/Pasos de ejecución: Se debe escoger el grupo al cual se va a evaluar y la batería de las pruebas que realizo el grupo. También se debe especificar si se desean utilizar las normativas establecidas por el INDER.
Resultado esperado: Se muestran las evaluaciones de cada estudiante del grupo obtenidas en cada prueba de la batería.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 75 Prueba 1 HU Evaluar datos de una prueba

Caso de Prueba de Aceptación	
Código: HU11_P1	**Historia de Usuario:** 11
Nombre: Evaluar datos de una prueba.	
Descripción: Prueba para la funcionalidad de evaluar datos de una prueba.	
Condiciones de ejecución: El jefe de año o un profesor deben estar autenticados. Se usará un usuario con datos válidos. El grupo al cual se va a evaluar debe estar inscrito al igual que la prueba.	
Entradas/Pasos de ejecución: Se debe escoger el grupo al cual se va a evaluar y la prueba que hizo el grupo. También se debe especificar si se desean utilizar las normativas establecidas por el INDER.	
Resultado esperado: Se muestran las evaluaciones de cada estudiante del grupo los resultados que obtuvieron en la prueba.	
Evaluación de la prueba: Prueba satisfactoria.	

3.5.1.3 Iteración tercera

Tabla 76 Prueba 1 HU Ranking por grupo

Caso de Prueba de Aceptación	
Código: HU6_P1	**Historia de Usuario:** 6
Nombre: Ranking por grupo.	
Descripción: Prueba para la funcionalidad del ranking por grupo.	
Condiciones de ejecución: El jefe de año, un profesor o un estudiante deben estar autenticados. Se usará un usuario con datos válidos. Es obligado que exista el grupo al cual se le va consultar el ranking.	
Entradas/Pasos de ejecución: Para consultar el ranking, se escoge el grupo y se determina el sexo por el cual se desea consultar el ranking	
Resultado esperado: Se muestran las posiciones de cada estudiante con los resultados que alcanzaron en las pruebas.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 77 Prueba 1 HU Autenticar

Caso de Prueba de Aceptación	
Código: HU7_P1	**Historia de Usuario:** 7
Nombre: Autenticar.	
Descripción: Prueba para la funcionalidad de autenticar.	
Condiciones de ejecución: El jefe de año, un profesor o un estudiante deben estar autenticados. Se usará un usuario con datos válidos. Es obligado que exista el grupo al cual se le va consultar el ranking.	

Entradas/Pasos de ejecución: Para consultar el ranking, se escoge el grupo y se determina el sexo por el cual se desea consultar el ranking
Resultado esperado: Se muestran las posiciones de cada estudiante con los resultados que alcanzaron en las pruebas.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 78 Prueba 1 HU Ranking por prueba

Caso de Prueba de Aceptación	
Código: HU12_P1	**Historia de Usuario:** 12
Nombre: Ranking por prueba.	
Descripción: Prueba para la funcionalidad del ranking por prueba.	
Condiciones de ejecución: El jefe de año, un profesor o un estudiante deben estar autenticados. Se usará un usuario con datos válidos. Es vital que exista tanto el grupo al cual se le va consultar el ranking como la prueba por la cual se medirá la consulta.	
Entradas/Pasos de ejecución: Para consultar el ranking, se escoge el grupo y se determina el sexo por el cual se desea consultar el ranking	
Resultado esperado: Se muestran las posiciones de cada estudiante con los resultados que alcanzaron en la prueba.	
Evaluación de la prueba: Prueba satisfactoria.	

3.5.1.3 Iteración cuarta

Tabla 79 Prueba 1 HU Posibles evaluaciones

Caso de Prueba de Aceptación	
Código: HU8_P1	**Historia de Usuario:** 8
Nombre: Posibles evaluaciones.	
Descripción: Prueba para la funcionalidad del posibles evaluaciones.	
Condiciones de ejecución: El jefe de año, un profesor o un estudiante deben estar autenticados. Se usará un usuario con datos válidos. Es vital que exista la batería de pruebas por la cual se medirá la consulta.	
Entradas/Pasos de ejecución: Para calcular las posibles evaluaciones se escoge la batería de pruebas. Luego se da paso a calcularlas.	
Resultado esperado: Se muestran las posibles evaluaciones que obtendrá el estudiante.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 80 Prueba 1 HU Ranking por año

Caso de Prueba de Aceptación	
Código: HU8_P1	**Historia de Usuario:** 8
Nombre: Ranking por año.	
Descripción: Prueba para la funcionalidad del ranking por año.	
Condiciones de ejecución: El jefe de año, un profesor o un estudiante deben estar autenticados. Se usará un usuario con datos válidos. Es vital que exista tanto el año al cual se le va consultar el ranking como la prueba por la cual se medirá la consulta.	

Entradas/Pasos de ejecución: Para consultar el ranking se escoge el año para realizar la consulta.
Resultado esperado: Se muestran las posibles evaluaciones que obtendrá el estudiante.
Evaluación de la prueba: Prueba satisfactoria.

3.6 Conclusiones

Con la finalización de este capítulo se dejan sentadas las bases necesarias para la implementación de la aplicación. Con la especificación de los estándares de codificación se abarca todo lo referente a la estructura que contendrá el desarrollo del sistema. Los diagrama posibilitan un mejor entendimiento de cómo quedará el sistema. Por otra parte se propusieron las pruebas de aceptación, que brindarán al cliente conformidad y seguridad ante las funcionalidades del sistema.

CONCLUSIONES GENERALES

Después de haber realizado un exhaustivo estudio enfocado en el campo de la educación física, se demostró la necesidad de construir un sistema capaz de gestionar toda la información referente a esta área y de esta forma lograr una mayor seguridad, facilidad y rapidez en el manejo de toda esta información. Mediante la metodología de desarrollo de software XP se documentó todo el ciclo de desarrollo del producto, lo cual facilitará el estudio del sistema por parte de otros desarrolladores para futuras mejoras. Se comprobó con la utilización del framework Django lo beneficiosa que es esta herramienta para la implementación de aplicaciones web, en conjunto con el lenguaje de programación Python, puesto que proporciona un marco de trabajo fácil y cómodo, lo que permitió un desarrollo rápido del sistema y que estuviese guiado por una arquitectura estandarizada. Con este sistema se brinda un entorno amigable que facilita la confortabilidad en el trabajo cotidiano de los profesores. Podrán crear sus propias normativas, lo cual permite contar en la UCI con un sistema de normativas inherentes a las necesidades de su estudiantado. También se incluye la manera de elaborar un ranking individual por prueba, y general considerando la batería con el conjunto de pruebas, lo cual aumentará la motivación de los estudiantes posibilitando incrementar el rendimiento de los alumnos en las clases de educación física. La versión de la aplicación obtenida fue validada y comprobada, demostrando así su utilidad y factibilidad, además de construir una base de datos para el almacenamiento de la información para el trabajo en Departamento de Educación Física.

RECOMENDACIONES

En el transcurso del proceso investigativo que se llevó a cabo han surgido algunas ideas que sería recomendable tener en cuenta para el mejoramiento del sistema:

- Agregar nuevas funcionalidades de acuerdo a las expectativas que surjan en el Departamento de Educación Física.
- Potenciar el uso de la aplicación en otros centros tanto de carácter deportivo como educativo.
- Realizar refactorización(o sea, reescribir ciertas partes del código para aumentar su legibilidad pero sin modificar su comportamiento) al código, con el objetivo de mejorar el sistema.

REFERENCIAS BIBLIOGÁFICAS

1. **Munuera Ruiz, Antonio.** *Nuevas Tecnologías de la información y la comunicación (TICs) en la Educación Física.* La Habana: 2006.

2. ***Universidad Católica de Valparaíso*** [Consultado el: 1 de marzo de 2010]. Disponible en: http://www.ucv.cl/web/estadistica/percentil.htm.

3. **Fuente Pérez, Armando.** *Sistema De Elaboración De Normativas, Baterías de Pruebas y Ranking para aplicar a Test Físicos Pedagógicos del Deporte y la Educación Física utilizando una Herramienta Web.* La Habana: 2007.

4. **Española, Real Academia.** *DICCIONARIO DE LA LENGUA ESPAÑOLA* Madrid: 2001.

5. **Ortega García, Fara Olivia.** *AUTOMATIZACIÓN DE LA PLANIFICACIÓN DEL ENTRENAMIENTO DEPORTIVO EN DIFERENTES DEPORTES* Facultad de Cultura Física Villa Clara

6. ***Sitio oficial de Administrador de datos*** [Consultado el: 10 de marzo de 2010]. Disponible en: http://ore.master-ore.com/sw/datos.html#arriba.

7. ***Sitio oficial de Informática & Deportes*** [Consultado el: 15 de febrero de 2010]. Disponible en: http://www.entrenar.com.ar.

8. ***Sitio oficial de Woplanner*** [Consultado el: 17 de febrero de 2010]. Disponible en: http://www.woplanner.com.

9. **Services, Centers for Medicare & Medicaid.** *SELECTING A DEVELOPMENT APPROACH*

10. **Mendosa, María de los Angeles.** *Metodologías de Desarrollo de Software* [Consultado el: 20 de febrero de 2010 de 2010]. Disponible en: http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html. .

11. **Jacobson, Ivar; Booch, Grady; Rumbaugh, James.** *El Proceso Unificado de Desarrollo de Software* La Habana: Félix Varela,

12. ***Sitio oficial de DSDM*** [Consultado el: 10 de marzo de 2010]. Disponible en: http://www.dsdm.org/.

13. **Cheesman, John; Daniels, John.** *UML components, a simple process for specifying component based software* Londres.

14. **Del Juncal Huerta, Jorge Luis; Gómez Landeros, Ruth Priscila.** [Consultado el: 22 de febrero de 2010]. Disponible en: http://www.monografias.com/trabajos14/herramicase/herramicase.shtml. .

15. **Pavón Mestras, Juan.** *Rational Rose: Ingeniería del Software 2* Facultad de Informática. Universidad Complutense Madrid (UCM). [Consultado el: 17 de febrero de 2010]. Disponible en: http://www.fdi.ucm.es/profesor/jpavon/is2/Lab01RationalRose.pdf. .

16. ***Sitio oficial de Visual Paradigm*** [Consultado el: 18 de febrero de 2010]. Disponible en: www.visual-paradigm.com/.

17. **Markiewicz, Marcus Eduardo; de Lucen, Carlos J.P**. *Sitio oficial The Association for Computing Machinery. The ACM Student Magazine* [Consultado el: 10 de marzo de 2010]. Disponible en: http://www.acm.org/crossroads/espanol/xrds7-4/frameworks.html. .

18. **Potencier, Fabien**. *Symfony la guía definitiva* [Consultado el: 16 de febrero de 2010]. Disponible en: http://www.librosweb.es/symfony_1_0. .

19. **Pronstad Thomas, Westerlund Vegar.***Mandatory Security Measures in Web Applications. Norwegian University of Science and Technology Faculty of Information Technology, Mathematics and Electrical Engineering Department of Computer and Information Science, 2007*

20. **Suárez Torres, Domingo; Reyes Z., José Juan**. *Sitio oficial de SpringHispano* [Consultado el: 8 de marzo de 2010]. Disponible en: http://www.springhispano.org/.

21. **Holovaty, Adrian; Kaplan-Moss, Jacob**. *El libro de Django.* España: 2007.

22. ***Sitio oficial de ASP .NET*** [Consultado el: 15 de marzo de 2010]. Disponible en: http://www.asp.net/.

23. **González Duque, Raúl.** *Python para todos.* España: 2006.

24. **Instituto Geográfico Agustín Codazzi -IGAC-.** , [Consultado el: 9 de marzo de 2010]. Disponible en: http://www.igac.gov.co:8080/igac_web/UserFiles/File/ciaf/TutorialSIG_2005_26_02/paginas/ctr_sistemasdegestiondebasededatos.htm. .

25. **Daniel, Pecos.** *PostgreSQL_MySQL* [Consultado el: 17 de febrero de 2010]. Disponible en: http://danielpecos.com/docs/mysql_postgres/x15.html. .

26. **Todo expertos desarrollado por Avanzis(2)**, [Consultado el: 11 de marzo de 2010]. Disponible en: http://www.todoexpertos.com/categorias/tecnologia-e-internet/bases-de-datos/oracle/respuestas/14706/vetajas-y-desventajas.

27. ***Sitio oficial de Glosario*** [Consultado el: 24 de febrero de 2010]. Disponible en: www.glosario.net.

28. **Montero Garrido, Jesús Manuel.** *Plataforma Eclipse. Introducción Técnica*

29. ***Sitio oficial de Wing IDE*** [Consultado el: 3 de marzo de 2010]. Disponible en: http://www.wingide.org.

BIBLIOGRAFÍA

1. ***Revista Chilena de Ingeniería.*** Desafíos en la integridad de señales ed. 2010, vol. 17 No. 1, ISBN 0718-3305

2. **Cristina Curto, Isabel Gelabert; Carles González, José Morales**. *Experiencias con éxito de aprendizaje cooperativo en Educación Física.* Celesa, 2009 ed. ISBN 849729145X, 9788497291453.

3. **Documentation, P. O.** *PostgreSQL 8.4 Server Programming* Fultus Corporation, Disponible en: http://books.google.com.cu/books?id=ICEje5Q5lfQC&pg=PA36&dq=postgres&lr=&as_drrb_is=b&as_minm_is=1&as_miny_is=2009&as_maxm_is=1&as_maxy_is=2010&as_brr=0&cd=3#v=onepage&q=postgres&f=false ISBN 1-59682-160-4.

4. **Heurtel, Olivier.** *PHP y MySQL Domine el desarrollo de un sitio Web dinámico e interactivo.* ENI-Enero 2009 ed. ISBN 978-2-7460-4502-6.

5. **Laurent Debrauwer, Fien Van der Heyde.** *UML 2 Iniciación, ejemplos y ejercicios corregidos* ENI. ed. Disponible en: http://books.google.com.cu/books?id=GkP8GxPUzK4C&pg=PA21&dq=RUP&lr=&as_drrb_is=b&as_minm_is=1&as_miny_is=2009&as_maxm_is=1&as_maxy_is=2010&as_brr=0&cd=1#v=onepage&q=RUP&f=false ISBN 978-2-7460-4741-9.

6. **Lutz, Mark.** *Learning Python* O´Reilly 4 Edición. ed. Disponible en: http://books.google.com.cu/books?id=1HxWGezDZcgC&printsec=frontcover&dq=Python&lr=&as_drrb_is=b&as_minm_is=1&as_miny_is=2009&as_maxm_is=1&as_maxy_is=2010&as_brr=0&cd=1#v=onepage&q&f=false ISBN 978-0-596-15806-4.

7. **Pekka Abrahamsson, Michele; Marchesi Maurer, Frank.** *Agile Processes in Software Engineering and Extreme Programming.* Springer-Verlag Berlin Heidelberg 20092009. ISBN 1865-1348.

8. **Pila Hernández, Hemeregildo.** *Selección de talentos para el deporte,27 años de experiencia en Cuba, metodología para evaluar las pruebas* Disponible en: http://www.efdeportes.com/efd69/talento.htm.

9. **Riso, Walter.** *Terapia cognitiva.* Paidós Ibérica ed. 2008. ISBN 978-84-493-2195-5.

10. **Teuber Henríquez, Pablo Sebastián.** *Propuesta de mejoras a la metodología de desarrollo de software de ORDEN integración en relación a las áreas de proceso de ingeniería del modelo de calidad CMMI. Chile.* Universidad de Concepción, 2008.

GLOSARIO DE TÉRMINOS

UCI: Universidad de las Ciencias Informáticas

Python: Lenguaje de programación creado por Guido van Rossum a principios de los años 90. Se trata de un lenguaje interpretado o de script, con tipado dinámico, fuertemente tipado, multiplataforma y orientado a objetos.

Código abierto: Es una tendencia internacional del desarrollo del software que profesa la distribución del código junto a las aplicaciones.

API: Del inglés *Application Programming Interface* – Interfaz de Programación de Aplicaciones, es el conjunto de funciones y procedimientos (o métodos si se refiere a programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

LDAP: (*Lightweight Directory Access Protocol*), (Protocolo Ligero de Acceso a Directorios) es un protocolo a nivel de aplicación que permite el acceso a un servicio de directorio ordenado y distribuido.

IDE: Integrated Development Enviroment (Ambiente Integrado de Desarrollo).

UML: Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, Unified Modelling Language) es el lenguaje de modelado de sistemas de software más conocido en la actualidad.

Feedback: Es el continuo seguimiento que recibimos a la hora de laborar en un entorno de desarrollo ágil.

Multiplataforma: Puede ser ejecutado en diferentes Sistemas Operativos (SO).

Plug-ins: Programas que deben instalarse en los ordenadores clientes

GPL: Es una licencia pública general de GNU o más conocida por su nombre en inglés *General Public License*. Está orientada principalmente a proteger la libre distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios.