Universidad de las ciencias informáticas

Centro de informática industrial (CEDIN)

Facultad 5

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas.

Tema: Sistema de monitoreo de bicicletas ergométricas mediante microcontroladores AVR.

Autor: Edward Cruz Rivery

Tutores: Ing. Julio Alberto Leyva Durán

Ing. Yordan Carlos Pérez Attanaff

Lic. Julián Hernández Domínguez

Declaración de autoría

Declaro por este medio que yo, Edward Cruz Rivery, con carné de identidad 91022622366, soy el autor principal del trabajo final de tesis “Sistema de monitoreo de las bicicletas ergométricas, mediante microcontroladores AVR” y que autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

Para que así conste firmo la presente a los __ días del mes de ____ del año ____

_____

Edward Cruz Rivery

Autor

_____

Ing. Yordan Carlos Pérez Attanaf

Tutor

_____

Lic. Julián Hernández Domínguez

Tutor

_____

Ing. Julio Alberto Leyva Durán

Tutor

Resumen

El desarrollo y perfeccionamiento de las capacidades físicas se fundamenta en cualquier proceso pedagógico, en un conjunto de regularidades, leyes y principios, los cuales constituyen una especie de guía metodológica para profesores de Educación Física y entrenadores deportivos. En el gimnasio de la UCI, la planificación y dosificación de la capacidad física de resistencia no logra un entrenamiento continuo, sistemático y pedagógicamente organizado debido a la variedad de métodos que se utilizan para la evaluación fisiológica del entrenamiento físico. A partir de lo anterior, se plantea como objetivo, desarrollar una aplicación web para supervisar el funcionamiento de las bicicletas ergométricas, para que, a partir de los datos que se recolecten de esta, se le pueda dar un seguimiento al deportista, evaluar y diagnosticar con más precisión la evolución de su entrenamiento.

Para llevar a cabo el desarrollo del sistema fue seleccionado el framework Symfony2, una herramienta potente para la creación de aplicaciones web, la plataforma de hardware abierto Arduino y el protocolo de comunicación Modbus en su variante TCP. Todo esto sobre las bases que brindan los lenguajes php5 para el servidor y Arduino para la tarjeta de adquisición de datos, guiando el proceso de desarrollo por la metodología AUP-UCI.

Palabras Clave: aplicación web, bicicleta ergométrica, diagnóstico, supervisión.

Indices

Índice de Ilustraciones

Índice de Tablas

Introducción

El desarrollo y difusión de la información ha alcanzado fuerza a nivel mundial en los últimos años, llegando a estar en cada evento actual de la sociedad. Esto ha llevado al ser humano a un nivel superior sobre sus antepasados ya que uno de los principales problemas de la sociedad es la necesidad de almacenar información y a partir de esta, tomar decisiones que le permitan remediar errores o mejorar decisiones, acciones o eventos que conllevan al desarrollo de la humanidad. Uno de los eventos sociales con mayor necesidad de información es el deporte, siendo esta de vital importancia para el entrenador y el atleta guardar toda información posible sobre su entrenamiento para a partir de esta tomar decisiones que les permitan mejorar los resultados del atleta. Dentro de los medios de entrenamientos que se emplean para entrenar la capacidad física de resistencia se encuentran las bicicletas ergométricas. Actualmente existen sistemas adjuntos a estos aparatos para realizar ejercicios que logran extraer los datos de los esfuerzos realizados por el atleta durante el entrenamiento que le permiten al cuerpo de entrenadores, a través de toda la información que se acumula tomar decisiones y a partir de ella ver cuál es camino que debe tomar la preparación del deportista.

La Universidad de las Ciencias Informáticas (UCI) como parte del proceso de informatización de la sociedad cubana, ha realizado varios proyectos para la satisfacción de la sociedad. Al Centro de informática industrial (CEDIN) se le ha propuesto resolver el problema que presentan los entrenadores de la universidad en la gestión de los entrenamientos en las bicicletas ergométricas para reforzar la resistencia de un atleta. Este proceso se realiza de manera manual ya que no existe ningún sistema informático que gestione la información de un atleta capturando de manera automática los esfuerzos realizados durante los entrenamientos, lo cual dificulta la mejora del proceso y obliga al entrenador a estar presente todo el tiempo durante el entrenamiento.

Teniendo en cuenta dicha situación se plantea el presente **problema de investigación**:

¿Cómo supervisar el proceso de entrenamiento de los atletas universitarios obteniendo los datos de manera automática de las bicicletas ergométricas?

Tomando este punto de partida se define como **objeto de estudio** las aplicaciones web para la supervisión de entrenamientos lo cual enmarca esta investigación en el **campo de acción** de las aplicaciones web.

Para dar solución a este problema se plantea como **objetivo** desarrollar una aplicación web que logre mostrar y almacenar de manera automática los datos extraídos del entrenamiento del atleta en las bicicletas ergométricas para que permita al profesor tomar decisiones más exactas para su desarrollo.

Las tareas investigativas a realizar para asegurar el cumplimiento al objetivo planteado son:

- Elaboración de un marco teórico de la investigación a través de un estudio del estado de las aplicaciones existentes.
- Definición de los requisitos para el desarrollo de la aplicación.
- Diseño de la aplicación según los requisitos obtenidos para el desarrollo.
- Implementación de la aplicación como propuesta de solución.
- Realización de pruebas a la aplicación para validar su correcto funcionamiento.

Para guiar la investigación se utilizarán los siguientes métodos científicos:

Métodos teóricos

- **Análisis histórico-lógico:** para el estudio de trabajos similares e investigaciones que abordan el tema de la supervisión de la evolución del ejercicio a nivel nacional e internacional, así como sistemas relacionados con este proceso, que sirvan como punto de partida para el desarrollo de la solución en su conjunto.
- **Método analítico-sintético:** para analizar desde diferentes aristas los conceptos asociados a las plataformas web y los microcontroladores AVR y así, sintetizar la información recopilada, permitiendo describir las características generales y las relaciones esenciales entre estas.
- **Método de la modelación:** para crear abstracciones con el objetivo de explicar la realidad, se utilizará para la modelación de los diversos diagramas necesarios en cada uno de los flujos de trabajo según la metodología seleccionada

.

Métodos empíricos

El método **observación científica** se empleará con el objetivo de observar el funcionamiento de algunas plataformas web para el control de la evolución de un deportista recibiendo un entrenamiento, para obtener un registro visual de las características comunes en estas y que pueden formar parte de la solución.

El presente trabajo consta con la siguiente estructura capitular:

Capítulo 1: “Marco teórico de la investigación” relacionado con el desarrollo de aplicaciones web para controlar la evolución de un deportista a través de un tratamiento deportivo. Se realiza un estudio de las soluciones existentes en el ámbito nacional e internacional. Además, se explica la metodología, herramientas y tendencias actuales a utilizar en el desarrollo de la aplicación web para controlar la evolución de dicho paciente a través de esta tecnología.

Capítulo 2: “Fases planificación y diseño”. Se presenta la descripción de la solución propuesta al problema de investigación. Se describe el funcionamiento general del mecanismo a desarrollar en relación con sus características y componentes. Se definen los elementos de documentación más significativos tras la aplicación de la metodología de desarrollo seleccionada.

Capítulo 3: “Fases desarrollo y pruebas”. Se presentan las tareas de ingeniería desarrolladas, se expone la convención de codificación utilizada en la obtención de la solución propuesta. Se realizan las pruebas de aceptación para validar los requerimientos del cliente. Se presenta una carta aceptación emitida por el cliente donde se expresa su conformidad la aplicación web realizada.

Capítulo 1: Marco teórico de la investigación.

Introducción del capítulo

Para formalizar una propuesta de solución es imprescindible acercarse al entorno actual en que se encuentra enmarcada la investigación. En el presente capítulo se incluyen los principales conceptos asociados al dominio del problema, un estudio del estado actual que llevan aplicaciones para supervisar la evolución de un individuo a través de las bicicletas ergométricas, los principales marcos de trabajo de desarrollo para PHP, los gestores de base de datos, el empleo de los microcontroladores como un recurso esencial para los Sistemas Embebidos y las características de los principales microcontroladores que más se emplean en la actualidad, las metodologías de desarrollo de software.

1.1 Fundamentación del tema

Para un mejor entendimiento, se hace necesario relacionar un conjunto de términos y conceptos asociados al dominio del problema:

Entrenamiento

"Se define al entrenamiento como un proceso sistemático repetitivo y progresivo de ejercicios, teniendo como objetivo el mejoramiento de la performance atlética" (1). El entrenamiento es un conjunto de procedimientos y actividades realizadas para aumentar la capacidad física, desarrollando las cualidades de un individuo de la forma más adecuada y en función de las circunstancias.

Todo programa de entrenamiento deberá atender a una serie de elementos: Nutrición, practicas defatigantes, procesos formativos y de aprendizaje de técnicas deportivas concretas.

Los programas de entrenamiento deben instruirse desde las primeras etapas de la vida iniciándose con actividades de educación física básica. Se han definido actividades físicas según edades, por ejemplo para un niño que inicias sus actividades motoras deben proponerse juegos y actividades tendentes para mejorar la psicomotricidad, la movilidad, el equilibrio, el sentido del ritmo y la percepción y control del espacio.

Los principios de los entrenamientos físicos vienen definidos por los siguientes conceptos:

- Especificidad: Si cada estimulo produce una reacción y adaptación específica, deberá buscarse el tipo de actividad concreta que conduzca a los resultados a conseguir. Si el objetivo es obtener resistencia, la carrera es la más adecuada que el levantamiento de pesas. Por el contrario si la fuerza es el objetivo prioritario, será el trabajo con pesas la esencia del entrenamiento.

- Supercompensación: Es el proceso de adaptación por el que el organismo a través del entrenamiento consigue un nivel superior de capacidad física. Tras un esfuerzo se produce fatiga y tras la recuperación que el descanso produce, el organismo alcanza un nivel superior de capacidad.
- Periodización: Los programas de entrenamiento deben referirse siempre a un periodo de tiempo que en general siguen una secuencia de temporada.
- Individualización: El entrenamiento debe siempre adaptarse a las condiciones del individuo considerando la edad, las características morfológicas y el trabajo físico previo.
- Variedad: Deben realizarse ejercicios de diversa índole para cubrir todo el espectro de necesidades musculoesqueléticas. La repetición exagerada de un mismo tipo de movimiento puede condicionar fatiga crónica y provocar disminución del rendimiento físico (2).

Con el trascurso del tiempo se han creado diversos tipos de aparatos mecánicos para hacer ejercicios con el fin de apoyar los entrenamientos. Estos ayudan a perfeccionar las condiciones físicas de quien los utilizan y permite que los ejercicios puedan realizarse en áreas limitadas. Ejemplos de estos son: Las máquinas de andar y correr, remos sentados en máquinas y las bicicletas ergométricas.

Bicicletas Ergométricas

La bicicleta ergométrica o bicicleta estática es una de las máquinas (totalmente mecánicas) más antiguas para hacer ejercicios. Unas de sus primeras variantes se vieron reflejadas por primera vez en Gran Bretaña en el año 1796 a partir de tener una bicicleta fija para prácticas deportivas en áreas limitadas. La popularidad de las bicicletas ergométricas viene de su simplicidad de construcción, de su tamaño relevante y compacto y de sus grandes aplicaciones, comenzando desde una terapia de rehabilitación y terminando en el uso de los astronautas en la gravedad cero.

Las bicicletas ergométricas tienen muchas funcionalidades además de ser el arma alternativa a las máquinas de andar y correr para quemar calorías. Puede ayudar en la tonicidad de la musculatura si se usa en sus niveles de resistencia más alto (3).

El uso de las bicicletas estáticas ha tomado gran auge en los últimos años, evolucionando las mismas mediante la aplicación de diversas tecnologías. Uno de los principales avances ha sido la extracción de los datos de los ejercicios que se realizan en estas para de forma tal que el usuario pueda apreciar o valorar los esfuerzos que está realizando y poder

planificarse de una mejor manera. Una de las tecnologías que utilizada para la extracción de datos es el uso de los microcontroladores mediantes sistemas embebidos.

Microcontroladores

"Un microcontrolador, a veces conocido como MCU (por sus siglas en ingles Micro-Controller Unit), es básicamente un circuito integrado por millones de semiconductores (transistores) interconectados entre sí para dar paso a módulos con funciones lógicas o analógicas previamente definidas por el diseñador del microcontrolador."

Aunque los microcontroladores pueden utilizarse en aplicaciones para procesamiento de datos o información compleja, como en una computadora o un analizador de imágenes, en realidad son más lentos que los microprocesadores. Los microcontroladores están pensados para aplicaciones de rápido manejo, atención y respuesta a eventos externos o internos del producto.

La arquitectura de los MCUs se basa en la composición de un circuito integrado por varios módulos y un núcleo. El núcleo es el que se conoce por "core" y es el que define las características principales del microcontrolador: velocidad y número de bits que se pueden controlar. Existen microcontroladores de 8-bits, 16-bits hasta los 32-bits. Hoy en día, los MCUs también tienen características de relevancia después del core, como los son el tamaño y el tipo de la flash (memoria no volátil), la RAM (por sus siglas en ingles Random Access Memory – memoria volátil) (4).

Hay, principalmente, tres familias de microcontroladores: Microchip, Freescale-Motorola y Atmel, cuyas características se exponen a continuación:

	ATMEL® ATMEGA16	Microchip® PIC16F877A	Freescale® MC68HC908AP16
RAM	1024	368	1024
ROM	16KBytes	14.3KBytes	16KBytes
EEPROM;	512Bytes	256Bytes	×
Puertos I/O	32	33	32
Oscilador	Interno 8MHz Externo 16MHz	Externo 20MHz	Interno 32MHz
Tiempos de instrucción	1-5 Cicl. reloj	4/8 Cicl. reloj	1-7 Cicl. reloj
Arquitectura	RISC	RISC	CISC
Registros de trabajo	32	1	1
USART	✓	✓	×
I^2C	✓	✓	✓
SPI	✓	✓	✓
ADC	8 Canales 10 bits	8 Canales 10 bits	8 Canales 10 bits
JTAG	✓	×	×
Multiplicador	✓	×	✓
Estado	Activo	Activo	Obsoleto

Ilustración 1: Tabla comparativa de las principales familias de microcontroladores

En el centro de informática industrial de la Universidad de las Ciencias Informáticas se utiliza un set de microcontroladores AVR disponibles para el desarrollo de sistemas embebidos.

Microcontroladores AVR

Los MCUs AVR son una familia de microcontroladores RISC (Reduced intruction Set Computer) del fabricante estadounidense Atmel. La arquitectura de los AVR fue concebida por dos estudiantes en el *Norwegian Institute of Technology*, y posteriormente refinada y desarrollada en Atmel Norway, la empresa subsidiaria de Atmel, fundada por los dos arquitectos del chip. Debido a su diseño simple y la facilidad de programación cuenta con bastantes aficionados hacia esta tecnología. El microcontrolador AVR es un CPU de arquitectura Harvard, tiene 32 registros de 8 bits. La concatenación de los 32 registros, los registros de entrada/salida y la memoria de datos conforman un espacio de direcciones unificado, al cual se accede a través de operaciones de carga/almacenamiento.
El AVR fue diseñado desde un comienzo para la ejecución eficiente de código C compilado. Como este lenguaje utiliza profusamente punteros para el manejo de variables en memoria, los tres últimos pares de registros internos del procesador son usados como punteros de 16 bit al espacio de memoria externa (5).

El set de instrucciones AVR está implementado físicamente y disponible en el mercado en diferentes dispositivos, que comparten el mismo núcleo AVR pero tienen distintos periféricos y cantidades de RAM y ROM: desde el microcontrolador de la familia *Tiny AVR* ATtiny11 con 1KB de memoria flash y sin RAM (sólo los 32 registros), y 8 pines, hasta el microcontrolador ATmega2560 de la familia *Mega AVR* con 256KB de memoria flash, 8KB de memoria RAM, 4KB de memoria EEPROM, conversor análogo digital de 10 bits y 16 canales, temporizadores, comparador analógico, JTAG, etc. La compatibilidad entre los distintos modelos es preservada en un grado razonable.

Los microcontroladores AVR tienen una cañería ('pipeline' en inglés) con dos etapas (cargar y ejecutar), que les permite ejecutar la mayoría de las instrucciones en un ciclo de reloj, lo que los hace relativamente rápidos entre los microcontroladores de 8-bit.

El desarrollo de sistemas mediante la utilización directa de MCUs se dificulta debido a las especificidades del lenguaje y configuraciones propias de este tipo unidades, por ello se opta por una capa superior basada en dichos dispositivos.

Arduino

Arduino es un proyecto de hardware libre, que ideó y desarrolló una plataforma completa de hardware y software compuestos por placas de desarrollo que integran un microcontrolador y un entorno de desarrollo (IDE), es diseñado para facilitar el uso de la electrónica en proyectos multidisciplinarios. Todos sus componentes tanto el software como el hardware son liberados bajo licencia de código abierto.

Consiste en una placa de circuito impreso con un microcontrolador, usualmente Atmel AVR, y puertos digitales y analógicos de entrada/salida, los cuales pueden conectarse a placas de expansión (shields) que amplían las características de funcionamiento de la placa arduino.

A partir de Octubre del año 2012, se incorporaron nuevos modelos de placas de desarrollo que hacen uso de microcontroladores CortexM3, ARM de 32 bits, que coexisten con los originales modelos que integran microcontroladores AVR de 8 bits. ARM y AVR no son plataformas compatibles a nivel binario, pero se pueden programar y compilar bajo el IDE clásico de Arduino sin ningún cambio (6).

1.2 Análisis de Sistemas similares.

En la actualidad existen múltiples sistemas que facilitan la planificación del entrenamiento deportivo. Estos varían en dependencia de la disciplina deportiva, ya que no todos los deportes entrenan de la misma forma ni con la misma intensidad. Actualmente existen

sistemas que han colaborado con la planificación del entrenamiento de la capacidad física de deportes en diversos países.

1.2.1 Sistemas similares a nivel internacional

En el mundo existen sistemas que se encargan de la planificación del entrenamiento deportivo para diferentes deportes.

***X-MEDALIST**:* Es un software que se encarga de planificar y controlar el entrenamiento para deportes individuales. Es un programa que está orientado a entrenadores de cualquier deporte individual. Permite llevar un registro y control de las lesiones, historia clínica y deportiva, y del análisis de la técnica individual y el biorritmo personal. También permite revisar toda la información que se encuentra en las planificaciones en forma de lista simple o bien combinada entre actividades o deportistas. Dispone de una herramienta que le permite comparar volúmenes de diferentes áreas entre dos o más deportistas y brinda la posibilidad de exportar todos los resultados a formato *Excel* (7) .

X-Training FUSSION: Es una versión de un programa de computación para la planificación, periodización y control del entrenamiento deportivo. Este programa está pensado y diseñado para cubrir todas las expectativas de los entrenadores personales, preparadores físicos, técnicos de diferentes deportes, clubes, instituciones deportivas, laboratorios de evaluaciones deportivas, etc. Además, brinda la posibilidad de confeccionar planificaciones para un deportista individual, o bien para un equipo de cualquier deporte.
Para ahorrarle tiempo y esfuerzo, tiene implementado un sistema que permite exportar las planillas de evaluaciones a *Excel* para poder cargar los resultados de las evaluaciones directamente en el campo de entrenamiento (7).

Espacio virtual para profesores y entrenadores de judo: Es un sitio que puede ser usado como guía por los entrenadores para la realización de diversas actividades entre las que se pueden citar: impartir cursos de judo, explicar las didácticas del judo, así como dar a conocer los parámetros a tener en cuenta para que un judoca alcance la forma óptima. Es simplemente una Web de información al entrenador (8).

Elatleta.com: Es la página Web oficial del club deportivo *Páris*, cuyo objetivo es informar de todo lo relacionado con el atletismo. Este plan de entrenamiento pretende fortalecer el organismo de las personas que practican el atletismo, para ello se centra en mejorar la capacidad aeróbica y la musculatura; especialmente las piernas y todos los músculos que protegen la columna. Es una aplicación Web de estructura dinámica que brinda mucha información a los entrenadores de atletismo (9).

1.2.2 Sistemas similares a nivel nacional

Automatización de la planificación del entrenamiento deportivo en diferentes deportes.

Es una aplicación realizada por la Facultad de Cultura Física de Villa Clara para el entrenamiento deportivo de diferentes deportes. Como resultado de la investigación se obtuvo el diseño y programación de sistemas automatizados para la planificación del atletismo, las pesas como deporte, las pesas como deporte auxiliar y la esgrima. Para la elaboración de cada una de las planificaciones se utilizó *Microsoft Access* (10).

Planificación del entrenamiento para el deporte de atletismo: Para este software se tuvieron en cuenta los elementos teóricos de la planificación en este deporte, los cuales permitieron definir los macro ciclos, períodos y etapas que se encuentran dentro del plan de entrenamiento requerido por los especialistas.

Planificación del entrenamiento para el deporte pesas: Para el software de pesas se tuvieron en cuenta los elementos teóricos y los pasos para realizar la misma de forma manual. El software tiene la ventaja de haber sido confeccionado para el equipamiento más moderno disponible en el país y facilita las entradas de datos utilizando las posibilidades de los cuadros combinados para la selección de textos, en lugar de tener que teclearlos, y da la posibilidad a los usuarios de seleccionar las variantes de semanas y días más usados y de incluir otras variantes o modificar las ya existentes.

Planificación del entrenamiento en las pesas como deporte auxiliar: En este software se tuvo en cuenta que los entrenadores de todas las disciplinas utilizan las pesas como deporte auxiliar, pero ello requiere de su planificación, la cual, si se hace con todo el rigor necesario, trae aparejada la realización de una gran cantidad de cálculos engorrosos, que

le restan tiempo de su labor fundamental: la atención de las diferencias individuales de los atletas. El software permite obtener el plan de entrenamiento de todos los micro ciclos del meso ciclo planificado (utilizando las pesas) por plano muscular y zona de intensidad en cada día de entrenamiento.

Planificación del entrenamiento deportivo en esgrima: Para el software de esgrima se tuvo en cuenta el criterio de los entrenadores, los cuales solicitaron el modelo de salida requerido. Basándose en este y en las transformaciones necesarias de los datos se programó el sistema. El software facilita las entradas de datos utilizando las posibilidades de los cuadros combinados para la selección de textos en lugar de tener que teclearlos.

Todos estos sistemas son de gran ayuda para la realización del software deseado, ya que tienen en común supervisar el entrenamiento de un atleta acumulando la mayor información posible de este ya sea de los entrenamientos, como de su historial clínico o de competencias. Pero, ninguno de ellos posee una funcionalidad que les permita extraer los datos de los esfuerzos del atleta durante el entrenamiento en una bicicleta ergométrica por lo que utilizar algunos de estos software no resolverá el problema existente llegando a la necesidad de desarrollar un nuevo sistema que le permita al atleta que además de guardar todo su información personal también almacene de manera automática los datos que se extraen durante su sección de ejercicio, ya sea para atletismo, pesas, esgrima, o cualquier otro deporte. Con el fin de que la persona que trabaje directamente con la aplicación se ahorre el tedioso trabajo de realizar grandes cantidades de cálculos, y así agilizar el proceso de forma organizada y sistemática.

1.3 Tecnologías, Herramientas y metodologías de desarrollos

1.3.1 Marcos de Trabajo

El desarrollo de aplicaciones web está actualmente condicionado por la utilización de los marcos de trabajo; si se concibe un producto relativamente grande es recomendable la utilización de una estructura de soporte definido, mediante la cual otro proyecto de *software* puede ser organizado y desarrollado. “Es una estructura conceptual y tecnológica compuesta por bibliotecas, componentes y clases que facilitan el desarrollo ágil, seguro y escalable de una aplicación” (11). “Simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes.

Ofrecen una infraestructura que permite tener un código ordenado, limpio y fácil de actualizar, un código seguro, robusto y eficiente" (12).

La aplicación que se propone desarrollar debe garantizar compatibilidad con diferentes sistemas operativos, gestión de información en bases de datos, trabajo con ficheros compactados y que los usuarios accedan de manera remota a los servicios. Para poder garantizar lo anterior, es necesario implementar una aplicación *web* que utilice PHP5 como lenguaje de programación, lo que conlleva a una investigación sobre los *frameworks* PHP existentes.

Estudio sobre *frameworks* PHP

Atendiendo a las necesidades de la aplicación a desarrollar, se considera la información de los cinco mejores *frameworks* para PHP (13):

- **PHP5**: Indica si el *framework* viene con soporte incorporado para la versión 5 de PHP. Esta versión garantiza manejar los ficheros compactados e incorpora nuevas funciones para manipular arreglos de datos, obtenidos a partir de consultas.
- **MVC**: Indica si el *framework* viene con soporte incorporado para una configuración del patrón Modelo-Vista-Controlador. Este patrón de diseño se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar las tareas de desarrollo de aplicaciones y su posterior mantenimiento.
- **ORM**: Indica si el *framework* es compatible con un asignador de objeto de registro, por lo general una implementación de *ActiveRecord*. Esta funcionalidad permite generar un CRUD a una tabla de la base de datos (BD) de manera rápida.
- **DB Objects**: Indica si el *framework* incluye otros objetos de base de datos, como un *TableGateWay*. Esta función es necesaria para acceder a los datos almacenados en el BD.
- **Caching**: indica si el *framework* incluye una caché de objetos o de alguna manera otra forma de almacenamiento en caché.
- **Validation**: Indica si el *framework* tiene una validación o filtrado de componente incorporado.
- **Ajax:** Indica si el *framework* viene con soporte incorporado para Ajax que permita establecer comunicación asincrónica en el sistema.

- **Auth Module**: Indica si el *framework* tiene un módulo incorporado para el manejo de la autenticación de usuario y el control de acceso basado en roles.
- **Modules**: Indica si el *framework* es extensible o tiene otros módulos que permitan, entre otras cosas, generar reportes estadísticos.

	Yii	CodeIgniter	CakePhp	Zend	Symfony
PHP5	SI	SI	SI	SI	SI
MVC	SI	SI	SI	SI	SI
ORM	SI	NO	SI	SI	SI
DB Objects	SI	SI	SI	SI	SI
Caching	SI	SI	SI	SI	SI
Validation	SI	SI	SI	SI	SI
Ajax	SI	NO	SI	SI	SI
Auth module	SI	NO	SI	SI	SI
Modules	SI	NO	SI	SI	SI

Tabla 1: Comparativa de framework PHP

Atendiendo a los parámetros medidos en la comparación anterior resaltan los frameworks Yii y Symfony2 como los candidatos más fuertes, debido a que presentan todas las características evaluadas. Para escoger definitivamente, se analiza la facilidad de acceso a documentación actualizada y soporte de ambos frameworks desde la UCI, seleccionando definitivamente Symfony2. Este ofrece una amplia gama de videos, documentos y tutoriales en idiomas tanto como el español como en inglés. Además, cuenta con una gran comunidad de desarrolladores que crece cada día y cuenta con un fórum excelente ante dudas específicas.

Del framework Symfony se va a utilizar su versión 2.8.6. “Symfony2 ha sido ideado para exprimir al límite todas las nuevas características de PHP 5.3 y por eso es uno de los frameworks PHP con mejor rendimiento. Su arquitectura interna está completamente desacoplada, lo que permite reemplazar o eliminar fácilmente aquellas partes que no encajan en tu proyecto.

Symfony2 también es el framework que más ideas incorpora del resto de frameworks, incluso de aquellos que no están programados con PHP. Si has utilizado alguna vez Ruby On Rails, django o Spring encontrarás muchas similitudes en algunos de los componentes de Symfony2" (14)

1.3.2 Lenguajes utilizados

PHP 5

Acrónimo de *Hypertext Preprocessor*, PHP es un lenguaje de código abierto interpretado, de alto nivel, ejecutado en el servidor. Puede ser utilizado en cualquiera de los principales sistemas operativos del mercado actual, incluyendo Linux, UNIX, Microsoft Windows, Mac OS X, RISC OS. Es soportado por la mayoría de servidores *web*, incluyendo Apache, Microsoft Internet Information Server, Personal Web Server, Netscape e iPlanet. Soporta los principales sistemas de bases de datos como InterBase, FrontBase, mSQL, Direct MS-SQL, MySQL, Oracle (OCI7 and OCI8), Ovrimos, PostgreSQL (15).

Las cuatro características más importantes de PHP son simplicidad, seguridad, estabilidad y velocidad. En cuanto a la velocidad, se integra muy bien a otro *software* y requiere pocos recursos de sistema. Garantiza la estabilidad pues utiliza su propio sistema de administración de recursos y posee un sofisticado método para manejar variables, lo que lo hace un sistema robusto. Posee protección contra ataques, proveyendo diferentes niveles de seguridad. La simplicidad es la posibilidad brindada al desarrollador para generar código rápidamente, para esto, el lenguaje incluye gran cantidad de funciones predefinidas (16).

HTML 5

"HTML o *Hypertext Markup Language* (lenguaje de marcado de hipertexto), es un lenguaje diseñado para crear páginas *web* y otros documentos que sean posibles visualizar en el navegador *web*, siendo esencial para la creación de páginas y mostrar el contenido en ellas. El HTML se encarga de desarrollar una descripción sobre los contenidos que aparecen como textos y sobre su estructura, complementando dicho texto con diversos objetos (como imágenes, animaciones, videos)" (17)**.**

JavaScript

JavaScript es un lenguaje de programación *script* multiplataforma, diseñado para una fácil incrustación en otros productos y aplicaciones, tales como los navegadores *web*. JavaScript

puede ser conectado a los objetos de su entorno para proveer un control programable sobre estos (18). El empleo de JavaScript para complementar el diseño de interfaz de usuario de la plataforma ofrecerá efectos de animación que mejoran la visualización de los elementos que la componen.

Como medio de optimización para Javascript se puede utilizar JQuery, que realiza funciones de script frecuentes y utiliza menos líneas de código. Además, para la generación de gráficas se puede utilizar JQChart, que es una de las variantes de JQuery.

CSS3

Cascading Style Sheets u Hojas de Estilo en Cascada: es un lenguaje que permite a los desarrolladores describir la presentación de las páginas *web*. Son incluidos los colores, el diseño, y las fuentes, permitiendo adaptar la presentación a los diferentes tipos de dispositivos, tales como diferentes resoluciones de pantallas o impresoras (19). La separación del código HTML de CSS hace que sea fácil mantener sitios, compartir hojas de estilos a través de páginas y adaptarlas a entornos distintos. Esto se conoce como la separación de la estructura (o contenido) de la presentación (20). Para optimizar CSS3, se puede utilizar Bootstrap, que permite que la plataforma se adapte a las distintas resoluciones de pantalla, aumente el atractivo visual, incorpore las nuevas tendencias del diseño *web* y garantice una fácil actualización del diseño visual.

1.3.3 Análisis de gestores de base de datos

Un Sistema Gestor de Bases de Datos (SGBD) es una colección de programas cuyo objetivo es servir de interfaz entre el BD, el usuario y las aplicaciones. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta. Un SGBD permite definir los datos a distintos niveles de abstracción y manipularlos, garantizando su seguridad e integridad (21).

“Las aplicaciones Symfony2 no gestionan su información accediendo directamente a la base de datos. Crean, modifican y borran información mediante objetos PHP, en vez de crear y ejecutar sentencias SQL. Esto es posible gracias a unas bibliotecas externas llamadas ORM u *Object-Relational Mapping”* (14). Entre los Sistemas de Gestión de Base de Datos (SGBD) soportados por Symfony2 y de mayor relevancia en la actualidad están

Oracle, PostgreSQL y MySQL. A continuación, se analizan las características de estos SGBD:

- **Oracle:** Es un sistema de gestión de base de datos relacional. Se considera como uno de los sistemas de bases de datos más completos, destacando su soporte de transacciones, estabilidad; lo que significa que su nivel de fallo disminuye en dependencia de la estabilidad que se requiera y escalabilidad; es decir, su posibilidad de estar preparado para hacerse más grande sin perder calidad en los servicios ofrecidos, siendo capaz de cambiar su tamaño o configuración para adaptarse a las circunstancias cambiantes (22). Además, es multiplataforma, pero el precio de su licencia es de varios miles de euros, unido a la dificultad que presenta para poder instalarse. Aunque su dominio en el mercado de servidores empresariales ha sido casi total.
- **PostgreSQL:** Es un sistema de gestión de bases de datos objeto-relacional. Está liberado bajo licencia BSD, además es extensible, multiplataforma y presenta modelos de negocios rentables con instalaciones a gran escala (23). Tiene ahorros considerables en costos de operación. El *software* ha sido diseñado y creado para tener un mantenimiento y ajuste mucho menor que los productos de los proveedores comerciales, conservando todas las características de rendimiento.
- **MySQL:** Es un SGBD relacional multiplataforma (Linux, Windows, Mac OS). Posee un sistema de privilegios y contraseñas que es muy flexible y seguro, apoyado por los algoritmos de encriptación que se ejecutan cuando se conecta al servidor. Soporta grandes BD. Utiliza un índice que puede usar prefijos de una columna para los tipos de columna CHAR, VARCHAR, BLOB o TEXT. MySQL tiene soporte para comandos SQL para chequear, optimizar y reparar tablas. Es muy rápido, seguro y fácil de usar, esto significa que es un servidor bastante apropiado para acceder a bases de datos en Internet (24). Presenta características como son: conectividad segura, transacciones y claves foráneas, replicación, búsqueda de indexación de campos de texto y disponibilidad en gran cantidad de plataformas y sistemas (25).

Luego de realizar un estudio de estos SGBD, se llega a la conclusión de que MySQL se ajusta para ser utilizado en la gestión de datos de la solución propuesta. Esta herramienta posee licencia libre y se integra con el lenguaje de programación seleccionado. Además, la integración con el *framework* Symfony2 es excelente y sencilla. Es válido señalar que es el SGBD que viene integrado por defecto en el paquete de programas XAMPP, que se utiliza

para poner en funcionamiento la aplicación, por lo que no se necesita realizar instalaciones extras para poner en pleno funcionamiento el sistema y posee una interfaz web para su administración.

1.3.3 Análisis de Entornos de Desarrollo Integrado

Un Entorno de Desarrollo Integrado (IDE, por sus siglas del inglés *Integrated Development Environment*) es un entorno de programación que ha sido empaquetado como un programa de aplicación. Consiste en un editor de código, un compilador, un depurador y en ocasiones un constructor de interfaz gráfica de usuario. Los IDE pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes (26). Una decisión importante a la hora de desarrollar con PHP es la selección del IDE, ya que el entorno de desarrollo que se elija puede suponer una verdadera diferencia en el tiempo de trabajo invertido. Debido a lo anterior se realiza un estudio sobre algunos de los IDE para PHP:

- **Zend Studio:** es el soporte en desarrollos y pruebas de PHP con el set más completo de herramientas para la creación de aplicaciones altamente fiables como lo requiera una empresa. Asegura el desarrollo de *software* mediante la combinación del IDE con un entorno de prueba que agiliza la seguridad de la calidad, integración y las etapas de los procesos. Brinda todo lo que necesita para construir, probar y entregar aplicaciones de alto rendimiento. Contiene distintas funcionalidades como generación de código, formateo de código configurable, pruebas unitarias y desarrollo remoto. Cuesta 299€ y es multiplataforma (Linux, Windows y Mac OS X) (27).
- **Netbeans:** es multiplataforma (Windows, Linux, Mac OS X y Solaris), gratuito, de código abierto (con licencia *CDDL*) y se puede utilizar para programas en otros lenguajes además de PHP. A parte de las funciones básicas con las que debería contar cualquier IDE, como resaltado de sintaxis, autocompletado, formateo de código o depurador (xDebug), también cuenta con otras funcionalidades menos comunes como la integración con PHPUnit para las pruebas unitarias y con CVS, Subversion y Mercurial para el control de versiones (28).
- **PhpStorm:** JetBrains PhpStorm es un IDE multiplataforma comercial para PHP desarrollado sobre la plataforma JetBrains IntelliJ IDEA. Un IDE que tiene un número de funcionalidades similar al Netbeans, la licencia de este entorno de desarrollo fue

adquirida por el centro de desarrollo de php de la uci a pesar de ser un software privado lo que conlleva a que tanto estudiantes como profesores puedan utilizar libremente este software. Proporciona un editor para PHP, HTML y JavaScript con el análisis de código sobre la marcha, la prevención de errores y refactorizaciones automáticas para PHP y JavaScript. Ofrece soporte para PHP 5, incluyendo generadores, lista en foreach, espacios de nombres, los cierres, los rasgos y la sintaxis de matrices corto. Incluye un editor de pleno derecho de SQL con resultados de la consulta editables (29).

A partir del estudio realizado sobre IDE, se llega a la conclusión que tanto como Netbeans como PhpStorm se ajustan para la realización de la aplicación, uno por tener la licencia gratis y otro por tener la universidad la licencia adquirida por lo tanto se elige el IDE PhpStorm para la creación de la aplicación.

1.3.4 Lenguaje Unificado de Modelado

El Lenguaje Unificado de Modelado (en lo adelante UML, por sus siglas en inglés, *Unified Modeling Language*) se utiliza para especificar, visualizar, construir y documentar artefactos de un sistema de *software*. Se emplea para diseñar, hojear, configurar, mantener y controlar la información sobre sistemas. Está pensado para emplearse con todos los métodos de desarrollo, etapas del ciclo de vida y dominios de aplicación. UML incluye conceptos semánticos, notación y principios generales. Tiene partes estáticas, dinámicas, de entorno y organizativas. Está pensado para ser utilizado en herramientas interactivas de modelado visual que tengan generadores de código así como generadores de informes (30) (JACOBSON *et al.*, 1999).

Es importante resaltar que UML es un “lenguaje” para especificar y no para describir métodos o procesos. Se utiliza para definir un sistema de *software*, para detallar los artefactos en el sistema y para documentar y construir. En otras palabras, es el lenguaje en el que está descrito el modelo. Se puede aplicar en una gran variedad de formas para dar soporte a una metodología de desarrollo de *software*, pero no especifica en sí mismo qué metodología o proceso utilizar.

Herramienta de ingeniería de software asistida por computadora

Las herramientas de ingeniería de *software* asistida por computadoras (en lo adelante CASE, por sus siglas del inglés, *Computer Aided Software Engineering*) brindan ayuda a los analistas, ingenieros de software y desarrolladores. Por ejemplo, permiten el modelado de los sistemas mediante diferentes diagramas, generación de código a partir de estos y viceversa. Las herramientas CASE de modelado con UML permiten aplicar la metodología de análisis y diseño orientados a objetos. Además permiten abstraerse del código fuente, en un nivel donde la arquitectura y el diseño se tornan más fáciles de entender (31).

Rational Rose Enterprise

IBM Rational Rose Enterprise proporciona un conjunto de prestaciones controladas por modelo para desarrollar muchas aplicaciones de software, incluidas aplicaciones Ada, ANSI C++, C++, CORBA, Java, Java EE, Visual C++ y Visual Basic.

- Modelado de las aplicaciones más habituales: proporciona prestaciones de modelado visual para desarrollar muchos tipos de aplicaciones de software.
- Desarrollo de aplicaciones para la web: contiene herramientas web y XML para el modelado de aplicaciones web.
- Integración del diseño de aplicaciones con el desarrollo: unifica el equipo del proyecto proporcionando una ejecución y una notación de modelos UML comunes (32).

Visual Paradigm

Visual Paradigm para UML es una herramienta CASE profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite dibujar todos los tipos de diagramas de clases, código inverso y generar documentación. También proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML. Brinda la posibilidad de generar código a partir de los diagramas para lenguajes como Java y PHP, así como obtener diagramas a partir de código (33). Esta es precisamente una gran ventaja puesto que el sistema será desarrollado en PHP.

La herramienta seleccionada para el modelado de la aplicación es Visual Paradigm, debido fundamentalmente a que es una herramienta multiplataforma que ayuda a una rápida construcción de aplicaciones de calidad. Además, la UCI posee licencia para su utilización.

1.3.5 Metodología de desarrollo de software

Una metodología de desarrollo de software es un conjunto de pasos y procedimientos que deben seguirse para desarrollar software. Define cómo dividir un proyecto en etapas, qué tareas se llevan a cabo en cada etapa, qué restricciones deben aplicarse, qué técnicas y herramientas se emplean y cómo se controla y gestiona un proyecto. Constituye un medio de estandarización que cubre por completo el proceso de desarrollo de software. Posibilita verificar trabajos realizados, corrección de los errores detectados y además provee un lenguaje común entre los analistas, programadores, clientes y usuarios (34). Además, especifica completamente el desarrollo de un sistema informático, pero es necesario tener en cuenta que no es lo mismo utilizar un modelo para la creación de un sitio *web* que para el desarrollo de una aplicación de escritorio, por lo que existe una gran diversidad de metodologías de desarrollo. Debido a lo anterior, es necesario hacer una buena elección, en dependencia de las características del sistema a desarrollar.

Metodología Programación Extrema

La Programación Extrema, o *Extreme Programming* (XP según siglas en inglés), surge como posible solución a los problemas derivados del cambio en los requerimientos; esta metodología ofrece la posibilidad de cambiar los requisitos en cualquier momento de la vida de un proyecto, ya que es adaptable a estos cambios. Se centra en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. El ciclo de vida ideal de XP consta de seis fases: la exploración, planificación de la entrega (*release*), iteraciones, producción, mantenimiento y muerte del proyecto (35).

XP es una de las metodologías ágiles más utilizadas en el mundo por sus características, pues se basa en una programación rápida y extrema. Tiene la particularidad de tener como parte del equipo al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto. Las características fundamentales son:

- Desarrollo iterativo e incremental: pequeñas mejoras, unas tras otras.
- Programación por parejas: se recomienda que las tareas de ingeniería se lleven a cabo por dos personas en un mismo puesto.
- Frecuente interacción del equipo de programación con el cliente o usuario: Se recomienda que un representante del cliente trabaje junto al equipo de desarrollo.
- Corrección de todos los errores antes de añadir nueva funcionalidad: Hacer entregas frecuentes.

Metodología AUP

El Proceso Unificado Ágil de Scott Ambler o *Agile Unified Process* (AUP, en su término en inglés) es una versión simplificada del Proceso Unificado de Racional (RUP). Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP (36).

Al igual que otras metodologías de desarrollo esta consta con fases de desarrollo que originalmente eran inicio, construcción, elaboración, transición, la universidad hizo adaptaciones para un mejor acoplamiento en la producción quedando de la siguiente forma.

Fases de AUP para la UCI

- **Inicio:** Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.

- **Ejecución:** En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elabora la arquitectura, el diseño, se implementa y se libera el producto. Durante esta fase el producto es transferido al ambiente de los usuarios finales o entregado al cliente. Además, en la transición se capacita a los usuarios finales sobre la utilización del software.

- **Cierre:** En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

A partir de haber desarrollado un resumen sobre estas metodologías de desarrollo se llegó a la conclusión que la metodología que se debe utilizar para la creación de esta aplicación sea la metodología AUP-UCI ya que es una metodología robusta que cumple con los requisitos necesarios para un mejor proceso de creación de la aplicación ya que es fácil y simple de trabajar, y principalmente, para cumplir el objetivo de la universidad que es que todos los proyectos que se realicen en esta tengan una misma organización y estén definidos por los mismos documentos y metodología.

1.3.5 Protocolo de comunicación

Debido a la organización del sistema se hace necesario adoptar un protocolo de comunicación para el intercambio de información entre el servidor web y la tarjeta de adquisición de datos que permita hacer un uso eficiente del canal de datos, entre ellos se destacan MQTT y Modbus.

MQTT

El protocolo MQ Telemetry Transport v3 se ha diseñado para intercambiar mensajes entre pequeños dispositivos con reducido ancho de banda y para enviar mensajes de forma segura sobre TCP/IP. Utiliza la opción de publicar/suscribir y da soporte a tres calidades de servicio: "transmitir y olvidar", "al menos una vez" y "exactamente una vez", ya que con tres calidades de servicio, puede buscar un punto intermedio entre la latencia baja y la fiabilidad.

Los mensajes son reducidos gracias al pequeño tamaño de las cabeceras del protocolo y a la carga del mensaje en matriz de bytes. Las cabeceras constan de una cabecera fija de 2 bytes y de hasta 12 bytes de otras cabeceras variables adicionales. El protocolo utiliza 12 bytes de cabeceras variables para suscribirse y conectarse y sólo 2 bytes para cabeceras variables para la mayoría de las publicaciones (37).

Modbus

Modbus es un protocolo de comunicaciones, se basa dos tipos de arquitectura maestro/esclavo (RTU) o cliente/servidor (TCP/IP), fue diseñado en 1979 por Modicon para su gama de controladores lógicos programables (PLCs). Está ubicado en el nivel 7 del

Modelo OSI. Es el que goza de mayor disponibilidad para la conexión de dispositivos electrónicos industriales (38).

Modbus permite el control de una red de dispositivos, por ejemplo un sistema de medida de temperatura y humedad, y comunicar los resultados a una PC. Modbus también se usa para la conexión de un ordenador de supervisión con una unidad remota (RTU) en sistemas de supervisión adquisición de datos (SCADA).

Existen versiones del protocolo Modbus para puerto serie y Ethernet (Modbus/TCP), con diferentes representaciones numéricas de los datos y detalles del protocolo ligeramente desiguales. Modbus RTU es una representación binaria compacta de los datos. Modbus ASCII es una representación legible del protocolo pero menos eficiente. Ambas implementaciones del protocolo son serie. El formato RTU finaliza la trama con una suma de control de redundancia cíclica (CRC), mientras que el formato ASCII utiliza una suma de control de redundancia longitudinal (LRC). La versión Modbus/TCP es muy semejante al formato RTU, pero estableciendo la transmisión mediante paquetes TCP/IP (puerto del sistema 502, identificador *asa-appl-proto*) (39).

De ellos se selecciona Modbus en su variante TCP ya que es protocolo estándar para este tipo de sistemas, asegura la integridad de la información y permite, además, la integración con el lenguaje PHP

Conclusiones del Capítulo

Después de haber realizado un estudio del estado del arte de las aplicaciones que están vinculadas al desarrollo de los entrenamientos para atletas, los principales *frameworks* para PHP, los SGBD, los IDE, los microcontroladores AVR, la plataforma basada en un microcontrolador (Arduino) y las metodologías de desarrollo de *software*, se concluye que:

- Las aplicaciones destinadas a la planificación y desarrollo de los entrenamientos en un atleta permiten implementar servicios de gran ayuda al entrenador para facilitarle y mejorarle su trabajo y para que el atleta alcance su mejor rendimiento.
- En Cuba no se dispone de ninguna aplicación que realice el proceso de supervisión de un entrenamiento en las bicicletas ergométricas.

- Se seleccionaron las herramientas, tecnologías y la metodología para el empleo de ellas durante el desarrollo de la aplicación.

De esta manera se está en condiciones de desarrollar la aplicación *web* sincronizada las bicicletas ergométricas a través de microcontroladores para visualizar la evolución de la capacidad física de resistencia de los atletas, utilizando las herramientas y tecnologías seleccionadas.

Capítulo 2: Fases Planificación y diseño

Introducción

En el presente capítulo se adquiere una visión práctica del sistema desarrollado. En el mismo se expondrán las reglas de negocio, así como los requisitos funcionales y no funcionales para el desarrollo de la solución del problema, definiendo que esperan los usuarios de la misma.

2.1 ¿Qué es el modelo de dominio?

El Modelo de Dominio es una representación visual de los conceptos u objetos del mundo real significativos para un problema o área de interés. Representa clases conceptuales del dominio del problema, conceptos del mundo real en lugar de componentes de software.

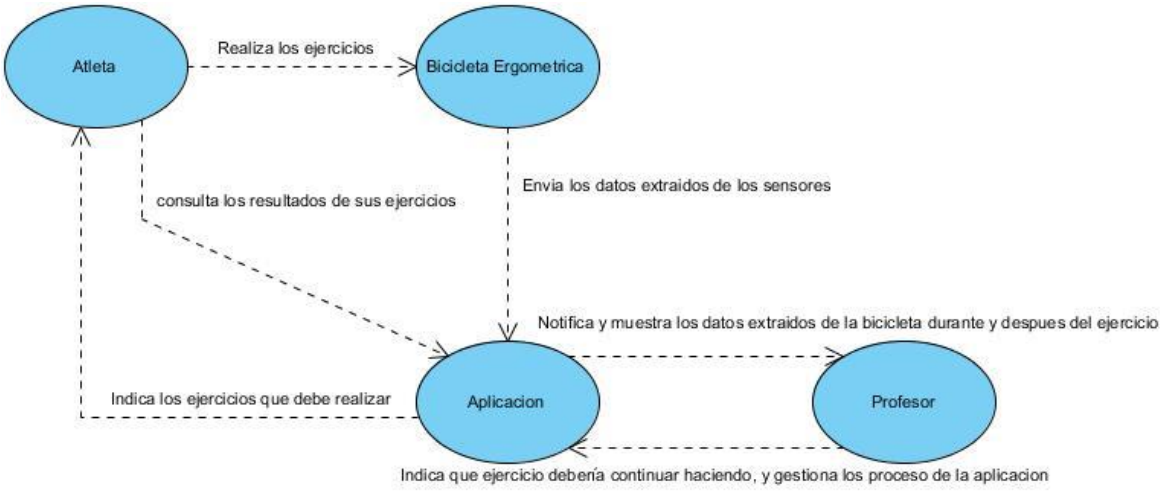


Ilustración 2: Modelo de dominio de sistema

2.2 Requisitos funcionales

RF1: Autenticar usuario.

RF2: Registrar usuario.

RF3: Modificar usuario.

RF4: Visualizar datos generales.

RF5: Ver orientaciones de entrenamiento a realizar.

RF6: Guardar los datos extraídos de la bicicleta durante el entrenamiento.

RF7: Ver datos durante el entrenamiento.

RF8: Orientar próximo entrenamiento.

2.3 Requisitos no funcionales

Requerimientos de Restricciones en el diseño e implementación

- El sistema debe ser desarrollado en el lenguaje de programación PHP para la programación de la aplicación web al servidor
- La implementación debe ser desarrollada utilizando el marco de trabajo Symfony para implementación de páginas Web

Requerimientos de ayuda y documentación

- Cada una de las etapas del ciclo de vida del proyecto deberá contar con una documentación según la metodología establecida.
- Requerimientos de licencias y patentes.
    - ✓ Se debe utilizar herramientas libres

Requerimientos de Portabilidad

- El sistema debe funcionar en sistemas de la familia GNU/Linux.

Requerimientos de seguridad

- Cuando se intente realizar cualquier acción irreversible, existirá una opción de advertencia antes realizar dicha acción.

Requerimientos de software

- Sistema operativo GNU/Linux, distribución Debian y derivados.

2.4 Historia de Usuario para los requisitos funcionales de la aplicación.

El modelado de negocio de la metodología AUP-UCI propone tres variantes a utilizar en los proyectos:

- Casos de usos de negocio (CUN)
- Descripción del proceso de negocio (DPN)
- Modelo conceptual (MC)

Y existen tres formas de encapsular los requisitos:

- Casos de usos del sistemas (CUS)
- Historias de usuario (HU)
- Descripción de requisitos por procesos (DRP)

A partir de ello surgen escenarios para modelar el sistema en los proyectos que utilizan esta metodología, manteniendo en dos de ellos el MC, quedando de la siguiente forma:

- **Escenario No1:** CUN + MC = CUS, para Proyectos que modelen el negocio con CUN solo pueden modelar el sistema con CUS.
- **Escenario No2:** MC = CUS, Proyectos que modelen el negocio con MC solo pueden modelar el sistema con CUS.
- **Escenario No3:** DPN + MC = DRP, Proyectos que modelen el negocio con DPN solo pueden modelar el sistema con DRP.
- **Escenario No4:** HU, Proyectos que no modelen negocio solo pueden modelar el sistema con HU.

Después de haber evaluado el negocio a informatizar dio como resultado un negocio muy bien definido y que el cliente va a estar siempre presente en cada paso que se de en la creación del proyecto, por lo tanto se elige para el modelado de negocio el escenario número 4 por lo que se utilizaran las historias de usuario para el modelado (36).

Las historias de usuario sustituyen a los documentos de especificación funcional y a los casos de usos. Estas historias son escritas por el cliente en su propio lenguaje en descripciones breves de lo que el sistema debe realizar.

Para la confección de las historias de usuarios se realizaron diferentes entrevistas a los entrenadores que son los clientes. Si bien el cliente no fue quien escribió personalmente las historias de usuario, fue el que diseño su contenido, asigno la prioridad de cada una de ellas y dirigió la redacción de las mismas.

Las historias de usuario se representan mediante tablas las cuales contienes las siguientes secciones:

- **Número:** En este campo se define el número del requisito que esta embebido en la tabla.

- **Nombre:** Nombre que identifica las historias de usuarios.
- **Usuario:** Roles de los usuarios que pueden acceder a esta funcionalidad.
- **Prioridad en Negocio:** Esta característica es dada por el cliente con los valores: alta, media o baja en dependencia de la importancia en que desean ser implementadas.
- **Iteración asignada:** Número de la iteración en la cual se desarrollara la HU.
- **Puntos estimados:** Tiempo estimado en semanas que se le asignara.
- **Descripción:** Breve descripción del proceso que define la HU.
- **Observaciones:** Alguna acotación importante de señalar acerca de la HU
- **Riesgo en desarrollo:** Esta característica es dada por el desarrollador con los valores: alta, media, baja en dependencia de la importancia que debe tener implementar sin errores este requisito para que no afecte a la funcionalidad del sistema.

Historia de Usuario	
Número: 1	**Nombre:** Autenticar usuario
Usuario: Usuarios registrados	**Iteración asignada:** 1
Prioridad en Negocio: Alta	**Puntos Estimados:** 1
Riesgo en Desarrollo: Alto	**Puntos Reales:** 1
Descripción: Un usuario desea autenticarse para poder acceder a sus funcionalidades en dependencia de que si es un profesor o un atleta, para ello debe acceder a la función del sistema de autenticarse. Datos para poder autenticarse: • Correo Electrónico (Obligatorio).	

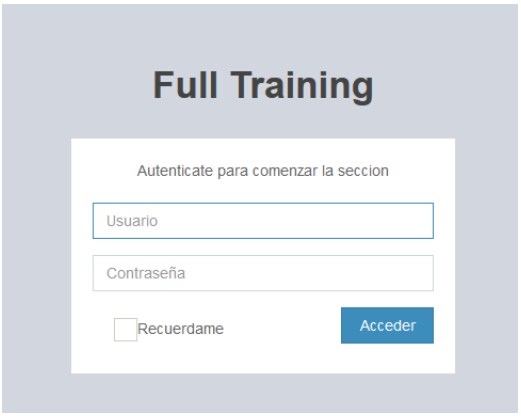
• Contraseña (Obligatorio).
Observaciones: Para que un usuario pueda autenticarse debe estar registrado en el sistema.
Prototipo de interfaz de usuario:  *Ilustración 3: Ejemplo de autenticar usuario*

Tabla 2: Historia de usuario – Autenticar usuario

Historia de Usuario	
Número: 2	**Nombre:** Registrar usuario
Usuario: Usuarios no registrados en el sistema	**Iteración asignada:** 1
Prioridad en Negocio: Alta	**Puntos Estimados:** 1
Riesgo en Desarrollo: Alto	**Puntos Reales:** 1

Descripción: Un usuario desea registrarse para poder acceder a las funcionalidades de la aplicación, para ello cuenta con la opción de registrar usuario. El sistema debe permitir al usuario registrarse de manera satisfactoria, para esto debe llenar los siguientes campos:

- Nombre(s) (Obligatorio)
- 1er Apellido (Obligatorio)
- 2do Apellido (Obligatorio)
- Correo electrónico (Obligatorio)
- Clave o contraseña (Obligatorio)
- Foto de perfil (Opcional)
- Atleta o Profesor (Obligatorio)

Observaciones: Posterior de la creación de un usuario, en dependencia de la opción que eligió en el campo de si es un ROLE_ADMIN (profesor) o si es un ROLE_USER (atleta) se le concederá los permisos pertenecientes a dicho rol.

Prototipo de interfaz de usuario:

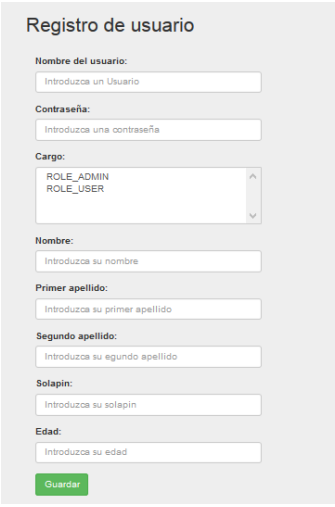


Ilustración 4: Ejemplo de registrar usuario

Tabla 3: Historia de usuario – Registrar usuario

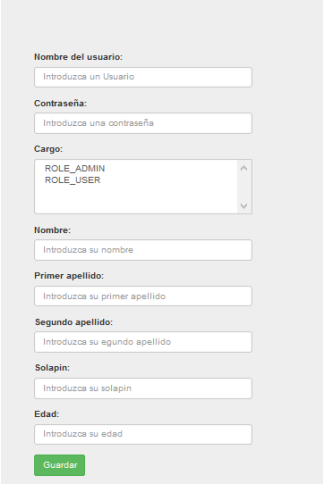
Historia de Usuario	
Número: 3	**Nombre:** Modificar Usuario
Usuario: Usuario registrado	**Iteración asignada:** 1
Prioridad en Negocio: Alta	**Puntos Estimados:** 1
Riesgo en Desarrollo: Alto	**Puntos Reales:** 1
Descripción: Un usuario desea modificar sus datos ya sea porque los ha introducido mal o porque desea actualizarlos, para esto el sistema debe permitir que el usuario pueda modificar sus datos de insertados en el sistema mediante la función “modificar usuario”.	
Observaciones: El usuario debe estar previamente registrado y autenticado en el sistema para poder acceder a esta función en el sistema	
Prototipo de interfaz de usuario: 	

Ilustración 5: Ejemplo de modificar usuario

Tabla 4: Historia de usuario – Modificar usuario

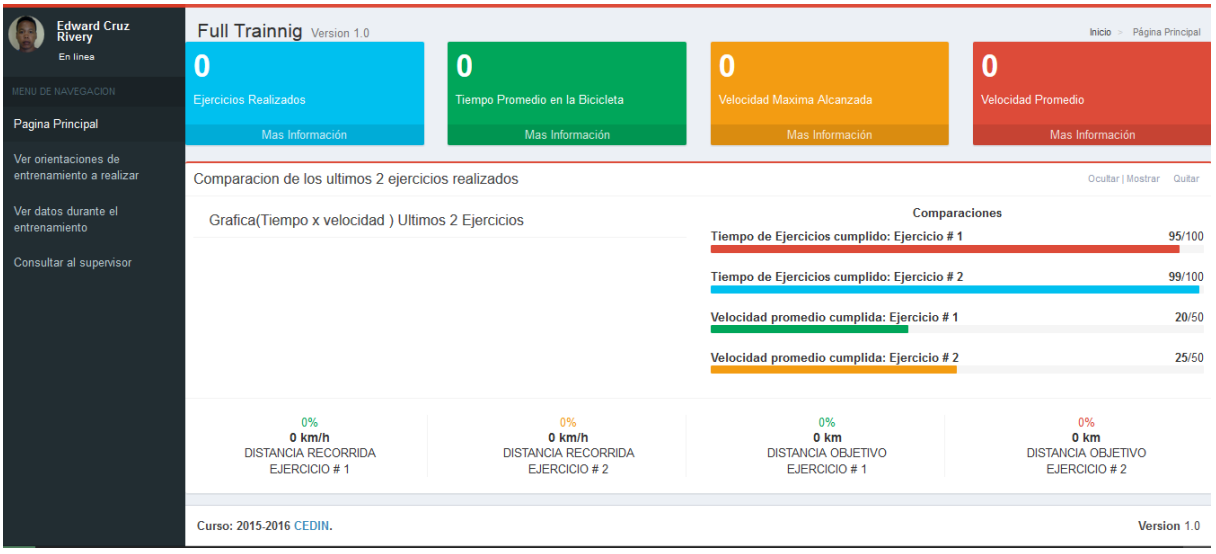
Historia de Usuario	
Número: 4	**Nombre:** Visualizar datos generales.
Usuario: Atleta	**Iteración asignada:** 1
Prioridad en Negocio: Alta	**Puntos Estimados:** 1
Riesgo en Desarrollo: Alto	**Puntos Reales:** 1
Descripción: Luego de que el usuario se haya autenticado, si el usuario es un atleta se le direccionará a la página principal del atleta.	
Observaciones: El usuario debe estar previamente registrados y tener los permisos para el rol de atleta.	
Prototipo de interfaz de usuario: 	

Ilustración 6: ejemplo de la página principal del atleta

Tabla 5: Historia de usuario – Ver datos generales

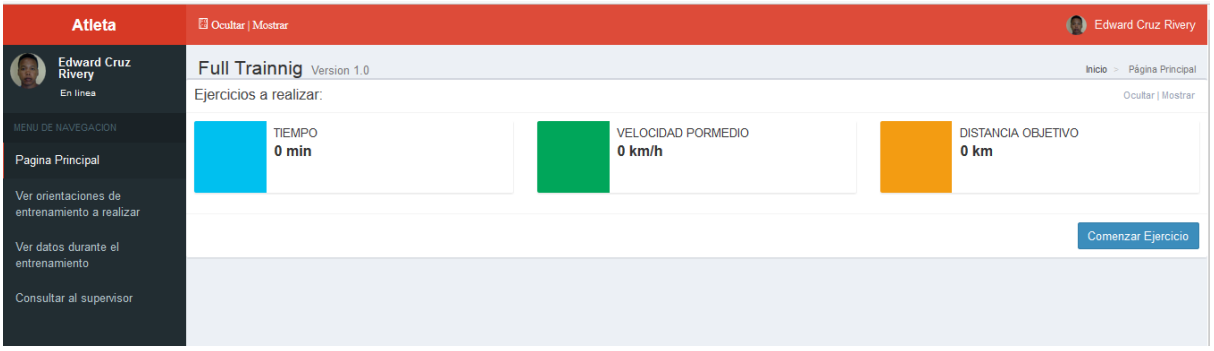
Historia de Usuario	
Número: 5	**Nombre:** Ver orientaciones de entrenamiento a realizar.
Usuario: Atleta	**Iteración asignada:** 1
Prioridad en Negocio: Alta	**Puntos Estimados:** 1
Riesgo en Desarrollo: Alto	**Puntos Reales:** 1
Descripción: El sistema debe permitir al atleta ver el entrenamiento que debe realizar durante la sección de ejercicios en la bicicleta estática como el tiempo de ejecución y velocidad promedio.	
Observaciones: El profesor previamente debe haberle introducido los ejercicios que debe realizar o sino no vera ningún dato para realizar los ejercicios el atleta.	
Prototipo de interfaz de usuario:  *Ilustración 7: Ejemplo de la página "Ejercicios a realizar"*	

Tabla 6: Historia de usuario - Ver orientaciones de entrenamiento a realizar

Historia de Usuario	
Número: 6	**Nombre:** Guardar los datos extraídos de la bicicleta durante el entrenamiento.
Usuario: Atleta	**Iteración asignada:** 1
Prioridad en Negocio: Alta	**Puntos Estimados:** 1
Riesgo en Desarrollo: Alto	**Puntos Reales:** 1
Descripción: El sistema debe almacenar los datos extraídos de la bicicleta de manera automática en una base de datos donde se van a almacenar: el identificador del usuario que está realizando el ejercicio, la fecha en el momento en que se le pidió los datos extraídos de la bicicleta y la cantidad de ciclos de pedaleo que dio el atleta durante los 10 segundos.	
Observaciones: Los datos se deben de guardar de manera automática a la vez que se le muestran al atleta.	
Prototipo de interfaz de usuario: No existe una interfaz para este requisito ya que su función simplemente es almacenar los datos de manera automática.	

Tabla 7: Historia de Usuario - Guardar los datos extraídos de la bicicleta durante el entrenamiento

Historia de Usuario	
Número: 7	**Nombre:** Ver datos durante el entrenamiento.
Usuario: Atleta	**Iteración asignada:** 1

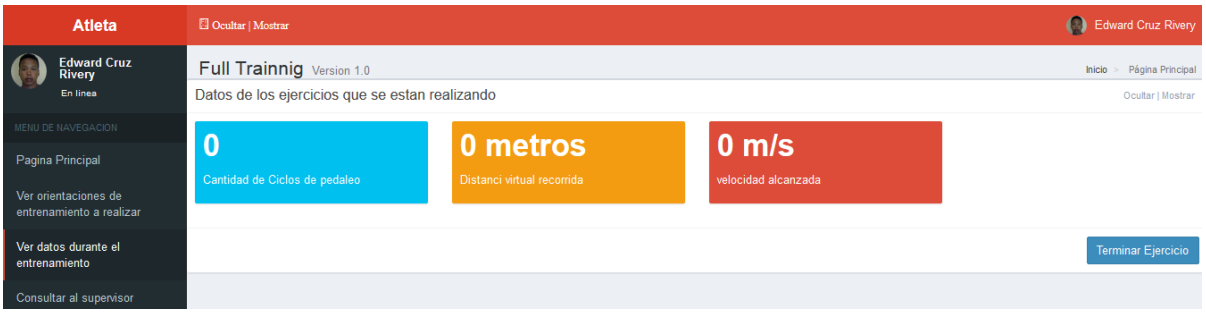
Prioridad en Negocio: Alta	**Puntos Estimados:** 1
Riesgo en Desarrollo: Alto	**Puntos Reales:** 1
Descripción: El sistema debe permitir al atleta ver su estado físico durante la sección de entrenamiento mientras que está realizando los ejercicios en la bicicleta actualizándose la página cada 10 segundos	
Observaciones:	
Prototipo de interfaz de usuario: 	

Ilustración 8: Ejemplo de la página "Ver datos durante entrenamiento"

Tabla 8: Historia de usuario - Ver datos durante el entrenamiento

Historia de Usuario	
Número: 8	**Nombre:** Orientar próximo entrenamiento.
Usuario: Profesor	**Iteración asignada:** 1
Prioridad en Negocio: Alta	**Puntos Estimados:** 1

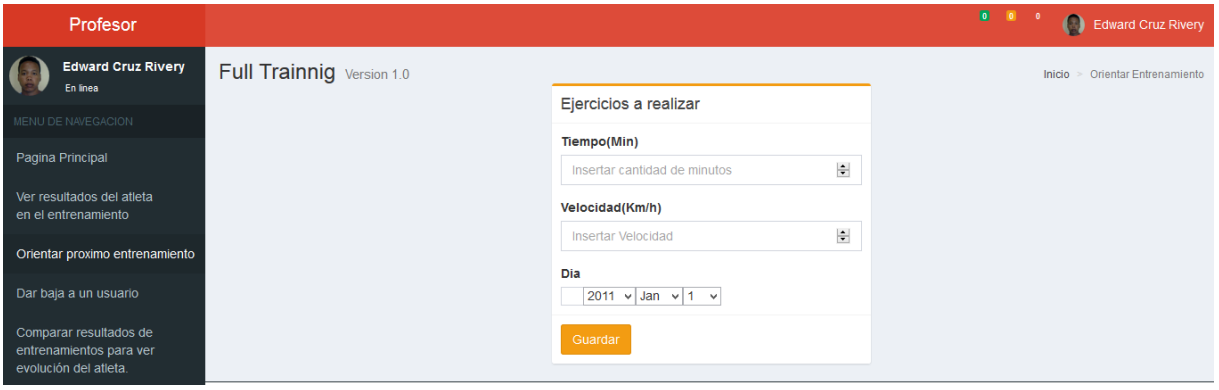
Riesgo en Desarrollo: Alto	**Puntos Reales:** 1
Descripción: El sistema debe permitir al supervisor orientar el próximo entrenamiento al atleta según lo visto por el supervisor en el entrenamiento pasado de este.	
Observaciones:	
Prototipo de interfaz de usuario:  *Ilustración 9: ejemplo de la página "Orientar próximo entrenamiento"*	

Tabla 9: Historia de usuario – Orientar próximo entrenamiento

2.5 Patrones de arquitectura de software

De acuerdo al Software Engineering Institute (SEI), la Arquitectura de Software se refiere a las estructuras de un sistema, compuestas de elementos con propiedades visibles de forma externa y las relaciones que existen entre ellos.

El término “elementos” dentro de la definición del SEI es vago a propósito, pues puede referirse a distintas entidades relacionadas con el sistema. Los elementos pueden ser entidades que existen en tiempo de ejecución (objetos, hilos), entidades lógicas que existen en tiempo de desarrollo (clases, componentes) y entidades físicas (nodos, directorios). Por otro lado, las relaciones entre elementos dependen de propiedades visibles (o públicas) de los elementos, quedando ocultos los detalles de implementación. Finalmente, cada conjunto de elementos relacionados de un tipo particular corresponde a una estructura distinta, de ahí que la arquitectura está compuesta por distintas estructuras.

Dentro de los objetivos de los patrones se encuentran:

- Proporcionar catálogos de elementos reusables en el diseño de sistemas software.
- Evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente.
- Formalizar un vocabulario común entre diseñadores.
- Estandarizar el modo en que se realiza el diseño.
- Facilitar el aprendizaje de las nuevas generaciones de diseñadores condensando conocimiento ya existente.

2.5.1 Patrón Utilizado

2.5.1.1 Modelo-Vista-Controlador

El patrón Modelo-Vista-Controlador (MVC) surge con el objetivo de reducir el esfuerzo de programación, necesario en la implementación de sistemas múltiples y sincronizados de los mismos datos, a partir de estandarizar el diseño de las aplicaciones. El patrón MVC es un paradigma que divide las partes que conforman una aplicación en el Modelo, las Vistas y los Controladores, permitiendo la implementación por separado de cada elemento, garantizando así la actualización y mantenimiento del *software* de forma sencilla y en un reducido espacio de tiempo. A partir del uso de *frameworks* basados en el patrón MVC se puede lograr una mejor organización del trabajo y mayor especialización de los desarrolladores y diseñadores.

2.5.1.2 Solución Arquitectónica del sistema

Symfony2 basa su funcionamiento interno en la famosa arquitectura Modelo-Vista–Controlador, utilizada por la mayoría de *frameworks* web. No obstante, según su creador Fabien Potencier: *"*Symfony2 no es un framework MVC. Symfony2 sólo proporciona herramientas para la parte del Controlador y de la Vista. La parte del Modelo es responsabilidad tuya, aunque existen librerías para integrar fácilmente los ORM más conocidos, como Doctrine y Propel*"* (40).

En cualquier caso, resulta esencial conocer cómo se aplican los principios fundamentales de la arquitectura MVC a las aplicaciones Symfony2. A continuación se muestra un pequeño esquema del funcionamiento interno de Symfony2 (14).

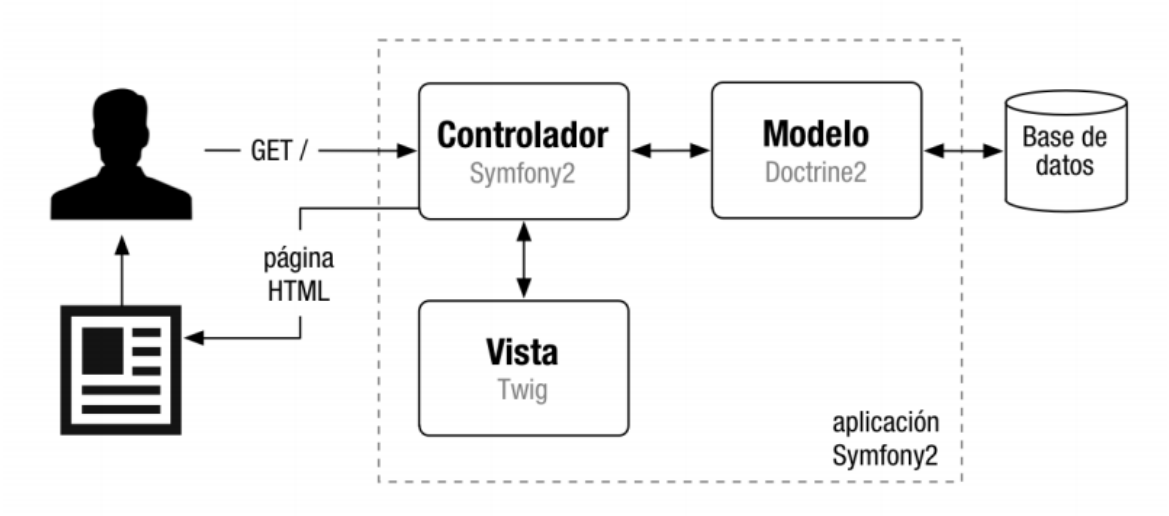


Ilustración 10: Representación de como se ve reflejado el MVC en el framework Symfony2

Cuando el usuario solicita ver la portada del sitio, internamente sucede lo siguiente:

1. El sistema de enrutamiento determina que controlador está asociado con la página de la portada.
2. Symfony2 ejecuta el controlador asociado a la portada. Un controlador no es más que una clase PHP en la que puedes ejecutar cualquier código que quieras.
3. El controlador solicita al modelo los datos de la tabla creada previamente. El modelo no es más que una clase PHP especializada en obtener información, normalmente de una base de datos.
4. Con los datos devueltos por el modelo, el controlador solicita a la vista que cree una página mediante una plantilla y que inserte los datos del modelo.
5. El controlador entrega al servidor la página creada por la Vista.

A pesar de que puedes llegar a hacer cosas muy complejas con Symfony2, el funcionamiento interno siempre es el mismo:

- El controlador manda y ordena
- El modelo busca la información que se le pide

- La vista crea páginas con plantillas y datos.

Patrones de diseño

Los patrones de diseño tratan los problemas que se repiten y que se presentan en situaciones particulares del diseño, con el fin de proponer soluciones a ellas. Por lo tanto, los patrones de diseño son soluciones exitosas a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. Los patrones dan nombre y forma a heurísticas abstractas, reglas y buenas prácticas de técnicas orientadas a objetos (41).

Los patrones de software para la asignación general de responsabilidades (GRASP, por las siglas del inglés *General Responsibility Assignment Software Patterns*) describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en formas de patrones. A continuación, se describen aquellos patrones utilizados de acuerdo a las soluciones que brindan dentro del contexto del sistema:

- **Experto:** es uno de los más utilizados, debido a que se debe tener en cuenta que solo se asignarán responsabilidades a las clases que contengan la información para llevarlas a cabo.

- **Bajo acoplamiento:** Es la meta principal que ha de tenerse en cuenta en cada momento en todas las decisiones de diseño. Es un patrón evaluativo que el desarrollador aplica al valorar sus decisiones de diseño. Mejora la claridad y la facilidad con que se entiende el diseño. Se simplifican el mantenimiento y las mejoras en funcionalidad. A menudo genera un bajo acoplamiento (41). Este patrón fue utilizado en el diseño de la aplicación de manera general, siguiendo la premisa de que cada clase debe contener operaciones que resuelvan necesidades a fines con ellas.

- **Alta cohesión:** Es un principio que se debe recordar durante las decisiones de diseño: es la meta principal que es preciso tener presente siempre. Estimula asignar una responsabilidad de modo que su colaboración no incremente el acoplamiento tanto que produzca los resultados negativos propios de un alto acoplamiento. Soporta el diseño de clases más independientes, que reducen el impacto de los

cambios (41) . Este patrón fue utilizado para el diseño de las clases utilizadas en el mecanismo de depuración.

- **Controlador:** se encarga de que una clase actúe como intermediaria para el manejo de eventos. Este patrón se evidencia en las clases controladoras definidas.

2.6 Modelo de datos del sistema

El diseño de la base de datos es de gran importancia para el almacenamiento de los datos y para permitir a los usuarios recuperar y actualizar la información en base a sus peticiones posteriormente.

Una característica fundamental del enfoque de BD es que proporciona cierto nivel de abstracción de los datos, al ocultar detalles de almacenamiento que la mayoría de los usuarios no necesitan conocer. Los modelos de datos son el principal instrumento para ofrecer esta abstracción. Además, describen la representación lógica y física de los datos persistentes en el sistema (42).

El modelo de datos que se utilizó en la aplicación es la que esta mostrada en la siguiente imagen:

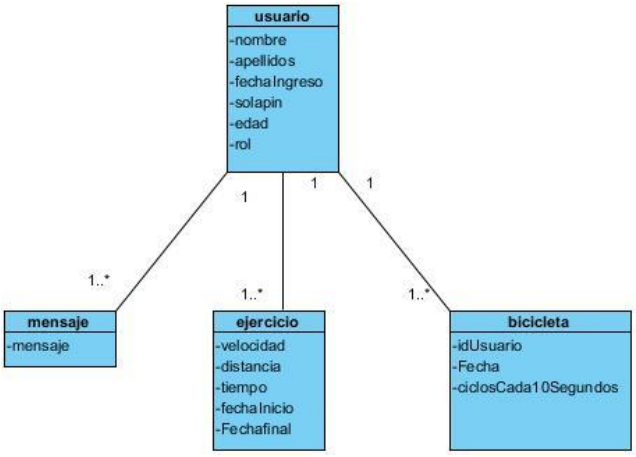


Ilustración 11: Modelo de datos

Es de vital importancia resaltar que el framework Symfony utiliza para el trabajo con las base de datos el ORM Doctrine, la cual trabaja con objetos en vez de trabajar directamente con las tablas. El modelo de datos de la aplicación tiene alta complejidad debido a que va a estar recibiendo datos externos a través de un microcontrolador y en ejercicios de larga duración el almacenamiento de los datos puede llegar a ser muy grande

A continuación se procede a describir las entidades principales dentro del sistema:

Entidad	Descripción
Ejercicio	En esta entidad se guardan los datos referentes a los ejercicios que debe realizar el atleta los cuales son insertados por el profesor. • Velocidad: se refiere a la velocidad promedio que debe tener el atleta en la bicicleta • Distancia; se refiere a la distancia virtual que debe alcanzar el atleta en la bicicleta • Tiempo: se refiere al tiempo mínimo que debe realizar el ejercicio • fechaInicio: almacena el inicio del plazo en el que el atleta debe de realizar el ejercicio • fechaFinal: se refiera al fin del plazo en el que el atleta debe de realizar el ejercicio
Bicicleta	En esta entidad se guardan los datos extraídos por el microcontrolador insertado en la bicicleta para que a partir de aquí, tanto el profesor como el atleta puedan visualizar todos los datos que se han podido extraer durante el entrenamiento. • CicloCada10Segundos: se refiere a la llamada que se le hace al microcontrolador para que devuelva el

	número de la cantidad de pedales que dio el atleta durante 10 segundos. • Fecha: se refiere a la fecha que va a tener el ejercicio en el momento en que se le pide los datos al microcontrolador integrado a la bicicleta. • id Usuario: se refiere al identificador del usuario que está realizando el ejercicio.

Tabla 10: Descripción de las entidades principales del sistema

Diagramas de clases propuesto como solución

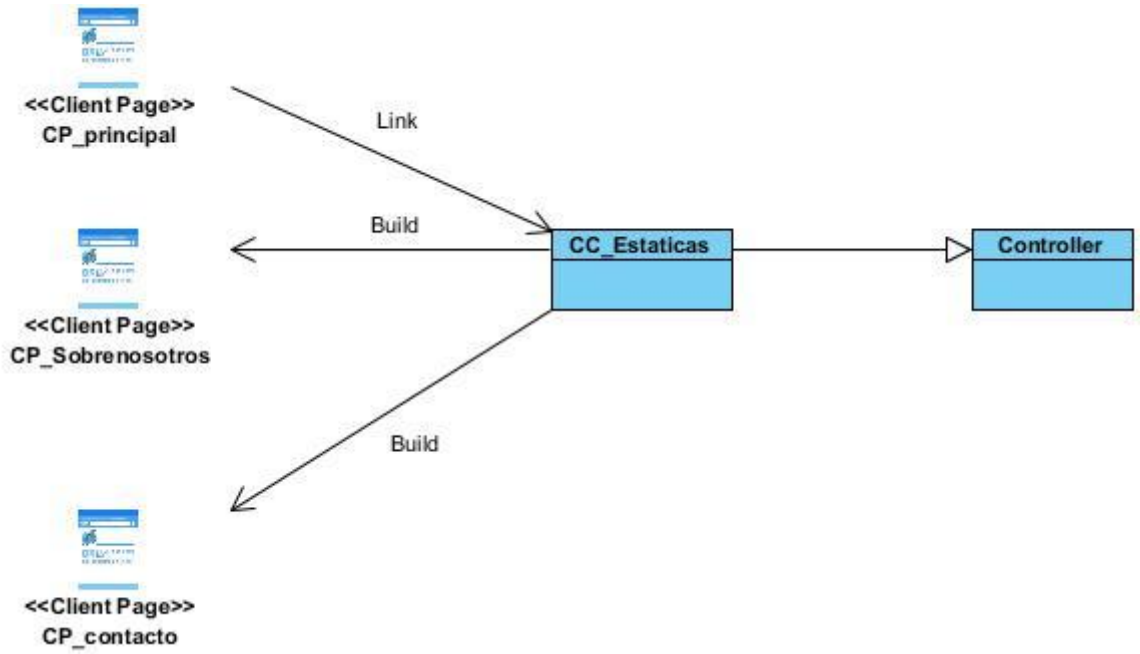


Ilustración 12: Diagrama de clases: CC_Estáticas

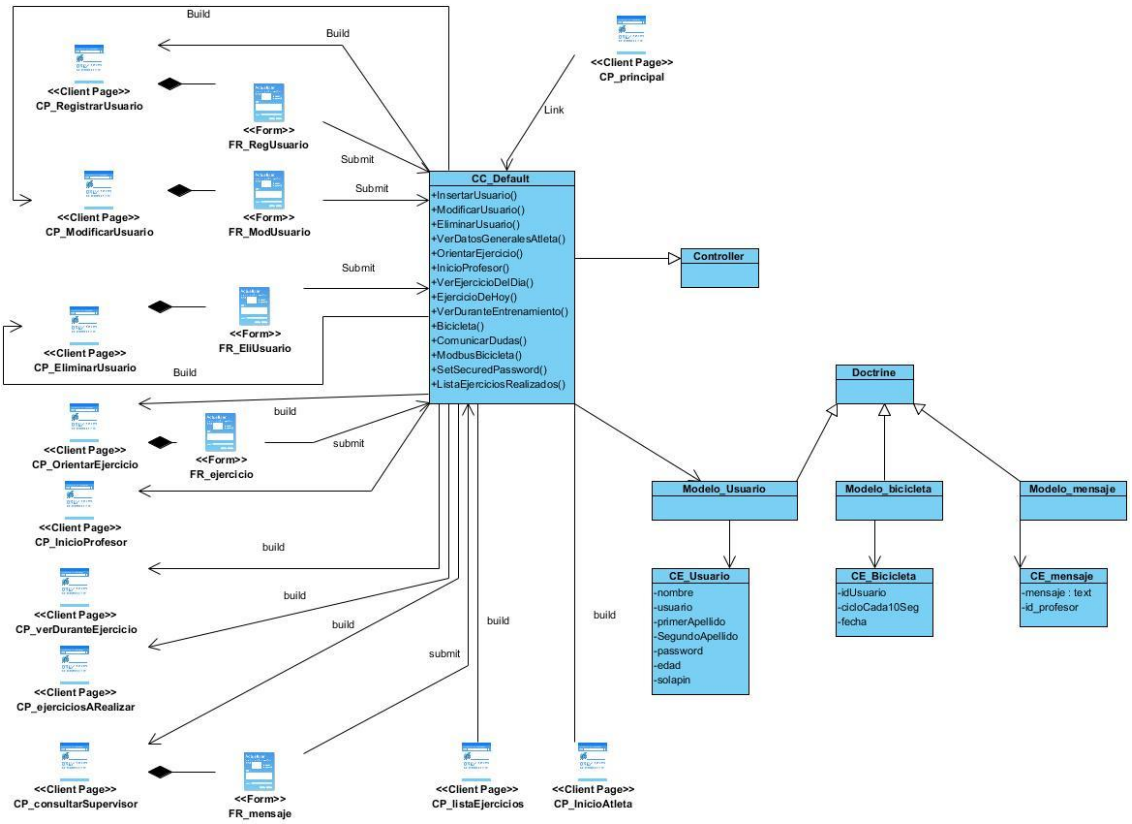


Ilustración 13: Diagrama de clases: CC_default

La estructura de datos propuesta es la distribución de las clases que están involucradas la aplicación, la cual se explica el proceso que realiza el sistema cuando el usuario desea acceder a alguna página en el sistema, por ejemplo: el usuario desde la página principal desea acceder a la página de modificar usuario, desde la página principal, pincha en el link, este envía la petición a la clase controladora y esta construye la página que desea el usuario.

Modelo de despliegue

Un diagrama de despliegue muestra las relaciones físicas entre los componentes hardware y software en el sistema final. Este diagrama es útil para ilustrar la arquitectura física de un sistema. A continuación de muestra una representación gráfica de donde se encuentra la aplicación situada en la PC cliente como funcionalidad adicional para que los datos extraídos del microcontrolador que se encuentra instalado en la bicicleta ergométrica.

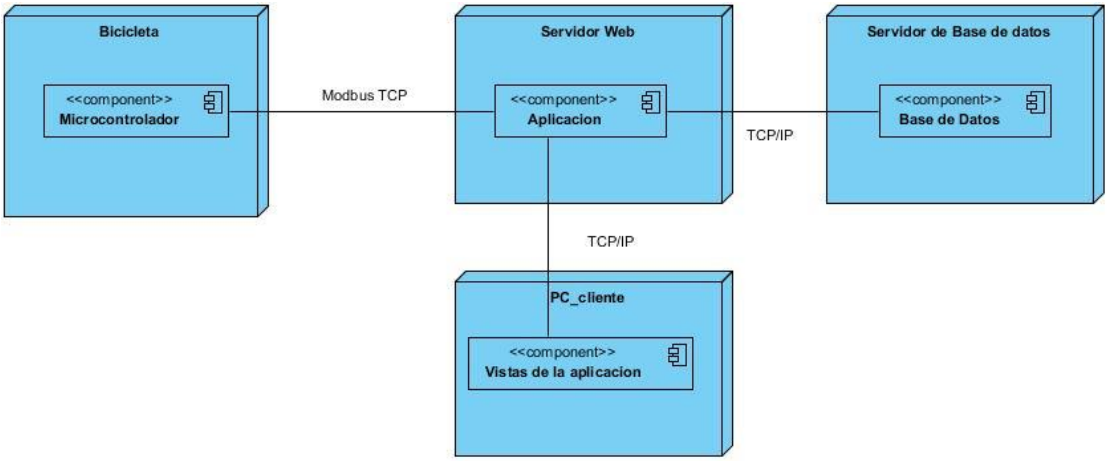


Ilustración 14: Modelo de despliegue

Diagrama de componentes

Los diagramas de componentes se utilizan para modelar la vista estática de un sistema. Muestran la organización y las dependencias lógicas entre un conjunto de componentes de software, sean éstos componentes de código fuente, librerías, binarios o ejecutables. No es necesario que un diagrama incluya todos los componentes del sistema, normalmente, se realizan por partes.

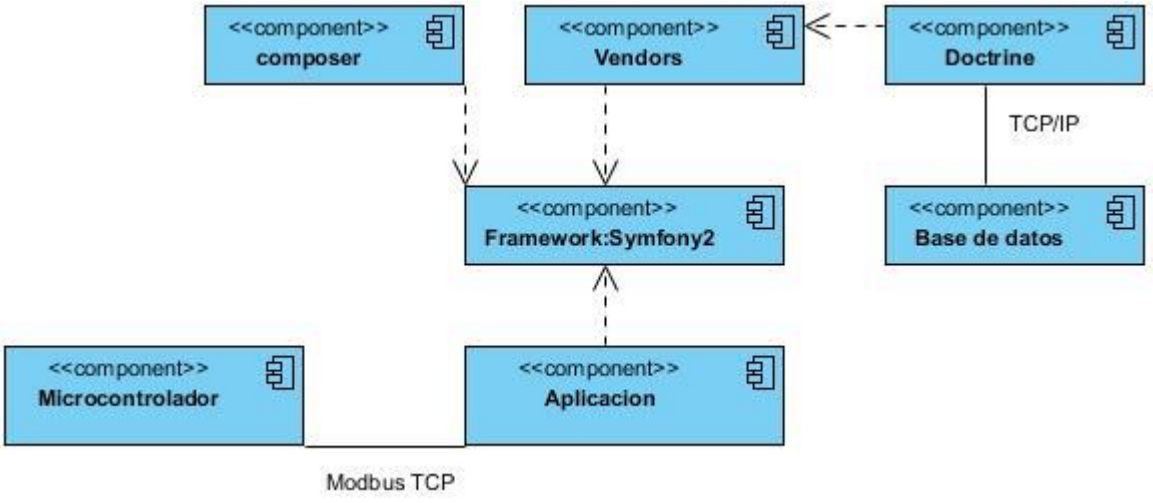


Ilustración 15: Diagrama de componentes

Solución

Se desarrolla una aplicación capaz de cumplir con los requisitos planteados, respondiendo a las funcionalidades siguientes:

1. Permite al atleta poder visualizar mientras realiza sus ejercicios los datos que se extraen durante su entrenamiento, además le permite comunicarle dudas al supervisor que está en frente por si no le queda claro alguna de las actividades ordenadas.
2. Permite al supervisor poder extraer todos los datos exactos de un atleta mientras realiza los ejercicios en la bicicleta lo cual le permitirá realizar mejores metodologías de preparación física para el atleta.
3. Mostrar tablas comparativas donde ayuda y facilita el proceso de entrenamiento.

Se propone implementar una aplicación inicial completamente funcional dando como salida una herramienta que puede integrarse al funcionamiento en tiempo real para la realización de estos entrenamientos.

Conclusiones parciales

En este capítulo se han elaborado los aspectos referentes a la concepción de la aplicación y sus características funcionales y no funcionales. Se detallaron las principales particularidades del funcionamiento de la aplicación, haciendo especial énfasis en la obtención de los datos extraídos de la bicicleta, almacenarlos y mostrarlos de forma automática, generando a partir de ellos gráficos y estadísticas para un mejor entendimiento de los datos extraídos por la bicicleta ergométricas.

Capítulo 3: Fases desarrollo y pruebas

En el presente capítulo se detallan la implementación del sistema perteneciente a la fase de desarrollo y las pruebas realizadas a mecanismo de obtención para el almacenamiento y muestreo de los datos extraídos de las bicicletas ergométricas pertenecientes a la fase de pruebas. Se identifican y definen las tareas de ingeniería y se llevan a cabo las pruebas de aceptación realizadas a la aplicación con el objetivo de que cumpla con lo esperado.

3.1 Fase de desarrollo

Dentro de la fase de desarrollo se encuentra la implementación o el desarrollo del código el cual se sustenta en buenas prácticas planteadas por el ciclo de vida de AUP_UCI para esta etapa entre las que se encuentran: el desarrollo del sistema y probado al completo en el ambiente de desarrollo.

3.1.1 Tareas de ingeniería o desarrollo

Asociado a cada iteración se encuentra la planificación de las tareas de ingeniería o programación, cada historia de usuario se transforma en estas tareas que son desarrolladas por programadores, dentro del equipo de desarrollo, aplicando la práctica de la programación en parejas. Para cada iteración se realizó la distribución de tareas en correspondencia con las historias de usuarios que se desarrollaron.

Historia de usuario	Tareas de ingeniería
HU1 Autenticar usuario.	1. Desarrollo de una plantilla para la función de autenticación en el sistema.
HU2 Registrar usuario.	1. Desarrollo de un formulario para registrar el usuario. 2. Desarrollo de una plantilla para la función de registro de usuario.
HU3 Modificar Usuario.	1. Desarrollo de una plantilla para la función de modificar usuario.

HU4 Visualizar datos Generales.	1. Desarrollo de una plantilla donde muestre los datos generales de un usuario.
HU5 Ver orientaciones de entrenamiento a realizar.	1. Desarrollo de una plantilla donde visualice las tareas que le están asignadas al atleta.
HU6 Guardar los datos extraídos de la bicicleta durante el entrenamiento.	1. Desarrollo de un método donde guarde automáticamente cada 10 segundos los datos extraídos por la bicicleta.
HU7 Ver datos durante el entrenamiento.	1. Desarrollo de una plantilla donde muestre los datos de los ejercicios que el atleta está realizando de manera asincrónica.
HU8 Orientar próximo entrenamiento.	1. Desarrollo de una plantilla donde pueda el entrenador insertar los ejercicios que debe hacer el atleta.

Tabla 11: Tabla de las tareas de ingeniería

3.2 Fases de pruebas

Las pruebas son consideradas una de las fases más importantes en todo proyecto que se lleve a cabo, ya que estas permiten corregir la mayor cantidad de errores posibles y entregar un producto de mayor calidad. Existen diferentes tipos de prueba, entre ellas están las pruebas unitarias y las pruebas de aceptación. Las pruebas unitarias son una forma de probar el correcto funcionamiento de un módulo de código. Esto sirve para asegurar que cada uno de los módulos funcione correctamente por separado. Por otra parte, las pruebas de aceptación evalúan el grado de calidad del software con relación a todos los aspectos relevantes para que el uso del producto se justifique. Para el caso del sistema desarrollado se realizaron

pruebas de aceptación, utilizando el método de caja negra con la técnica de partición de equivalencia.

3.2.1 Pruebas de aceptación

Las pruebas de aceptación son creadas en base a las historias de usuario, en cada ciclo de la iteración del desarrollo. El cliente debe especificar uno o diversos escenarios para comprobar que una historia de usuario ha sido correctamente implementada. Las pruebas de aceptación son consideradas como “pruebas de caja negra”, Los clientes son responsables de verificar que los resultados de estas pruebas sean correctos (43).

Como criterio de aprobación de cada iteración se tomó que el 100% de los casos de prueba sean exitosos para pasar de iteración. El objetivo de estas pruebas no es tener un conjunto de casos escritos que cubran el 100% del código, sino poder realizarle pruebas al sistema desde el punto de vista del usuario.

Para la realización de cada una de las pruebas de aceptación se siguieron una serie de pasos que se muestran a continuación:

- Identificar todas las acciones en la historia de usuario.
- Para cada acción escribir al menos una prueba.
- Para algunos datos, reemplazar las entradas que hacen que la acción ocurra y llenar en la casilla resultados esperados los resultados obtenidos.
- Para otros datos, reemplazar las entradas que hacen que la acción falle, y registrar los resultados.

Para representar las pruebas de aceptación se definieron los siguientes elementos:

- **Código:** Representa al caso de prueba, incluye el número de HU, de la prueba y si posee diferentes escenarios.
- **HU:** Número de la HU a la cual pertenece.
- **Nombre:** Junto al código conforma el identificador del caso de prueba.
- **Descripción:** Acción que debe realizar el sistema.
- **Condiciones de ejecución:** Describe las características y elementos que debe contener el sistema para realizar el caso de prueba.

- **Entrada:** Incluye las entradas necesarias para realizar el sistema.
- **Pasos de ejecución:** Pasos para realizar el caso de prueba**.**
- **Resultados Esperados:** Descripción de la respuesta del sistema ante el caso de prueba.
- **Resultado Obtenido:** Respuesta visual del sistema después de realizar el caso de prueba.
- **Evaluación de la prueba:** Clasificación de la prueba en satisfactoria o insatisfactoria.
- **Iteración 1:**

Caso de prueba de aceptación	
Código: HU1_P1	**HU:** 1
Nombre: Autenticar usuario.	
Descripción: Prueba para la funcionalidad autenticar usuario.	
Condiciones de ejecución: El usuario debe estar previamente registrado. El usuario y contraseña deben ser válidos.	
Pasos de ejecución: Se intenta autenticar un usuario en el sistema con los datos válidos.	
Resultados esperados: El usuario se autentica correctamente en el sistema.	
Evaluación de la prueba: Satisfactoria.	

Tabla 12: Caso de prueba de aceptación - Autenticar usuario

Caso de prueba de aceptación	
Código: HU2_P2	**HU:** 2
Nombre: Registro de usuario.	
Descripción: Prueba para la funcionalidad registrar un usuario.	
Condiciones de ejecución: El usuario que se va a registrar debe introducir correctamente el nombre, los apellidos, el nombre del usuario y la clave o contraseña a crear. En el caso de la clave, esta debe contener al menos 6 caracteres, que pueden ser letras, dígitos o los caracteres (#@$%).	

Pasos de ejecución: Se intenta crear un usuario en el sistema con los datos válidos.
Resultados esperados: El usuario se crea correctamente en el sistema.
Evaluación de la prueba: Satisfactoria.

Tabla 13: Caso de prueba de aceptación - Registro de usuario

Caso de prueba de aceptación	
Código: HU3_P2	**HU:** 3
Nombre: Modificar usuario.	
Descripción: Prueba para la funcionalidad modificar usuario.	
Condiciones de ejecución: El usuario que va a modificar sus datos de perfil debe introducir correctamente el nombre, los apellidos, el nombre del usuario y la clave o contraseña a cambiar. En el caso de la clave, esta debe contener al menos 6 caracteres, que pueden ser letras, dígitos o los caracteres (#@$%).	
Pasos de ejecución: Se intenta crear un usuario en el sistema con los datos válidos.	
Resultados esperados: El usuario se crea correctamente en el sistema.	
Evaluación de la prueba: Satisfactoria.	

Tabla 14: Caso de prueba de aceptación - Modificar usuario

Caso de prueba de aceptación	
Código: HU4_P2	**HU:** 4
Nombre: Visualizar datos generales.	
Descripción: Prueba para la funcionalidad de visualizar los datos generales del atleta.	
Condiciones de ejecución: Para acceder a esta página tiene que ser un usuario con el rol de atleta.	
Pasos de ejecución: Se intenta acceder a la página visualizar datos generales.	
Resultados esperados: El usuario con el rol de atleta acceder correctamente y la página le carga todos los datos necesarios.	

Evaluación de la prueba: Satisfactoria.

Tabla 15: Caso de prueba de aceptación - Visualizar datos generales

Caso de prueba de aceptación	
Código: HU5_P3	**HU:** 5
Nombre: Ver orientaciones de entrenamiento a realizar.	
Descripción: Prueba para la funcionalidad de ver orientaciones de entrenamiento a realizar.	
Condiciones de ejecución: El usuario debe estar previamente registrado y tener los permisos del rol de atleta para poder usar esta funcionalidad.	
Pasos de ejecución: Se intenta consultar los ejercicios que debe realizar el usuario para hoy.	
Resultados esperados: Se consulta correctamente los ejercicios preparados para ese día.	
Evaluación de la prueba: Satisfactoria.	

Tabla 16: Caso de prueba de aceptación - Ver orientaciones de entrenamiento a realizar

Caso de prueba de aceptación	
Código: HU6_P4	**HU:** 6
Nombre: Guardar los datos extraídos de la bicicleta durante el entrenamiento.	
Descripción: Prueba para la funcionalidad de guardar los datos extraídos de la bicicleta durante el entrenamiento.	
Condiciones de ejecución: El usuario debe estar previamente registrado, tener los permisos del rol de atleta para poder usar esta funcionalidad y estar en la funcionalidad de ver durante el ejercicio.	
Pasos de ejecución: Se intenta consultar los datos de los ejercicios que está realizando de manera asincrónica y almacenarlos simultáneamente.	
Resultados esperados: Se almacena correctamente los datos de los ejercicios que está realizando de manera asincrónica.	

Evaluación de la prueba: Satisfactoria.

Tabla 17: Caso de prueba de aceptación - Guardar los datos extraídos de la bicicleta durante el entrenamiento

Caso de prueba de aceptación	
Código: HU7_P5	**HU:** 7
Nombre: Ver datos durante el entrenamiento.	
Descripción: Prueba para la funcionalidad ver los datos durante el entrenamiento.	
Condiciones de ejecución: El usuario debe estar previamente registrado y tener los permisos del rol de atleta para poder usar esta funcionalidad.	
Pasos de ejecución: Se intenta consultar los datos de los ejercicios que está realizando de manera asincrónica.	
Resultados esperados: Se consulta correctamente los datos de los ejercicios que está realizando de manera asincrónica.	
Evaluación de la prueba: Satisfactoria.	

Tabla 18: Caso de pruebas de aceptación - Ver durante el entrenamiento

Caso de prueba de aceptación	
Código: HU8_P6	**HU:** 8
Nombre: Orientar próximo entrenamiento.	
Descripción: Prueba para la funcionalidad de orientar el próximo entrenamiento del atleta.	
Condiciones de ejecución: El usuario debe estar previamente registrado y tener los permisos del rol de supervisor para poder usar esta funcionalidad.	
Pasos de ejecución: Se intenta guardar las próximas orientaciones que el atleta debe de realizar	
Resultados esperados: Se guarda correctamente los datos.	

Evaluación de la prueba: Satisfactoria.

Tabla 19: Caso de prueba de aceptación - Orientar próximo entrenamiento

Resultados de las pruebas de aceptación

Durante el proceso de pruebas, se realizaron un total de 8 casos de pruebas de aceptación (caja negra), de ellos todos resultaron satisfactorios, lo cual representa el 100% de satisfacción. La aplicación de estas pruebas permitió comprobar la correcta implementación de las historias de usuario definidas con anterioridad.

Conclusiones parciales

En este capítulo se especificó el proceso de implementación del sistema. Se detallaron las tareas de ingeniería asociadas a las historias de usuarios definidas. Además, se definió el estándar de programación a utilizar y se detallaron las pruebas de aceptación que brindaran al cliente la conformidad y seguridad ante las funcionalidades del sistema.

Conclusiones Generales

En el presente trabajo de diploma se presentó la investigación y el posterior proceso de desarrollo de la plataforma para la supervisión de los ejercicios en las bicicletas ergométricas y se llegó a las siguientes conclusiones:

- Con la implementación de una aplicación capaz de almacenar y mostrar todos los datos de manera automática que se puedan extraer de las bicicletas ergométricas se dió cumplimiento al objetivo de la investigación presentada.
- El proceso de desarrollo guiado por la metodología AUP-UCI e implementado con el framework Symfony2 en conjunto con MySQL y Arduino facilitaron la creación de la plataforma en un periodo corto de tiempo.
- Se mejora el rendimiento de los atletas a través de entrenamientos más eficientes generados a partir de los datos obtenidos por el sistema.
- Se pone a disposición del departamento de educación física una aplicación extensible para mejorar la calidad de la enseñanza.
- Se ahorra recursos económicos del país ya que no se necesita adquirir la tecnología que necesitan están bicicletas ergométricas y se dispone de una alternativa nacional desarrollado con tecnologías libres.

Recomendaciones

- Continuar con el estudio de las aplicaciones para la gestión de los datos extraídos de un entrenamiento con el objetivo de encontrar mejoras e incluirlas en futuras versiones.
- Perfeccionar la funcionalidad del almacenamiento e presentación de los datos para hacer a la aplicación más eficiente en su rendimiento y utilización.
- Estudiar sobre otros aparatos mecánicos para hacer ejercicios para ver la posibilidad de insertarlos en la aplicación y hacer de esta más integral para realizar el proceso de entrenamiento.

Referencias Bibliográficas

1. **Bompa, Tudor O.** *Periodización de la fuerza.* s.l. : Biosystem Servicio Educativo, 1995.

2. **Sanitas.es. [En línea] http://www.sanitas.es/sanitas/seguros/es/particulares/biblioteca-de-salud/ejercicio-deporte/Consejos-para-correr/conceptos-entrenamiento.html.**

3. MasMusculos. [En línea] 25 de mayo de 2011. http://www.masmusculo.com.es/workout/los-beneficios-de-la-bicicleta-ergometrica/.

4. Ramirez, Francisco. Electronicos Online.com. [En línea] http://www.electronicosonline.com/2008/09/18/Conceptos-basicos-de-los-Microcontroladores/?imprimir=true.

5. Atmel. Atmel. [En línea] http://www.atmel.no.

6. Arduino. Arduino. [En línea] [Citado el: 15 de abril de 2016.] http://www.arduino.cc.

7. P. H., Palomeque. Informática & Deportes. [En línea] 2009. [Citado el: 1 de noviembre de 2015.] http://www.entrenar.com.ar/contacto.php?action=consulta&ref=.

8. Sariola, J. A. M. Espacio virtual para profesores y entrenadores de judo. [En línea] [Citado el: 2015 de noviembre de 1.] From http://www.mirallas.org/.

9. Seco, I. Elatleta.com. [En línea] 2000. [Citado el: 2015 de noviembre de 1.] http://www.elatleta.com.

10. Ortega, F. O. G. *Automatización de la planificación del entrenamiento deportivo en diferentes deportes.* 2005.

11. AGUIRRE, C. B. E. y ESQUIVEL, P. Comparativa de Frameworks para el desarrollo de aplicaciones con php. [En línea] [Citado el: 1 de diciembre de 2015.] http://dspace.uazuay.edu.ec/bitstream/datos/2125/1/08920.pdf.

12. RIEHLE, D. y GROSS, T. *Role model based framework design and integration.* s.l. : ACM SIGPLAN Notices, 1998. 117-133.

13. PHPFRAMEWORK. Top 10 php frameworks. [En línea] [Citado el: 2015 de noviembre de 5.] http://www.phpframeworks.com/top-10-php-frameworks/.

14. Equiluz, Javier. *Desarrollo web ágil con Symfony2.* 2013.

15. ACHOUR, M. y BETZ, F., et al. Manual de PHP. [En línea] [Citado el: 10 de noviembre de 2015.] http://docs.php.net/manual/es.

16. MARIÑO, C. V. Programación en PHP5. Nivel Básico. [En línea] 2008. [Citado el: 10 de noviembre de 2015.] https://n-1.cc/file/download/745373. .

17. LAWSON, B. y SHARP, R. Introducing HTML5. [En línea] 2011. [Citado el: 10 de noviembre de 2015.] http://sunshine.prod.uci.cu/gridfs/sunshine/books/Introducing-HTML5---New-Riders.pdf..

18. ZELDMAN, J. Diseño con estándares web. [En línea] [Citado el: 10 de noviembre de 2015.] http://books.google.com.ar/books/about/Diseño_con_estándares_web.html?id=tMJ_AAAACAAJ.

19. PÉREZ, J. E. Introducción a CSS. [En línea] 2007. [Citado el: 10 de noviembre de 2015.] http://www.librosweb.es/css.

20. (W3C), T. W. W. W. C. HTML & CSS. [En línea] 2012. [Citado el: 11 de noviembre de 2015.] http://www.w3.org/standards/webdesign/htmlcss#whatcss..

21. ALVAREZ, S. Sistemas gestores de bases de datos. [En línea] 31 de julio de 2007. [Citado el: 14 de noviembre de 2015.] http://www.desarrolloweb.com/articulos/sistemas-gestores-bases-datos.html..

22. ORACLE. Productos y servicios Oracle. [En línea] [Citado el: 2015 de novimbre de 14.] http://www.oracle.com/es/products/index.html..

23. POSTGRESQL. PostgreSQL Global Development Group. [En línea] [Citado el: 1 de diciembre de 2015.] http://www.postgresql.org/about/..

24. GROUP, T. P. API MySQL original. [En línea] [Citado el: 12 de diciembre de 2015.] http://php.net/manual/es/book.mysql.php..

25. PECOS, D. PostgreSQL vs. MySQL. [En línea] http://danielpecos.com/documents/postgresql-vs-mysql/.

26. ROUSE, M. integrated development environment (IDE) definition. [En línea] http://searchsoftwarequality.techtarget.com/definition/integrated-development-environment.

27. STUDIO, Z. The PHP IDE for Smarter Development | Zend Studio. [En línea] [Citado el: 12 de diciembre de 2015.] http://www.zend.com/en/products/studio..

28. MICROSYSTEMS, S. NetBeans IDE 8.0. [En línea] https://netbeans.org/community/releases/80.

29. JETBRAINS. PHP IDE :: JetBrains PhpStorm. [En línea] [Citado el: 12 de diciembre de 2015.] https://www.jetbrains.com/phpstorm/.

30. JACOBSON, I. y BOOCH, G., et al. El Lenguaje Unificado de Modelado (UML). [En línea] http://ingenieriasoftware2011.files.wordpress.com/2011/07/el-lenguaje-unificado-de-modelado-manual-de-referencia.pdf.

31. THAMES, J. P. B. Ingeniería del software asistida por computadora (case). [En línea] http://es.slideshare.net/jpbthames/ingeniera-del-software-asistida-por-computadora-case..

32. IBM. [En línea] http://www-03.ibm.com/software/products/es/enterprise.

33. LTD, V. P. I. Visual Paradigm for UML. [En línea] http://www.visual-paradigm.com..

34. HERNANDO, R. Metologías de desarrollo de software. [En línea] http://www.rhernando.net/modules/tutorials/doc/ing/met_soft.html.

35. BECK, K. *Extreme programming explained: embrace change. Addison-Wesley Professional.* s.l. : ISBN , 2000. 0201616416.

36. Sánchez, Tamara Rodríguez. *Metodología de desarrollo para la actividad productiva de la UCI.* 2015.

37. IBM. [En línea] 21 de septiembre de 2015. https://www.ibm.com/support/knowledgecenter/es/SSFKSJ_7.5.0/com.ibm.mq.pro.doc/q002870_.htm.

38. modbus. modbus. [En línea] http://www.modbus.org.

39. Modbus. service names port numbers. [En línea] http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xml.

40. Potencier, Fabien. [En línea] http://fabien.potencier.org/article/49/what-is-symfony2.

41. Larman, Craig. *UML y patrones. Introducción al análisis y diseño orientado.* 2003.

42. ALCALÁ, J. y PELTA, D. Fundamentos de Informática, modelado de Bases de Datos. [En línea] http://www.ugr.es/~jalcala/teaching/informatica/teoria/Tema3.pdf.

43. Joskowicz, Jose. *Reglas y prácticas en Extreme Programing.* Universidad de Vigo : s.n., 2008.

44. Morales, Frank. FrankMolrales. [En línea] http://frankmorales.webcindario.com/trabajos/planificacion.html.