

Universidad de Valladolid

E.U. DE INFORMÁTICA (SEGOVIA)

Ingeniería Técnica en Informática de Gestión

Gestión Integral de Clubes Deportivos.

Alumno: Diego Cebrián

Tutor: Jesús Cordobés Puertas

Índice de contenido

1. Introducción.....	1
1.1 Idea inicial.....	2
1.2 Estructura de los clubes.....	2
1.3 Ampliación de funciones.....	2
1.4 Funcionamiento de la aplicación.....	3
2. Análisis del sistema.....	4
2.1 Ámbito de la aplicación.....	5
2.1.1 Gestión de asociaciones y clubes deportivos.....	5
2.1.2 Gestión de organizadores de eventos deportivos.....	6
2.1.3 Gestión de empresas de que ocio y deporte.....	6
2.2 Objetivos del proyecto.....	6
2.3 El lenguaje de modelado unificado (UML).....	7
2.4 Metodología de análisis.....	8
2.5 Glosario de términos deportivos.....	9
2.6 Análisis de Requisitos.....	9
2.6.1 Requisitos Funcionales.....	9
2.6.1.1 Gestión deportiva.....	10
2.6.1.2 Gestión Logística.....	13
2.6.1.3 Gestión social.....	14
2.6.1.4 Gestión Económica.....	14
2.6.2 Requisitos no funcionales.....	15
2.7 Identificación de las clases de análisis.....	16
2.7.1 Clases candidatas.....	16
2.7.1.1 Del análisis de la gestión deportiva.....	16
2.7.1.2 Del análisis de la gestión logística.....	16
2.7.1.3 Del análisis de la gestión social.....	16
2.7.1.4 Del análisis de la gestión económica.....	17
2.7.2 Clases del dominio del problema.....	17
2.7.2.1 Del análisis de la gestión deportiva.....	17
2.7.2.2 Del análisis de la gestión logística.....	18
2.7.2.3 Del análisis de la gestión social.....	18
2.7.2.4 De análisis de la gestión económica.....	18
3. Planificación.....	19
3.1 Desarrollo temporal.....	20
3.2 Ciclo de vida y metodología.....	20
3.3 El diseño del interfaz de usuario.....	21
3.4 Presupuesto y financiación.....	21
4. Diseño de la solución final.....	23
4.1 Elección de la plataforma.....	24
4.1.1 Soluciones multiplataforma.....	24
4.1.1.1 Lenguajes mono-plataforma con interfaz de usuario común.....	24
4.1.1.2 Interfaz web a través de navegador.....	25

4.1.1.3 Tecnologías multiplataforma.....	26
4.1.2 La gestión de los datos.....	26
4.1.2.1 Gestores de Bases de Datos Orientados a Objetos.....	26
4.1.2.2 Gestores de Bases de Datos Relacionales.....	27
4.1.2.4 Gestores de Bases de Datos Relacional-Objeto.....	27
4.1.2.3 Sistemas intermediarios entre Base de Datos Relacional o Relacional Objeto y Programación Orientada a Objetos.....	28
4.1.3 El servidor.....	29
4.1.3.1 Windows.....	29
4.1.3.2 *NIX.....	30
4.1.4 La plataforma elegida.....	30
4.1.5 Diseño de la infraestructura.....	31
4.2 Diseño de las clases de objetos.....	33
4.2.1 Diccionario de clases.....	34
4.2.2 Diagramas de clases.....	34
4.3 Diseño de la base de datos.....	34
4.4 Diseño del interfaz gráfico de usuario.....	34
4.5 Diseño de las pruebas.....	34
5. Manuales de usuario y administrador.....	36
5.1 Manual de instalación.....	37
5.2 Manual de usuario.....	37
5.3 API de extensión de funcionalidades.....	37
6. Índices de anejos.....	38
Índice de Ilustraciones.....	39
7. Bibliografía.....	41

1. Introducción

Descripción de las diferentes ideas que dan lugar a la realización de este proyecto, ideas iniciales, evoluciones y posibilidades de la aplicación.

1.1 Idea inicial

La idea inicial de este proyecto nace de la necesidad detectada por el alumno que lo realiza al enfrentarse a la tesorería del club de atletismo en que milita. Al ver las dificultades que existen cuando se trata de gestionar económicamente un club deportivo con un programa de gestión comercial generalista se plantea la creación de uno especializado en clubes deportivos. El programa, aparte de la gestión económica propiamente dicha debería tener la capacidad de controlar el activo de socios, deportistas, instalaciones, material deportivo, seguros médicos, organización de competiciones propias, desplazamientos a competiciones, publicación de resultados en la web y prensa. Además debería proporcionar métodos para poder incluir módulos que amplíen las funciones del programa de una forma relativamente fácil. Aunque las necesidades iniciales se atribuyen a un club de atletismo este programa debería ser perfectamente válido para la gestión de cualquier club deportivo, ya sea este de fútbol o de tenis de mesa.

1.2 Estructura de los clubes

Como hemos podido comprobar en el punto anterior las áreas de gestión de un club son muy amplias. Probablemente a los deportistas no les importen muchos aspectos económicos, y a los entrenadores no les interese conocer los entresijos de la organización de una prueba. Así pues, cada persona que forme parte del club deberá tener acceso a unos datos diferentes. Ya veremos más adelante la manera de conseguirlo, pero de lo que si que podemos hablar es de una diferenciación entre diferentes estamentos de la organización, con sus permisos y para ver, modificar y agregar datos a esta aplicación.

En la estructura de todo club deportivo hay muchas relaciones internas, flujos económicos y de trabajo, que se han de representar correctamente. El programa que vamos a hacer deberá cumplir estos flujos de trabajo de manera que a cada persona asociada a los mismo se les deberá dar la información necesaria para cumplir con su labor, pero debidamente filtrada para que el exceso de información no dificulte las tareas y se completen lo más pronto posible.

Muchas de las figuras que trabajan en club deportivo tienen acceso a datos confidenciales de otras, como por ejemplo los entrenadores a datos médicos de los deportistas, o el tesorero a datos bancarios. El programa debe cuidar también de conservar esta confidencialidad, e impedir que puedan producirse filtraciones.

1.3 Ampliación de funciones

La gestión de un club deportivo se puede extrapolar a otras organizaciones que tienen que ver con el deporte sin ser necesariamente clubes. Estos otros posibles destinatarios objetivos de la aplicación son los gimnasios, empresas organizadoras de eventos deportivos, ayuntamientos, empresas de gestión de recintos deportivos. Al tener la idea de ser un programa completamente modular, cualquier otra entidad que pueda necesitar de un software especializado podría beneficiarse de las características.

Inicialmente se comenzará con la gestión de deportes individuales, pasando en una segunda fase del proyecto a la de deportes de equipo. La razón de esta separación viene dada por las grandes diferencias entre estos tipos de preparación física. Posteriormente se incluiría la gestión de instalaciones deportivas desde el punto de vista de la explotación de las mismas y el registro de socios con cuotas para la explotación de gimnasios, piscinas, centros de alto rendimiento, residencias de deportistas y otros recintos habilitados para la práctica deportiva.

1.4 Funcionamiento de la aplicación

Para este proyecto se ha pensado en la diversidad de tipos de clubes deportivos que hay en el mundo. Se puede tratar de deportes de equipo, individuales, una agrupación que incluya varios deportes (F.C. Barcelona tiene secciones de Fútbol, Baloncesto, Balonmano, Fútbol Sala, Atletismo, Béisbol, Ciclismo, Hockey sobre hierba y hielo, Patinaje artístico, Rugby y Voleibol, cada una de ellas con sus diferentes categorías y secciones masculina y femenina). Esta condición hace que la aplicación deba poder ser ejecutada desde muchos lugares diferentes, como las diferentes sedes de cada deporte, a pesar de tener que tener los datos centralizados en un solo lugar. Tampoco conocemos los sistemas operativos utilizados por los diferentes posibles clientes en sus instalaciones. Es por eso que el programa deberá funcionar, al menos en cliente, en casi cualquier sistema operativo, así que deberá ser un sistema multiplataforma, ya sea a través de web con navegador, o utilizando tecnologías multiplataforma propiamente dichas, como entornos de desarrollo Qt, programación en .NET (con soporte para plataformas no windows con Mono), o la opción más conocida del lenguaje Java, que funciona en casi todo tipo de dispositivos, desde ordenadores con diferentes sistemas operativos, Linux, Solaris, Windows, MacOS y dispositivos móviles.

2. Análisis del sistema

Estudio de las necesidades del proyecto.

2.1 Ámbito de la aplicación

En nuestra sociedad los valores han cambiado mucho en los últimos 30 años. Hoy en día el deporte es visto como algo accesible a todos los públicos, y la proliferación de instalaciones deportivas en todas las ciudades y en los más pequeños municipios indican que su práctica ha pasado a formar parte del común de la sociedad. Debido a esto los gestores de estas instalaciones han tenido que buscar los recursos adecuados para gestionar el buen uso e instalación de las mismas.

Asociado a este gran crecimiento de la práctica del deporte también se ha generado la necesidad de agruparse, además de en el deporte de equipo donde esto es obvio, para obtener ventajas o compartir costes en la instalaciones. Así se han generado federaciones, clubes y asociaciones relacionadas con las prácticas deportivas.

También ha crecido en un gran porcentaje el tiempo dedicado al deporte en los medios de comunicación, de manera que la gente se siente más cercana al deporte. Esto ha generado que muchas personas tomen como pasatiempo la práctica de diferentes modalidades deportivas esporádicas. En torno a esta nueva necesidad han surgido empresas que se dedican a gestionar actividades deportivas, organizarlas, o que han creado instalaciones vacacionales basadas en el ejercicio físico.

Las empresas, como organizaciones dedicadas a hacer dinero a cambio de servicios, han realizado diferentes sistemas para el control de muchos de estos aspectos, centrándose sobre todo en la gestión de instalaciones, generación de carnés de socios y almacenamiento de unos pocos datos como control de accesos y uso de máquinas. Estos usos de las aplicaciones solo tienen una vertiente económica y no entran a valorar más allá de si las personas tienen derecho a usar la máquina una vez más o si han venido 30 veces al mes a la instalación. Ignoran por completo otros aspectos como los que se refieren a la gestión de los entrenamientos, las comunicaciones entre el club y sus deportistas, socios y empleados, la organización de los eventos, la comunicación pública, y las gestiones sociales, como almacenamiento de actas o relaciones de socios no deportistas.

Gesport intentará cubrir ese hueco dejado en los ERP, enterprise resource planning o sistemas de planificación de recursos empresariales, o crear un nuevo concepto de programa: sistema de planificación de recursos deportivos.

2.1.1 Gestión de asociaciones y clubes deportivos.

Las asociaciones y clubes deportivos son entidades sin ánimo de lucro que tienen entre sus objetivos el facilitar la práctica deportiva a sus socios o afiliados. Son el objetivo principal del mercado del proyecto. Las necesidades de estas organizaciones pueden ser muy diferentes dependiendo de su tamaño y masa social. En los últimos años la industria del ocio ha crecido mucho. Cada vez son más los clubes de golf que inundan nuestros campos y en ellos no solo se realiza la práctica de ese deporte, sino que incluyen servicios de gimnasio, tenis, entrenamientos personales. La calidad de vida viene ligada al ejercicio deportivo regular y en todos los ámbitos sociales se practica en la medida de lo posible.

Los clubes deportivos pueden dedicarse tanto a una labor de socialización del deporte, como pueden ser los relativos a golf o tenis, a la competición, o a la formación de nuevos deportistas de categorías inferiores. Además de cumplir varios de estos objetivos en un mismo club. Aunque los a los clubes deportivos se les suponga entidades sin ánimo de lucro son numerosos los casos en los que la rentabilidad es importante.

Pueden poseer instalaciones, teniendo una necesidad de gestionarlas en horarios adecuados para que todos sus socios o deportistas las aprovechen. Estas instalaciones pueden ser en propiedad o alquiladas.

Cuentan con una junta directiva que toma decisiones en los aspectos económicos, deportivos generales, organizativos y de estructura de club. La información que llega a esas juntas y la que sale de ella debe ser correspondientemente filtrada. Esta información y la toma de decisiones se debe poder aplicar y consultar en cualquier momento.

En los clubes existen equipos de profesionales, desde entrenadores, hasta médicos, fisioterapeutas, psicólogos y los propios deportistas en el caso de que compita en círculos de élite. Cada uno de estos profesionales tiene que tener acceso a todos los datos necesarios para realizar su trabajo, así como la posibilidad de comunicarse entre ellos, de manera que los masajistas o fisioterapeutas tengan claro cuando viene mejor un masaje de descarga o relajación, o ejercicios de fortalecimiento, o cuando los médicos tienen que realizar mayor aporte de vitaminas y estimulantes del sistema inmunológico, dependiendo de las cargas de entrenamiento, descansos.

También se debe contar con una agenda de plazos administrativos que facilite la información a la hora de rendir cuentas al estado, impuestos, seguros obligatorios, presentación de fichas federativas de las diferentes especialidades que se practiquen el club. Esto puede afectar en parte a la junta y en parte a los empleados.

En el aspecto logístico los problemas que puede solucionar la aplicación pueden venir dados por la organización de material de entrenamiento y competición, transportes internos y externos, y localización y repartición de los almacenes para ese control.

2.1.2 Gestión de organizadores de eventos deportivos

En la práctica del deporte existen eventos, ya sean competiciones o exhibiciones, donde se muestra a un público como se realiza la actividad. Si bien los clubes desarrollan esta actividad por si mismos, también es cierto que existen empresas, federaciones y asociaciones que organizan este tipo de acontecimientos.

Gesport deberá cumplir con esta necesidad y facilitar las gestiones, organizaciones de personal, arbitraje, seguridad, material necesario, premios, publicidad de patrocinadores de la prueba y de la propia prueba.

2.1.3 Gestión de empresas de que ocio y deporte

Las empresas que se dedican al mundo del deporte y ocio (gimnasios, piscinas, escuelas de danza, etc...) tienen que tener la misma capacidad de organización de un club deportivo con el añadido de la posibilidad de facturación, control de tiempos de accesos a instalaciones, nóminas de trabajadores o gestión de otros productos que puedan tener en tienda, como complementos alimenticios o máquinas para el trabajo personal en casa.

2.2 Objetivos del proyecto

El objetivo de este proyecto es crear un aplicación modular y extensible que posibilite la gestión de todos los aspectos deportivos, logísticos, económicos, sociales, legales y organizativos de cualquier organización, ya sea empresa, asociación o club, que tenga que ver con el deporte.

Se trata también de que la gestión de estos recursos sea eficiente y provea las herramientas

necesarias de conocimiento de todas las áreas del club a los diferentes usuarios, cada uno en su medida, de manera que tengan un acceso fácil y sencillo a cualquier información de su incumbencia.

Deberá ser de fácil manejo, pues al usuario final no se le presuponen capacidades del manejo de la informática más allá de una alfabetización básica.

Tendrá que ser multiplataforma, pues no se puede asegurar que todos los componentes de una organización tendrán el mismo tipo de ordenador, y menos cuando se piensa en esta aplicación para múltiples organizaciones.

Estará obligado por ley a guardar la privacidad de los datos de los usuarios en los niveles que así lo requiera, tendrá que cumplir con la Ley orgánica de Protección de datos y los usuarios podrán borrar o rectificar los datos personales contenidos en los ficheros del programa. Deberá permitir la importación o exportación de firmas digitales. La codificación de datos para permitir la lectura de los mismos por las personas adecuadas será asimétrica.

Deberá presentar la información justa para la toma de decisiones en cada momento. Los datos deberán poder simplificarse en resúmenes, gráficos o esquemas.

2.3 El lenguaje de modelado unificado (UML)

El lenguaje de modelado unificado es un lenguaje gráfico que se usa principalmente para visualizar, especificar, construir y documentar componentes de software y sistemas de software. El UML no es de uso exclusivo del software ya que con el mismo lenguaje se pueden documentar gráficamente todo tipo de sistemas, desde empresariales hasta de cualquier ingeniería. El UML permite el modelado del sistema completo como modelo en diferentes diagramas, que pueden ser de diferentes tipos. Existen otras técnicas de ingeniería del software, como el método Jackson basado en la estructura de los datos, pero se ha elegido esta por la simplicidad que tiene el representar a los diferentes elementos que componen la organización real con objetos de software de parecida estructura interna.

El UML no interviene solo en la fase de diseño de software, a la hora de establecer la clases, objetos, relaciones y contenidos, sino que a través de los diferentes tipos de diagramas también se usa en el análisis, con los diagramas de casos de uso, y en la implementación, si el software usado es el apropiado permite la generación de las clases para facilitar la tarea del programados. Aunque este último caso no sea propiamente UML hoy en día casi todas las herramientas de ingeniería de software propietarias o libres proveen esta característica por defecto.

- Los tipos de diagramas que componen el modelo software son los siguientes:
- Diagramas de Casos de Uso, donde se describen a los diferentes actores y casos de uso, funcionalidades muy concretas del sistema que pueden formar parte de un proceso más largo, que entran en contacto con el sistema modelado y que mensajes se transmiten entre ellos.
- Diagramas de colaboración y de secuencia, que definen las relaciones entre los distintos elementos del programa y los actores. Con esto quiero decir que establecen de manera formal a que llamadas de los usuarios van a responder y como los diferentes módulos del programa. En este caso interviene el factor temporal y cada caso de uso debería tener una secuencia para su realización. Hasta este punto estamos trabajando en el ámbito del análisis, a partir de aquí se trata el diseño del modelo.
- Diagramas de clases, que consisten en la representación de los elementos de la organización como clases, conjuntos de estructuras de datos con instrucciones para su manejo interno.

Se trabaja con diferentes diagramas según sean necesarios. Puede que en un modelo no haya diagrama de casos de usos (algunas referencias bibliográficas desaconsejan su uso para el modelado de software orientado a objetos), y puede que tenga múltiples diagramas de un mismo tipo. Para que el modelo sea correcto todos los diagramas de un mismo tipo podrían transformarse en uno solo que sería el diagrama del modelo. En este diagrama no habría funciones de clases ni clases duplicadas.

Con un buen modelado en todos los aspectos tendremos una gran base para construir un programa sólido y coherente tanto con las necesidades de los usuarios, como con una estructura interna fácil de mantener y ampliar.

2.4 Metodología de análisis

Para la realización del análisis de las posibles necesidades organizativas, técnicas y económicas se han realizado una serie con Pedro Lorenzo. Actualmente es el Director Técnico del Club Deportivo Atlético Joaquín Blume – Caja-Segovia. Es Licenciado en Ciencias de la Actividad Física y del Deporte por la Universidad Europea de Madrid con la especialización en Atletismo y Golf, Entrenador Nacional de Atletismo por la Real Federación Española de Atletismo y monitor de diferentes disciplinas de ocio y deportivas como Natación, Tiempo Libre, Piragüismo y Polideportivo. Además ha realizado cursos de especialización en Dietética y Nutrición, Gestión de Instalaciones Deportivas, Esquí, Fútbol, Fútbol Sala, Aquatic Fitness y Fisiología del Ejercicio y a publicado artículos en diferentes revistas especializadas. Por todo ello, y por su dilatada experiencia laboral en diferentes clubes deportivos como el Real Club de Golf Puerta de Hierro, creo que es una persona apropiada y que conoce suficientemente los entresijos de este dominio que vamos a abarcar en el proyecto.

Las entrevistas se realizaron en tres fases diferentes.

En la primera se definieron las diferentes necesidades del personal técnico y deportistas a nivel deportivo. En esta fase se definieron los diferentes documentos que intervienen en la planificación deportiva, sus fases y periodos, los datos necesarios en cada periodización y la información que el sujeto debe transmitir al entrenador en forma de realimentación para que este pueda adaptar los futuros entrenamientos.

En la segunda se abordaron los problemas de organización interna del club, desde la masa social hasta los diferentes grupos organizativos: categorías, deportes, grupos de entrenamiento. También se trataron en esta fase todo el tema de instalaciones deportivas, siempre desde un punto de vista de club y no de empresa dedicada a la explotación.

En el último punto se abordó la gestión económica de un club. Esto era relativamente sencillo en comparación con otro tipo de empresas, pero se requirió una completa integración con la organización del club para la gestión correcta del cobro y pago de fichas deportivas, solicitud de subvenciones, pago de los medios de transporte, personal interno del club, y colaboradores ocasionales.

Tras estas entrevistas se va a seguir una metodología de análisis orientada a objetos. La primera parte consistirá en una especificación de requisitos, donde se van a recoger todos los requisitos funcionales y no funcionales que va tener la aplicación.

Sobre estos requisitos se estudiarán las clases y objetos que van a intervenir en ella. A este proceso se le llama identificación de las clases del dominio. Para este proceso se utilizará la técnica de la identificación de sustantivos. Consiste en extraer los nombres y grupos nominales de la descripción del problema y las entrevistas con los usuarios, expertos en el dominio y clientes. Sobre estos nombres obtenidos se realizará una criba donde se eliminarán los que estén duplicados en

significado o que representen lo mismo, los que no intervengan realmente como elementos del problema a resolver, los elementos propios del sistema software que vaya a tratar el problema, pantalla, teclado, etcétera, y los que indiquen frecuencia temporal. Además de estos que se eliminan otros tendrán que transformarse pues pueden ser atributos de una clase, en vez de una clase propiamente dicha, representar una acción, que en el sistema se convertirán en métodos, o un estado, que será un valor de un atributo.

2.5 Glosario de términos deportivos.

Durante el transcurso de las entrevistas surgieron términos técnicos del área del deporte y de la salud que vamos a definir en este punto para que el resto del texto de análisis y posterior diseño y desarrollo se entiendan de la mejor manera posible.

Temporada

Objetivos

Periodización

Ejercicio

Espirometría

Test de Esfuerzo

Consumo máximo de oxígeno

Marca

Test

Carga

Intensidad

Sesión

Tipos de trabajo (fuerza resistencia, anaeróbico, aeróbico)

Microciclo

Mesociclo

Macro ciclo

2.6 Análisis de Requisitos

2.6.1 Requisitos Funcionales

En esta sección vamos a describir las necesidades del cliente que tiene que cumplir el programa en su manera de funcionar. El primer detalle a destacar es que será un software que podrán usar todos los miembros de la organización. Cada uno tendrá acceso a realizar diferentes tareas de cada área según sea su interés real en las mismas. Los diferentes funcionamientos de cada área vienen dados por el cliente.

2.6.1.1 Gestión deportiva.

El programa debe soportar diferentes tipos de personas encargadas de la gestión deportiva. Cada una de estas personas tendrá acceso a ver, añadir, modificar y borrar ciertos datos de las otras personas que intervienen en la gestión. De la misma manera existen determinados tipos de documentos diferenciados que también pueden ser objeto de uso de diferentes personas.

Documentos necesarios para la correcta gestión deportiva:

1. Documento de datos personales.
2. Documento de datos del deportista.
3. Documento de objetivos a conseguir durante la temporada presente.
4. Documento de periodización de la temporada. Donde se describen por semanas los diferentes aspectos a trabajar durante la misma, así como el número de kilómetros, kilos o repeticiones trabajados en cada semana.
5. Documento de progresión. Es el documento que describe los ejercicios que hay que hacer dependiendo del aspecto a trabajar y la intensidad deseada para el mismo.
6. Documento de programación. Es el que nos describe los ejercicios a realizar en cada día de la semana. A través de este documento se produce la realimentación hacia el entrenador, pues tiene campos abiertos para que el sujeto entrenado describa la realización del ejercicio. En este documento también aparecen los gráficos de las zonas de trabajo físico realizadas.
7. Documento de testings. Es un documento donde se acumulan los diferentes test físicos, como realizarlos y las condiciones necesarias.
8. Documento de marcas. Contienen todas las marcas y resultados de la vida deportiva del deportista entrenado, así como los resultados de los test.
9. Documento de datos médicos. Contiene informes médicos de los diferentes sujetos entrenados. Los informes vendrán dados por la necesidad del entrenador de cierto conocimiento a la hora de planificar la temporada. Así se utilizarán plantillas de espirometría, analítica completa, test de esfuerzo, consumo máximo de oxígeno. Al ser estos datos médicos y de carácter personal será el atleta quien decida para quien estarán disponibles y el entrenador no tiene porque no verlos si así lo desea el deportista.
10. Documento de tablas de ejercicios. Tiene la información de diferentes ejercicios que van a realizarse conjuntamente en una sesión para realizar un trabajo de un tipo concreto.

Además de los documentos citados podrían añadir otros documentos médicos.

En la organización, o club habrá varios entrenadores que entrenarán a cada uno de sus deportistas. Existen documentos propios por entrenador que son iguales independientemente del atleta entrenado: 5. Documento de progresión, 7. Documento de Testings y 10. Documento de tablas de ejercicios. Existen documentos que tiene cada atleta personalizado: 1. Documento de datos personales, 2. Documento de datos del deportista, 3. Documento de objetivos, 4. Documento de periodización, 6. Documento de Programación, 8. Documento de marcas y 9. Documento de datos médicos.

Estos últimos datos, al ser de carácter personal médico del deportista, tendrían una visibilidad exclusiva para el propietario y necesitarían de un permiso expreso establecido para que pudieran ser vistos por otras personas, como por ejemplo el entrenador.

Vamos a proceder a describir las informaciones completas que contienen estos

documentos:

2.6.1.1.1 Documento de datos personales:

Al crear un nuevo deportista, socio, o trabajador se generará este documento que contiene todos los datos como NIF, nombre, apellidos, dirección, teléfono, número de cuenta bancaria y el tipo o tipos de papeles que representa en el club (socio, entrenador, deportista, trabajador, director técnico, presidente, secretario, vicepresidente, tesorero, vocal).

Los entrenadores podrán añadir o editar personas de tipo deportista, el director técnico a entrenadores y trabajadores del club, aparte de deportistas, el secretario a los socios.

2.6.1.1.2 Documento de datos del deportista:

En este documento se describe el grupo al que pertenece, el deportista, su entrenador. El propio deportista podrá cambiar su documento.

2.6.1.1.3 Documentos de objetivos:

Un documento de objetivos por temporada. Contiene un listado de objetivos principales, uno de objetivos operativos y uno de objetivos secundarios. El entrenador define el periodo que dura la temporada. Puede añadir, editar y borrar el documento de objetivos que tiene cada atleta para esa temporada.

2.6.1.1.4 Documento de periodización:

Consiste en una planificación total de la temporada, luego existe uno diferente por cada deportista y temporada. Contiene:

- Nombre de la temporada
- Tipo de progresión
- Listado de Macrociclos, Periodos, Mesociclos y Microciclos .
- Kilómetros, toneladas, ejercicios por microciclo.
- Fechas correspondientes a cada microciclo.
- Listado de ejercicios diarios del documento de progresión, con el nivel correspondiente del trabajo en cada microciclo.
- Listado de tests (del documento de tests) que se realizarán coincidiendo en los microciclos.

El entrenador del atleta es el que edita este documento. No está visible para el atleta normalmente, si bien el entrenador puede hacerlo visible si lo desea.

2.6.1.1.5 Documento de progresión

Listado de tipos de entrenamientos con diferentes niveles. En cada nivel de cada tipo de ejercicio se presentará una ejercicio diario a desarrollar. Este ejercicio diario se mostrará en el documento de planificación en la lista de ejercicios a desarrollar y puede ser modificado por el entrenador en ese documento. Este documento en la teoría varía muy poco. En el sistema cada entrenador tiene un solo documento de progresión por deporte o especialidad. El entrenador puede

añadir, editar y borrar tipos de entrenamientos, deportes y especialidades, y añadir editar y borrar ejercicio diario para cada nivel de cada tipo de entrenamiento.

2.6.1.1.6 Documento de programación

Es el documento donde se tiene el entrenamiento que tiene que hacer cada día el deportista. Está estructurado en microciclos, y en cada día que dure el microciclo se pueden poner hasta 3 sesiones de entrenamiento, que si bien no es lo usual, si que pueden llegar a alcanzarse en deportistas de muy alto nivel.

Cada microciclo tiene:

- Las cantidades de ejercicio (kilómetros, kilos, ...) de la periodización
- Las cantidades de ejercicio realizados según los datos de realimentación del sujeto entrenado.
- Las horas de entrenamiento.
- El ejercicio diario que tiene que realizarse según las periodización
- El ejercicio que al buen criterio del entrenador deba realizarse en vez de el propuesto por la tabla.
- Entrada de realimentación para que el atleta ofrezca resultados de su entrenamiento para que el técnico pueda comprobar su estado y evolución. Consiste en que el deportista introduzca en una tabla el tiempo dedicado a cada tipo de trabajo de cada sesión.
- El índice de carga, que se calculará a partir de la información que introduce el deportista y que depende de la intensidad del ejercicio.

Según lo indicado anteriormente es el entrenador el que introduce los datos posteriormente a que el programa haya presentado unos ejercicios teóricos según la periodización.

2.6.1.1.7 Documento de testing

Contiene la información de los diferentes test que realiza cada entrenador Para cada test se guarda: el código del test, la capacidad que mide, el nombre del test, la medida en que se realiza, porque se realiza, material necesario, otros. Existe uno por entrenador. El entrenador puede añadir, editar y borrar test de su documento únicamente.

2.6.1.1.8 Documento de estadísticas en competición

Hay un documento de estadísticas del deportistas por temporada. Se recogen la fecha, la prueba, y el resultado objetivo de su realización. En el resultado objetivo pueden ser tanto los goles marcados, como los pases bien o mal dados en los deportes de balón, o las marcas de una prueba así como los diferentes pasos por las distancias o el puesto. También se pueden incluir los test realizados con sus resultados.

2.6.1.1.9 Documento de datos médicos

El entrenador podrá solicitar a sus deportistas los datos médicos que considere necesarios. Estos datos estarán en un fichero encriptado del que solo se podrán extraer con una clave privada de la persona que tenga permiso para verlos, y que el deportista haya concedido el permiso. Este documento consistirá es el conjunto de informes médicos de un deportista del club. Cada tipo de informe tiene diferentes campos de datos médicos. Un deportista puede crear,editar y borrar sus

informes médicos. Los documentos médicos serán estándares para toda la organización y los 4 básicos que vendrán configurados de serie son: Espirometría, Test de Esfuerzo, Consumo máximo de oxígeno y analítica sanguínea y de orina completa.

El sujeto analizado es el propietario del documento.

Puede permitir y denegar acceso al documento, al informe o a alguno de sus datos concretos a cualquiera del personal técnico del club.

2.6.1.1.10 Documento de tablas de ejercicios

Contiene una lista de conjuntos de ejercicios. Cada conjunto de ejercicios es una tabla. Cada ejercicio tendrá un nombre, un músculo que trabajar, una descripción, un dibujo explicativo y una zona de trabajo.

Tiene la posibilidad de añadir, borrar y editar los ejercicios diferentes para las tablas.

Se puede generar una tabla de ejercicios idéntica a una ya existente con nombre diferente.

Se pueden añadir y quitar ejercicios de una tabla individualmente o por músculo o zona de trabajo.

El orden de realización de la tabla importa.

2.6.1.2 Gestión Logística

Dentro de las actividades de cualquier club deportivo se realizan operaciones de logística, aprovisionamiento de material, contratación de servicios, organización de viajes. El tener un inventario de material deportivo disponible, un listado de proveedores por servicios ofrecidos y el control de las ubicaciones de los distintos materiales clasificado por almacenes resulta en un mejor aprovechamiento del tiempo en entrenamientos, preparación y dedicación a otros aspectos deportivos mejor que organizativos.

Los requisitos funcionales de área logística son los siguientes:

- Creación, edición y borrado de nuevos almacenes.
- Creación, edición y borrado de nuevos tipos de artículos.
- Creación edición y borrado de nuevos artículos. Esto artículos podrán ser únicos (una jabalina marca ACME de 750g de entrenamiento del año 2008), en los que si hay dos iguales se diferenciarán como artículo por la singularidad de los mismos, o seriados (15 camisetas de competición de la talla M).
- Creación, edición y borrado de nuevos proveedores.
- Creación, edición y borrado de nuevos tipos de solicitudes de servicios.
- Creación, edición y borrado de nuevas solicitudes de servicios.
- Creación, edición y borrado de nuevos tipos de eventos. Cada tipo de evento conllevará una serie de materiales y servicios necesarios para su realización. Al crear el evento de un tipo estos se mostrarán como pendientes hasta que se marque que se ha realizado. En servicios como viajes, inscripciones, etc, se requerirán las condiciones del servicio como horas, deportistas inscritos, colaboradores, etc, para conseguir una documentación completa de la organización del evento. En los eventos que se organicen a terceros se especificará el precio por cada uno de los servicios ofrecidos por el club.

- Creación, edición y borrado de nuevos eventos.

- En cualquiera de los materiales y servicios se podrá generar factura por la compra que deberá poder pasar al sistema de gestión económica automáticamente. Se crearán cuentas de gastos e ingresos asociadas al evento si este lo quiere (de ahí el marcar los diferentes tipos con sus características) y las facturas de ingresos y gastos harán referencia al mismo.

- En cualquiera de los eventos organizados se podrá emitir una factura de venta o recibo por los gastos ocasionados al club en la organización.

2.6.1.3 Gestión social

La estructura de un club deportivo comprende muchos aspectos. En el aspecto del club como una sociedad de personas se debe tener conocimiento del grupo humano que lo compone. Así como tenerlo bien clasificado y diferenciado quien es que. En asociaciones pequeñas una misma persona podrá ocupar múltiples puestos y será a la vez deportista, entrenador, socio, miembro de la junta directiva, llevará las cuentas y participará como colaborador en ciertos aspectos de eventos. Aunque en un club grande no suelen darse estas situaciones de forma tan exagerada si que pueden verse puestos de entrenador y director técnico de un área encargados a la misma persona. Así el programa deberá soportar esa multiplicidad de papeles para una misma persona.

Las necesidades del club en cuanto a la gestión interna de sus documentos, actas de reuniones y listados de socios también se reflejan en este área.

Los requisitos funcionales de la parte de gestión social son los siguientes:

- Añadir, editar y borrar tipos de documentos.
- Añadir, editar, anular y borrar documentos
- Añadir, editar y borrar tipos de personas que pueden formar parte del club.
- Añadir, editar, borrar y anular personas.
- Listar personas de múltiples maneras.
- Listar documentos de múltiples maneras
- Añadir y editar temas de actas de reuniones.
- Añadir y editar el contenido de un tema del acta de una reunión.
- Añadir, editar y borrar convocatorias de reuniones por tipos de personas.
- Añadir, editar y borrar plantillas de convocatorias

2.6.1.4 Gestión Económica.

La gestión económica de un club estará dirigida por el tesorero, si bien habrá personas que tengan acceso a la presentación de gastos según sus puestos en las direcciones de las diferentes secciones del club. Así de esta manera se presentan los siguientes requerimientos funcionales en el área de gestión económica:

- Emisión de facturas, por los servicios prestados por el club, publicidad en los eventos realizados, cobro de premios.
- Recepción de facturas de los proveedores.

- Cobro de facturas por diferentes métodos. Se deben contemplar por defecto al menos el recibo, efectivo y transferencia bancaria.

- Gestión de medios de cobro, que permita al gerente económico añadir, quitar y editar las diferentes formas de cobro a los clientes.

- Pago de facturas de proveedores por los siguientes métodos: efectivo, transferencia bancaria, recibo en banco o talón bancario.

- Gestión de medios de pago. Al igual que los de cobro para permitir la inclusión de nuevos métodos o modificar los ya existentes.

- Listados y relaciones de facturas pendientes de pago y de cobro.

- Contabilidad formal del club.

- Informe de ingresos y gastos por conceptos agrupados.

- Relación con listado de socios, deportistas y trabajadores del club para comprobar los pagos y cobros pendientes referentes a los mismos. Estos pagos y cobros no tienen porque ser facturas. Los pagos pueden ser los sueldos de los entrenadores, fisioterapeutas, médicos, monitores, técnicos y personal que organice los eventos, las fichas de los atletas destacados, las subvenciones para material deportivo, pago de desplazamientos a campeonatos y competiciones, premios de carreras por equipos o cobradas a través del club. Cada uno de estos pagos deberá ser autorizado por una persona diferente en el club, y sin su autorización no se podrá generar la orden del pago al afectado. Los cobros a estas personas pueden ser por las fichas federativas, material deportivo uniformado del club, inscripciones a eventos.

- Calendario de subvenciones para asociaciones, documentación que presentar y plazos. También las firmas de los miembros de la junta directiva necesarias para la presentación, ya que cada entidad necesita una diferente: tesorero, presidente, secretario.

- Acceso al programa a organizadores de eventos para que puedan hacer asientos contables en cuentas relacionadas con la organización de su evento. Se trataría de agilizar el tratamiento de la información económica para evitar retrasos en pagos y cobros.

- Cierres y aperturas anuales para hacer balance.

- Módulo de realización de presupuestos, tanto generales anuales como de eventos concretos. En base a datos anteriores y con las estimaciones de receso o crecimiento en cada una de las áreas de gastos e ingresos.

2.6.2 Requisitos no funcionales

Para el análisis se han determinado las siguientes condiciones en las que se va a mover la aplicación:

- La aplicación resultante deberá ser accesible a todos los usuarios a través de Internet. Esta disponibilidad podrá ser va web o por descarga directa de un programa que permita el acceso a los datos pertinentes del servidor central.

- Tendrá un módulo que funcione desde terminales móviles, teléfonos, agendas electrónicas, reproductores musicales digitales, etcétera, para la toma de tiempos, resultados y variación de entrenamientos durante el trabajo de campo para los entrenadores y técnicos. Las funcionalidades de este módulo serán las mínimas posibles para permitir un trabajo con varios deportistas o grupos.

- Tendrá un entorno gráfico de usuario de fácil manejo y preparado para una adaptación a

tecnologías de accesibilidad avanzada (personas con problemas de vista, oído o movilidad en sus extremidades).

- Soportará la internacionalización. En la base tendrá 3 idiomas, Español, Inglés y Francés, y tendrá la posibilidad de añadir idiomas sin necesidad de recompilar el código. El idioma por defecto de funcionamiento será el Inglés, pero el software detectará del sistema operativo el idioma de la persona que está trabajando y en el arranque lo adaptará. Permitirá la elección manual de idioma.

2.7 Identificación de las clases de análisis

Con los datos que tenemos hasta ahora vamos a identificar las clases del programa. Primero se muestra un listado de candidatas que luego se filtrará. Estas clases que identificamos ahora son las que componen el dominio del problema a resolver, que no tienen porque ser clases efectivas en el diseño y programación de software. Se corresponden a las clases de objetos que intervienen en todo el ámbito del problema en el mundo real. Así mismo en la fase de diseño se crearán nuevas clases para el intercambio de mensajes, las relaciones, los listados y el trabajo interno de software. Estas últimas clases de objetos se diseñan pensando en la solución software y en esta fase del proyecto estamos analizando el problema.

2.7.1 Clases candidatas

2.7.1.1 Del análisis de la gestión deportiva.

Documento de datos personales, deportista, socio, trabajador, NIF, nombre apellidos dirección, teléfono, número de cuenta bancaria, entrenador, presidente, vicepresidente, secretario, vocal, tesorero, director técnico, personal técnico, director de área, coordinador, secretario técnico, documento de datos del deportista, grupo de entrenamiento, categoría, equipo, documento de objetivos, objetivo principal, objetivo secundario, objetivo, temporada, documento de periodización, tipo de progresión, macrociclo, periodo, mesociclo, microciclo, sesión, ejercicio diario, documento de progresión, tipo de entrenamiento, nivel de trabajo, test, documento de programación, cantidad de ejercicio, cantidad de ejercicio de la periodización, cantidad de ejercicio realizado, horas de entrenamiento, deporte, especialidad, entrada de realimentación, índice de carga (tipos), índice de carga, documento de testing, código de test, capacidad física básica, medidas de test, Documento de estadísticas en competiciones, fecha, prueba, resultado, documento de datos médicos, tipo de informe, informe médico, dato médico, documento de tablas de ejercicios, tabla de ejercicios, músculo, dibujo explicativo, zona de trabajo, ejercicio.

2.7.1.2 Del análisis de la gestión logística.

Inventario de material deportivo, proveedor, ubicación, almacén, tipo de artículo, artículo, servicio, tipo de servicio, tipo de evento, evento, factura de compra, recibos recibidos, factura de venta, recibos emitidos.

2.7.1.3 Del análisis de la gestión social.

Miembro de la junta directiva, colaborador, área, documento interno, acta, listado de socios, tema de junta, tipo de persona, convocatoria, plantilla de convocatoria.

2.7.1.4 Del análisis de la gestión económica

Presentación de gastos, aprobación de gastos, factura emitida, servicios prestados, publicidad en eventos, cobro de premios, cobro de factura, medios de cobro, pago de facturas, medios de pago, efectivo, transferencia bancaria, emisión de recibos, recepción de recibos, talón bancario, listado de facturas pendientes de pago, listado de facturas pendientes de cobro, estado de factura emitida (pro-forma, pendiente de cobro, cobrada), estado de factura recibida (pro-forma, pendiente de pago, pagada), cuenta, sub-cuenta, tipo de cuenta (ingreso, gasto, activo, pasivo, banco, caja,), listado de cuentas, informe, informe ingresos – gastos, informe de pagos y cobros internos, calendario, subvención, documento requerido de subvención, plazo, firma de directivo, cuentas para organizadores, cierre y apertura anual, presupuesto general anual, presupuesto de área o sectorial, presupuesto de evento.

2.7.2 Clases del dominio del problema

2.7.2.1 Del análisis de la gestión deportiva

Del listado de clases candidatas hay algunas que por la propia definición del problema no plantean ninguna duda de son clases efectivas en la solución del problema. Las que me han planteado la duda son:

- NIF: es un tipo de documento personal que está formado por una serie de números y una letra. Si bien cabría contemplar otros tipos de identificadores, pues es común entre los clubes contar con deportistas extranjeros. El hecho de si es clase o no viene definido por la capacidad para realizar operaciones. Si en la resolución del problema se va a necesitar comprobar la veracidad y la corrección del número introducido y esa comprobación la va a hacer una operación propia del documento entonces el Documento identificativo tendría el estatus de clase. Digo Documento identificativo, que ya no es NIF, por la variedad de documentos que pueden admitirse por la situación de los extranjeros, y aprovechando la coyuntura de la comprobación se podría utilizar para hacer lo mismo con los datos empresariales de terceros.

- Nombre, apellidos: en este caso los consideraré atributos de persona, como clase base para todos los tipos de persona que habrá después en el club.

- Dirección: Aquí viene la duda de si se va a trabajar la dirección como una serie de caracteres o va a contener más campos como código postal (comprobable al igual que NIF para hallar la provincia correcta), de si cada persona va a poder tener más de una dirección y si esta misma dirección va a servir para otras entidades como empresas. Como estas características se dan (las empresas tienen una o varias direcciones y va a reutilizarse), entonces consideramos a dirección como clase, y añadimos lista de direcciones también, pues a cada persona o tercero le corresponderá una lista de direcciones.

- Teléfono, numero de cuenta bancaria: al igual que dirección, por poder verificar la correcta disposición de dígitos de control, prefijos o poder enlazar directamente con programas de llamada (asterisk, Skype, Ekiga con protocolo SIP, etcétera) consideramos ambos datos clases, puesto que pueden tener operaciones internas.

- Deportista, socio, trabajador y el resto de diferentes puestos en la infraestructura del club. Se podría pensar que son objetos de la clase persona con diferentes características. En esta fase de análisis son clases de personas que intervienen en la organización y en la fase de diseño se decidirá la implementación apropiada para ellos estudiando cada caso particular de individualidad en instanciación, o por áreas.

- Objetivo puede ser perfectamente una cadena de texto que pertenezca a uno de los objetivos principales o secundarios, la problemática es que a lo largo de la vida deportiva de los deportistas los objetivos se repiten, o se comparten por varios de ellos (y más aún en los deportes de equipo). Visto de esta manera los objetivos deberían poderse almacenar y guardar de una temporada para otra y así constituir una clase efectiva en la resolución del problema. Además propongo que sea posible listarles a la hora de seleccionar, pues es probable que un objetivo haya sido propuesto antes una vez se haya manejado el programa durante un tiempo. Así se añadiría la clase lista de objetivos.

- Ejercicio diario se refiere al ejercicio realizado en una sesión por lo que estaría mejor denominado “ejercicio de la sesión”. Constituiría una clase en la que se podrían incorporar ejercicios predeterminados en el documento de progresión y otros que el entrenador del deportista considere oportunos.

- Tipo de entrenamiento y nivel de trabajo son dos conceptos cruzados. Para un tipo de trabajo aeróbico de nivel 1 se corresponde una carrera continua de 15 minutos con una recuperación andando a la mitad del tiempo de 5 minutos, y para ese mismo tipo de trabajo en un nivel superior puede hacer una hora de carrera continua lenta a un ritmo determinado que una persona sin preparación física no consideraría, ni por asomo, lento. Con esto quiero decir que la clase en si vendría representada por cada relación tipo de entrenamiento y nivel, o bien cada tipo de entrenamiento tendría el carácter de clase con una serie de “ejercicio de la sesión” asociados a cada nivel. Teniendo en cuenta que hay que definir en algún momento las características del tipo de entrenamiento opto por declarar clase al tipo de entrenamiento, con los diferentes niveles asociados como atributos de esta y que corresponden a ejercicio de la sesión.

- Fecha. No aparece en los datos personales pero debería aparecer con la fecha de nacimiento. En la fase de análisis es un atributo de la clase persona (fecha de nacimiento) y de la clase prueba o test (fecha de la prueba, competición o test). En la fase de diseño probablemente se convierta en una clase, aunque venga dado por el lenguaje de programación en su especificación, porque necesitará de métodos y condiciones para que la fecha sea válida.

2.7.2.2 Del análisis de la gestión logística

En la gestión logística todos los conceptos que aparecen parecen apropiados para constituir una clase, si bien en las facturas y recibos me faltarían las líneas con las que se calcula el importe final.

2.7.2.3 Del análisis de la gestión social

En la gestión social igual que en la logística se consideran correctas todas las definiciones de clases candidatas.

2.7.2.4 De análisis de la gestión económica

Todas las clases candidatas se consideran efectivas.

3. Planificación

Detalles organizativos a la hora de resolver el problema.

3.1 Desarrollo temporal

El tiempo estimado de desarrollo de esta aplicación son 9 meses realizada por una sola persona con una dedicación de 7 horas al día.

La primera parte de este tiempo (1 meses) corresponde a la realización de estudios previos de mercado y de funcionamiento de las organizaciones para las que está destinada Gesport. Durante este periodo se han entrevistado a directivos, entrenadores, deportistas y trabajadores de clubes de atletismo y empresas de organización de actividades deportivas. Se han estudiado las necesidades anteriormente detalladas.

La segunda fase (3 meses) ha consistido en plasmar todo esa información obtenida en la redacción del análisis, en la creación de los diferentes gráficos y esquemas que provee el Lenguaje de Modelado Unificado para facilitar una cómoda codificación de la aplicación. Se han seguido manteniendo las entrevistas para corregir los posibles errores en los conceptos iniciales. En esta parte se han aportado al proyecto todos los diagramas de Clases y Actividades. Aparte de los gráficos propiamente dichos se han desarrollado las explicaciones de los términos más complejos de temas deportivos y de las relaciones en los diagramas que pudieran no llegarse a entender. Se ha hecho el estudio de las soluciones posibles para los problemas planteados y se han tomado las decisiones relativas a metodología y tecnología de clientes y servidores a usar. En esta etapa también se han diseñado las pruebas a las que se habría de someter cada parte de la aplicación y la aplicación en general.

La tercera fase (4 meses) ha consistido en la implementación y escritura de todos los códigos fuentes necesarios, búsqueda de bibliotecas que ya proveyeran de funciones necesarias (evitar inventar la rueda, que ya está inventada), diseño de los interfaces gráficos de usuario y pruebas de conexiones entre bases de datos, servidor de aplicaciones e interfaces de usuarios. Es la parte más larga y tediosa del proyecto.

El último mes se ha dedicado a completar la presentación de diapositivas, los manuales de usuario y administrador, terminar detalles y flecos sueltos, y probar más combinaciones posibles que puedan generar error para evitar efectos indeseados en la presentación ante el jurado.

Durante todo este periodo de tiempo el alumno ha dedicado una media de 2 horas al día, todos los días de la semana y los fines de semana y festivos 4 o 5 horas.

3.2 Ciclo de vida y metodología

Se ha utilizado un desarrollo en espiral, porque, aunque se estudió la posibilidad de realizarlo en cascada por que se tenía conocimiento de todas las áreas del proyecto desde el estado inicial, he creído más apropiado hacer un ciclo de la espiral con cada funcionalidad añadida al programa. Así de esta manera tendremos versiones funcionales de partes del programa en cada ciclo que podrán ser utilizadas por los miembros del club mientras se sigue el desarrollo de otras áreas.

El orden a seguir en esta metodología ha sido el siguiente:

- Sistema gestor de entrenamientos para entrenador.
- Aportes del sujeto entrenado.
- Organización de desplazamientos a competiciones.
- Organización de almacenes y materiales.

- Gestión económica.
- Gestión de socios.
- Gestión de actas de reuniones, convocatorias y documentos internos
- Gestión de datos médicos con datos cifrados.

3.3 El diseño del interfaz de usuario

Este programa tiene como objetivo el facilitar la gestión de los aspectos deportivos, organizativos y económicos a un club deportivo. Así pues el manejo del mismo tiene que ser lo más sencillo e intuitivo posible. Para que resulte sencillo se utilizará un sistema de ventanas gráfico como puede ser el de Windows, Qt, Gtk, o alguno de los múltiples posibles para Java. La elección del sistema de ventanas vendrá determinada por sistema base que soportará la aplicación.

Para que sea intuitivo se estudiarán las operaciones de trabajo en el campo de los profesionales de forma que la manera de introducir y obtener los datos sea parecida a la que se realiza en vivo y en directo en la instalación deportiva.

Además se tendrán en cuenta las recomendaciones de usabilidad recogidas en las diferentes organizaciones que tratan sobre el tema, como en las normas de GUI de Gnome Foundation, o las recomendaciones para el diseño de aplicaciones Java.

3.4 Presupuesto y financiación

Para el desarrollo de esta aplicación se ha estimado un coste por hora de 45 €. Divido en las diferentes fases de la creación del programa se ha contado con el siguiente presupuesto

TODO

- Horas de análisis:
 - *Entrevistas con cliente:
 - *Análisis de los requisitos:
- Horas de diseño del software:
- Horas de diseño de pruebas:
- Horas de diseño de la interfaz:
- Horas de programación:
- Horas de pruebas:
- Hardware necesario para pruebas:
 - * PDA
 - * Ordenador personal para el desarrollo de la aplicación
 - * Servidor para pruebas
 - * Servidor de producción para versiones finales en funcionamiento
- Servidores de control de versiones, copia de seguridad y distribución.
-

El proyecto, según la normativa de la Universidad de Valladolid, será licenciado como software libre al amparo de la GPL v. 3. El tiempo de desarrollo del proyecto se considera una inversión para dar un posterior mantenimiento y servicio personalizado a clubes, gimnasios, entrenadores personales. De esta manera la financiación se obtendrá de la repercusión del coste del mantenimiento del software en los potenciales clientes que puedan hacer uso de el, así como de las ventas e instalaciones de hardware en los locales de los mismos. Se prevé una amortización del coste de la inversión en TODO.

4. Diseño de la solución final

Elección de la plataforma de servidor, lenguaje de programación, diagramas de clases, de secuencia y colaboración, algoritmos de cálculos de índices de trabajo físico y relaciones entre las diferentes áreas del programa.

4.1 Elección de la plataforma.

4.1.1 Soluciones multiplataforma

Uno de los requisitos no funcionales de la solución final es que el programa del cliente funcionase en varias plataformas. El motivo de esto era que en una organización concreta suele usarse la misma plataforma en todos los ordenadores, pero la aplicación debe ser exportable a futuros posibles clientes. Si bien la mayoría de los usuarios del posible mercado en el que nos encontramos usa sistemas operativos Microsoft Windows ®, en el caso concreto planteado del Club de Atletismo Joaquín Blume existen al menos dos puestos en los que el sistema operativo es Linux o MacOS. Ante este problema se han barajado las siguientes soluciones:

4.1.1.1 Lenguajes mono-plataforma con interfaz de usuario común

Consiste en crear la aplicación con un interfaz de usuario basado en librerías libres utilizables en cualquiera de estos sistemas operativos de escritorio. El lenguaje de base es mono-plataforma, pero las librerías que usa son comunes a varios sistemas operativos, de manera que el mismo código fuente con diferente compilador (para cada sistema operativo) darían lugar al mismo programa.

El que sean libres viene dado por tres razones principalmente:

- El coste del uso de librerías propietarias.
- El soporte por una comunidad de usuarios y desarrolladores de estas librerías muy involucrados por su voluntariedad.
- Por último, y no menos importante, por el aporte de componentes al movimiento del software libre que puede generar el proyecto.

Esta posibles librerías de interfaz de usuario son GTK (Gimp Tool Kit) o Qt.

- GTK es una implementación de un sistema de ventanas en las que se basa el entorno de ventanas para sistemas UNIX Gnome. Comenzó su desarrollo por los creadores de Gimp, un software de edición gráfica de mapa de bits para sistemas UNIX. Hoy en día tiene capacidades parecidas a Adobe Photoshop ® gracias a las extensiones programables en diferentes lenguajes, aunque con una velocidad de ejecución más lenta debido a que estas extensiones se hacen con lenguajes interpretados. La base de la programación para GTK es C, aunque tiene extensiones para C++ y últimamente ha salido, a través del proyecto Mono, para C#. Esta última manera de programas se considerará en las plataformas interpretadas, puesto que, al igual que con Java, una misma compilación sirve para los tres sistemas operativos.

- Qt es una implementación de una empresa privada llamada Trolltech. En sus inicios tenía licencia privativa, pero su uso para el desarrollo del entorno de ventanas KDE, también para sistemas UNIX y actualmente funcionando en Windows y en Mac sustituyendo a los sistemas por defecto de estos sistemas operativos, hizo que al final Trolltech liberara el código con un doble licenciamiento: si la aplicación que usa Qt es libre el licenciamiento es libre, si la aplicación tiene una licencia no libre entonces hay que pagar para usar las librerías. El lenguaje que se usa con Qt es C++, que permite orientación a objetos y supone una ventaja en el desarrollo sobre GTK.

Es posible que existan otras alternativas, pero estas son las más conocidas, más usadas y con más soporte de la comunidad. Existe la posibilidad de usar GTK+ que es GTK orientado a

objetos. El uso de estas tecnologías tiene ciertas ventajas e inconvenientes.

- Ventajas: Se basan en lenguajes compilados, lo que implica una mayor velocidad de ejecución, tienen un interfaz común en todos los sistemas operativos, lo que simplifica, tanto los manuales de usuario, como el aprendizaje de los usuarios,

- Desventajas: Al ser lenguaje compilado necesita una compilación por cada sistema operativo destino. GTK en MacOS solo funciona con el servidor X de UNIX instalado. Qt 4, que es la versión que soporta correctamente todos los sistemas operativos de forma nativa, acaba de salir prácticamente y no hay mucha documentación ni experiencia sobre ella. Al tener la base de datos en un lugar diferente al programa, que se conectaría remotamente, el trabajo sobre ella será lento y en las transferencias con bloqueo se puede bloquear la aplicación. Las diferentes especificaciones de los lenguajes de programación para los diferentes compiladores, no es lo mismo el C de Visual Studio que el de GCC, harían necesarias modificaciones en la escritura del programa para transformarlo de una plataforma a otra. Si bien estas diferencias serían mínimas, harían necesaria una revisión casi completa del código escrito.

4.1.1.2 Interfaz web a través de navegador

Esta solución consiste en introducir el software en un servidor web al que tengan acceso los usuarios. Este servidor web ejecutaría los programas en algún lenguaje y ofrecería las respuestas al que consulta o trabaja. Los lenguajes que se soportarían en web son múltiples, ya sean compilados a través de CGI, PHP, Ruby on Rails, JSP o ASP. La base de datos que almacenaría los datos del programa se ejecutaría en el mismo servidor o en un servidor dedicado en la misma red, de manera que las consultas a la base de datos serían bastante rápidas.

Las opciones que se han contemplado para el desarrollo sobre servidor web son:

- PHP: lenguaje orientado a objetos de tipado débil. Se puede ejecutar sobre servidores Windows, Linux, MacOS, Solaris, y casi todos los posibles. Se necesita un servidor web sobre el que se ejecute el interprete PHP. Este puede ser Apache, Roxen, Microsoft Internet Information Server y muchos otros, entre ellos todos los que soporten CGI, pues se puede configurar como programa que funciona sobre CGI. Es un lenguaje de scripting que ha añadido el soporte de objetos desde la versión 4 y que desde la 5 el soporte es casi total.

- ASP: Actualmente el desarrollo de ASP se realiza en .NET en su mayoría. Los desarrollos en ASP basados en Visual Basic compilado y puesto en servidor ya no se usan mucho por la mayor facilidad de trabajo en .NET. El servidor debe ser necesariamente Windows salvo que se trabaje con el proyecto Mono, en cuyo caso se perdería la posibilidad de trabajar con Visual Studio, que sería una de las mayores ventajas que tendría, ya que tiene asistente para diseño de formularios web.

- JSP: Se trabaja prácticamente igual que en ASP. El único cambio es que ya no depende tanto del sistema operativo ya que la base del software es Java. El programa se ejecuta en un servidor de aplicaciones y sale a la web con un Apache Tomcat u otro servidor web sobre Java.

Al igual que las tecnologías de lenguajes mono-plataforma tienen sus ventajas e inconvenientes:

- Ventajas: Interfaz para el usuario reconocida de antemano, porque casi todos los usuarios tienen la experiencia de navegar por la web anteriormente. Mayor velocidad de conexión a las bases de datos desde el programa pues se encuentran localizadas en la misma red o servidor (dependiendo del caso del cliente). Experiencia del programador en este ámbito de varios años.

- Desventajas: Necesidad de conexión a la red en todo momento durante el manejo del

software. Dificultad en la creación de formularios por la falta de asistentes para ello. Informes y reportes manuales en HTML, peor presentación que con un asistente de reportes o en PDF, o si no, una gran dificultad para realizarlos en PDF con librerías no comerciales. Gran consumo de ancho de banda en las descargas de gráficas. Responsabilidad de cálculo casi exclusivamente en servidor, salvo control de introducción de información en formularios con javascript. Control de errores y excepciones muy rudimentario en la entrada de formularios.

4.1.1.3 Tecnologías multiplataforma

Las tecnologías multiplataforma son sistemas que se componen de un compilador del lenguaje y un interprete para cada plataforma destino. Están realizados de manera que una misma compilación sirva para todos los sistemas operativos y arquitecturas de procesadores, pues los mensajes con el sistema operativo los realiza el interprete. Así si en un programa usamos un la posibilidad de varios hilos de ejecución y el procesador y el sistema operativo lo soportan será la maquina virtual que interpreta el programa compilado la que cree las nuevas instancias de ejecución y no el programa por si mismo. De esta manera y con una fuente de datos común se puede trabajar en diferentes ordenadores, con diferentes sistemas operativos y procesadores con un mismo programa.

Las tecnologías multiplataforma disponibles actualmente en el mercado y que cuentan con una documentación abundante son Java y el proyecto Mono, una especificación libre del estándar ECMA para programación, equivalente a la tecnología .NET.

- Ventajas: Un mismo código fuente sirve para todos los sistemas operativos. Hay Plataformas de desarrollo lo suficientemente avanzadas como para integrar la programación de servidor, con el diseño de interfaces de usuario y ayuda en la búsqueda de funciones de librerías avanzadas. Java permite el almacenamiento de certificados de seguridad en sus almacenes, esto es una ventaja respecto a la tecnología de navegador, donde el almacén es compartido por varios usuarios si no se usa correctamente, y respecto a lenguajes compilados, donde el almacenamiento es externo o hay que implementarlo a mano. El interfaz está integrado en el sistema operativo destino, de manera que el usuario no percibe el funcionamiento de la máquina virtual y le resulta más fácil encontrar todo (con los lenguajes multiplataforma y librerías gráficas comunes la mayoría de las veces la ventana se adapta a la librería gráfica correspondiente, ignorando si en el sistema operativo los botones y las fuentes son de una determinada manera).

- Inconvenientes: Es ligeramente más lento en ejecución puesto que el programa funciona sobre una máquina virtual, otro programa, si bien con Java este inconveniente se está solucionando desde la apertura de su código al empezar a incluirlo en los núcleos de los sistemas operativos. La conexión de la base de datos tiene el mismo problema que los lenguajes mono-plataforma.

4.1.2 La gestión de los datos

Esta aplicación va a contener una gran cantidad de datos de diferentes deportistas. El diseñador de la aplicación no concibe el producto sin la potencia de un Sistema Gestor de Bases de Datos (SGBD). Para la elección del sistema gestor de bases de datos se han tenido en cuenta diferentes opciones, desde bases de datos orientadas a objetos hasta las soluciones comerciales más conocidas. A continuación vamos a pasar a describir las diferentes posibilidades:

4.1.2.1 Gestores de Bases de Datos Orientados a Objetos

Estos sistemas de bases de datos son una relativa novedad. A pesar de que las

metodologías orientados a objetos tienen ya cierta solera en el ámbito de la informática, el almacenamiento de la información como objetos no se ha desarrollado mucho. Hay que tener en cuenta que un objeto no es un conjunto de datos, sino los datos de ese conjunto, las operaciones que se pueden realizar con ellos, los mensajes que puede recibir para devolver información y los mensajes que puede recibir para alterar su estado. Con todos estos detalles se nos hace muy difícil la representación de un objeto en una base de datos.

Aún así el mundo de la informática ha evolucionado lo suficiente como para que exista este tipo de software en el mercado. Así tenemos la casi recién nacida JavaDB, DB4o (DataBase for Objects, del 2004), o con mucha más experiencia en el mercado, pero con carácter comercial Cache. Puesto que es un sistema aún bastante nuevo y, a pesar del estándar declarado OQL (Object Query Language) , aún las diferentes opciones son demasiado diferentes como para pensar en la posibilidad de una migración de una a otra y demasiado jóvenes como para pensar que en un futuro van a persistir. Es por esta razón, por algunos motivos tecnológicos (se necesitan ejecutar como una clase dentro del sistema, salvo Cache, con lo que tendríamos que programar un servidor de manera imprescindible, y no permiten conexión remota si no es programada en la aplicación a través de un socket) y algunos otros de carácter teórico (el lenguaje de consulta no está basado en una matemática formal como en los sistemas de bases de datos relacionales y relacional – objeto) que se ha descartado este tipo de base de datos en la realización del proyecto.

4.1.2.2 Gestores de Bases de Datos Relacionales

Las bases de datos más comunes en el mercado son las bases de datos relacionales. Son sistemas gestores de bases de datos basados en el modelo entidad-relación. Su principal característica es que los lenguajes de consulta, edición y administración se basan en casi todos los casos en el estándar SQL (Structured Query Language). Esto significa que en cualquier base de datos relacional se puede realizar una consulta estándar de búsqueda de datos, de creación o edición de datos, o de administración de permisos de usuarios de una manera única para todas ellas. Este estándar no se cumple al completo en todas las bases de datos, aunque es cierto que los cambios son debido a las mejoras y nuevas aplicaciones de las sistemas gestores que lo modifican.

En este grupo de bases de datos se encuentran MySQL, SQL Server, SQLite o sybase. Soportan mejor el trabajo en red porque tienen integrado un servidor que atiende a las consultas a través de un socket (excepto SQLite) abierto en el puerto en el programa cliente (en el caso anterior había que hacer un programa en servidor que era el que recogía las consultas y las pasaba a la base de datos). El problema es que no se pueden representar objetos y las diferentes herencias se tienen que construir en tablas relacionadas con condicionantes de claves foráneas. Esto dificulta mucho el desarrollo del diseño de la base datos en relación con el diseño orientado a objetos de la funcionalidad de la aplicación.

4.1.2.4 Gestores de Bases de Datos Relacional-Objeto

Una solución intermedia a estas dos es aplicar las extensiones al SQL que tienen ciertas bases de datos relacionales (Oracle, PostgreSQL) para trabajar con relaciones entre clases. El SQL de estas bases de datos se sale del estándar (lo que no significa mucho problema, pues entre las que son solo relacionales ya hemos comentado que también existen diferencias) y aporta nuevas palabras clave para trabajar con herencia, creación de nuevas implementaciones de tipos de datos y, aún sin estar relacionado con las tablas donde se almacenan estos objetos, permiten extensiones de funcionalidad con lenguajes de programación integrados en la misma base datos (Extensiones PL o Procedural Language). Tanto Oracle como PostgreSQL permiten realizar estas extensiones en Java, C, y algún otro lenguaje de programación.

El licenciamiento de una y otra base de datos es una de las grandes diferencias. Oracle, a pesar de tener una licencia de uso gratuito, no es libre. En la licencia de uso gratuito ofrecen soporte para un solo procesador y funcionalidad limitada. La variedad de pago es una licencia de soporte anual del producto sin derecho a actualizaciones en el caso de no renovación. PostgreSQL por el contrario es libre. Si bien no tiene el respaldo de una comunidad de usuarios como pueda ser el conjunto de programadores web que tiene MySQL, en su desarrollo participan multinacionales como Fujitsu o Sun Microsystems, y almacena los datos del sistema de telefonía IP Skype.

Este último tipo de base de datos sería el más apropiado para esta aplicación, ya que, en sistemas de producción, estimo que es más rentable la estabilidad de un producto estandarizado con posibles extensiones tecnológicas, que la extrema innovación de las bases de datos orientadas a objetos y que el trabajoso proceso de transformar un diseño orientado a objetos en un diagrama entidad-relación.

4.1.2.3 Sistemas intermediarios entre Base de Datos Relacional o Relacional Objeto y Programación Orientada a Objetos

Además de las diferentes posibilidades de sistemas gestores de bases de datos, existen en el mercado diferentes interfaces que hacen posible la persistencia de los objetos de una manera sencilla al programador. Con esto quiero decir que se trabaja con el objeto directamente sin tener que hacer declaraciones SQL para obtenerlo, guardarlo o modificarlo. Existen varios programas que realizan esta función de interface. Me he fijado en dos por ser software libre: Hibernate, de RedHat, e iBatis, de Apache. Aparte está el sistema de persistencia de objetos de Java.

Las ventajas de estos programas consisten en que el diseño entidad-relación para la creación y gestión de la base de datos no son necesarios. Partes de los objetos y son los propios objetos los que tienen la persistencia adquirida como propiedad, de manera que, con una clase que genera objetos a partir de los datos almacenados, podemos editar, modificar y borrar las propiedades persistentes (almacenadas en la base de datos) de estos objetos sin tener que ejecutar las sentencias SQL. Así conseguimos un programa totalmente orientado a objetos sin constructores ni métodos extraños con lenguajes externos al de programación (SQL, OQL).

Cada uno de estos sistemas utiliza un método para garantizar una persistencia correcta. Hibernate forma parte del proyecto de JBoss de RedHat. Aparte del sistema de persistencia en si mismo tiene añadidos que permiten la indexación de campos para facilitar la búsqueda en diferentes campos de la tabla. Además, a partir de la definición de las clases de datos, el fichero de configuración de conexión, y la equivalencia de las clases con la base de datos, te crea en el sistema gestor de base de datos la estructura de las tablas que se van a utilizar.

iBatis tiene la ventaja de que es más simple de manejar, pues no tiene tantos añadidos. A pesar de ello, la documentación es mucho mayor, así como la comunidad de usuarios, en Hibernate.

La decisión de usar un sistema intermediario va a permitir escalabilidad en la aplicación. No va a depender de un sistema gestor de base de datos almacenado en un servidor. Cambiando el fichero de configuración el mismo esquema valdría para una base de datos pequeña almacenada en el propio disco de la aplicación (por ejemplo SQLite) como para una base de datos Oracle distribuida en varios servidores. Esto permite además al desarrollador que, una vez que tiene definidas las clases y los esquemas, se dedique solo a programar en el lenguaje de programación elegido, sin tener que preocuparse de la corrección de las sentencias de consulta a la base de datos. Además este sistema tiene la ventaja de que en búsquedas complejas y muy detalladas (en el caso de que no se use el servidor de indexación de Hibernate, que las realizaría con métodos orientados a objetos) se pueden pasar partes de la sentencia del lenguaje de consulta a la base de datos para concretar los resultados.

4.1.3 El servidor

A pesar de que el programa se ejecutará en los ordenadores de cada uno de los usuarios en sus casas u oficinas, los datos actualizados estarán almacenados en el servidor de base de datos. Además del propio servidor de los datos es muy posible existan procesos automáticos de cálculos, o incluso la posibilidad de que se consulte, a través de web ciertos datos públicos contenidos en las bases de datos. Dada esta situación la elección de un servidor apropiado es un punto crítico en el posterior funcionamiento correcto del sistema. En esta elección entran en juego el tipo de procesador, el procesador o procesadores, la memoria, la capacidad de almacenamiento y el sistema operativo.

La necesidad de capacidad de cálculo, almacenamiento en memoria y conexiones múltiples dependerá en todo momento del club deportivo y su magnitud. Puede que en ciertos clubes multidisciplinarios se necesite un sistema de dos servidores, uno de almacenamiento de base datos y otro de procesamiento y ejecución del programa, o una granja de varios servidores con las tareas compartidas, de manera que si falla uno el sistema completo siga funcionando, y se pueda añadir capacidad de proceso y almacenamiento según la necesidad.

Uno de los puntos imprescindibles en el funcionamiento de la aplicación, sea cual sea la magnitud de la organización, es la virtualización. La capacidad de ejecutar el servidor y la base de datos en un mismo equipo, pero con sistemas operativos virtualizados da la posibilidad de cambiar en cualquier momento la estructura del sistema con impacto mínimo en el tiempo de migración. Además la seguridad que representaría el poder volver a levantar cualquiera de los servidores (el de la aplicación o el de los datos) en cualquier ordenador que tengamos a mano con la máquina virtual instalada en caso de errores nos da la confianza de tener un tiempo de respuesta a fallos muy rápido.

El caso que nos ocupa es el de un club pequeño, que puede tener 10 o 20 conexiones simultáneas a lo sumo, unos 150 los socios, deportistas, directivos y trabajadores, y un uso del sistema informático por parte de 2 o 3 de los técnicos. Luego las necesidades son muy pequeñas, pero escalables, puesto que puede ser muy interesante la capacidad de crecimiento que aporta al club la aplicación.

La tarjeta de red no será necesariamente más que la estándar de 100Mb/s pues el cuello de botella de la transferencia de datos se encuentra en la línea de conexión a internet del propio servidor. En el caso de que la infraestructura sea virtualizada se tratará de dar un interfaz físico de red a cada máquina virtual.

En la elección del sistema operativo tenemos que tener en cuenta, además de que soporte virtualización, que sea totalmente compatible con la base de datos elegida (Oracle o PostgreSQL) y con el servidor de aplicaciones. Si bien el sistema operativo del servidor de aplicaciones podría ser diferente al de almacenamiento de datos, en futuro la formación del mantenedor de sistemas se podría encarecer, luego la elección sería la misma para ambos sistemas. Aquí tenemos las opciones:

4.1.3.1 Windows

Sistema operativo de Microsoft ® que en sus diferentes versiones de servidor ha evolucionado positivamente. Windows Server 2008 representa grandes avances de seguridad para un servidor que por la naturaleza cerrada de su código siempre ha sido objeto de ataques. La experiencia del diseñador con Windows Server (2003 en este caso) es positiva en medios de producción con software de gestión de terceros. Sus mejores características son la posibilidad de tener asistentes de todo tipo para configuración de software y hardware y la facilidad de manejo por su gran parecido al los Windows Home y Pro. Su gran desventaja es el incremento del consumo

que representa cada incremento de versión, que hace que de una versión de a otra de Windows el hardware que la soporta quede obsoleto. La gran ventaja vendría si la elección del lenguaje de programación es .NET.

4.1.3.2 *NIX

Los sistemas operativos de la familia *NIX (UNIX, Linux, Solaris, *BSD, MacOS) tienen un origen común en los laboratorios de AT&T y se han ramificado dando lugar a versiones tanto propietarias, como libres. Tienen la ventaja de que entre la gran gama de núcleos de sistema operativo deferentes todos tienen las mismas herramientas de aplicación en la superficie, consola bash, servidor de ventanas X11, aplicaciones basadas en GTK y Qt y comandos de administración de sistema parecidos. El diseñador de la aplicación tiene larga experiencia en la administración de sistemas Linux de producción y MacOS de escritorio. La experiencia con estos sistemas operativos es muy buena en el ámbito de la estabilidad. Además, la reciente liberación de Java y las incorporaciones al núcleo del sistema operativo Solaris y Linux hacen que si el lenguaje es finalmente Java, se considere esta opción como la más aceptable. Además PostgreSQL nació para sistemas *NIX y, aunque la última versión ya lo soporte, en Windows está aún lejos de tener la misma estabilidad.

4.1.4 La plataforma elegida

Después de haber considerado todas estas opciones lo primero que se descartó fue cualquier plataforma basada en Windows tanto por el alto coste total de la implantación y mantenimiento para el cliente, como por las pocas ventajas económicas que aportaría para hipotética empresa que explotara la aplicación. TODO Precios El coste de la estación de trabajo donde se programaría la aplicación representaría en coste de licenciamiento 128€ del Windows Vista Bussines y 600€ de Visual Studio. En este precio no se ha incluido ningún antivirus, mantenimiento ni seguro por parte de otras empresas que incrementarían notablemente el precio. El coste del licenciamiento de la estación de trabajo es necesario, pues no se puede programar para un servidor Windows sin un cliente Windows. El coste del sistema operativo de servidor mínimo son 300€ pero es la versión que no tiene servidor web, que probablemente se use como complemento de consulta a la aplicación, así que habría que buscar la opción más cara de 600€. Como hemos mencionado antes la base de datos PostgreSQL no funciona del todo bien en Windows Server y tendríamos que hacer la inversión en Oracle. Además de todo ello para hacer funcionar esta serie de características es necesario un hardware muy potente.

Entre los sistemas operativos *NIX se plantea la duda entre Open Solaris y Linux. Mac se ha descartado por la obligatoriedad de hardware de Apple ® para hacerlo funcionar, y los sistemas BSD por tener un soporte más pequeño de la comunidad.

Puesto que ya no nos ata la tecnología propietaria de Windows, podemos elegir libremente y en igualdad de condiciones entre Oracle y PostgreSQL. Como las características de ambas bases de datos son muy parecidas, tanto en potencia como en modularidad, capacidad de escalar con clusters o granjas de ordenadores, y lenguajes procedurales internos optamos por PostgreSQL que tiene la propiedad de ser libre, además de contar con una capa de compatibilidad con Oracle, de manera que si deseamos hacer la aplicación compatible con ambas bases de datos no tenemos más que usar esta aplicación que convierte el SQL de Oracle en el de PostgreSQL.

La capa de abstracción de base de datos será Hibernate, que permite, al igual que la capa de compatibilidad de Oracle con PostgreSQL, las migraciones sencillas entre bases de datos, el ahorrar el diseño entidad - relación y el considerar en todo momento objetos en el software sin tener en cuenta las sentencias con el lenguaje de consultas de la base de datos.

Para la elección del lenguaje de programación hemos tenido en cuenta la capacidad de soporte externo, tanto de la comunidad como de una empresa, la facilidad de diseño con IDE o plataforma completa de desarrollo (que abarque desde el diseño de las clases, hasta de la interfaz gráfica, completamente enlazada con el editor de código), la posibilidad de reutilizar todo o casi todo lo desarrollado en una aplicación de servidor, y la mayor abstracción y parecido de las capacidades del lenguaje con la metodología orientada a objetos, siempre sin perder el tipado fuerte para que los errores surjan en compilación. Descartando las diferentes opciones (los lenguajes mono-plataforma no tienen posibilidad de reutilizarse en servidor web, los lenguajes ejecutados directamente en servidor no cumplen con tener una plataforma completa de desarrollo (salvo JSP y ASP), tan solo nos quedan .NET y Java, pero como hemos descartado previamente .NET por ser tecnología de Microsoft y tener un coste total de adquisición de la propiedad muy alto, tan solo nos queda Java.

El sistema operativo de servidor será Linux, por tener una comunidad de usuario mayor que Solaris. Esta decisión se toma porque encontrar un técnico competente en Linux en el mercado laboral será bastante más fácil y barato que uno de Solaris.

4.1.5 Diseño de la infraestructura

A continuación se presentan los diferentes diagramas de la infraestructura sin virtualización (Diagrama 4.1) y virtualizada(Diagrama 4.2).

Se ha valorado la inclusión de diferentes sistemas de réplica de la base de datos como pgpool que permite la realización de las consultas en múltiples bases de datos. En el caso de que alguna de ellas fallara las demás responderían. Esto podría causar fallos de integridad referencial de los datos contenidos al reiniciar la base de datos parada, pero se evitaría con el módulo Slony, que automatiza la actualización de datos entre diferentes servidores, de manera que podríamos ejecutar un script que recuperara los datos perdidos durante la caída del servicio. Así pues se ha desestimado esta opción, prefiriendo delegar la seguridad de los datos en discos con sistemas de ficheros RAID invisibles tanto al SGBD como al sistema operativo.

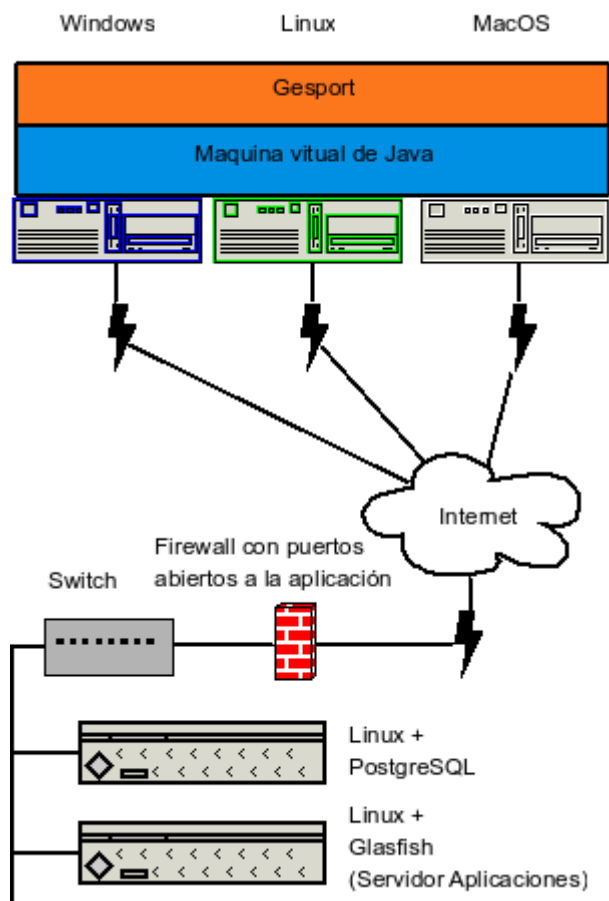


Ilustración 1: Infraestructura sin virtualización

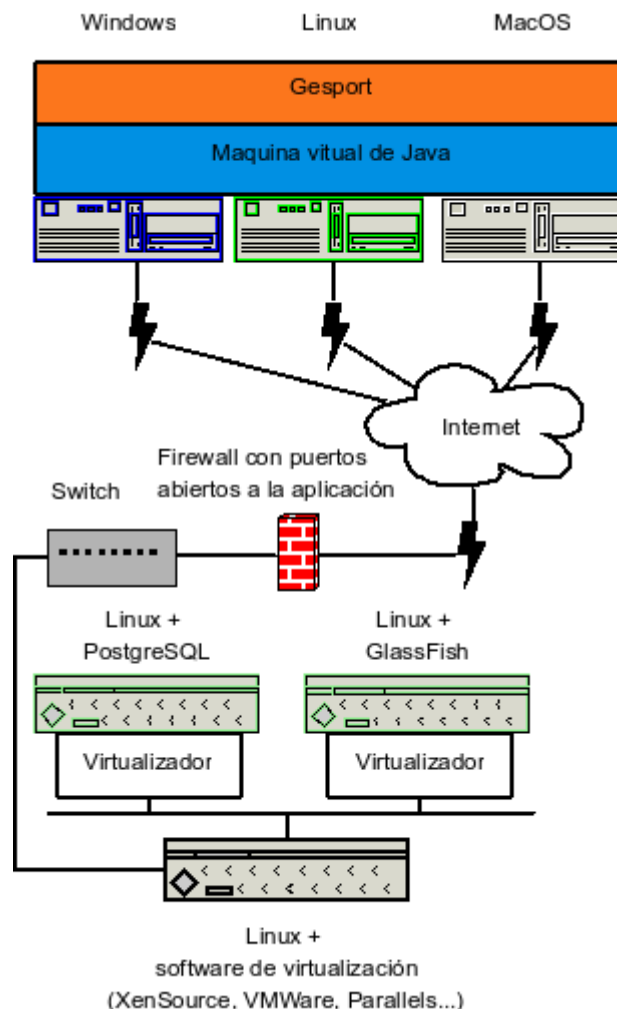


Ilustración 2: Infraestructura virtualizada

4.2 Diseño de las clases de objetos

Tomando como base las clases de análisis se realiza un proceso de diseño de las aplicación basada en clases de objetos efectivas en el programa. En este diseño ya se tienen en cuenta las propiedades del lenguaje de programación elegido. Además de las clases que intervienen en el dominio del problema habrá que definir las clases auxiliares de almacenamiento de datos, de ventanas y paneles del interface gráfico de usuarios o de intercambio de mensajes complejos entre clases del dominio. En las siguientes secciones se describen las clases del dominio del problema, las clases de utilidad. Las clases de interface gráfico se describen en la sección 4.4 Diseño del interfaz gráfico de usuario. En las clases se puede comprobar que se han omitido todos los get y set

de todos los tributos tal y como están definidos, así como los métodos hashCode() y toString(). En este diccionario si que se me

4.2.1 Diccionario de clases

4.2.2 Diagramas de clases

4.3 Diseño de la base de datos

El diseño de la base de datos viene dado por las definiciones en XML de las clases. Se convierten en sentencias SQL al iniciar la ejecución del programa y quedan grabadas en la base de datos. El XML de la definición de las clases de Hibernate permite definir las relaciones entre las diferentes tablas, asignar los eventos que suceden en diferentes cambios, representar la herencia de las clases. Es por esta razón que no se ha representado ningún diagrama entidad relación en esta sección. En vez de ello se va redactar el funcionamiento y configuración de Hibernate para la aplicación Gesport en cuestión.

4.4 Diseño del interfaz gráfico de usuario

Para el interface gráfico de usuario se utilizará el conjunto de clases gráficas de Java Swing, construido sobre AWT. Tendrá un aspecto de programa estándar con un menú superior, un panel principal con los elementos que pertenezcan al trabajo seleccionado en el menú superior, y una barra de estado donde se indicarán la corrección de la conexión al servidor, el nombre del usuario conectado, el estado de seguridad en la transmisión de los datos.

A continuación muestro esbozos de la idea con la que voy a trabajar. El objetivo es que cumpla en mayor medida las normas de usabilidad de Gnome, el entorno de escritorio de GNU, porque tiene uno de los mejores estudios de usabilidad de cuantos sistemas de escritorio hay.

A pesar de usar este sistema en la colocación de botones, menús, aplicaciones sobre elementos con botones derecho e izquierdo, la aplicación usará el entorno de usuario del sistema operativo en el que se ejecute (GTK en Linux, Windows Forms en Windows, el entorno predeterminado de Solaris en caso de que alguno de los usuarios tenga este sistema operativo). Esto se consigue gracias a las librerías gráficas de java, que soportan las diferentes apariencias de los sistemas operativos.

4.5 Diseño de las pruebas

Las pruebas que se van a realizar para comprobar el correcto funcionamiento del programa son las siguientes.

- Dada la especificación de las clases se van a probar cada uno de los métodos de clase con valores al azar, así como con valores fuera del rango predeterminado de los elementos y valores extremos. Esta prueba se hará con programas que trabajarán exclusivamente con la clase en pruebas y las clases dependientes devolverán los valores extremos, al azar o fuera de rango (eligiendo tipos que soporten más decimales, espacio en memoria más grande, o valores que no tengan que ver con los esperados).

- Dada la especificación de las clases y su asignación de persistencia en base de datos se van a probar cada uno de los métodos de clase con valores al azar, así como con valores fuera del rango predeterminado de los elementos y valores extremos tanto de la base de datos como de la clase (en el caso de que los extremos no sean comunes). Al igual que en la prueba anterior las clases que entren en contacto con la de pruebas serán generadoras de datos al azar, extremos fuera de rango. La base de datos se utilizará para estudiar la recuperación de los datos igual que las clases extremas, y la clase enviará en, otra fase de esta prueba, datos al azar, extremos y fuera de rango a la base de datos.

- Se realizarán pruebas simultaneas de varias ejecuciones de clases con persistencia para comprobar la integridad de los datos en tiempo de ejecución, bloqueos de acceso y modificación de informaciones, y concurrencia de acceso a datos.

- Dada la especificación de los paquetes de clases se generará una prueba donde se comprueben todas las interacciones de las clases de un paquete. Se realizará la prueba ya con persistencia y después de haber realizado las tres baterías de pruebas anteriores.

- Se realizarán pruebas del funcionamiento correcto del programa. Para ello se crearán datos de ejemplo desde cero en una base de datos totalmente vacía. Con esta prueba también se comprobará la seguridad de la instalación, pues el programa en su primera ejecución creará un usuario administrador con una contraseña al azar.

- Se realizará otra prueba de concurrencia de acceso a datos con el programa completo para comprobar que el aumento de funcionalidades no afecta al rendimiento de las clases.

- Se realizarán pruebas de funcionamiento del programa con monitores de consumos de procesador, memoria, referencias circulares, consumo de ancho de banda y otras herramientas de optimización de rendimiento de la aplicación.

- Se realizarán pruebas con servidor remoto de base de datos accedido desde una línea ADSL con programas estándar funcionando al tiempo de la ejecución. El servidor a su vez estará situado en un ADSL con velocidad de subida de 1Mb. Se irán aumentando las instancias del software Gesport para comprobar el límite de conexiones en estas circunstancias. Se hará un estudio de escalabilidad aumentando los anchos de banda de servidor, memoria, número de conexiones aceptadas por el servidor de base de datos, con un gestor de conexiones como aplicación de servidor, enviando y recibiendo los datos con XML comprimido en vez de abrir conexiones contra la base de datos (esta última prueba se hará en caso de que el sistema de persistencia utilizado lo permita como un tipo de conexión más como origen de datos).

5. Manuales de usuario y administrador

5.1 Manual de instalación

5.2 Manual de usuario

5.3 API de extensión de funcionalidades

6. Índices de anejos

Relación de ilustraciones y tablas del proyecto

Índice de Ilustraciones

Ilustración 1: Infraestructura sin virtualización.....	31
Ilustración 2: Infraestructura virtualizada.....	32

7. Bibliografía

Requirements Engineering [Ian Sommerville & Pete Sawyer, 1997].