|  |
| --- |
| **GESTOR DE EVENTOS DEPORTIVOS** |
|  |
| OLIMPIADAS2MIL |
|  |
| Francisco Huertas  Gonzalo Ortíz  Alicia Pérez  Laura Reyero  Gabriela Ruíz  Rubén Tarancón |
|  |
| CURSO 2009/2010 |
|  |

Contenido

[Introducción 5](#_Toc261445942)

[Propósito 5](#_Toc261445943)

[Componentes del sistema 5](#_Toc261445944)

[Roles y perfiles 5](#_Toc261445945)

[Requisitos 7](#_Toc261445946)

[Casos de uso 7](#_Toc261445947)

[Diagrama de clases 8](#_Toc261445948)

[Observaciones 8](#_Toc261445949)

[Restricciones 8](#_Toc261445950)

[Diagramas de estado 9](#_Toc261445951)

[Eventos 9](#_Toc261445952)

[Observaciones 9](#_Toc261445953)

[Entradas 9](#_Toc261445954)

[Observaciones 9](#_Toc261445955)

[Diagramas de actividad 10](#_Toc261445956)

[Eventos 10](#_Toc261445957)

[Observaciones 10](#_Toc261445958)

[Entradas 10](#_Toc261445959)

[Observaciones 10](#_Toc261445960)

[Diagramas de secuencia 11](#_Toc261445961)

[Obtener clasificación 11](#_Toc261445962)

[Observaciones 11](#_Toc261445963)

[Restricciones 11](#_Toc261445964)

[Consultar horarios personalizados 12](#_Toc261445965)

[Observaciones 12](#_Toc261445966)

[Restricciones 12](#_Toc261445967)

[Crear acta 13](#_Toc261445968)

[Observaciones 13](#_Toc261445969)

[Restricciones 13](#_Toc261445970)

[Reservar entradas 14](#_Toc261445971)

[Observaciones 14](#_Toc261445972)

[Restricciones 14](#_Toc261445973)

[Diagramas de componentes 15](#_Toc261445974)

[Diagramas de Componentes 15](#_Toc261445975)

[Base de datos 15](#_Toc261445976)

[Gestores 16](#_Toc261445977)

[Interfaces 17](#_Toc261445978)

[Modelo de datos 19](#_Toc261445979)

[Diagrama entidad-relación 19](#_Toc261445980)

[Modelo relacional 20](#_Toc261445981)

[Descripción de las tablas 20](#_Toc261445982)

[Funciones de la base de datos 22](#_Toc261445983)

[Patrones 29](#_Toc261445984)

[PATRÓN SINGLETON 29](#_Toc261445985)

[Interfaz de usuario 30](#_Toc261445986)

[Investigación sobre los componentes 31](#_Toc261445987)

[Propuesta para la interfaz 31](#_Toc261445988)

[Propuesta para la base de datos 32](#_Toc261445989)

[Propuesta para el blog 32](#_Toc261445990)

[Propuesta para el componente de pago 33](#_Toc261445991)

[Planificación estimada 34](#_Toc261445992)

[Propuesta de calidad 37](#_Toc261445993)

[Cuestionario CMMI 38](#_Toc261445994)

[Evolución del proyecto 43](#_Toc261445995)

# Introducción

## Propósito

Queremos ofrecer una herramienta que facilite la gestión de eventos deportivos de grandes magnitudes. Para ello elaboramos una aplicación que administra los eventos desde su adjudicación hasta la finalización de los mismos.

Una vez definido el tipo de evento del que se trata, se solicita al Ayuntamiento del lugar de la celebración todos los elementos necesarios para la organización. Éste nos suministra una serie de bases de datos en las que se encuentra almacenada la información de: los emplazamientos existentes con el tipo de emplazamiento, tipo de instalaciones y el número de espectadores posibles.

La entidad organizadora del evento deportivo nos proporcionará información de las distintas competiciones que se disputarán, el equipamiento necesario, y la forma en la que tendrán que programarse las competiciones de los mismos. También nos proporcionará una base de datos actualizada de los deportistas y árbitros que participarán.

Una vez recopilada toda esta información, nuestra aplicación se encarga de organizar en el tiempo y el espacio todas las competiciones, atendiendo a las limitaciones impuestas por cada deporte. Centrándonos en los requisitos de los mismos, podremos decidir la ubicación de los eventos, y proceder a la venta de entradas para los espectadores interesados.

Nuestra aplicación se encargará también de asignar los árbitros a cada evento atendiendo a las restricciones pertinentes.

Así mismo, nuestra aplicación actúa de portal de noticias y de visualización de las características del evento deportivo (calendario, clasificación, etc.) según el perfil al que pertenezca el usuario.

## Componentes del sistema

## Roles y perfiles

##### USUARIO NO REGISTRADO

Otorgaremos este perfil a cualquier individuo que acceda a nuestra aplicación sin identificarse mediante una clave de usuario y una contraseña.

##### USUARIO REGISTRADO

Otorgaremos este perfil a cualquier individuo ajeno a la organización que acceda a nuestra aplicación identificándose mediante una clave de usuario y una contraseña.

##### ÁRBITRO

Otorgaremos este perfil a aquellos individuos identificados por la aplicación como árbitros.

##### COI

Otorgaremos este perfil a aquellos individuos identificados por la aplicación como miembros del Comité Olímpico Internacional.

##### AYUNTAMIENTO

Otorgaremos este perfil a aquellos individuos identificados por la aplicación como miembro del ayuntamiento de la localidad organizadora.

##### ADMINISTRACIÓN – GESTOR DE EVENTOS

Otorgaremos este perfil a aquellos usuarios con poderes de administración total en la aplicación.

Figura: Diagrama de herencia entre los perfiles

## Requisitos

## Casos de uso

# Diagrama de clases

### Observaciones

Desde la clase evento, se gestionan todos los detalles necesarios para la organización del mismo. Éste tiene lugar en un emplazamiento concreto atendiendo a las restricciones impuestas por los deportes que participan en el evento, y en algunos casos pueden celebrarse varios eventos en un mismo emplazamiento.

Desde esta clase, también se gestiona la asignación de los equipos que participarán, y los árbitros que colaborarán. El número de equipos y árbitros que tendrá el evento dependerá de la competición a la que pertenezca.

Cada emplazamiento tiene un número de localidades que servirá al sistema para poner a la venta entradas para el evento que se celebre en un momento concreto. Una vez el evento ha finalizado, las entradas creadas para éste son eliminadas del sistema.

Tras finalizar el evento, el árbitro elabora un acta, que recoge .

También dispone de la información del siguiente evento que le sucederá en orden cronológico.

La clase competición se encarga de unificar las condiciones que se tiene que dar para que se pueda llevar a cabo. Determina el número de árbitros y equipos necesarios, además de imponer el tipo de instalaciones que deberá incorporar el emplazamiento donde se vaya a ubicar el evento de la competición.

Los equipos están formados por uno o más deportistas, todos de la misma nacionalidad. Tanto los deportistas como los árbitros que participan, heredan de la clase participante.

El emplazamiento posee información sobre sus instalaciones, para la posterior asignación. Tiene asientos para los espectadores en función de su aforo. Cada asiento tener asignado o no una entrada para asistir al evento. Dado que en un mismo emplazamiento, a lo largo del tiempo, se pueden celebrar varios eventos, el asiento tendrá asignado una entrada por cada evento que se celebra.

Todas las entradas se crean con estado disponible, y cuando este estado cambia, se le asigna un comprador. Una persona identificada como comprador puede adquirir varias entradas, para el mismo evento, o para eventos diferentes. El precio de la entrada depende del evento para el cual se pone a la venta. Además, se pueden obtener descuentos de hasta el 25% si se trata de un estudiante o jubilado. Las entradas de tipo vip, se ofrecerán a personalidades sin coste alguno.

Un acta está relacionada con al menos dos resultados, que almacenarán el identificador del equipo y el resultado que ha obtenido en la competición en la que ha participado. En ella también se anotan las lesiones o sanciones que se han podido producir durante el evento.

### Restricciones

# Diagramas de estado

## Eventos

### Observaciones

## Entradas

### Observaciones

# Diagramas de actividad

## Eventos

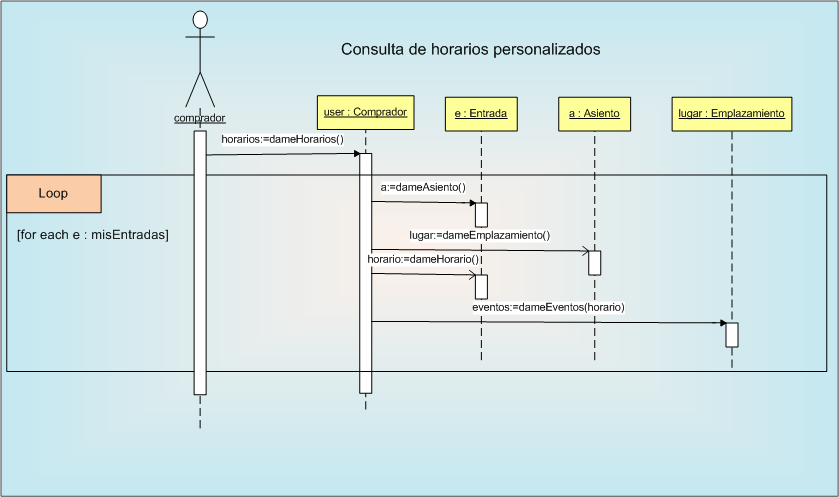
### Observaciones

## Entradas

### Observaciones

# Diagramas de secuencia

## Obtener clasificación

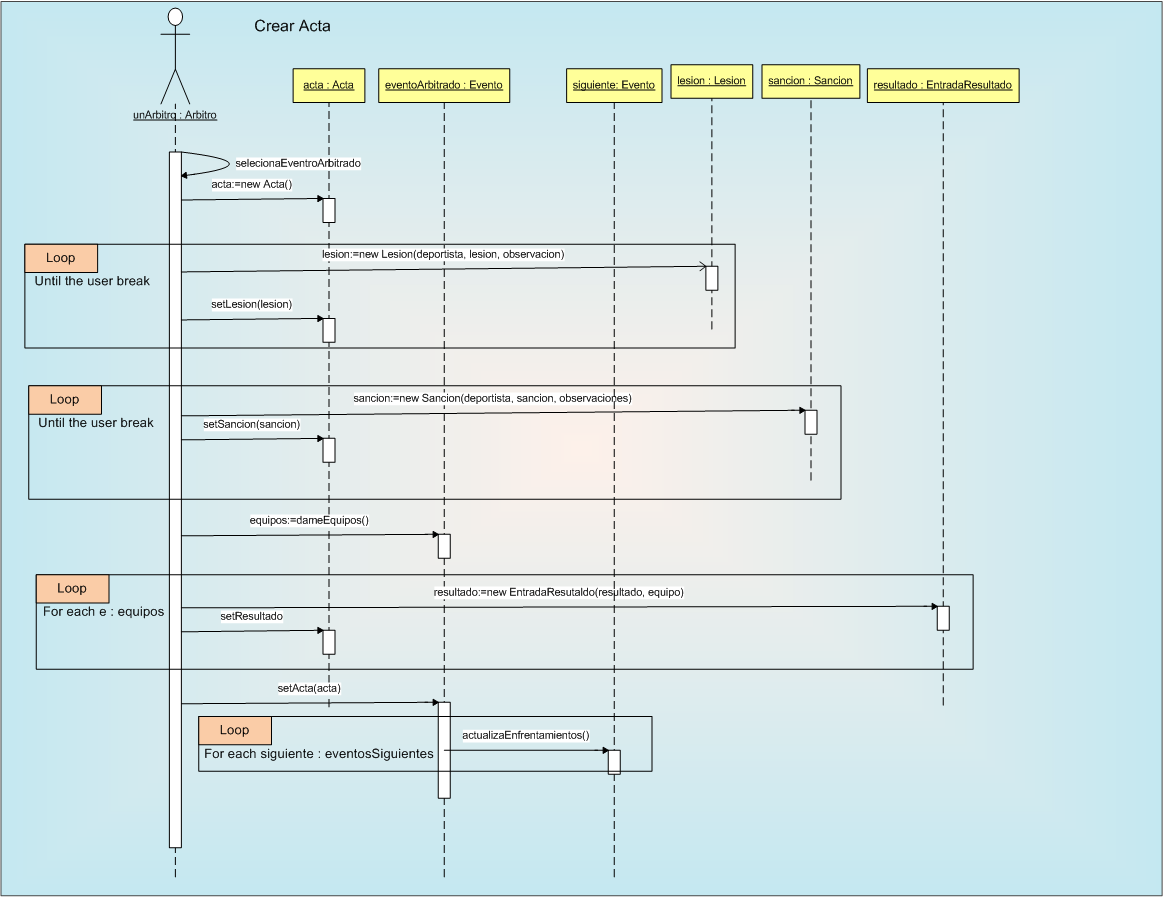


### Observaciones

Este diagrama muestra cómo un usuario puede consultar el estado de una competición. Esto incluye resultados de partidos anteriores, próximos eventos, actas de los equipos etc.

### Restricciones

## Consultar horarios personalizados

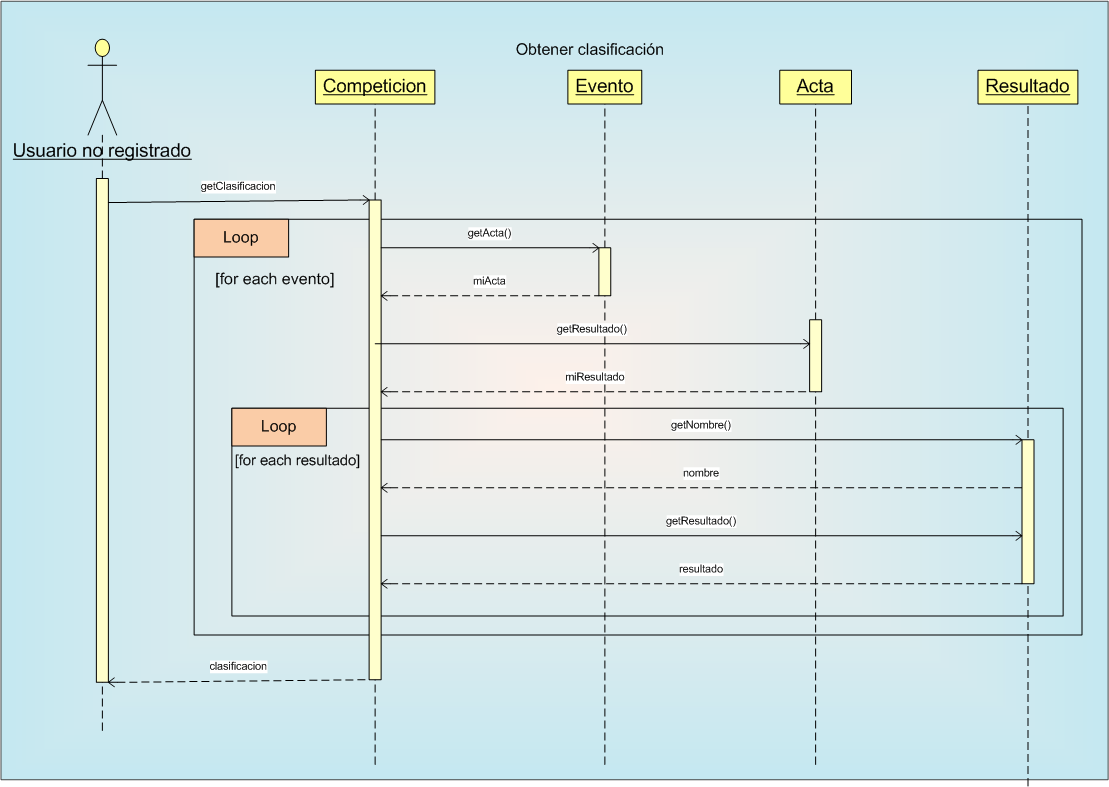


### Observaciones

Este diagrama de secuencia representa cómo se genera un horario personalizado dependiendo de las entradas. El usuario solicita los horarios que le serán representados en la interfaz web. Esto se realiza con el objeto comprador, que será el que consultando todas las entradas y sus respectivos asientos y emplazamientos, dará una lista detallada de todo el horario.

### Restricciones

## Crear acta

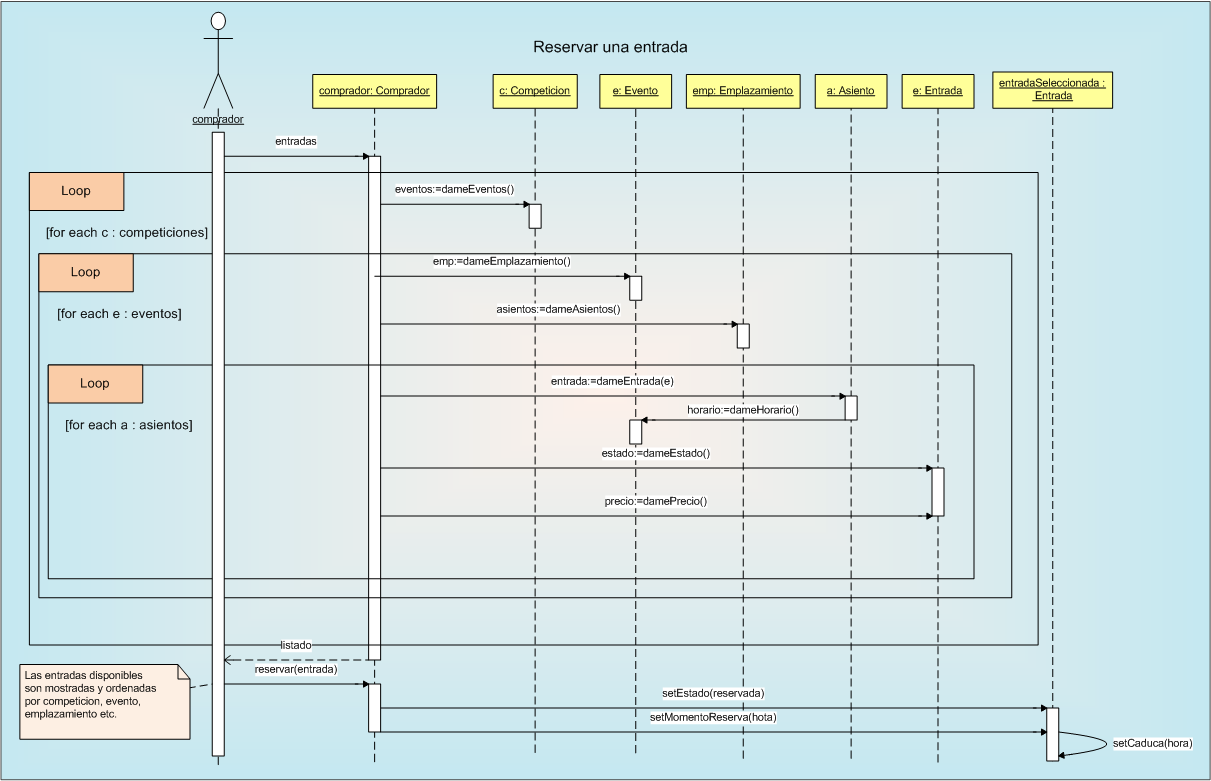


### Observaciones

Este diagrama de secuencia muestra como se crea un acta. Las actas una vez creadas no pueden ser modificadas y quedan almacenadas en el sistema. Al introducir el acta hay que especificar los atributos del acta y al almacenarla se actualizan, en caso de existir, los siguientes eventos de la competición.

### Restricciones

## Reservar entradas



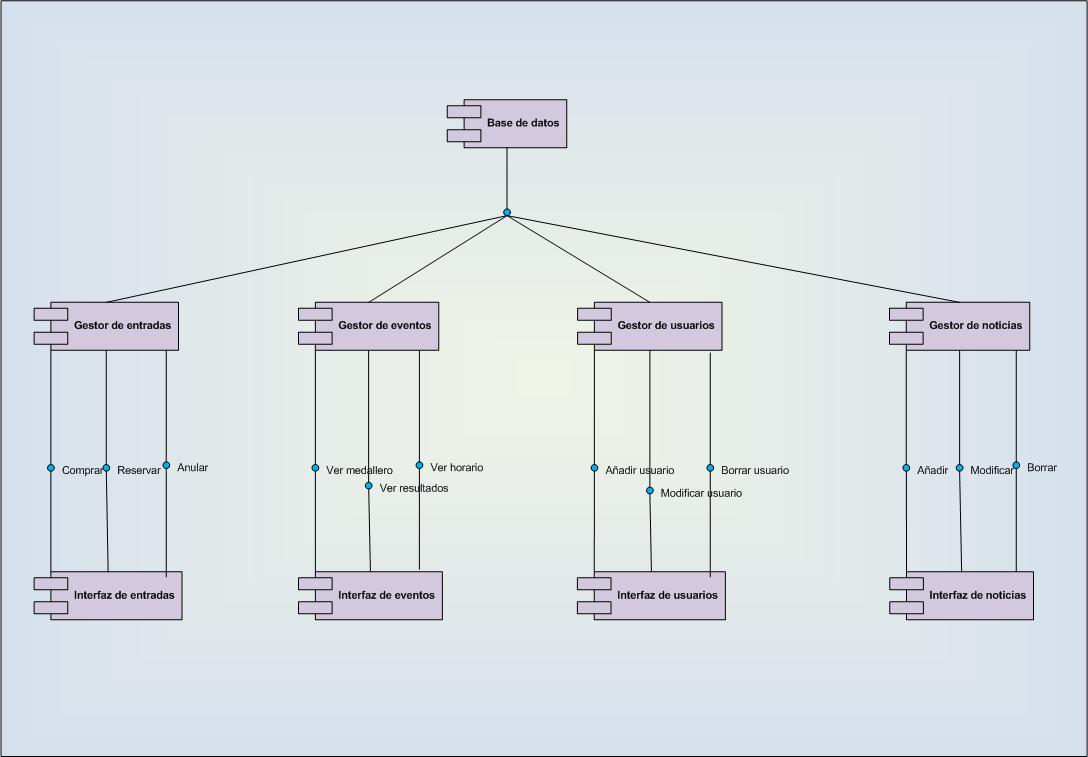
### Observaciones

Para realizar una reserva lo primero que se hace es comprobar el estado de las entradas disponibles, haciendo un listado con todas las entradas disponibles y seleccionables. En el caso de existir entradas de varios tipos, si la entrada no se ajusta al perfil de comprador (ya sea porque no está disponible para él o porque ese tipo de entrada es específico para determinado tipo de personas) no aparecerán en el listado. Una vez realizada la petición de reserva, ésta es realizada.

### Restricciones

# Diagramas de componentes

## Diagramas de Componentes



## Base de datos

* La base de datos de nuestra aplicación constará de varias tablas para los diferentes usos del sistema. Habrá tablas para el registro de los usuarios, para la venta de entradas, equipos participantes, instalaciones, etc.
* Alguna de esas tablas nos la enviará el COI (como la tabla equipos y la tabla competiciones) y en nuestra aplicación solo se podrán utilizar, nunca modificar. Otras tablas nos la dará el ayuntamiento de la ciudad organizadora (tabla instalaciones) y al igual que antes, nuestra aplicación no modificara esa tabla.
* El resto de tablas son las que mediante el sistema de gestión de bases de datos, abajo explicado, y los gestores harán que nuestra aplicación tenga una funcionalidad completa.
* En el apartado relativo a modelo de datos se explicara más detalladamente el formato de cada una de las tablas

###### SISTEMA DE GESTION DE BASE DE DATOS

* Es el componente que nos ayuda a gestionar la base de datos de la aplicación, el espacio físico y nos proporciona unas interfaces de acceso primitivas. Es el puente entre los gestores y las tablas de la base de datos.
* La interfaz consta de métodos de acceso a tablas (select), de creación de tablas (create), de borrado de tablas (drop) , borrado de información de una tabla (delete), modificación de información contenida en tablas (update) y de las propias tablas (alter) , y por ultimo un método para insertar información en las bases de datos (insert).

## Gestores

###### GESTOR DE ENTRADAS

* Es el componente a través del cual vamos a gestionar las entradas de los espectadores a las distintas disciplinas olímpicas.
* Va a ser el que va a recibir las ordenes de los compradores a través de la interfaz de entradas y el que va a tener que utilizar el SGDB para acceder y modificar las tablas. Gracias a él, los datos que los usuarios puedan introducir, serán coherentes con las tablas de la base de datos relativas a las entradas.
* Las operaciones que realizara sobre las tablas de la base de datos son:
* Reservar entrada : añadirá una fila a la tabla de entradas reservadas
* Comprar entrada: añadirá una fila a la tabla de entradas vendidas si la entrada que se intenta comprar pertenece a la tabla entradas reservadas y en caso afirmativo borrará la entrada de la tabla entradas reservadas.
* Anular entrada : eliminará una fila de la tabla de entradas reservas

###### GESTOR DE USUARIOS

* Es el componente a través del cual vamos a gestionar los usuarios que van a poder disfrutar de las ventajas de esta aplicación.
* A través del formulario de registrarse de la interfaz el gestor va a recibir todos los datos necesarios para que el usuario sea registrado en la aplicación. A partir de ahí el gestor hará las operaciones necesarias para que en la tabla de usuario queden reflejados los cambios.
* Las operaciones que va a realizar el gestor son:
* Registrar un usuario: añadir una fila a la tabla de usuarios.
* Modificar usuario: modificar los datos en la fila de la tabla de usuarios.
* Eliminar usuario: eliminar la fila de la tabla de usuarios.
* Login: comprobar que los datos introducidos están en la tabla de usuarios.

###### GESTOR DE NOTICIAS, FOTOS

* Es el componente a través del cual vamos a gestionar las noticias y fotos que van aparecer en la herramienta.
* En la interfaz de noticias, el usuario podrá introducir el titular y el cuerpo de la noticia, o si lo desea, fotos. Las noticias y fotos se almacenaran en el servidor y el encargado será el gestor.
* Las operaciones que realiza son:
* Añadir noticia
* Modificar noticia
* Añadir foto
* Eliminar foto
* Poner noticia en ultima hora
* En la parte relativa a componentes software se dará más información de este gestor y de cómo se va a integrar en nuestra aplicación

###### GESTOR DE EVENTO

* Es sin duda, el gestor más importante de nuestra aplicación a la vez que el más sofisticado.
* Utilizará las tablas de la base de datos de deportes, equipos e instalaciones que le vendrán dadas por el COI y el ayuntamiento para realizar un organigrama de toda la competición (en el diagrama de actividad Busca emplazamiento se detalla). Gracias a él se construirá el calendario completo de todos los eventos que se van a realizar en las olimpiadas y les podrá horario e instalaciones. También gestionará la tabla de resultados que después de cada partido se rellenará gracias a los actas y gracias a ello podremos visualizar inmediatamente las clasificaciones y el medallero. Por último el gestor tendrá operaciones más simples, pero no por ello menos importantes como pueden ser: modificar la información de un evento o modificar las actas de un partido.

## Interfaces

* Es el componente a través del cual el usuario va a poder acceder a todas las utilidades que ofrece el sistema, por lo tanto debe ser sencillo y muy intuitivo.
* Habrá una serie de interfaces, cada una la gestionará uno de los componentes arriba descritos.
* Más adelante se podrán ver con más detalle estas interfaces.

# Modelo de datos

## Diagrama entidad-relación



## Modelo relacional

**Equipo**(idEquipo, país, info, idCompeticion)

**Deportista**(idDeportista, nacionalidad, nombre, apellido1, apellido2)

**Pertenece\_Deportista**(idEquipo, idDeportista)

**Competicion**(idCompeticion, nombre, numPart, numArbitros, maxEquipos, minEquipos,

necPiscina,necPista, necCircuito)

**Evento**(idEvento, fase, fecha\_inicio, fecha\_fin)

**Resultado**(idResultado, idEquipo, resultado)

**Lesion**(idLesion, idEvento, lesión, idDeportista, observaciones)

**Sancion**(idSancion, idEvento, idDeportista, penalización, observaciones)

**Evento**\_Resultado(idEvento, idResultado)

**Arbitro**(idArbitro, nombre, apellido1, apellido2, nación)

**ArbitrosEvento**(idArbitro, idEvento)

**Emplazamiento**(idEmplazamiento, idMunicipio, idVia, cp, numero, numCircuitos, numPistas,

numPiscinas)

**Lugar\_Evento**(idEmplazamiento, idEvento, fecha)

**Asiento**(idAsiento, localizador, idEmplazamiento)

**Entrada**(idEntrada, idAsiento, idEmplazamiento, fecha, estado, precio, comprador,

fechaReserva, fechaCaducidad)

## Descripción de las tablas

###### TABLA DE EQUIPO

* El COI nos facilitará los equipos que van a participar en los diferentes deportes olímpicos. Cada equipo será identificado por “idEquipo”, y llevara información como el deporte al que pertenece, la nacionalidad del equipo e información adicional del equipo, como puede ser el nombre de los participantes del equipo.

###### TABLA DE DEPORTISTA

* La tabla de deportistas, al igual que la anterior, nos vendrá dada por el COI. Los únicos atributos relevantes para nuestra aplicación son el nombre, los apellidos y la nacionalidad.

###### TABLA DE PERTENECE\_DEPORTISTA

* Esta tabla se encarga de relacionar los equipos con los deportistas que van a formar cada equipo.

###### TABLA DE COMPETICION

* En esta tabla se alberga la información principal de cada competición. Los datos nos los facilitará el COI.

###### TABLA DE EVENTO

* La tabla de eventos va a recoger la información de cada uno de los partidos que se van a realizar en la olimpiada. Nuestro sistema, a partir de las tablas de deportes, instalaciones y equipos gestionará esta tabla automáticamente. El identificador de cada evento va a ser “idEvento” y va a llevar información básica, como puede ser los equipos participantes, los árbitros, el deporte, la instalación, la fase de competición o la fecha

###### TABLA DE RESULTADO

* Esta tabla va a ser utilizada por los árbitros al introducir las actas de los eventos. Se guardara el resultado en tuplas de equipo – resultado. Esto es así para facilitarnos el almacenamiento de resultados de eventos donde participen más de un equipo, ya que, por ejemplo, en atletismo se pueden almacenar el corredor y su tiempo, en fútbol el equipo junto a su resultado etc.

###### TABLA DE LESION

* Esta tabla va a ser utilizada por los árbitros al introducir las actas de los eventos. Se guardará la lesión y el deportista que la sufre. En el campo observaciones se puede añadir información adicional como la duración estimada de la lesión o la gravedad.

###### TABLA DE SANCION

* Esta tabla va a ser utilizada por los árbitros al introducir las actas de los eventos. Se guardará la sanción, el deportista que ha sido sancionado y en qué deporte. También contiene un campo observaciones donde introducir información adicional como una descripción de la acción sancionada.

###### TABLA DE EVENTO\_RESULTADO

* La misión de esta tabla es relacionar los eventos con el resultado de cada uno de los equipos que ha participado en el evento

###### TABLA DE ÁRBITRO

* Es otra tabla que nos dará el COI y simplemente se guardará el nombre y los apellidos del árbitro, el deporte que va a arbitrar y la nacionalidad de éste, muy importante a la hora de que no arbitre partidos de su propia nación.

###### TABLA DE ARBITROS\_EVENTO

* En esta tabla habrá una relación entre los árbitros y los eventos que van a arbitrar.

###### TABLA DE EMPLAZAMIENTO

* En este caso la tabla del emplazamiento nos la dará el ayuntamiento de la ciudad organizadora y contendrá la información de las instalaciones que posee la ciudad para albergar las olimpiadas. Cada instalación viene identificada por “id” y lleva información como restricciones que tienen que ser compatibles con el atributo restricciones del deporte que se va a realizar en esta instalación. También tiene la capacidad de espectadores.

###### TABLA DE LUGAR\_EVENTO

* En esta tabla se relacionara el evento con el emplazamiento que lo va a albergar.

###### TABLA DE ASIENTOS

* Cada estadio tendrá una tabla de asientos en la que cada asiento estará designado por un identificador “idAsiento”, que pertenecerá a un estadio concreto, “idEstadio”. Todos sabemos que en cada instalación hay asientos mejores y asientos peores, por eso hay que distinguirlo y como es obvio si el asiento es mejor tendrá un precio más caro. También es lógico que un asiento sea más costoso en una fase de competición que en otra, por ello esto también debemos de almacenarlo.

###### TABLA DE ENTRADA

* Existirá una tabla de entradas por cada instalación de las olimpiadas. Inicialmente estará vacía y a medida que se va reservando se va añadiendo una fila. Recordemos que para que una entrada sea vendida primero ha de ser reservada, por lo que las entradas vendidas son las que están en la tabla con el atributo “esReservada” a false. Por último es necesario añadir el identificador del usuario que compra la entrada y la fecha en la que lo hace, ya que hay que recordar que las entradas se compra por instalación y por día.

## Funciones de la base de datos

###### TABLA DE EQUIPOS

* El Access Control de esta tabla lo va tener el COI, las operaciones que va a poder realizar son:

añadirEquipo( idE, id, pais, inf)

pre:

not(Equipo.filas -> exists(f1: fila |

f1.idEquipo=idE and f1.idCompeticion=idC))

post:

Equipo.filas -> exists(f1:fila |

f1.idEquipo = idE and

f1.idCompeticion = idC and

f1.pais = pais and

f1.info = inf)

eliminarEquipo(idE, idC)

pre:

Equipo.filas -> exits(f1:fila |

f1.idEquipo=idE and f1. idCompeticion =idC)

post:

not(Equipo.filas -> exists(f1:fila |

f1.idEquipo=idE and f1. idCompeticion =idC)

modificarDeporte(idE, idC, pais, inf)

pre:

Equipo.filas -> exits(f1:fila |

f1.idEquipo=idE and f1.idCompeticion=idC)

post:

Equipo.filas -> exists(f1:fila |

f1.idEquipo = idE and

f1.idCompeticion = idC and

f1.pais = pais and

f1.info = inf)

###### TABLA DE DEPORTISTA

* El Access control en este caso también lo va a tener el COI, las operaciones que va a realizar son:

añadirDeportista(idD,nacion,nomb,ape1,ape2)

pre:

not(Deportista.filas -> exists(f1: fila |

f1.idDeportista=idE))

post:

Deportista.filas -> exists(f1:fila |

f1.idEquipo = idD and

f1.nacionalidad = nacion and

f1.nombre = nomb and

f1.apellido1 = ape1

f1.apellido2 = ape2)

eliminarEquipo(idD)

pre:

Deportista.filas -> exits(f1:fila |

f1.idDeportista=idD)

post:

not(Deportista.filas -> exists(f1:fila |

f1.idDeportista=idD)

modificarDeporte(idD,nacion,nomb,ape1,ape2)

pre:

Deportista.filas -> exits(f1:fila |

f1.idDeportista=idD)

post:

Deportista.filas -> exists(f1:fila |

f1.idEquipo = idD and

f1.nacionalidad = nacion and

f1.nombre = nomb and

f1.apellido1 = ape1

f1.apellido2 = ape2)

###### TABLA DE PERTENECE\_DEPORTISTAS

* Esta tabla al relacionar los deportistas con el equipo al que pertenecen, el Access control es también del COI. Las operaciones que se pueden utilizar son:

añadirPertenece\_Deportista(idE,idD)

eliminarPertenece\_Deportista(idE,idD)

modificarPertenece\_Deportista(idE,idD)

* Todas las operaciones anteriores son muy similares a añadirDeporte, modificarDeporte y eliminarDeporte, respectivamente. En esta tabla también van a ser importantes las operaciones de consulta como:

Equipo\_de\_deportista(idD)

pre:

Pertenece\_Deportista.filas->exits( f1:filas |

f1.idDeportista = iD

post:

return Pertenece\_Deportista.f1.idEquipo

Deportistas\_de\_Equipo(idE)

pre:

Pertenece\_Deportista.filas->exits( f1:filas |

f1.idEquipo = idE

post:

return Pertenece\_Deportista.f1.idDeportista

###### TABLA DE COMPETICION

* Esta tabla contiene toda la información relativa a la competición por lo que va a ser gestionada por el COI. Las operaciones son muy a las de la tabla deportista:

añadirCompeticion(idCompeticion, nombre, numPart, nArbitros, maxEquipos, minEquipos, necPiscina,necPista, necCircuito)

modificarCompeticion(idCompeticion, nombre, numPart, nArbitros, maxEquipos, minEquipos, necPiscina,necPista, necCircuito)

eliminarCompeticion(idCompeticion)

###### TABLA DE EVENTO

* Las operaciones que se pueden realizar son:

crearEvento(idEvento, equipos, árbitros, idDeporte, Emplazamiento, faseCompeticion, fecha)

modificarEvento(idEvento, equipos, árbitros, idDeporte, id Emplazamiento, faseCompeticion, fecha)

EliminarEvento(idEvento)

###### TABLA DE RESULTADO

* Los árbitros emplean las siguientes consultas para poner los resultados:

crearActa(idE, res, obs)

pre:

not(Resultados.filas->exits(f1:filas |

f1.idEvento = idE)

post:

Resutados.filas->exits(f1:filas |

f1.idEvento = idE and

f1.resultado = res and

f1.observaciones = obs)

modificarActa(idE, res, obs)

pre:

Resutados.filas->exits(f1:filas |

f1.idEvento = idE)

post:

Resutados.filas->exits(f1:filas |

f1.idEvento = idE and

f1.resultado = res and

f1.observaciones = obs)

eliminarActa(idE)

pre:

Resutados.filas->exits(f1:filas |

f1.idEvento = idE)

post:

not(Resultados.filas->exits(f1:filas |

f1.idEvento = idE)

###### TABLA DE LESION

* El Access control de esta tabla pertenece al árbitro, las operaciones que puede realizar son:

añadirLesion(idLesion, idEvento, lesión, idDeportista, observaciones)

modificarLesion(idLesion, idEvento, lesión, idDeportista, observaciones)

eliminarLesion(idLesion, idEvento)

###### TABLA DE SANCION

* El Access control de esta tabla pertenece al árbitro, las operaciones que puede realizar son similares a las de la tabla lesión:

añadirSancion(idSancion, idEvento, idDeportista, penalización, observaciones)

modificarSancion(idSancion, idEvento, idDeportista, penalización, observaciones)

eliminarSancion(idSancion, idEvento)

###### TABLA DE EVENTO\_RESULTADO

* La tabla está controlada también por el árbitro. Las operaciones que se pueden utilizar son:

añadirEvento\_Resultado(idE,idR)

eliminarEvento\_Resultado(idE,idR)

modificarEvento\_Resultado(idE,idR)

* Todas las operaciones anteriores son muy similares a añadirDeporte, modificarDeporte y eliminarDeporte, respectivamente. En esta tabla también van a ser importantes las operaciones de consulta como:

Resultado\_Evento(idE)

pre:

Evento\_Resultado.filas->exits( f1:filas |

f1.idEvento = idE

post:

return Evento\_Resultado.f1.idResultado

###### TABLA DE ÁRBITRO

* El Access Control de esta tabla es del COI. Las operaciones que se van a poder realizar son:

añadirArbitro(idArbitro, nombre, apellidos, idDeporte)

eliminarArbitro(idArbitro)

modificarArbitro(idArbitro, nombre, apellidos, idDeporte)

* Estas operaciones son muy similares a añadirDeporte, modificarDeporte y eliminarDeporte respectivamente.

###### TABLA DE ARBITROS\_EVENTO

* El Access control de la tabla es de Ayuntamiento. Las operaciones que se pueden utilizar son:

añadirArbitros\_Evento(idA,idE)

eliminarArbitroEvento(idA,idE)

modificarArbitro\_Evento(idA,idE)

* Todas las operaciones anteriores son muy similares a añadirDeporte, modificarDeporte y eliminarDeporte, respectivamente. En esta tabla también van a ser importantes las operaciones de consulta como:

Arbitros\_Evento(idE)

pre:

Arbitros\_Evento.filas->exits( f1:filas |

f1.idEvento = idE

post:

return Arbitros\_Evento.f1.idArbitro

###### TABLA DE EMPLAZAMIENTO

* La tabla relativa al emplazamiento nos la proporcionara el ayuntamiento de la ciudad organizadora, por lo que el Access control es del Ayuntamiento. Nosotros solo vamos a poder acceder a la tabla para hacer consultas. Las operaciones que se van a poder realizar son:

añadirEmplazamiento(idE, idD, rest, cap)

eliminarEmplazamiento(idE, idD)

modificarEmplazamiento(idE, idD, rest, cap)

* Estas operaciones son muy similares a añadirDeporte, modificarDeporte eliminarDeporte respectivamente.

exiteEmplazamiento(rest)

pre:

not( Emplazamiento.filas->isEmpty() )

post:

return Emplazamiento.filas->exits(f1:fila|

f1.restriccion=rest)

###### TABLA DE LUGAR\_EVENTO

* El Access control de la tabla es de Gestor de Eventos. Las operaciones que se pueden utilizar son:

añadirLugar\_Evento(idEmplazamiento, idE, fecha)

eliminarLugar\_Evento(idEmplazamiento, idE, fecha)

modificarLugar\_Evento(idEmplazamiento, idE, fecha)

* Todas las operaciones anteriores son muy similares a añadirDeporte, modificarDeporte y eliminarDeporte, respectivamente. En esta tabla también van a ser importantes las operaciones de consulta como:

Lugar\_Evento(idE,fecha)

pre:

Lugar\_Evento.filas->exits( f1:filas |

f1.idEvento = idE and f1.fecha = fecha)

post:

return Lugar\_Evento.f1.idEmplazamiento

###### TABLA DE ASIENTOS

* El Access control pernecene al gestor de Entradas. Las operaciones que se puede realizar en esta tabla son:

insertarAsiento(idAsiento, localizador, idEmplazamiento)

modificarAsiento(idAsiento, localizador, idEmplazamiento)

eliminarAsiento(idAsiento)

###### TABLA DE ENTRADA

* El Access Control es el gestor de entradas y las operaciones que puede realizar son:

insertarEntrada(idEntrada, idAsiento, idEmplazamiento, fecha, estado, precio, comprador, fechaReserva, fechaCaducidad)

modificarEntrada(idEntrada, idAsiento, idEmplazamiento, fecha, estado, precio, comprador, fechaReserva, fechaCaducidad)

eliminarEntrada(idEntrada, idAsiento, idEmplazamiento)

verPrecio(idE, idA, idEmpl)

pre:

Entrada.filas->exits( f1:filas |

f1.idEntrada = idE and f1.idEmplazamiento = idEmpl and f1.idAsiento = idA)

post:

return Entrada.f1.precio

# Patrones

## PATRÓN SINGLETON

El objeto de este patrón es garantizar que una clase sólo tenga una instancia y proporcionar un punto de acceso global a ella.



Usaremos este patrón cuando deba haber exactamente una instancia de una clase y deba ser accesible a los clientes desde un punto de acceso conocido. Por ejemplo:

* Acceso al archivo de configuración de una aplicación
* Gestor de impresión
* Una fábrica de objetos

# Interfaz de usuario

# Investigación sobre los componentes

Primeramente exponer que vamos a utilizar una forma de programación en 3 capas. La capa de presentación, la de negocios y la de datos. La capa de presentación va a ser el componente de la interfaz, la capa de negocios va a ser la propia aplicación y la capa de datos va a ser la base de datos.

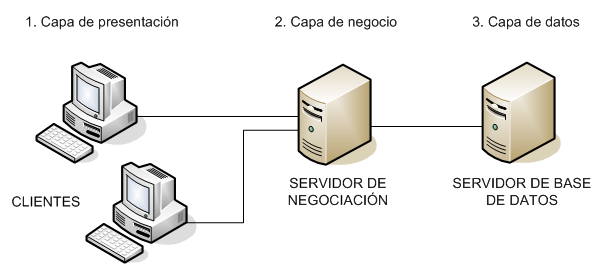


Figura: Modelo en tres capas

## Propuesta para la interfaz

La interfaz se va a desarrollar en lenguaje PHP. Algunas de las ventajas que hemos visto en utilizar este lenguaje para desarrollar nuestra interfaz son:

* Es un lenguaje multiplataforma.
* El código fuente escrito en PHP es invisible al navegador y al cliente ya que es el servidor el que se encarga de ejecutar el código y enviar su resultado HTML al navegador. Esto hace que la programación en PHP sea segura y confiable.
* Posee una amplia documentación en su página oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
* Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.

La interfaz va a consistir básicamente en la parte visual, todos los accesos a la base de datos se van hacer a través de la capa de negocio, la interfaz solo va a llamar a las funciones de esa capa para que se ejecuten. El creador de la interfaz no tiene porque saber el funcionamiento exacto del sistema.

El lenguaje que vamos a utilizar para la aplicación es Java, por lo que tenemos que llamar desde PHP a java, el siguiente enlace nos muestra como se puede llevar a cabo:

<http://gonetil.wordpress.com/2008/07/28/instalar-phpjava-bridge-en-linux/>

## Propuesta para la base de datos

Para desarrollar la nuestra base de datos nos hemos decantado por la última versión estable de Oracle, la Database 11g Enterprise Edition, ya que Oracle tiene el record mundial en rendimiento y en la relación calidad/precio, lo cual nos viene muy bien para nuestro sistema.

Las características más importantes que posee Oracle son:

* Es el gestor de bases de datos más utilizado por las compañías más grandes del mundo, por su robustez y por la seguridad.
* Tiene múltiples versiones para cada sistema operativo, entre ellos UNIX, LINUX, NT, Win 2000- 98 - 95, Novel etc.
* Ofrece soporte mundial a través de sus centros de soporte y sus sitios Web donde se puede encontrar desde scripts hasta documentos de instalación.
* Proporciona actualización de versiones gratis al cliente; siempre y cuando la licencia de soporte lo cubra.
* El 80% o más de los sitios Web en Internet tienen bases de datos Oracle.
* Oracle es más que un gestor de bases de datos. Oracle Corporation ofrece otras soluciones a la plataforma de Negocio, EBusiness, Ecommerce etc.
* Puedes programar con las últimas herramientas del mercado.

## Propuesta para el blog

Nuestra página web posee una parte de noticias, donde se actualizarán las novedades que vayan surgiendo y que gestionaremos por medio de un blog. Para esta parte nos hemos decantado por Wordpress, ya que sólo con instalarlo tienes acceso a los blogs.

Para instalar el componente blog en nuestra página web únicamente hemos incorporando Wordpress a nuestra página en PHP. La forma de hacerlo es bastante sencilla tal y como se muestra en el siguiente enlace:

<http://ezekinho.com.ar/?p=28>

Al registrarse en nuestra aplicación, damos opción al usuario de suscribirse a un servicio de noticias y actualizaciones para que las pueda consultar en su correo o en su teléfono móvil. Para ello vamos a incluir en nuestro blog el protocolo RSS, que es una herramienta para publicar o visualizar artículos de una sola fuente en diferentes medios al mismo tiempo, generalmente al instante de publicarlos. Muchos medios en línea ponen a disposición su feed (dirección única del RSS del sitio) para mostrar los últimos titulares.

En el siguiente enlace se puede ver cómo hacerlo:

<http://www.vidadigital.net/blog/2008/04/21/rss-y-tu-blog-en-wordpresscom/>

## Propuesta para el componente de pago

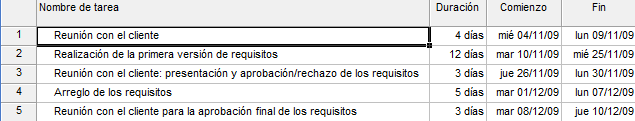
PayPal es un sistema que permite a cualquier persona que tenga una dirección de correo electrónico enviar o recibir dinero online utilizando su tarjeta de crédito de manera totalmente segura.

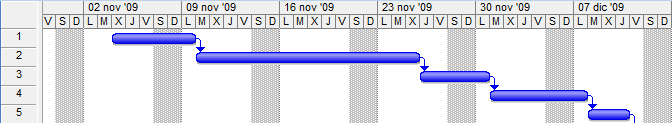
Hemos pensado que sería una forma amena y sencilla de incorporar pagos online para nuestra aplicación, ya que dispone de herramientas guiadas y sencillas para la integración del paquete en la aplicación web. Además, es gratuita.

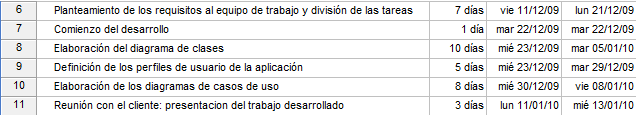
La forma de integrarlo se ve en la siguiente página:

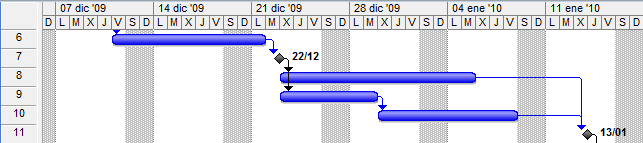
<http://www.elwebmaster.com/articulos/un-script-php-para-integrar-pagos-con-paypal-a-tu-sitio-web>

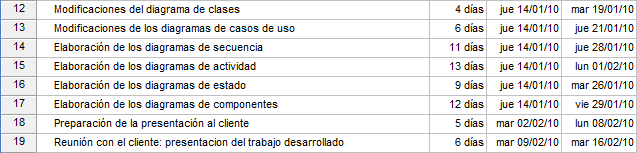
# Planificación estimada

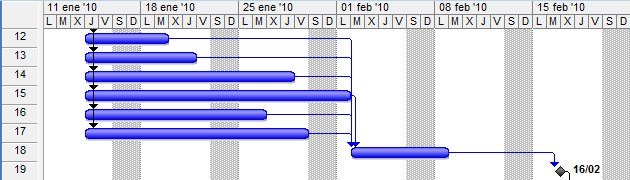


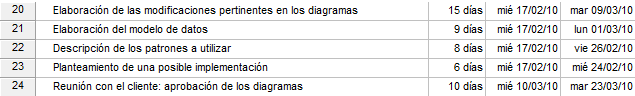


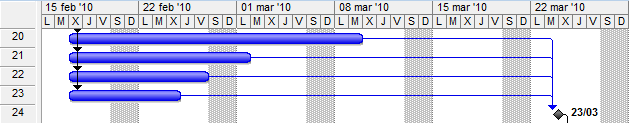




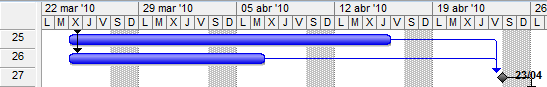


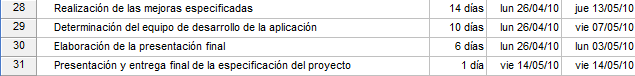


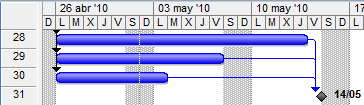












# Propuesta de calidad

# Cuestionario CMMI

###### NIVEL 2 (REPETIBLE)

**A lo largo del desarrollo del proyecto, ¿la forma de trabajo ha sido clara y controlada? ¿Por qué? ¿Se ha seguido algún método, procedimiento y/o entrenamiento?**

Se distinguen dos líneas de opinión muy diferenciadas.

* La forma de trabajo no ha sido clara y controlada desde el principio, y al obtener una visión de cómo y cuánto se avanzaba, se eligió como metodología de trabajo divide y vencerás, es decir, la repartición del trabajo de forma equitativa y con ciertas limitaciones, evitando así tener que consensuar todas las cuestiones que se presentasen (relevantes o irrelevantes), comunicando únicamente las cuestiones de importancia global y con repercusiones.
* La forma de trabajo ha sido clara pero descontrolada, ya que no se eligió ningún método de trabajo determinado y ni se realizo entrenamiento previo a la realización de tareas, lo que conllevo a múltiples suposiciones erróneas y por tanto, a un retroceso (rehacer) en el trabajo.

**¿El proyecto ha tenido una documentación correcta y coherente en cada fase? Argumenta tu respuesta**

Se manifiesta la existencia de opiniones contrapuestas:

* El proyecto no ha tenido una documentación correcta y coherente, ya que los cambios realizados y requerimientos nuevos se comunicaban en las reuniones y no se reflejaban, así como sus motivos y soluciones, en ningún documento escrito. Esta forma de trabajo tuvo como resultado la necesidad de retomar debates ya concluidos.
* El proyecto ha tenido una buena documentación, acorde al desarrollo de las tareas. Se mantuvo una documentación visual que facilitaba la necesidad de otras explicaciones, y se controlaron las versiones de la documentación mediante el repositorio donde se alojo el proyecto durante su desarrollo.

**En todo momento, ¿se ha conocido la documentación a presentar en la entrega final?**

Se distinguen dos líneas de opinión muy diferenciadas.

* Durante el desarrollo del proyecto no ha sido conocida la documentación a presentar, hasta el final del proceso.
* La documentación a presentar ha sido conocida pero de forma inexacta, es decir, no en su totalidad; ya que se desconocía, de forma exacta, el alcance y nivel de profundidad que se deseaba obtener, porque siempre iba en aumento y la principal fuente de conocimiento a lo que se accedió fueron proyectos de años anteriores.

**¿La documentación final ha sido elaborada desde el principio del proyecto?**

Se manifiesta la existencia de opiniones contrapuestas:

* La documentación ha sido elaborada desde el principio, sentando las bases más solidas posibles para obtener la coherencia y consistencia del documento, que ha sido modificado en numerosas ocasiones según aumentaban los requisitos de la entrega así como las modificaciones propias del proyecto, en cuanto a su definición y su alcance.
* La documentación final se ha relegado al último paso en el desarrollo, ya que se entiende como un documento recopilatorio de todo el trabajo realizado, que brinda la oportunidad de realizar una revisión total de todo el proyecto

**¿El tiempo ha sido distribuido de una forma correcta y necesaria en las distintas fases del proyecto?**

La distribución del tiempo no ha sido la idónea, por diversos factores:

* El método de trabajo al principio no era el adecuado, ya que todos los miembros no podían dedicarse a la misma tarea. Por esto motivo, se opto por la repartición de trabajo de forma equitativa, asumiendo una cantidad de tiempo desperdiciada por la mala organización
* La fase inicial (Especificación de Requisitos) se dio por concluida estando en un estado inexacta y demasiado ambicioso, lo que conllevo a múltiples debates acerca de las funcionalidades a desarrollar y el rediseño que obligaba a realizar los cambios decididos en los debates mencionados.

**¿La planificación del trabajo ha sido la adecuada?**

La planificación no ha sido la adecuada, por diversos motivos;

* La falta de un método de trabajo. El intento de que todos los miembros abarcasen todas las tareas conllevo al retraso y a tomar la iniciativa de dividir el trabajo, estrategia que debió emplearse desde el inicio del proyecto.
* Los innumerables debates sobre cualquier cambio. Se debería debatir solo lo imprescindible, es decir, aquellos cambios que tengan una repercusión real y significativa; y comunicar a todos los miembros en reuniones cortas los avances realizados por cada subgrupo.
* Falta de especificación en la idea a desarrollar. Se debería dedicar más tiempo en la especificación de requisitos dejando aclarado en su totalidad, y en medida de lo posible, sin lugar a ambigüedades, las funcionalidades y repercusiones de cada requisito, haciendo consciente al equipo del nivel de ambición del proyecto y el tiempo necesario para desarrollarlo, como también evitando la necesidad de rediseñar.
* Mayor conocimiento del trabajo a realizar desde el inicio del proyecto, facilitando de esta forma la metodología de repartición de tareas.
* Diseñar un estructura o jerarquía con roles bien definidos, existiendo una figura encargada de organizar al resto del equipo según las necesidades del proyecto.

**¿La especificación de requisitos ha sido bien definida desde el inicio?**

La especificación de requisitos no ha estado bien definida, debido a la falta de detalle en su definición y el exceso de ambición debido a las funcionalidades que se planteaban, lo que generó la necesidad de reducir dichas funcionalidad a medida que avanzaba el proyecto, y los innumerables debates causados por las ambigüedades o falta de exactitud en las definiciones; debiendo realizar reestructuraciones y rediseños.

**Porcentaje de requisitos modificados**

Los requisitos han sido modificados en tordo al 60%, siendo una gran parte de estas modificaciones eliminaciones de requisitos. Las modificaciones realizadas han supuesto una gran pérdida de tiempo y esfuerzo, dando lugar a replanteamientos y rediseños, pasando por extensos debates sobre cada modificación.

###### NIVEL 3 (DEFINIDO Y ESTRUCTURADO)

**¿La forma de trabajo ha funcionado bien?**

Se distinguen dos líneas de opinión muy diferenciadas.

* La forma de trabajo ha funcionado bien al final, después de la evolución del grupo hasta conseguir su total integración.
* La forma de trabajo no ha funcionando bien porque no ha sido la adecuada, el proyecto se desarrollo bajo un grupo con una metodología desordenada, en la que todos los miembros trabajaban y opinaban sobre todo generando grandes pérdidas de tiempo y exceso de modificaciones

**¿La definición de cada componente del proyecto ha sido la adecuada?**

Cada componente ha tenido una definición correcta en un punto cercano a la versión definitiva. El principal obstáculo para obtener la definición adecuada eran las distintas modificaciones que se realizaban sobre la misma, ya que cada miembro del grupo tenía una visión diferente, lo que sumía al equipo en deliberaciones hasta obtener un punto en común (olvidado y posteriormente retomado en diversas ocasiones por la falta de documentación)

**¿Las herramientas utilizadas en cada parte del desarrollo del proyecto han sido adecuadas?**

Las herramientas han tenido que reemplazarse en varias ocasiones ya que no abarcaban todas las tareas que se requerían. Entre las limitaciones de las herramientas destacan:

* Incapacidad intrínseca para dividir el trabajo con Subversion (Trabajo en paralelo).
* Incapacidad de integración con otros modelos (OCLs)
* Ausencia de corrección sintáctica (OCLs)

**¿El estado y avance del proyecto ha sido conocido, en todo momento, por cada miembro del equipo?**

Cada miembro tenía a su disposición, en todo momento y en exactitud, conocer el estado y modificaciones realizadas en el proyecto mediante el envío de correos y actualizaciones del repositorio

Aunque el equipo es unánime en cuanto a la disponibilidad para conocer el estado del proyecto, se distinguen opiniones opuestas basadas en la dificultad para conseguirlo:

Todas las modificaciones eran consensuadas y en su mayoría realizadas en las horas lectivas de la asignatura (laboratorios) con la presencia de la mayoría (o totalidad) de los miembros del equipo.

Las actualizaciones en el repositorio se realizaban sin documentar (comentarios explicativos) y de forma autoritaria o no consensuada (ya que se suponía el concilio de opiniones entre todos los miembros), generando discusiones y una gran complejidad por desconocer los motivos respecto a modificaciones hechas.

**¿El método de trabajo ha sido centralizado (reuniones periódicas)?**

El método de trabajo ha sido centralizado ya que las horas lectivas en el laboratorio facilitan esta tarea. Se realizaban reuniones 2 veces por semana de 2 horas de duración para la puesta en común y nueva división de tareas.

**¿Qué porcentaje (%) de tiempo se ha invertido en el aprendizaje necesario para desarrollar el proyecto?**

El aprendizaje necesario se aproxima a un 20 %, ya que gran parte de esta tarea se realizaba en paralelo con las tareas a las que se destinaba el aprendizaje.

En oposición a esta opinión:

* No se invirtió tiempo en aprendizaje ya que lo explicado en las horas lectivas de teoría era suficiente, en suma con otras asignaturas anteriores.

**¿Existe coordinación entre las diferentes tareas de los distintos miembros?**

Existe coordinación ya que al tener un método de trabajo centralizado, todos los miembros conocen las tareas que se desarrollan en cada etapa así como el subgrupo o miembro encargado de dicha tarea.

**¿Qué herramientas han favorecido la coordinación entre los miembros?**

* Repositorio GoogleCode: Olimpiadas2mil
* Grupo de correo (GoogleGroups): Olimpiadas2020
* Eclipse (herramienta de modelado gratuita que interactúa con Subversion)

###### NIVEL 4 (GESTIONADO)

**¿Qué porcentaje (%) de tiempo dedicado al proyecto han sido fructífero?**

La mayoría del equipo estima que aproximadamente un 60% del tiempo dedicado ha sido fructífero, mientras que existe una estimación que considera un 30%.

**¿Qué porcentaje (%) de tiempo se ha dedicado a cada fase del proyecto? ¿Ha sido el adecuado?**

* Especificación de requisitos: 20%
* Diseño (Diagramas, invariantes, precondiciones, postcondiciones,…): 60%
* Memoria y documentación: 20%

No ha sido la distribución adecuada, por no generar un resultado óptimo. La escasa dedicación en la especificación de requisitos conllevo a gran número de debates y forzó a múltiples etapas de rediseño por cambios realizados.

**¿Qué tareas han significado una pérdida de tiempo?**

* Cambios de herramientas (rehacer lo que ya estaba hecho)

Las modificaciones y correcciones en los requisitos han significado un retraso significativo y las discusiones que generaban, un aumento en gran medida de este tiempo desperdiciado, ya que cada debate significaba el replanteamiento de alguno o varios elementos y su correspondiente funcionamiento.

* Diseños realizados (inconclusos) desechados.

**¿El tiempo para realizar una tarea ha podido ser estimado/predicho?**

La estimación de una tarea dependía en gran parte de las complicaciones que generará, debido a la realización de algún cambio y las correspondientes modificaciones que se deben realizar; por este motivo, el equipo se divide en dos posturas:

* Subgrupo que considera que podrían estimarse las tareas (aproximadamente)
* Subgrupo que considera que el tiempo dedicado a realizar una tarea no es predecible.

**¿En cada parte del proyecto, se ha distinguido la figura de un líder?**

No se distingue ninguna figura líder, cada miembro era consciente y responsable de la tarea que había elegido a realizar. Existían miembros más conscientes de las tareas a realizar que servían de guía para el desarrollo del proyecto.

**¿Se han distinguido especialistas en el proyecto?**

No existen especialistas declarados. Durante gran parte del desarrollo del proyecto el trabajo se ha dividido de forma equitativa en subgrupos, permitiendo que todos los miembros pudiesen participar en distintas tareas en las diversas fases.

**¿Existen diferentes puntos de vista sobre la gestión del proyecto?**

Existen tantos puntos de vista como miembros. Al no tener una estructura jerarquizada ni una especificación de requisitos totalmente clara y definida, cada miembro intentaba desarrollar el proyecto bajo su visión personal lo que generaba enfrentamientos (grandes pérdidas de tiempo). A medida que el proyecto tomaba forma y consistencia los debates disminuían, ya que toda la idea pasaba a tener claridad, consistencia y precisión.

**¿El progreso en el proyecto ha sido lineal? Argumenta tu respuesta**

Se divide en dos subgrupos que difieren en:

* El progreso ha sido lineal.
* El progreso del proyecto ha sido por ráfagas, siendo estas positivas (avances significativos en periodos cortos) o negativos (replanteamiento de requisitos, rediseño,… lo que generaba retrasos)

Aunque las ideas de ambos subgrupos se armonizan en la opinión de que en los últimos meses, sujetos a la presión de realizar una entrega, la progresión ha sido exponencial.

**Nivel de madurez obtenido 1,4**

**MODIFICACIONES A APLICAR PARA MEJORAR EL NIVEL DE MADUREZ ALCANZADO**

* Elegir un modelo de desarrollo.
* Revisar cada fase antes de darla por concluida e intentar esclarecer lo más posible cada punto o tarea, para evitar posibles confusiones, discusiones y la necesidad de realizar cambios en tareas concluidas.
* Elección de herramientas, basadas en documentación y experiencia.
* Establecer una jerarquía (elegir un líder), formar especialistas y generar una planificación global y otra dedicada a cada fase.
* Elaborar el documento final desde el principio, para hacernos más conscientes del trabajo ya realizado y las tareas que quedan por hacer.

## Evolución del proyecto

En un principio pensábamos gestionar:

Gestión de las competiciones:

* Creación del cuadro de la competición de cada disciplina.
* Inscripción de los deportistas en las competiciones.

Gestión del emplazamiento:

* Que cumpla con los requisitos del deporte (natación – piscina, tenis – pista de tenis….)
* Que esté bien comunicado con el alojamiento
* Que tenga el aforo adecuado
* Que todos los emplazamientos de la misma disciplina estén lo más cerca posible.

Gestión del horario:

* Que no se solape con una competición de las mismas características
* Cercano a los horarios de máxima audiencia

Gestión de entradas

* Venta de entradas a espectadores
* Pases de prensa para los medios
* Invitaciones para organizadores

Gestión de los participantes:

* Gestión de alojamiento, de forma que el alojamiento este lo más cerca posible o lo mejor comunicado posible de la competición y que este lo más cerca posible del resto de participantes de tu país en otras competiciones.
* Gestión de transporte, para que desde los hoteles a las pistas deportivas haya buena comunicación.

Gestión de los árbitros:

* Gestión de alojamiento: deberían de estar en una zona de la ciudad, aislados de los participantes y todos juntos.
* Gestión del transporte: tendrían un transporte privado para ellos desde su lugar de hospedaje hasta la competición deportiva.

Gestión del personal contratado y voluntario.

* Definición de sueldos.
* Asignación tareas y horarios así como el organigrama de la organización.

Gestión de patrocinadores:

* Asignación de recursos publicitarios en base al grado de colaboración, estos recursos serán gestionados por terceros para la administración de publicidad.
* Gestión de los espacios de publicidad, que incluye el control del cumplimiento de las empresas encargadas de la gestión.
* Acuerdos de colaboración con empresas de transportes para reforzar sus servicios.

Venta de entradas por Internet.

Portal deportivo:

* Mantiene la información de los resultados, medallero, clasificaciones, así como el horario y lugar de las competiciones.

Íbamos a tener los siguientes roles y perfiles:

* ROL: Administración: Posee el control total de la aplicación.
  + - PERFIL: Administrador del Comité de la Sede
* ROL: Control parcial: Solo en algunos aspectos de la aplicación
  + - PERFIL: Autoridad pública (policías, bomberos…)
    - Ejemplo: Hay un accidente grave en la nacional de acceso al evento 3 horas antes. El policía introduce la incidencia y el sistema alerta a la organización, participantes y espectadores que hayan adquirido una entrada por mail o sms. Además en la consulta de rutas de acceso al evento muestra una ruta alternativa para llegar al estadio
    - PERFIL: Federación
    - Ejemplo: Se lesiona 3 horas antes un participante. La aplicación gestiona esa baja e informa a la administración.
    - PERFIL: Espectador
    - Ejemplo: Al comprar una entrada se modifican las localidades libres, se incrementa la recaudación del evento…
    - PERFIL: Medio de comunicación
    - Ejemplo: TVE informa que sus 2 pases de prensa los van a ocupar Pepito Pérez y Juan Rodríguez.
* ROL: Consulta
  + - Cualquier perfil tiene como mínimo el perfil de consulta.
    - Sector servicios: transporte, alojamiento, restaurantes, cines, teatros…
    - Deportivos: participantes, clasificaciones, horarios…

Tras una reunión con personal ajeno a nuestro equipo de desarrollo, decidimos centrarnos en gestionar los eventos y las entradas para estos eventos, y dejar de lado todo lo relacionado con el transporte y alojamiento de los deportistas, árbitros y participantes en general.

También eliminamos la gestión de federaciones (inscripciones de participantes en competiciones etc.)

Eliminamos la publicidad y los pases de prensa.

También eliminamos la gestión de personal porque suponemos que todos esos servicios los subcontratamos.

Añadimos el concepto de ACTA como un informe que el árbitro rellenará al final de cada partido y que proporcionará información relativa a cada evento y que servirá para obtener la información de las competiciones.

En cuanto a los ROLES:

* El COI mantendría un rol de administración
* Hemos eliminado los perfiles de Prensa, Deportista y Autoridad pública, por lo que solo tendría un rol de “Control parcial” el usuario registrado (en la gestión de entradas) y el árbitro (en la gestión de actas)

En cuanto a la funcionalidad:

* Al implementar la interfaz hemos añadido opciones de consulta (noticias, última hora, fotos, villa olímpica, etc)

Conflictos encontrados:

* Antes de eliminar la gestión de transporte hubo discusiones acerca de introducir una clase gestora del transporte que realizara las tareas de reserva de recursos, rutas etc. Pero se descartó porque no se adaptaba a la filosofía de la orientación a objetos.
* En la gestión de entradas pasó algo parecido, teníamos un gestor de entradas que se dedicaba a vender, reservar etc. Decidimos delegar esa funcionalidad en el asiento y esto también generó polémica, pero decidimos dejarlo así.
* En la mitad del proyecto nos dimos cuenta de que estábamos abordando demasiados ámbitos y no nos iba a permitir realizar un buen trabajo con todo, así que decidimos quitar funcionalidades. Sopesamos la situación y decidimos quedarnos únicamente con las entradas, eventos y competiciones.
* En un principio trabajábamos con Bouml, pero daba problemas al trabajar con el SVN porque almacenaba todos los diagramas en un mismo fichero y esto daba conflictos al reconciliar las versiones. Probamos con ArgoUml pero le resultado era el mismo, así que al final nos decantamos por Eclipse, que al guardar cada diagrama en un fichero no nos daba ese problema. Aun así, no sabemos hacer los diagramas de secuencia con el Eclipse y los hemos hecho con ArgoUML.