UNIVERSIDAD DE ORIENTE
NÚCLEO DE ANZOÁTEGUI
ESCUELA DE INGENIERÍA Y CIENCIAS APLICADAS
DEPARTAMENTO DE COMPUTACIÓN Y SISTEMAS

"DESARROLLO DE UN SISTEMA DE GESTIÓN DE LOS PROCESOS ADMINISTRATIVOS DE LA COMISIÓN NACIONAL DE FÚTBOL SALA DE LA FEDERACIÓN VENEZOLANA DE FÚTBOL"

REALIZADO POR:

_____

Pereira Ginestra, Israel

Trabajo de Grado presentado como requisito parcial para obtener el Título de
INGENIERO EN COMPUTACIÓN

Barcelona, junio de 2009

UNIVERSIDAD DE ORIENTE
NÚCLEO DE ANZOÁTEGUI
ESCUELA DE INGENIERÍA Y CIENCIAS APLICADAS
DEPARTAMENTO DE COMPUTACIÓN Y SISTEMAS

“DESARROLLO DE UN SISTEMA DE GESTIÓN DE LOS PROCESOS ADMINISTRATIVOS DE LA COMISIÓN NACIONAL DE FÚTBOL SALA DE LA FEDERACIÓN VENEZOLANA DE FÚTBOL”.

ASESOR:

_____

Ing. José Guevara

Barcelona, junio de 2009

UNIVERSIDAD DE ORIENTE

NÚCLEO DE ANZOÁTEGUI

ESCUELA DE INGENIERÍA Y CIENCIAS APLICADAS

DEPARTAMENTO DE COMPUTACIÓN Y SISTEMAS

"DESARROLLO DE UN SISTEMA DE GESTIÓN DE LOS PROCESOS ADMINISTRATIVOS DE LA COMISIÓN NACIONAL DE FÚTBOL SALA DE LA FEDERACIÓN VENEZOLANA DE FÚTBOL".

JURADO CALIFICADOR:

_____

Ing. José Guevara

Asesor Académico

_____ _____

Ing. Gabriela Veracierta

Jurado Principal

Ing. Víctor Mujica

Jurado Principal

Barcelona, junio de 2009

RESOLUCIÓN

ARTÍCULO Nº 44 Del Reglamento de Trabajo de Grado

"Los trabajos de grado son de exclusiva propiedad de la Universidad y sólo podrán ser utilizados para otros fines con el conocimiento del Consejo de Núcleo respectivo, quién lo participará al Consejo Universitario".

DEDICATORIA

A Ana Isabel y José Luis.

A Daniel y Vanessa.

A ti Preciosa.

A Todos Aquellos que Nunca Creyeron En Mí.

Y por Último, a Todos Aquellos que Nunca Dejaron de Hacerlo.

AGRADECIMIENTOS

A mis padres, por ser tan diferentes y por tener tanta variedad de virtudes y defectos, por traerme a este mundo y guiarme.

A mis hermanos, por enseñarme lo que debo y no debo ser y hacer. A Daniel sobre todo por introducirme en este mundo y por siempre estar ahí cuando lo necesité.

A mis familiares, que siempre creyeron en mí y estuvieron demasiado pendientes de todo lo concerniente a mí y mi carrera.

A ti Bella! Que a pesar de la distancia y los problemas, siempre estuviste ahí para apoyarme, para impulsarme y para halarme las orejas cuando me hizo falta…

A todos mis amigos, por hacer este trayecto un poco más agradable, aunque en algunos casos todo lo contrario hehehe... Por acompañarme todos los días en la universidad, por las eternas charlas en el pasillo y por el “Fantasmeo” en general. Cuenten conmigo para lo que necesiten…

A todos los profesores que eventualmente a su vez se transformaron en amigos, por su apoyo y sus enseñanzas.

CONTENIDO

RESOLUCIÓN ..... **IV**

DEDICATORIA ..... **V**

AGRADECIMIENTOS ..... **VI**

CONTENIDO ..... **VII**

INDICE DE TABLAS ..... **XIV**

INDICE DE FIGURAS ..... **XVI**

RESUMEN ..... **XIX**

INTRODUCCIÓN **1**

CAPÍTULO I. EL PROBLEMA **3**

1.1 PLANTEAMIENTO DEL PROBLEMA 3

1.2 OBJETIVOS 3

1.2.1 Objetivo General *3*

1.2.2 Objetivos Específicos *3*

1.3 EL PROPÓSITO 4

1.4 IMPORTANCIA 5

1.5 ALCANCE 5

1.6 ORIGINALIDAD 5

CAPÍTULO II. MARCO TEÓRICO **6**

2.1 ANTECEDENTES 6

2.2 BASES TEÓRICAS 8

2.2.1 Fútbol Sala *8*

2.2.1.1 Historia 8

2.2.1.2 Definición 8

2.2.1.3 Aspectos Físicos 9

2.2.1.4 Aspectos Técnicos 10

2.2.1.5 Aspectos Tácticos 10

2.2.2 Programación Orientada a Objetos *11*
2.2.2.1 Clases 12
2.2.2.1.1 Atributos 12
2.2.2.1.2 Métodos 12
2.2.2.2 Propiedades 12
2.2.2.2.1 Abstracción 13
2.2.2.2.2 Encapsulamiento 13
2.2.2.2.3 Principio de Ocultación 13
2.2.2.2.4 Polimorfismo 13
2.2.2.2.5 Herencia 14
2.2.3 C# *14*
2.2.4 Bases de Datos *15*
2.2.4.1 Clasificación 15
2.2.4.2 SQL 17
2.2.4.3 Sistemas Manejadores de Bases de Datos 17
2.2.5 Aplicaciones WEB *19*
2.2.5.1 ASP.NET 20
2.2.5.2 Silverlight 20
2.2.6 Ingeniería de Software *21*
2.2.7 Proceso Unificado de Rational (RUP): *23*
2.2.7.1 Principios 23
2.2.7.2 Ciclo de Vida 24
2.2.8 Lenguaje Unificado de Modelado (UML) *25*
2.2.8.1 Definición 25
2.2.8.2 Elementos de UML 26
2.2.8.3 Diagramas de UML 27

CAPÍTULO III. FASE DE INICIO 32

3.1 INTRODUCCIÓN 32
3.2 REQUERIMIENTOS 32
3.3 DIAGRAMA DE DOMINIO 33
3.4 DIAGRAMAS DE CASOS DE USO DEL SISTEMA 35
3.4.1 Actores *35*
3.4.2 Diagrama de Caso de Uso *35*
3.4.3 Descripción del flujo de sucesos del Caso de Uso Torneos *38*
3.4.4 Descripción del flujo de sucesos del Caso de Uso de Equipos *41*

3.4.5 Descripción del flujo de sucesos del Caso de Uso de Jugadores *45*
3.4.6 Descripción del flujo de sucesos del Caso de Uso de Entrenadores *47*
3.5 DIAGRAMAS DE CLASES DE ANÁLISIS DEL SISTEMA 50
3.5.1 Diagrama de Clase de Análisis del Sistema *51*
3.5.2 Diagrama de Clase de Análisis de Torneos *52*
3.5.3 Diagrama de Clase de Análisis de Equipos *53*
3.5.4 Diagrama de Clase de Análisis de Jugadores *54*
3.5.5 Diagrama de Clase de Análisis de Entrenadores *55*
3.6 DIAGRAMAS DE COLABORACIÓN DEL SISTEMA 56
3.6.1 Diagrama de Colaboración General del Sistema *57*
3.6.2 Diagrama de Colaboración "Administrar Torneos" *57*
3.6.3 Diagrama de Colaboración "Administrar Equipos" *57*
3.6.4 Diagrama de Colaboración "Administrar Jugadores" *58*
3.6.5 Diagrama de Colaboración "Administrar Entrenadores" *58*
3.7 DIAGRAMAS DE PAQUETES DE ANÁLISIS 60
3.7.1 Diagrama de Paquetes de Análisis *61*
3.7.2 Diagrama de Paquetes de Análisis Torneos *61*
3.7.3 Diagrama de Paquetes de Análisis Equipos *62*
3.7.4 Diagrama de Paquetes de Análisis Jugadores *62*
3.7.5 Diagrama de Paquetes de Análisis Entrenadores *62*
3.8 CONCLUSIÓN DE LA FASE DE INICIO 63

CAPÍTULO IV. FASE DE ELABORACIÓN 65

4.1 INTRODUCCIÓN 65
4.2 ANÁLISIS 65
4.2.1 Casos de Uso *66*
4.2.1.1 Descripción del flujo de sucesos del Caso de Uso Torneos 66
4.2.2 Diagrama de Clases de Análisis *68*
4.2.2.1 Diagrama de Clase de Análisis de Torneos 68
4.2.3 Diagrama de Colaboración *69*
4.2.3.1 Diagrama de Colaboración "Torneos" 69
4.2.4 Identificación de los Paquetes de Análisis *70*
4.3 DISEÑO DE LA ARQUITECTURA DEL SISTEMA 70
4.3.1 Identificación de las Clases de Diseño *70*
4.3.1.1 Clase de Diseño Torneos 71

4.3.1.2 Clase de Diseño Equipos ... 72
4.3.1.3 Clase de Diseño Jugadores ... 72
4.3.1.4 Clase de Diseño Entrenadores ... 73
4.3.2 Identificación de los Paquetes de Diseño ... 74
4.3.2.1 Diagrama de Paquetes de Diseño Torneos ... 74
4.3.2.2 Diagrama de Paquetes de Diseño Equipos ... 74
4.3.2.3 Diagrama de Paquetes de Diseño Jugadores ... 75
4.3.2.4 Diagrama de Paquetes de Diseño Entrenadores ... 75
4.3.3 Diagrama de Clases ... 75
4.3.3.1 Diagrama de Clases del Sistema ... 76
4.3.3.1.1 Clase Torneos ... 76
4.3.3.1.2 Clase Jugadores ... 77
4.3.3.1.3 Clase Equipos ... 77
4.3.3.1.4 Clase Entrenadores ... 77
4.3.3.1.5 Clase Admin ... 77
4.3.4 Diagrama de Capas ... 78
4.3.5 Diagramas de Secuencia ... 78
4.3.5.1 Diagrama de Secuencia “Agregar Estadísticas A Torneo” ... 80
4.3.5.2 Diagrama de Secuencia “Agregar Jugador a Equipo” ... 82
4.3.6 Diseño de la Base de Datos ... 83
4.3.6.1 Tablas de la Base de Datos ... 83
4.3.6.1.1 Tabla Admin ... 83
4.3.6.1.2 Tabla CantEquiGrupos ... 84
4.3.6.1.3 Tabla Clasificados ... 84
4.3.6.1.4 Tabla Entrenadores ... 85
4.3.6.1.5 Tabla EntrenadoresEquipos ... 86
4.3.6.1.6 Tabla Equipos ... 86
4.3.6.1.7 Tabla EquiposParticipantes ... 87
4.3.6.1.8 Tabla EstadisticasEquipo ... 87
4.3.6.1.9 Tabla EstadisticasJugador ... 88
4.3.6.1.10 Tabla Jugadores ... 89
4.3.6.1.11 Tabla JugadoresEquipos ... 89
4.3.6.1.12 Tabla Partidos ... 90
4.3.6.1.13 Tabla PosicionesLiga ... 90
4.3.6.1.14 Tabla PosicionesLigaNacional ... 91
4.3.6.1.15 Tabla Sancionados ... 92
4.3.6.1.16 Tabla PosicionesLigaNacional ... 92

4.3.7 Diseño de la Interfaz de Usuario ... *93*
4.3.7.1 Entrada ... 93
4.3.7.2 Torneos ... 94
4.3.7.3 Jugadores ... 95
4.3.7.4 Entrenadores ... 96
4.3.7.5 Equipos ... 97
4.3.7.6 Admin ... 98
4.3.8 Diseño del Entorno WEB ... *98*
4.3.8.1 Interfaz del Entorno WEB ... 99
4.4 IMPLEMENTACIÓN ... 99
4.4.1 Procedimientos Almacenados ... *100*
4.4.1.1 Obtener Torneos ... 101
4.4.1.2 Obtener Jugadores ... 101
4.4.1.3 Obtener Equipos ... 101
4.4.1.4 Obtener Agentes Libres ... 101
4.4.1.5 Obtener Entrenadores Libres ... 102
4.4.1.6 Obtener Plantilla ... 102
4.4.1.7 Obtener Partidos de una Jornada ... 102
4.4.1.8 Obtener Estadísticas de Partido ... 103
4.4.1.9 Obtener Estadísticas de Jugadores en Partido ... 103
4.4.1.10 Obtener Tabla ... 103
4.4.1.11 Obtener Tabla Liga Nacional ... 104
4.4.1.12 Obtener Tabla Clasificados ... 104
4.4.1.13 Obtener Equipos No Participantes ... 104
4.4.2 Implementación del Entorno Web ... *105*
4.4.2.1 Implementación de la Interfaz Gráfica ... 105
4.4.2.2 Implementación de los Servicios Web ... 111
4.4.2.2.1 Obtener Jugadores de Equipo ... 111
4.4.2.2.2 Obtener Próximos Partidos de Torneo ... 112
4.4.2.2.3 Obtener Últimos Partidos Disputados de Torneo ... 113
4.5 PRUEBAS ... 113
4.5.1 Partición Equivalente ... *114*
4.5.1.1 Identificación de las Clases de Equivalencia ... 114
4.5.1.2 Grupo de Tipos de Entrada de Datos Almacenar Jugador ... 114
4.5.1.2.1 Aplicación de Casos de Prueba ... 114
4.5.1.3 Grupo de Tipos de Entrada de Datos Almacenar Entrenador ... 115
4.5.1.3.1 Aplicación de Casos de Prueba Almacenar Entrenador ... 115

4.5.1.4 Grupo de Tipos de Entrada de Datos Almacenar Torneo 116
4.5.1.4.1 Aplicación de Casos de Prueba Almacenar Torneo 116
4.5.1.5 Grupo de Tipos de Entrada de Datos Almacenar Equipo 117
4.5.1.5.1 Aplicación de Casos de Prueba Almacenar Equipo 117
4.5.1.6 Grupo de Tipos de Entrada de Datos Almacenar Estadísticas de Equipo 118
4.5.1.6.1 Aplicación de Casos de Prueba Almacenar Estadísticas de Equipo 118
4.5.1.7 Grupo de Tipos de Entrada de Datos Administrador 118
4.5.1.7.1 Aplicación de Casos de Prueba Almacenar Estadísticas de Equipo 119
4.5.2 Consistencia de Datos 119
4.6 CONCLUSIÓN DE LA FASE DE ELABORACIÓN 120

CAPÍTULO V. FASE DE CONSTRUCCIÓN 121

5.1 INTRODUCCIÓN 121
5.2 ANÁLISIS 121
5.3 DISEÑO 121
5.3.1 Diagrama de Clases 122
5.3.1.1 Diagrama de Clases del Sistema 122
5.3.1.1.1 Clase GenerarPDF 122
5.3.1.1.2 Clase Imagen 123
5.3.1.1.3 Clase Conexión 123
5.3.2 Diagrama de Secuencia 124
5.3.2.1 Diagrama de Secuencia “Generar Reporte de Torneo” 124
5.3.2 Diseño de la Interfaz de Usuario 124
5.3.2.1 Ver Torneos 125
5.3.2.2 Ver Partidos Disponibles de Torneo 126
5.3.2.3 Agregar Estadísticas de Equipos en Partido 126
5.3.2.4 Agregar Estadísticas de Jugadores en Partido 127
5.3.3 Diseño del Entorno WEB 128
5.3.3.1 Agenda 128
5.3.3.2 Selecciones 129
5.4 IMPLEMENTACIÓN 130
5.4.1 Procedimientos Almacenados 130
5.4.1.1 Obtener Equipo 131
5.4.1.2 Obtener Plantilla No Sancionada 131
5.4.1.3 Obtener Partidos Jugados 131
5.4.1.4 Obtener Partidos No Jugados 132
5.4.1.5 Obtener Sancionados 132

5.4.2 Implementación de la Interfaz de Usuario ... *132*
5.4.2.1 Ver Partidos Disponibles de Torneo ... 132
5.4.2.2 Agregar Estadísticas de Equipos en Partido ... 136
5.4.2.3 Agregar Estadísticas de Jugadores en Partido ... 140
5.4.3 Implementación del Entorno WEB ... *153*
5.4.3.1 Agenda ... 154
5.4.3.2 Selecciones ... 155
5.5 PRUEBAS ... 160
5.5.1 Pruebas de Unidad ... *160*
5.5.1.1 Administrar Torneos ... 161
5.5.1.2 Administrar Equipos ... 161
5.6 CONCLUSIÓN DE LA FASE ... 162

CAPÍTULO VI. FASE DE TRANSICIÓN ... 163

6.1 INTRODUCCIÓN ... 163
6.2 ANÁLISIS ... 163
6.3 DISEÑO ... 163
6.4 IMPLEMENTACIÓN ... 164
6.4.1 Procedimientos Almacenados ... *164*
6.4.1.1 Modificar Estadísticas de Equipos ... 164
6.4.1.2 Modificar Estadísticas de Jugadores ... 165
6.4.1.3 Modificar Sanción ... 165
6.5 PRUEBAS ... 165
6.6 CONCLUSIÓN DE LA FASE DE TRANSICIÓN ... 166

CONCLUSIONES ... 167

RECOMENDACIONES ... 169

BIBLIOGRAFÍA CITADA ... 170

BIBLIOGRAFÍA ADICIONAL ... 172

ANEXO A. MANUAL DE USUARIO ... ¡ERROR! MARCADOR NO DEFINIDO.

METADATOS PARA TRABAJOS DE GRADO, TESIS Y ASCENSO: ... 173

INDICE DE TABLAS

TABLA 3.1. MATRIZ DE REQUERIMIENTOS DEL SISTEMA 33
TABLA 3.2. ACTORES DEL SISTEMA..... 35
TABLA 3.3. CASO DE USO JUGADORES 36
TABLA 3.4. CASO DE USO EQUIPOS..... 37
TABLA 3.5. CASO DE USO ENTRENADORES 37
TABLA 3.6. CASO DE USO TORNEOS 37
TABLA 4.1 TABLA ADMIN..... 84
TABLA 4.2 TABLA CANTEQUIGRUPOS 84
TABLA 4.3 TABLA CLASIFICADOS..... 85
TABLA 4.4 TABLA ENTRENADORES..... 85
TABLA 4.5 TABLA ENTRENADORESEQUIPOS..... 86
TABLA 4.6 TABLA EQUIPOS 86
TABLA 4.7 TABLA EQUIPOS 87
TABLA 4.8 TABLA ESTADISTICASEQUIPO..... 87
TABLA 4.9 TABLA ESTADISTICASJUGADOR..... 88
TABLA 4.10 TABLA JUGADORES..... 89
TABLA 4.11 TABLA JUGADORESEQUIPOS 90
TABLA 4.12 TABLA PARTIDOS..... 90
TABLA 4.13 TABLA POSICIONESLIGA..... 91
TABLA 4.14 TABLA POSICIONESLIGANACIONAL 91
TABLA 4.15 TABLA SANCIONADOS..... 92
TABLA 4.16 TABLA POSICIONESLIGANACIONAL 93

TABLA 4.17. CASOS DE PRUEBA ALMACENAR JUGADOR 115
TABLA 4.18. CASOS DE PRUEBA ALMACENAR ENTRENADOR 116
TABLA 4.19. CASOS DE PRUEBA ALMACENAR TORNEO 116
TABLA 4.20. CASOS DE PRUEBA ALMACENAR EQUIPO 117
TABLA 4.21. CASOS DE PRUEBA ALMACENAR ESTADÍSTICAS DE EQUIPO 118
TABLA 4.22. CASOS DE PRUEBA ALMACENAR ESTADÍSTICAS DE EQUIPO 119

INDICE DE FIGURAS

FIGURA 3.1. DIAGRAMA DE DOMINIO 34
FIGURA 3.2. CASO DE USO DEL SISTEMA 36
FIGURA 3.3. CASO DE USO TORNEOS 38
FIGURA 3.4 CASO DE USO EQUIPOS 42
FIGURA 3.5 CASO DE USO JUGADORES 45
FIGURA 3.6 CASO DE USO ENTRENADORES 48
FIGURA 3.7. DIAGRAMA DE CLASE DE ANÁLISIS DEL SISTEMA 51
FIGURA 3.8 DIAGRAMA DE CLASE DE ANÁLISIS DE TORNEOS 52
FIGURA 3.9 DIAGRAMA DE CLASE DE ANÁLISIS DE EQUIPOS 53
FIGURA 3.10 DIAGRAMA DE CLASE DE ANÁLISIS DE JUGADORES 54
FIGURA 3.11 DIAGRAMA DE CLASE DE ANÁLISIS DE ENTRENADORES 55
FIGURA 3.12 DIAGRAMA DE COLABORACIÓN GENERAL DEL SISTEMA 57
FIGURA 3.13 DIAGRAMA DE COLABORACIÓN “ADMINISTRAR TORNEOS” 58
FIGURA 3.14 DIAGRAMA DE COLABORACIÓN “ADMINISTRAR EQUIPOS” 59
FIGURA 3.15 DIAGRAMA DE COLABORACIÓN “ADMINISTRAR JUGADORES” 59
FIGURA 3.16 DIAGRAMA DE COLABORACIÓN “ADMINISTRAR ENTRENADORES” 60
FIGURA 3.17 DIAGRAMA DE PAQUETES DE ANÁLISIS 61
FIGURA 3.18 DIAGRAMA DE PAQUETES DE TORNEOS 61
FIGURA 3.19 DIAGRAMA DE PAQUETES DE EQUIPOS 62
FIGURA 3.20 DIAGRAMA DE PAQUETES DE JUGADORES 62
FIGURA 3.21 DIAGRAMA DE PAQUETES DE ENTRENADORES 63
FIGURA 4.1. CASO DE USO ADMINISTRAR TORNEOS 66

FIGURA 4.2. DIAGRAMA DE CLASE DE ANÁLISIS DE TORNEOS 68
FIGURA 4.3. DIAGRAMA DE COLABORACIÓN “TORNEOS” 69
FIGURA 4.4 DIAGRAMA DE PAQUETES DE TORNEOS 70
FIGURA 4.5 CLASE DE DISEÑO TORNEOS 71
FIGURA 4.6 CLASE DE DISEÑO EQUIPOS 72
FIGURA 4.7 CLASE DE DISEÑO JUGADORES 73
FIGURA 4.8 CLASE DE DISEÑO ENTRENADORES 73
FIGURA 4.9 DIAGRAMA DE PAQUETES DE DISEÑO TORNEOS 74
FIGURA 4.10 DIAGRAMA DE PAQUETES DE DISEÑO EQUIPOS 74
FIGURA 4.11 DIAGRAMA DE PAQUETES DE DISEÑO JUGADORES 75
FIGURA 4.12 DIAGRAMA DE PAQUETES DE DISEÑO ENTRENADORES 75
FIGURA 4.13. DIAGRAMA DE CLASES DEL SISTEMA GENERAL 76
FIGURA 4.14 DIAGRAMA DE CAPAS DE LA APLICACIÓN 79
FIGURA 4.15 DIAGRAMA DE SECUENCIA “AGREGAR ESTADÍSTICAS A TORNEO” 81
FIGURA 4.16 DIAGRAMA DE SECUENCIA “AGREGAR JUGADOR A EQUIPO” 82
FIGURA 4.17. VENTANA PRINCIPAL 94
FIGURA 4.18 INTERFAZ TORNEOS 95
FIGURA 4.19 INTERFAZ JUGADORES 96
FIGURA 4.20 INTERFAZ ENTRENADORES 97
FIGURA 4.21 INTERFAZ EQUIPOS 98
FIGURA 4.22 INTERFAZ ADMIN 99
FIGURA 4.23 FORMATO DE LA PÁGINA WEB 100
FIGURA 5.1. DIAGRAMA DE CLASES DEL SISTEMA GENERAL 123
FIGURA 5.2 DIAGRAMA DE SECUENCIA “GENERAR REPORTE DE TORNEO” 124
FIGURA 5.3 INTERFAZ VER TORNEOS 125
FIGURA 5.4 VER PARTIDOS 126

FIGURA 5.5 AGREGAR ESTADÍSTICAS DE EQUIPOS EN PARTIDOS 127

FIGURA 5.6 AGREGAR ESTADÍSTICAS DE JUGADORES EN PARTIDO 128

FIGURA 5.7 AGENDA 129

FIGURA 5.8. SELECCIONES 130

FIGURA 5.9 PRUEBA ADMINISTRAR TORNEOS 161

FIGURA 5.10. PRUEBA ADMINISTRAR EQUIPOS 162

RESUMEN

En el presente trabajo se desarrolla una aplicación que permite gestionar todos los procesos administrativos de la Comisión Nacional de Fútbol Sala. El software desarrollado brinda al usuario la posibilidad de administrar torneos, equipos, jugadores y entrenadores, y a su vez permite la generación de reportes que faciliten la visualización de estadísticas para el público en general. La aplicación fue realizada utilizando el lenguaje de programación orientado a objetos C# y el sistema manejador de bases de datos SQLServer pertenecientes al entorno de desarrollo integrado Microsoft Visual Studio 2008. Para el desarrollo del proyecto se utilizó la metodología del Proceso Unificado de Rational (RUP), utilizando el análisis y diseño iterativo por medio del Lenguaje Unificado de Modelado (UML).

INTRODUCCIÓN

En Venezuela el Fútbol Sala se practica a nivel semi-profesional desde 1993, cuando se creó la Liga Especial de Fútbol 5, con una participación de 8 clubes. Actualmente la F.V.F. a través de su Comisión Nacional de Fútbol Sala organiza todos los años la Liga Nacional, la cual constituye el evento de mayor relevancia de esta disciplina deportiva. Junto con el baloncesto e incluso el beisbol, es uno de los deportes más populares del país, de hecho en los últimos años, esta disciplina deportiva ha alcanzado un grandísimo nivel de popularidad entre sus aficionados.

La tecnología puede ayudar en gran parte a la difusión de información acerca de este deporte ya que cuenta con el medio de información más grande del mundo, como lo es el Internet, que permite llevar toda la actualidad del fútbol sala nacional a la comodidad de los hogares del país a un costo muy bajo y en tiempo real. Otra ventaja de utilizar esta plataforma, proviene del uso de aplicaciones o programas de computadoras, que facilitan cualquier tipo de tareas, y que para este caso, pueden optimizar todo tipo de procesos que actualmente se realizan de forma manual, siendo posible ejecutar dichos programas en cualquier computador personal, permitiendo así la fácil y cómoda distribución de los mismos.

El Capítulo I, presenta la problemática que da origen a la investigación, aquí se muestran los objetivos que se persiguen en la investigación, así como también la justificación del trabajo y sus limitaciones.

El Capítulo II, muestra la base de sustanciación documental de la presente investigación, las investigaciones precedentes en el entorno de los sistemas

administrativos y seguidamente las teorías, preceptos y conceptos relevantes para entender la terminología de este estudio.

En el Capítulo III, se presenta la primera fase del proceso unificado (RUP) Inicio (Inception), donde se hace un plan de fases, se identifican los principales casos de uso y riesgos del sistema a desarrollar, permitiendo esto establecer los requisitos que debe cumplir la aplicación y elaborar el conjuntos de modelos que describirán el comportamiento de la misma.

El Capítulo IV, muestra la segunda fase del proceso unificado de desarrollo de software, conocida como elaboración, el objetivo de esta fase es la definición de la estructura interna de la aplicación el cual proporcionará unas buenas bases para las siguientes fases de diseño e implementación. La definición de la arquitectura debe tener en cuenta los requerimientos obtenidos durante la fase anterior de inicio o concepción.

El Capítulo V, muestra la fase de construcción del RUP, en la que se observan los componentes de software, la integración y las pruebas realizadas luego de la misma.

CAPÍTULO I. EL PROBLEMA

1.1 Planteamiento del Problema

Uno de los problemas que se presentan en la actualidad con respecto a este deporte, es que no se le ha dado cabida a la plataforma tecnológica en el mismo, y por lo tanto, la gestión de los torneos (ya sean regionales o nacionales), se realiza mediante un proceso manual de rellenado de planillas, lo cual supone un trabajo tedioso e ineficiente. Aunado a esto, se dificulta la difusión de las estadísticas del torneo, de los equipos y de los jugadores, al público en general y a los entrenadores, ya que estos datos se almacenan en papel, evitando así poder observar las progresiones atléticas de los deportistas, su rendimiento en el torneo o su información personal. El cálculo de todas las estadísticas del torneo como los goles por partido de un jugador, minutos jugados, asistencias por partido, se deben realizar también de forma manual lo cual no es óptimo y significa un retraso considerable a la presentación de dichas estadísticas a los entrenadores, jugadores y público en general.

1.2 Objetivos

1.2.1 Objetivo General

- Desarrollar una aplicación, que permita la automatización de la gestión de eventos y torneos de futbol sala regionales y nacionales, para la Federación Venezolana de Fútbol.

1.2.2 Objetivos Específicos

- Realizar el análisis y la especificación de requerimientos del sistema.

- Diseñar el modelo de datos que sustentará la base de datos relacional del sistema, la interfaz de la aplicación que permitirá la interacción de los usuarios con el sistema y el entorno WEB de la aplicación.

- Diseñar los subsistemas que se encargarán de la gestión administrativa y deportiva de los procesos de la Comisión Nacional de Fútbol Sala.
- Codificar los diferentes subsistemas que componen la totalidad de la aplicación.
- Integrar los subsistemas en una sola aplicación, para la depuración y corrección de fallas.
- Elaborar la documentación técnica que sirva de apoyo a los usuarios del sistema.

1.3 El Propósito

Este trabajo de investigación se plantea el desarrollo de una aplicación WEB que será utilizada en el sitio virtual de la Comisión Nacional de Fútbol Sala de la Federación Venezolana de Fútbol, que ayudará a dar publicidad al deporte, que permitirá ofrecer una gran gama de información (incluidos eventos, tablas de los torneos, calendario deportivo, etc.) a todos los aficionados del mismo, y que a su vez, automatizará el proceso de actualización de la página WEB en todos sus diferentes módulos, haciendo más fácil el trabajo de los administradores del mismo.

También se plantea el desarrollo de una aplicación Stand-Alone, para computadores personales, que ayude a los organizadores de los torneos (nacionales y regionales), a la gestión de los mismos, permitiendo agilizar y automatizar el proceso de creación de torneos nuevos, almacenamiento de información de los equipos, de los jugadores, y de los entrenadores, y que también sirva para informar a todos sus

usuarios y público en general, de la actualidad del torneo en disputa, de su calendario de partidos, y de próximos torneos a realizar.

1.4 Importancia

Con la puesta en marcha del nuevo software se contribuirá significativamente en la eficiencia del proceso de creación de torneos, de generación de estadísticas, de almacenamiento de datos e información de jugadores, equipos y cuerpo técnico, optimizando las horas de trabajo al máximo al reducir notablemente el tiempo que toma realizar los procesos anteriormente mencionados, llevándolos de ser manuales a totalmente automatizados.

1.5 Alcance

Se busca a la vez, impulsar el desarrollo del deporte en el país, y esto se hará mediante el uso del nuevo software para la página WEB de la Federación, el cual buscará atraer grandes volúmenes de personas, para proveerlas así de toda la información del fútbol sala nacional.

1.6 Originalidad

Este proyecto es el primero a nivel nacional que se propone gestionar torneos de fútbol sala ya que el mayor ente concerniente a este deporte en el país como lo es la Comisión Nacional de Fútbol Sala, realiza actualmente todos los trámites y procesos de forma manual. Representa un punto de partida para futuras ampliaciones y actualizaciones que se puedan agregar por otros desarrolladores conforme con requerimientos que puedan surgir.

CAPÍTULO II. MARCO TEÓRICO

2.1 Antecedentes

- Para el año 2004 el estudiante Adrian Moya realizó el proyecto titulado "**Desarrollo de un Sistema para la Automatización del proceso de Admisión y Control de Estudios en un Instituto Universitario"** para optar al título de Ingeniero en Computación. Este proyecto se realizó con el fin de automatizar las actividades de registro y control de calificaciones de los estudiantes en el Instituto Universitario Tecnológico Superior de Oriente, motivado debido al crecimiento de la población estudiantil. Se utilizó la metodología del Proceso Unificado de Desarrollo de Software, y el lenguaje de programación Delphi. [1]

- En el año 2004 el estudiante Miguel Simoni realizó el proyecto titulado **"Desarrollo de un Sistema de Información para la Automatización de los procesos realizados en el Departamento de Ciencias de la Unidad de Estudios Básicos de la Universidad de Oriente, Núcleo Anzoátegui"** para optar al título de Ingeniero en Computación. La finalidad del proyecto fue automatizar los procesos administrativos gestionados por dicho departamento, tales como: la creación de la programación académica, la realización de inscripciones por departamento, cambios de sección, retiros de carga académica y los diferentes procesos realizados a las actas de evaluación. Se utilizó la metodología del Proceso Unificado de Desarrollo de Software, el lenguaje de programación orientado a objetos Visual C# .NET y el manejador de bases de datos SQL Server 2000. [2]

- En el año 2005 las estudiantes Andreína Díaz y Zulhermys Guerra desarrollaron el proyecto titulado **"Actualización del Software Educativo Introducción a la Ingeniería en Computación, creado como herramienta didáctica para el proceso Enseñanza-Aprendizaje de la unidad curricular Introducción a la Ingeniería en Computación"** para optar al título de Ingeniero en Computación. El proyecto se realizó con el fin de actualizar el software anteriormente creado y añadirle 5 capítulos que no fueron desarrollados para la primera versión, con el propósito de hacer del software una herramienta educativa completa. Se utilizó la metodología del Proceso Unificado de Desarrollo de Software, Flash MX Professional para la creación de escenarios y películas en el ambiente WEB, Fireworks MX para la edición de imágenes, y la tecnología de Microsoft ASP.NET para la manipulación de una base de datos de SQL Server. [3]

- Para el año 2007 el estudiante Leoncio Núñez realizó el proyecto titulado **"Desarrollo de una aplicación WEB para la visualización en tiempo real de los parámetros operacionales de producción de la empresa PDVSA"** para optar al título de Ingeniero en Computación. La finalidad del proyecto fue el utilizar la plataforma de software libre, para el monitoreo en tiempo real de los parámetros operacionales de producción de la empresa PDVSA, tales como la temperatura, la presión y el voltaje. Se utilizó la metodología del Proceso Unificado de Desarrollo de Software, Macromedia Dreamweaver para la programación de los módulos, la tecnología SVG para la creación en tiempo real de gráficas, PHP para la capa de negocios del servidor, ECMAScript para la lógica de la aplicación y MySQL como manejador de bases de datos. [4]

2.2 Bases Teóricas

2.2.1 Fútbol Sala

2.2.1.1 Historia

La Federación Internacional de Fútbol Asociación (F.I.F.A) tiene admitida dentro de su organización la modalidad "Fútbol Sala" organizando las competencias Internacionales en los años 1989, 1993, 1996, 2000, 2004 y 2008, participando nuestra Selección Nacional en las eliminatorias de los Mundiales de 1996, 2000, 2004 y 2008.

La Federación Venezolana de Fútbol (FVF) creó la Comisión de Fútbol 5, actualmente denominada Comisión Nacional de Fútbol Sala (CONAFUTSALA), con sus respectivas Comisiones Regionales que desarrollan torneos locales en diferentes categorías y conforman las Selecciones Estadales para participar en los Campeonatos Nacionales.

A partir del año 2000 numerosas organizaciones que venían desarrollando la actividad de Fútbol en "Cancha reducida", voluntariamente se han incorporado a la organización participando en todas las competencias, de acuerdo a las reglas de juego de la F.I.F.A y bajos sus parámetros, conformándose así un movimiento de atletas y entidades deportivas de volumen considerable. [5]

2.2.1.2 Definición

El fútbol sala o futsal, es un deporte derivado de la unión de otros varios deportes: el fútbol, que es la base del juego; el waterpolo; el voleibol, el balonmano y el

baloncesto, tomando de éstos no sólo parte de las reglas, sino también algunas técnicas de juego. Este es un deporte de asociación, con oponente, mínimo contacto y móvil donde los jugadores precisan de una gran habilidad técnica y dominio sobre el balón, así como velocidad y precisión en la ejecución de los gestos técnicos que continuamente se realizan durante el desarrollo de un juego de fútbol sala.

Desde el mismo momento en que todos estos elementos se agrupan, se consolidan con la fijación de unas reglas propias, y desarrollan unas capacidades ya propias e independientes, se puede decir que el fútbol sala se establece como deporte autónomo, diferente, donde priman unos factores que forman su propia idiosincrasia: la rapidez, la espectacularidad, la intensidad, el dominio técnico del balón y el desarrollo de unas condiciones físicas, técnicas, tácticas, estratégicas, sociológicas y psicológicas condicionadas por las particularidades del juego.

2.2.1.3 Aspectos Físicos

El fútbol sala es un deporte en el que priman la intensidad y el alto ritmo de juego, provocados por el carácter reducido del espacio en el que se desarrolla, y la presencia constante del balón en zonas cercanas a la posición de cada jugador. Es por eso que una adecuada condición física resulta imprescindible para un óptimo desarrollo del juego.

Básicamente, en el fútbol sala se desarrollan todos los componentes físicos habituales, si bien unos priman sobre otros: su intensidad, constante movimiento, con marcado acento en cambios de ritmo, hacen que factores como la velocidad de reacción, el trabajo de resistencia anaeróbica, la agilidad, así como un adecuado uso de la potencia sean las capacidades más a tener en cuenta en la preparación física específica del fútbol sala.

2.2.1.4 Aspectos Técnicos

A nivel general, la técnica es considerada como la motricidad híper-especializada, específica de cada actividad y que se desarrolla según unos gestos en el medio que el jugador utiliza para conseguir su objetivo, siendo determinado por sus propias capacidades y la dificultad de la tarea a emprender.

En el fútbol sala, la técnica es interpretada como la ejecución de los fundamentos básicos de juego:

- El pase.
- El tiro.
- La recepción o control.
- El regate.
- La parada del portero, el saque.
- Elementos de técnica sin balón.
- Elementos de técnica defensiva.

2.2.1.5 Aspectos Tácticos

Básicamente, la táctica está relacionada con todo aquello que implique pensar. El aspecto táctico se puede dividir en tres fases:

- Percepción: En esta fase el jugador recibe información de todo el campo de juego, de la ubicación de sus compañeros y de los oponentes y de su distancia hacia la portería y hacia el balón.

- Decisión: Luego de percibir toda la información en la fase anterior, en base a esta, el jugador debe decidir qué acciones va a emprender para favorecer a su equipo.
- Ejecución: Por último, el jugador debe ejecutar las decisiones anteriormente tomadas en la fase previa.

2.2.2 Programación Orientada a Objetos

La Programación Orientada a Objetos es un paradigma de programación que usa objetos y sus interacciones para diseñar aplicaciones y programas de computadora. Está basado en varias técnicas, incluyendo herencia, modularidad, polimorfismo y encapsulamiento. Su uso se popularizó a principios de la década de 1990. Actualmente son muchos los lenguajes de programación que soportan la orientación a objetos.

Los objetos son entidades que combinan estado, comportamiento e identidad:

- El estado está compuesto de datos, serán uno o varios atributos a los que se habrán asignado unos valores concretos (datos).
- El comportamiento está definido por los procedimientos o métodos con que puede operar dicho objeto, es decir, qué operaciones se pueden realizar con él.
- La identidad es una propiedad de un objeto que lo diferencia del resto, dicho con otras palabras, es su identificador (concepto análogo al de identificador de una variable o una constante).

La programación orientada a objetos introduce nuevos conceptos, que superan y amplían conceptos antiguos ya conocidos. Entre ellos destacan los siguientes:

2.2.2.1 Clases

Las clases son declaraciones o abstracciones de objetos, lo que significa, que una clase es la definición de un objeto. Cuando se programa un objeto y se definen sus características y funcionalidades, realmente se programa una clase. Una clase es un contenedor de uno o más datos (atributos o variables miembro) junto a las operaciones de manipulación de dichos datos (funciones o métodos).

2.2.2.1.1 Atributos

Los atributos o variables son las características de los objetos. Cuando se define una propiedad normalmente se especifica su nombre y su tipo. Habitualmente, las variables miembro son privadas al objeto (siguiendo las directrices de diseño del Principio de ocultación) y su acceso se realiza mediante propiedades o métodos que realizan comprobaciones adicionales.

2.2.2.1.2 Métodos

Son funciones que implementan la funcionalidad asociada al objeto. Cuando se desea realizar una acción sobre un objeto, se dice que se le manda un mensaje invocando a un método que realizará la acción.

2.2.2.2 Propiedades

La programación orientada a objetos posee una gran cantidad de características que merecen ser mencionadas:

2.2.2.2.1 Abstracción

Cada objeto en el sistema sirve como modelo de un "agente" abstracto que puede realizar trabajo, informar y cambiar su estado, y "comunicarse" con otros objetos en el sistema sin revelar cómo se implementan estas características.

2.2.2.2.2 Encapsulamiento

Significa reunir a todos los elementos que pueden considerarse pertenecientes a una misma entidad, al mismo nivel de abstracción. Esto permite aumentar la cohesión de los componentes del sistema.

2.2.2.2.3 Principio de Ocultación

Cada objeto está aislado del exterior, es un módulo natural, y cada tipo de objeto expone una interfaz a otros objetos que especifica cómo pueden interactuar con los objetos de la clase. El aislamiento protege a las propiedades de un objeto contra su modificación por quien no tenga derecho a acceder a ellas, solamente los propios métodos internos del objeto pueden acceder a su estado.

2.2.2.2.4 Polimorfismo

Comportamientos diferentes, asociados a objetos distintos, pueden compartir el mismo nombre; y al llamarlos por ese nombre se utilizará el comportamiento correspondiente al objeto que se esté usando. O dicho de otro modo, las referencias y las colecciones de objetos pueden contener objetos de diferentes tipos, y la invocación de un comportamiento en una referencia producirá el comportamiento correcto para el tipo real del objeto referenciado.

2.2.2.2.5 Herencia

Las clases no están aisladas, sino que se relacionan entre sí, formando una jerarquía de clasificación. Los objetos heredan las propiedades y el comportamiento de todas las clases a las que pertenecen. La herencia organiza y facilita el polimorfismo y el encapsulamiento permitiendo a los objetos ser definidos y creados como tipos especializados de objetos preexistentes. Estos pueden compartir (y extender) su comportamiento sin tener que volver a implementarlo. [6]

2.2.3 C#

C# es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET, que después fue aprobado como un estándar por la ECMA e ISO. Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma.NET el cual es similar al de Java aunque incluye mejoras derivadas de otros lenguajes (entre ellos Delphi).

El símbolo # viene de sobreponer "++" sobre "++" y eliminar las separaciones, indicando así su descendencia de C++. C#, como parte de la plataforma.NET, está normalizado por ECMA desde diciembre de 2001 (ECMA-334 "Especificación del Lenguaje C#"). El 7 de noviembre de 2005 salió la versión 2.0 del lenguaje que incluía mejoras tales como tipos genéricos, métodos anónimos, iteradores, tipos parciales y tipos anulables. El 19 de noviembre de 2007 salió la versión 3.0 de C# destacando entre las mejoras los tipos implícitos, tipos anónimos y el LINQ (Language Integrated Query). [7]

2.2.4 Bases de Datos

Una base de datos es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso. En la actualidad, y debido al desarrollo tecnológico de campos como la informática y la electrónica, la mayoría de las bases de datos están en formato digital (electrónico), que ofrece un amplio rango de soluciones al problema de almacenar datos.

Las aplicaciones más usuales son para la gestión de empresas e instituciones públicas. También son ampliamente utilizadas en entornos científicos con el objeto de almacenar la información experimental.

2.2.4.1 Clasificación

Según la variabilidad de sus datos, las bases de datos se dividen en 2 tipos:

- **Estáticas:** Las bases de datos incluidas en este renglón son aquellas cuyos datos son de sólo lectura, es decir, la información que está contenida en ella, no es variable. Se utilizan primordialmente para almacenar datos históricos, que luego serán extraídos con el fin de observar el comportamiento de los datos con el paso del tiempo, y así realizar proyecciones y tomar decisiones.

- **Dinámicas:** Éstas son bases de datos donde la información almacenada se modifica con el tiempo, permitiendo operaciones como actualización y adición de datos, además de las operaciones fundamentales de consulta.

Las bases de datos también se pueden clasificar de acuerdo a su modelo de administración de datos, como se indica a continuación:

- **Jerárquicas:** almacenan su información en una estructura jerárquica. En este modelo los datos se organizan en una forma similar a un árbol, en donde un nodo padre de información puede tener varios hijos. El nodo que no tiene padres es llamado raíz, y a los nodos que no tienen hijos se los conoce como hojas. Son especialmente útiles en el caso de aplicaciones que manejan un gran volumen de información y datos muy compartidos permitiendo crear estructuras estables y de gran rendimiento. Una de las principales limitaciones de este modelo es su incapacidad de representar eficientemente la redundancia de datos.

- **De Red:** Éste es un modelo ligeramente distinto del jerárquico; su diferencia fundamental es la modificación del concepto de nodo: se permite que un mismo nodo tenga varios padres. Fue una gran mejora con respecto al modelo jerárquico, ya que ofrecía una solución eficiente al problema de redundancia de datos; pero, aun así, la dificultad que significa administrar la información en una base de datos de red ha significado que sea un modelo utilizado en su mayoría por programadores más que por usuarios finales.

- **Relacional:** Éste es el modelo más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente. Su idea fundamental es el uso de "relaciones". Estas relaciones podrían considerarse en forma lógica como conjuntos de datos llamados "tuplas". En este modelo, el lugar y la forma en que se almacenen los datos no tienen relevancia (a diferencia de otros modelos como el jerárquico y el de red). Esto tiene la considerable ventaja de que es más fácil de entender y de utilizar para un usuario esporádico de la base de datos. La información puede ser

recuperada o almacenada mediante "consultas" que ofrecen una amplia flexibilidad y poder para administrar la información. El lenguaje más habitual para construir las consultas a bases de datos relacionales es SQL (Structured Query Language), un estándar implementado por los principales motores o sistemas de gestión de bases de datos relacionales. [8]

2.2.4.2 SQL

El Lenguaje de consulta estructurado (SQL) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones sobre las mismas. Una de sus características es el manejo del álgebra y el cálculo relacional permitiendo lanzar consultas con el fin de recuperar de una forma sencilla, información de interés de una base de datos, así como también hacer cambios sobre la misma. Es un lenguaje de cuarta generación (4GL). [9]

2.2.4.3 Sistemas Manejadores de Bases de Datos

Existen unos programas denominados sistemas gestores de bases de datos (DBMS, por sus siglas en inglés), que permiten almacenar y posteriormente acceder a los datos de forma rápida y estructurada. Las propiedades de estos DBMS, así como su utilización y administración, se estudian dentro del ámbito de la informática.

Los DBMS son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. El propósito general de los sistemas de gestión de base de datos es el de manejar de manera clara, sencilla y ordenada un conjunto de datos que posteriormente se convertirán en información relevante, para un buen manejo de datos.

Existen distintos objetivos que deben cumplir los DBMS:

- **Abstracción:** Los DBMS ahorran a los usuarios detalles acerca del almacenamiento físico de los datos.
- **Independencia:** La independencia de los datos consiste en la capacidad de modificar el esquema (físico o lógico) de una base de datos sin tener que realizar cambios en las aplicaciones que se sirven de ella.
- **Consistencia:** En aquellos casos en los que no se ha logrado eliminar la redundancia, será necesario vigilar que aquella información que aparece repetida se actualice de forma coherente, es decir, que todos los datos repetidos se actualicen de forma simultánea.
- **Seguridad:** La información almacenada en una base de datos puede llegar a tener un gran valor. Los DBMS deben garantizar que esta información se encuentra segura frente a usuarios malintencionados, que intenten leer información privilegiada; frente a ataques que deseen manipular o destruir la información; o simplemente ante las torpezas de algún usuario autorizado pero despistado.
- **Integridad:** Se trata de adoptar las medidas necesarias para garantizar la validez de los datos almacenados. Es decir, se trata de proteger los datos ante fallos de hardware, datos introducidos por usuarios descuidados, o cualquier otra circunstancia capaz de corromper la información almacenada.
- **Respaldo:** Los DBMS deben proporcionar una forma eficiente de realizar copias de respaldo de la información almacenada en ellos, y de restaurar a partir de estas copias los datos que se hayan podido perder.
- **Control de la Concurrencia:** un DBMS debe controlar el acceso concurrente a la información, que podría derivar en inconsistencias.

- **Tiempo de Respuesta:** es deseable minimizar el tiempo que el DBMS tarda en darnos la información solicitada y en almacenar los cambios realizados. [10]

2.2.5 Aplicaciones WEB

En la ingeniería software se denomina aplicación web a aquellas aplicaciones que los usuarios pueden utilizar accediendo a un servidor web a través de Internet o de una intranet mediante un navegador. En otras palabras, es una aplicación software que se codifica en un lenguaje soportado por los navegadores web (HTML, JavaScript, Java, etc.) en la que se confía la ejecución al navegador.

Las aplicaciones web son populares debido a lo práctico del navegador web como cliente ligero, así como a la facilidad para actualizar y mantener aplicaciones web sin distribuir e instalar software a miles de usuarios potenciales.

Una ventaja significativa es que las aplicaciones web deberían funcionar igual independientemente de la versión del sistema operativo instalado en el cliente. En vez de crear clientes para Windows, Mac OS X, GNU/Linux, y otros sistemas operativos, la aplicación web se escribe una vez y se ejecuta igual en todas partes.

Aunque existen muchas variaciones posibles, una aplicación web está normalmente estructurada como una aplicación de tres-capas. En su forma más común, el navegador web ofrece la primera capa y un motor capaz de usar alguna tecnología web dinámica (PHP, Java Servlets o ASP, ASP.NET, CGI, ColdFusion, etc.) constituye la capa de en medio. Por último, una base de datos constituye la tercera y última capa.

2.2.5.1 ASP.NET

ASP.NET es un framework para aplicaciones web desarrollado y comercializado por Microsoft. Es usado por programadores para construir sitios web dinámicos, aplicaciones web y servicios web XML. Apareció en enero de 2002 con la version 1.0 del .NET Framework, y es la tecnología sucesora de la tecnología Active Server Pages (ASP). ASP.NET está construido sobre el Common Language Runtime, permitiendo a los programadores escribir código ASP.NET usando cualquier lenguaje admitido por el .NET Framework.

Las páginas de ASP.NET, conocidas oficialmente como "web-forms" (formularios web), son el principal medio de construcción para el desarrollo de aplicaciones web. Los formularios web están contenidos en archivos con una extensión ASPX; en jerga de programación, estos archivos típicamente contienen etiquetas HTML o XHTML estático, y también etiquetas definiendo Controles Web que se procesan del lado del servidor y Controles de Usuario donde los desarrolladores colocan todo el código estático y dinámico requerido por la página web. [11]

2.2.5.2 Silverlight

Microsoft Silverlight es un complemento para navegadores de Internet basado en la plataforma Windows que agrega nuevas funciones multimedia como la reproducción de vídeos, gráficos vectoriales, animaciones y de entorno de desarrollo; en forma similar a lo que hace Adobe Flash.

Silverlight compite con Adobe Flex, Nexaweb, OpenLaszlo y algunas presentaciones de componentes AJAX. La primera versión de Silverlight fue lanzada en septiembre de 2007 y actualmente su versión 2.0 se distribuye de forma gratuita.

Silverlight conserva un modo de gráficos de sistema, similar al del WPF e integra en un solo complemento multimedia, gráficos de computador, animaciones e interactividad. La base de su programación es XAML y el acceso a los objetos esta dado por Java Script. El XAML puede ser usado para marcar los gráficos vectoriales y las animaciones.

Una aplicación de Silverlight comienza por invocar el control de Silverlight mediante una la página HTML, para generar dicha página (en lo que a estructura se refiere), usa el archivo XAML. El archivo XAML puede contener múltiples objetos, pero normalmente el objeto padre suele ser del tipo Canvas, el cual actúa como contenedor de otros elementos. Silverlight ofrece la posibilidad de usar símbolos geométricos básicos como: líneas, elipses, elementos de texto, imágenes y multimedia. Los elementos están propiamente posicionados para alcanzar la disposición deseada. Cualquier figura arbitraria puede ser creada si es requerida. Estos elementos pueden ser animados usando el reproductor de eventos; algunos efectos de animaciones están predeterminados, mientras que otros pueden ser compuestos de otros efectos pre-definidos. Eventos como el movimiento del teclado o del ratón pueden ser manejados por scripts personalizados o manejadores de eventos.

2.2.6 Ingeniería de Software

Es la rama de la ingeniería que crea y mantiene las aplicaciones de software aplicando tecnologías y prácticas de las ciencias computacionales, manejo de proyectos, ingeniería, el ámbito de la aplicación, y otros campos.

El software es el conjunto de instrucciones que permite al hardware de la computadora desempeñar trabajo útil. En las últimas décadas del siglo XX, las

reducciones de costo en hardware llevaron a que el software fuera un componente ubicuo de los dispositivos usados por las sociedades industrializadas.

La ingeniería del software es multidisciplinaria. Utiliza las matemáticas para analizar y certificar algoritmos; la ingeniería para estimar costo; la administración para definir requerimientos, evaluar riesgos, administrar personal, etc.

La ingeniería de software se inicia con la necesidad para solucionar las inhabilidades de las organizaciones para predecir el tiempo, esfuerzo y costo para el desarrollo de software y la pobre calidad del software que se producía, además de esto hay otros factores que forzaron el surgimiento de la ingeniería de software como una disciplina.

Los factores que contribuyeron al surgimiento de la ingeniería de software y continúan influenciando en el desarrollo y refinamiento de la ingeniería de software son:

- Cambio en la relación de costos de hardware y software.
- Incremento importante del rol de mantenimiento.
- Avances en Hardware.
- Avances en técnicas de software.
- Incremento de la demanda de Software.
- La demanda de sistemas de software más largos y complejos.

La meta de la ingeniería de software es producir un sistema de software de calidad, a tiempo y a bajo costo.

2.2.7 Proceso Unificado de Rational (RUP):

El Proceso Unificado de Rational (Rational Unified Process, RUP) es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado (UML), constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. El RUP no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización.

2.2.7.1 Principios

El RUP está basado en 5 principios clave que son:

- **Adaptar el Proceso:** El proceso deberá adaptarse a las características propias del proyecto u organización. El tamaño del mismo, así como su tipo o las regulaciones que lo condicionen, influirán en su diseño específico.
- **Balancear Prioridades:** Los requerimientos de los diversos participantes pueden ser diferentes, contradictorios o disputarse recursos limitados. *Debe encontrarse un balance que satisfaga los deseos de todos*.
- **Demostrar Valor Iterativamente:** Los proyectos se entregan, aunque sea de un modo interno, en **etapas iteradas**. En cada iteración se analiza la opinión de los inversores, la estabilidad y calidad del producto, y se refina la dirección del proyecto así como también los riesgos involucrados
- **Elevar el Nivel de Abstracción:** Este principio dominante motiva el uso de conceptos reutilizables tales como patrón del software, lenguajes 4GL o marcos de referencia (frameworks) por nombrar algunos. Esto evita que los ingenieros de software vayan directamente de los requisitos a la

codificación de software a la medida del cliente, sin saber con certeza qué codificar para satisfacer de la mejor manera los requerimientos y sin comenzar desde un principio pensando en la reutilización del código.

- **Enfocarse en la Calidad:** El control de calidad no debe realizarse al final de cada iteración, sino en **todos** los aspectos de la producción.

2.2.7.2 Ciclo de Vida

El ciclo de vida RUP es una implementación del Desarrollo en espiral. Fue creado ensamblando los elementos en secuencias semi-ordenadas. El ciclo de vida organiza las tareas en fases e iteraciones. El RUP divide el proceso en cuatro fases, dentro de las cuales se realizan varias iteraciones en número variable según el proyecto y en las que se hace un mayor o menor énfasis en los distintas actividades. En la Figura 2.1 se muestra cómo varía el esfuerzo asociado a las disciplinas según la fase en la que se encuentre el proyecto RUP.

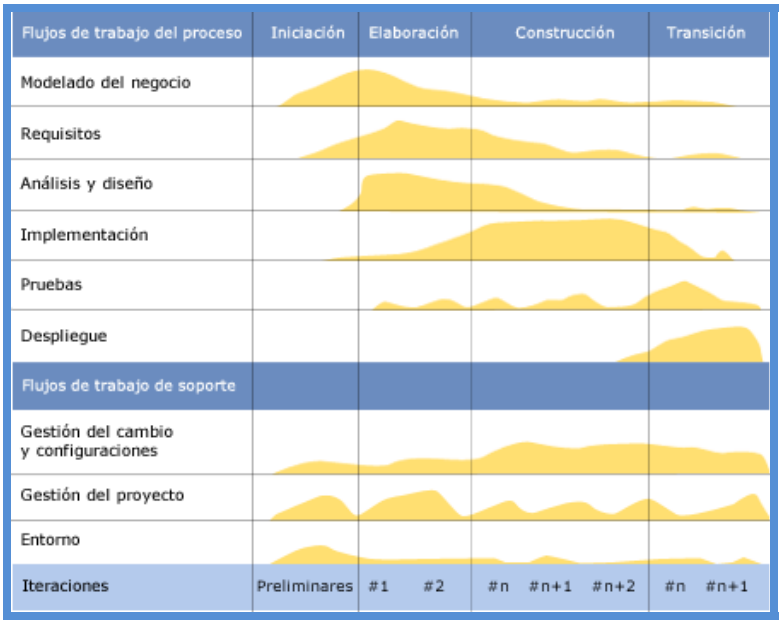

Figura 2.1. Ciclo de vida del RUP

Fuente: Propia

Las primeras iteraciones (en las fases de Inicio y Elaboración) se enfocan hacia la comprensión del problema y la tecnología, la delimitación del ámbito del proyecto, la eliminación de los riesgos críticos, y al establecimiento de una línea base de la arquitectura. Durante la fase de inicio las iteraciones hacen mayor énfasis en actividades de modelado del negocio y de requerimientos.

En la fase de elaboración, las iteraciones se orientan al desarrollo de la línea base de la arquitectura, abarcan más los flujos de trabajo de requerimientos, modelo de negocios (refinamiento), análisis, diseño y una parte de implementación orientado a la línea base de la arquitectura. En la fase de construcción, se lleva a cabo la construcción del producto por medio de una serie de iteraciones.

Para cada iteración se seleccionan algunos Casos de Uso, se refina su análisis y diseño y se procede a su implementación y pruebas. Se realiza una pequeña cascada para cada ciclo. Se realizan tantas iteraciones hasta que se termine la implementación de la nueva versión del producto. Y por último, en la fase de transición se pretende garantizar que se tiene un producto preparado para su entrega a la comunidad de usuarios. [12]

2.2.8 Lenguaje Unificado de Modelado (UML)

2.2.8.1 Definición

El Lenguaje de Modelado Unificado contiene una notación robusta para el modelado y desarrollo de sistemas orientados a objeto. Proporciona la tecnología necesaria para apoyar la práctica de la ingeniería del software orientada a objetos.

Como resultado de la aplicación de UML se puede producir un arreglo de modelos y documentos de trabajo. Sin embargo, éstos los reducen los ingenieros de software para lograr que el desarrollo sea más ágil y reactivo ante el cambio.

Otros métodos de modelaje como OMT (Object Modeling Technique) o Booch sí definen procesos concretos. En UML los procesos de desarrollo son diferentes según los distintos dominios de trabajo; no puede ser el mismo el proceso para crear una aplicación en tiempo real, que el proceso de desarrollo de una aplicación orientada a gestión, por poner un ejemplo.

2.2.8.2 Elementos de UML

- **Actor**: es una entidad externa (de fuera del sistema) que interacciona con el sistema participando (y normalmente iniciando) en un caso de uso. Los actores pueden ser gente real (por ejemplo, usuarios del sistema), otros ordenadores o eventos externos. Los actores no representan a personas **físicas** o a sistemas, sino su **papel**. Esto significa que cuando una persona interacciones con el sistema de diferentes maneras, estará representado por varios actores.

- **Clases**: En una clase se agrupan todos los objetos que comparten los mismos atributos, métodos y relaciones. Los atributos son características y propiedades comunes en todos los objetos de la clase. Los métodos son operaciones que deben cumplir las instancias de la clase. Las clases se representan como un rectángulo donde figuran el nombre de la clase, sus atributos y sus métodos.

- **Artefacto:** es una información que es utilizada o producida mediante un proceso de desarrollo de software. Pueden ser artefactos un modelo, una descripción o un software. Los artefactos de UML se especifican en forma de diagramas, éstos, junto con la documentación sobre el sistema constituyen los artefactos principales que el modelador puede observar.

- **Estado:** Los estados son los ladrillos de los diagramas de estado. Un estado pertenece a exactamente una clase y representa un resumen de los valores y atributos que puede tener la clase. Un estado UML describe el estado interno de un objeto de una clase particular**.**

- **Actividad**: es un único paso de un proceso**.**

2.2.8.3 Diagramas de UML

Un diagrama es una representación gráfica de una colección de elementos del modelo, que habitualmente toma forma de grafo donde los arcos que conectan sus vértices son las relaciones entre los objetos y los vértices se corresponden con los elementos del modelo.

Los distintos puntos de vista de un sistema real que se quieren representar para obtener el modelo se dibuja de forma que se resaltan los detalles necesarios para entender el sistema**.**

- **Diagramas de Casos de Uso**: Un diagrama de casos de uso es un diagrama que muestra un conjunto de casos de uso con sus relaciones y los actores implicados. Es un diagrama que sirve para modelar la vista estática de un programa. La vista estática nos permite visualizar el comportamiento externo

del programa; de esta forma se consigue conocer qué es lo que debe hacer el programa independientemente de cómo lo haga y sabremos los elementos que interactúan con el sistema. Los elementos implicados en un diagrama de casos de uso son los casos de uso, las relaciones y los actores. Un actor es un rol que interactúa con el sistema. Se define como rol porque un actor puede ser tanto un usuario de la aplicación como otro sistema o dispositivos externos. Los diagramas de casos de uso se utilizan para ilustrar los requerimientos del sistema al mostrar cómo reacciona una respuesta a eventos que se producen en el mismo. En este tipo de diagrama intervienen algunos conceptos nuevos: un actor es una entidad externa al sistema que se modela y que puede interactuar con él; un ejemplo de actor podría ser un usuario o cualquier otro sistema. Las relaciones entre casos de uso y actores pueden ser las siguientes:

  - Un actor se comunica con un caso de uso.
  - Un caso de uso extiende otro caso de uso.
  - Un caso de uso usa otro caso de uso.

- **Diagramas de Secuencia**: Un diagrama de secuencia es un diagrama de interacción UML. Estos diagramas muestran la secuencia de mensajes que se van lanzando los objetos implicados en una determinada operación del programa. Dentro del diagrama los objetos se alinean en el eje X respetando su orden de aparición. En el eje Y se van mostrando los mensajes que se envían, también respetando su orden temporal. Cada objeto tiene una línea de vida donde se sitúa su foco de control. El foco de control es un rectángulo que representa el tiempo durante el que un objeto está activo ejecutando una acción. Con este sencillo esquema podemos visualizar la comunicación y sincronización bajo un estricto orden temporal de los objetos implicados en las distintas funcionalidades de un sistema.

- **Diagrama de Clases de Análisis:** Es utilizado por los desarrolladores de software para determinar los requerimientos funcionales, considerando una o varias clases, o sub-sistemas del sistema a desarrollar. Los casos de uso se describen mediante clases de análisis y sus objetos. El diagrama de clases de análisis se construye examinando los casos de usuarios, cerrando sus reacciones e identificando los roles de los clasificadores.

- **Diagrama de Clases de Diseño:** Se emplean para modelar la estructura estática de las clases en el sistema, sus tipos, sus contenidos y las relaciones que se establecen entre ellos. A través de este diagrama se definen las características de cada una de las clases, interfaces, colaboraciones y relaciones de dependencia y generalización**.**

- **Diagramas de actividad**: Son similares a los diagramas de flujo de otras metodologías Orientadas a Objetos. En realidad se corresponden con un caso especial de los diagramas de estado donde los estados son estados de acción (estados con una acción interna y una o más transiciones que suceden al finalizar esta acción, o lo que es lo mismo, un paso en la ejecución de lo que será un procedimiento) y las transiciones vienen provocadas por la finalización de las acciones que tienen lugar en los estados de origen. Siempre van unidos a una clase o a la implementación de un caso de uso o de un método (que tiene el mismo significado que en cualquier otra metodología OO). Los diagramas de actividad se utilizan para mostrar el flujo de operaciones que se desencadenan en un procedimiento interno del sistema**.**

- **Diagramas de Implementación**: Se derivan de los diagramas de proceso y módulos de la metodología de Booch, aunque presentan algunas

modificaciones. Los diagramas de implementación muestran los aspectos físicos del sistema. Incluyen la estructura del código fuente y la implementación, en tiempo de implementación. Existen dos tipos:

- **Diagramas de componentes:** Muestra la dependencia entre los distintos componentes de software, incluyendo componentes de código fuente, binario y ejecutable. Un componente es un fragmento de código software (un fuente, binario o ejecutable) que se utiliza para mostrar dependencias en tiempo de compilación.

- **Diagrama de plataformas o despliegue:** Muestra la configuración de los componentes hardware, los procesos, los elementos de procesamiento en tiempo de ejecución y los objetos que existen en tiempo de ejecución. En este tipo de diagramas intervienen nodos, asociaciones de comunicación, componentes dentro de los nodos y objetos que se encuentran a su vez dentro de los componentes. Un nodo es un objeto físico en tiempo de ejecución, es decir una máquina que se compone habitualmente de, por lo menos, memoria y capacidad de procesamiento, a su vez puede estar formado por otros componentes.

- **Modelo de Dominio**: es un artefacto de la disciplina de análisis, construido con las reglas de UML durante la fase de concepción, en la tarea construcción del modelo de dominio, presentado como uno o más diagramas de clases y que contiene, no conceptos propios de un sistema de software sino de la propia realidad física. Los modelos de dominio pueden utilizarse para capturar y expresar el entendimiento ganado en un área bajo análisis como paso previo al diseño de un sistema, ya sea de software o de otro tipo. Similares a los

mapas mentales utilizados en el aprendizaje, el modelo de dominio es utilizado por el analista como un medio para comprender el sector industrial o de negocios al cual el sistema va a servir.

- **Diagrama de Paquetes**: Los diagramas de paquetes se usan para reflejar la organización de paquetes y sus elementos. Cuando se usan para representaciones, los diagramas de paquete de los elementos de clase se usan para proveer una visualización de los espacios de nombres. Los elementos contenidos en un paquete comparten el mismo espacio de nombre, el hecho de compartir espacios de nombres requiere que los elementos contenidos en un espacio de nombre específico tengan nombres únicos. Los paquetes se pueden construir para representar relaciones tanto físicas como lógicas. [13]

CAPÍTULO III. FASE DE INICIO

3.1 Introducción

En este capítulo se desarrollará la primera fase del proceso unificado (RUP) Inicio (Inception), donde se hace un plan de fases, se identifican los principales casos de uso y riesgos del sistema a desarrollar, permitiendo esto establecer los requisitos que debe cumplir la aplicación y elaborar el conjuntos de modelos que describirán el comportamiento de la misma.

La fase de inicio es la más importante ya que es aquí donde se establece un acuerdo entre todos los interesados acerca de los objetivos del proyecto. Esta fase es significativamente primaria para el desarrollo de nuevo software, ya que se asegura de identificar los requerimientos y los riesgos relacionados con el negocio. Para proyectos de mejora de software existentes esta fase es más breve y se centra en asegurar que vale la pena y es posible desarrollar el proyecto.

3.2 Requerimientos

Los requerimientos son una descripción de las necesidades de un producto. La meta de analizar los requerimientos es identificar y documentar lo que en realidad se espera del software claramente, de forma que se le pueda comunicar al cliente y a los miembros del equipo de desarrollo.

Los requerimientos del sistema son tomados de una matriz de requerimientos donde se depositan todas las necesidades manifestadas por los usuarios que apoyan la aplicación. Dicha matriz se presenta en la Tabla 3.1.

Tabla 3.1. Matriz de Requerimientos del Sistema

Fuente: Propia

Ref.	Funcionalidad
R1	El sistema debe permitir agregar torneos de diferentes tipos.
R2	El sistema debe permitir agregar jugadores nuevos.
R3	El sistema debe permitir agregar equipos nuevos.
R4	El sistema debe permitir agregar entrenadores nuevos.
R5	El sistema debe permitir agregar jugadores existentes a equipos existentes.
R6	El sistema debe permitir agregar entrenadores existentes a equipos existentes.
R7	El sistema debe permitir agregar equipos participantes a los torneos.
R8	El sistema debe generar los partidos para los torneos de forma aleatoria.
R9	El sistema debe permitir añadir resultados y estadísticas a cada partido.
R10	El sistema debe permitir añadir nuevos usuarios administradores.
R11	El sistema debe permitir observar las estadísticas de los torneos.
R12	El sistema debe permitir generar reportes por cada jornada de los torneos.

3.3 Diagrama de Dominio

Un modelo de dominio es una representación visual de las clases conceptuales u objetos del mundo real en un dominio de interés. También se les domina modelos conceptuales, modelo de objetos del dominio y modelos de objetos de análisis.

Para la descripción del modelo de dominio debe realizarse diagramas de clases basado en Lenguaje de Modelado UML. **[20]**

En la Figura 3.1 se representan las clases más importantes del modelo de dominio.

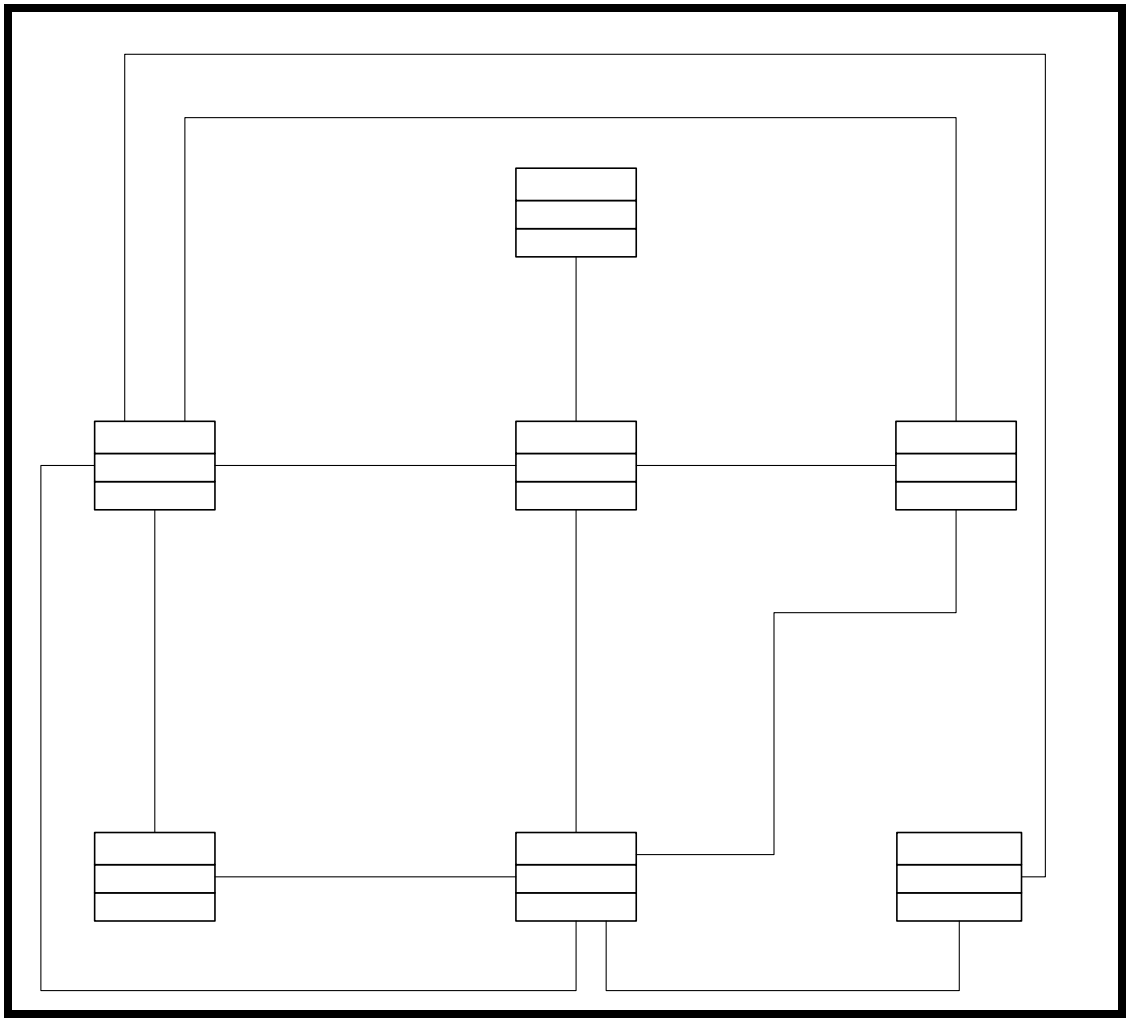

Figura 3.1. Diagrama de Dominio

Fuente: Propia

3.4 Diagramas de Casos de Uso del Sistema

3.4.1 Actores

Partiendo de los requerimientos del sistema, se identificaron los actores para la aplicación, con sus respectivas funciones y fueron representados en la Tabla 3.2:

Tabla 3.2. Actores del Sistema

Actores	Funciones
Administrador	Este usuario es el responsable de la administración del sistema: Poseerá todos los permisos del sistema. El administrador tiene privilegios para; manejar la información contenida, estar pendiente del buen funcionamiento de la aplicación y modificar datos de ser requerido.
Usuario Público	Este usuario es aquel que puede utilizar el programa únicamente para recibir información. Puede ver estadísticas y progresión de los jugadores, torneos, equipos y entrenadores. Este usuario no puede alterar valores ni introducir valores nuevos a la base de datos.

3.4.2 Diagrama de Caso de Uso

Un diagrama de casos de uso es una representación gráfica de parte o el total de los actores y casos de uso del sistema, incluyendo sus interacciones. Todo sistema tiene como mínimo un diagrama de caso de uso, que es una representación gráfica del entorno del sistema (actores) y su funcionalidad principal (casos de uso).

Un diagrama de casos de uso muestra, por tanto, los distintos requisitos funcionales que se esperan de una aplicación o sistema y cómo se relaciona con su entorno (usuarios u otras aplicaciones).

En la Figura 3.2 se representa el diagrama de caso de uso general, desde el punto de vista del usuario del sistema.

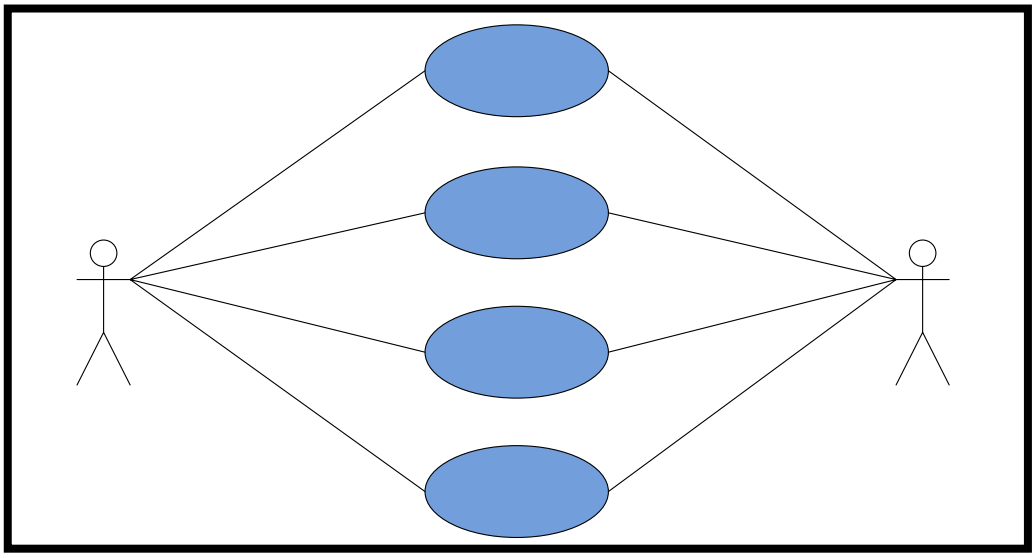

Figura 3.2. Caso de Uso del Sistema

Identificación de los casos de uso:

Tabla 3.3. Caso de uso Administrar Jugadores

Fuente: Propia

Caso de Uso	Administrar Jugadores
Descripción	Permite al administrador ingresar nuevos jugadores y añadirlos a equipos, previa identificación, mientras que al público en general le permite ver los datos de los jugadores.
Actores Implicados	Administrador, Público

Tabla 3.4. Caso de uso Administrar Equipos

Fuente: Propia

Caso de Uso	Administrar Equipos
Descripción	Permite al administrador ingresar nuevos equipos y añadirlos a torneos, previa identificación, mientras que al público en general le permite ver los datos de los equipos.
Actores Implicados	Administrador, Público

Tabla 3.5. Caso de uso Administrar Entrenadores

Fuente: Propia

Caso de Uso	Administrar Entrenadores
Descripción	Permite al administrador ingresar nuevos entrenadores y añadirlos a equipos, previa identificación, mientras que al público en general le permite ver los datos de los entrenadores.
Actores Implicados	Administrador, Público

Tabla 3.6. Caso de uso Administrar Torneos

Fuente: Propia

Caso de Uso	Administrar Torneos
Descripción	Permite al administrador ingresar nuevos torneos, agregar equipos a torneos y añadir estadísticas pertenecientes a los torneos, previa identificación, mientras que al público en general le permite ver los datos de los torneos.
Actores Implicados	Administrador, Público

La representación general por medio de un caso de uso muestra el contexto de trabajo de la aplicación, el tipo de información que se maneja, los actores que intervienen en las operaciones y las interacciones entre los procesos y los actores.

Para la visualización más detallada de los procesos representados de forma general en el diagrama, se presenta a continuación, de forma más detallada, la estructura interna de cada proceso a través de su representación en casos de uso:

3.4.3 Descripción del flujo de sucesos del Caso de Uso Torneos

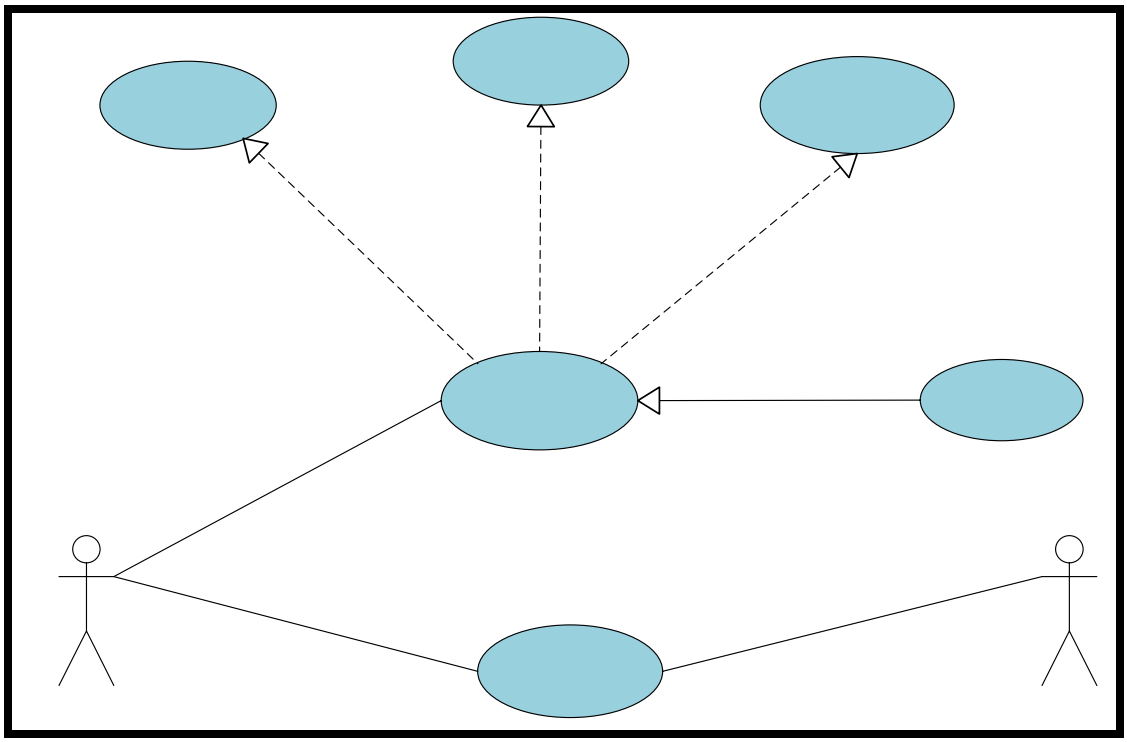

Figura 3.3. Caso de uso Administrar Torneos

Fuente: Propia

Nombre del Caso de Uso: "Administrar Torneos"

Agregar Torneo

Actores: Administrador

Descripción: Permite al administrador agregar torneos, agregar equipos a torneos, agregar estadísticas a torneos.

Pre-Condición: Este proceso es invocado por el usuario desde el proceso "Menú", al pulsar la opción.

Flujo de Eventos:

Flujo Principal:

1. El usuario invoca al caso de uso.
2. Se muestran las fichas para agregar torneos, añadir estadísticas y equipos sólo si anteriormente el usuario inició sesión como administrador.
3. El usuario visualiza la información y selecciona la operación que desea realizar.
4. Finaliza el caso de uso.

Flujo Alterno:

El usuario tiene la opción de seleccionar otra operación o puede salir del sistema.

Nombre del Caso de Uso: "Agregar Torneo"

Actores: Administrador

Descripción: Permite al administrador agregar torneos.

Pre-Condición: El usuario realiza la selección desde el proceso "Administrar Torneos"

Flujo de Eventos:

Flujo Principal:

1. El usuario invoca al caso de uso.
2. Se llenan los distintos formularios con la información del torneo, para posteriormente añadirlo.
3. Finaliza el caso de uso.

Flujo Alterno:

El usuario tiene la opción de seleccionar otra operación o puede salir del sistema.

Nombre del Caso de Uso: "Agregar Equipo a Torneo"

Actores: Administrador

Descripción: Permite al administrador agregar equipos a torneos.

Pre-Condición: El usuario realiza la selección desde el proceso "Administrar Torneos"

Flujo de Eventos:

Flujo Principal:

1. El usuario invoca al caso de uso.
2. Se selecciona el equipo a agregar y se pulsa el botón para hacerlo.
3. Finaliza el caso de uso.

Flujo Alterno:

El usuario tiene la opción de seleccionar otra operación o puede salir del sistema.

Nombre del Caso de Uso: "Agregar Estadísticas a Torneo"

Actores: Administrador

Descripción: Permite al administrador agregar estadísticas de los torneos.

Pre-Condición: El usuario realiza la selección desde el proceso "Administrar Torneos"

Flujo de Eventos:

Flujo Principal:

1. El usuario invoca al caso de uso.
2. Se selecciona el partido al que se desean agregar las estadísticas.
3. Se rellenan los formularios
4. Finaliza el caso de uso.

Flujo Alterno:

El usuario tiene la opción de seleccionar otra operación o puede salir del sistema.

Nombre del Caso de Uso: "Ver Datos Torneo"

Actores: Administrador, Público

Descripción: Permite al administrador y al público ver las estadísticas del torneo.

Pre-Condición: El usuario realiza la selección desde el proceso "Administrar Torneos"

Flujo de Eventos:

Flujo Principal:

1. El usuario invoca al caso de uso.
2. Se selecciona el torneo del que se desean ver los datos.
3. Finaliza el caso de uso.

Flujo Alterno:

El usuario tiene la opción de seleccionar otra operación o puede salir del sistema.

3.4.4 Descripción del flujo de sucesos del Caso de Uso de Equipos

Nombre del Caso de Uso: "Administrar Equipos"

Actores: Administrador

Descripción: Permite al administrador agregar equipos, agregar jugadores a los equipos, agregar entrenadores a los equipos.

Pre-Condición: Este proceso es invocado por el usuario desde el proceso "Menú", al pulsar la opción.

Flujo de Eventos:

Flujo Principal:

1. El usuario invoca al caso de uso.

2. Se muestran las fichas para agregar equipos, añadir jugadores y entrenadores sólo si anteriormente el usuario inició sesión como administrador.
3. El usuario visualiza la información y selecciona la operación que desea realizar.
4. Finaliza el caso de uso.

Flujo Alterno:

El usuario tiene la opción de seleccionar otra operación o puede salir del sistema.

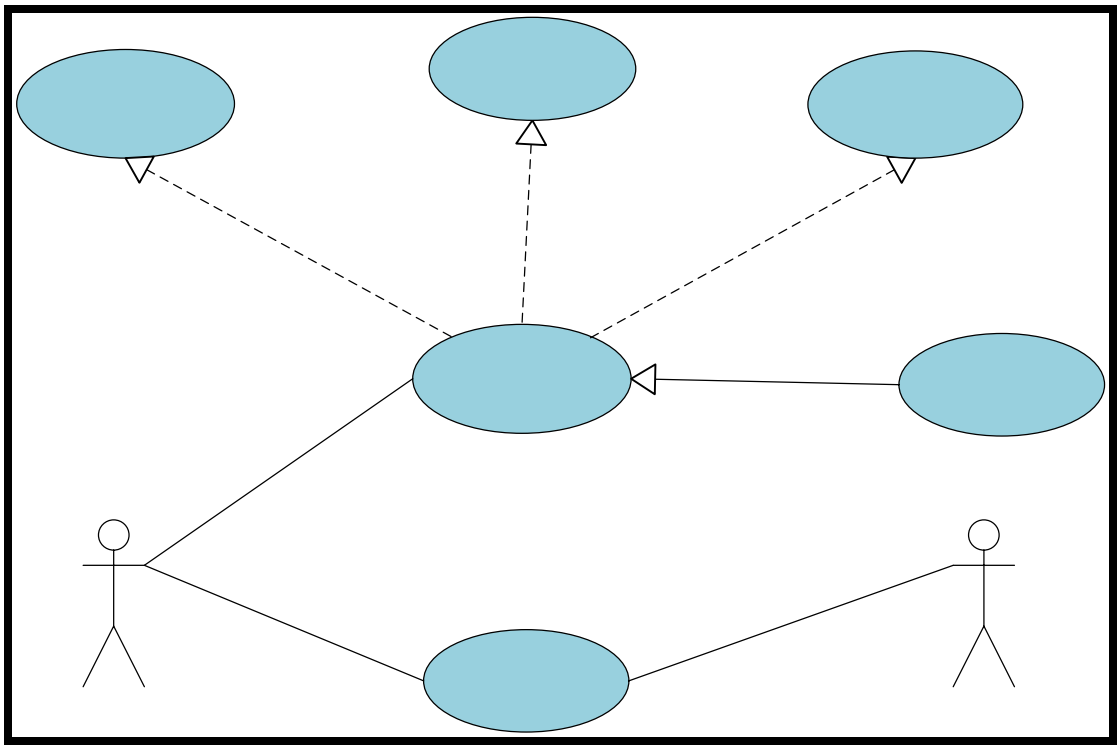

Figura 3.4 Caso de uso Administrar Equipos

Fuente: Propio

Agregar Equipo

Nombre del Caso de Uso: "Agregar Equipo"

Actores: Administrador

Descripción: Permite al administrador agregar equipos.

«extends»

Pre-Condición: El usuario realiza la selección desde el proceso “Administrar Equipos”

Flujo de Eventos:

Flujo Principal:

1. El usuario invoca al caso de uso.
2. Se llenan los distintos formularios con la información del equipo, para posteriormente añadirlo.
3. Finaliza el caso de uso.

Flujo Alterno:

El usuario tiene la opción de seleccionar otra operación o puede salir del sistema.

Nombre del Caso de Uso: “Agregar Jugador a Equipo”

Actores: Administrador

Descripción: Permite al administrador agregar jugadores a los equipos.

Pre-Condición: El usuario realiza la selección desde el proceso “Administrar Equipos”

Flujo de Eventos:

Flujo Principal:

1. El usuario invoca al caso de uso.
2. Se selecciona el jugador a agregar y se pulsa el botón para hacerlo.
3. Finaliza el caso de uso.

Flujo Alterno:

El usuario tiene la opción de seleccionar otra operación o puede salir del sistema.

Nombre del Caso de Uso: “Agregar Entrenadores a Equipo”

Actores: Administrador

Descripción: Permite al administrador agregar entrenadores a los equipos.

Pre-Condición: El usuario realiza la selección desde el proceso "Administrar Equipos"

Flujo de Eventos:

Flujo Principal:

1. El usuario invoca al caso de uso.
2. Se selecciona el entrenador a agregar y se pulsa el botón para hacerlo.
3. Finaliza el caso de uso.

Flujo Alterno:

El usuario tiene la opción de seleccionar otra operación o puede salir del sistema.

Nombre del Caso de Uso: "Ver Datos Equipo"

Actores: Administrador, Público

Descripción: Permite al administrador y al público ver los datos de los equipos.

Pre-Condición: El usuario realiza la selección desde el proceso "Administrar Equipos"

Flujo de Eventos:

Flujo Principal:

1. El usuario invoca al caso de uso.
2. Se selecciona el equipo del que se desean ver los datos.
3. Finaliza el caso de uso.

Flujo Alterno:

El usuario tiene la opción de seleccionar otra operación o puede salir del sistema.

3.4.5 Descripción del flujo de sucesos del Caso de Uso de Jugadores

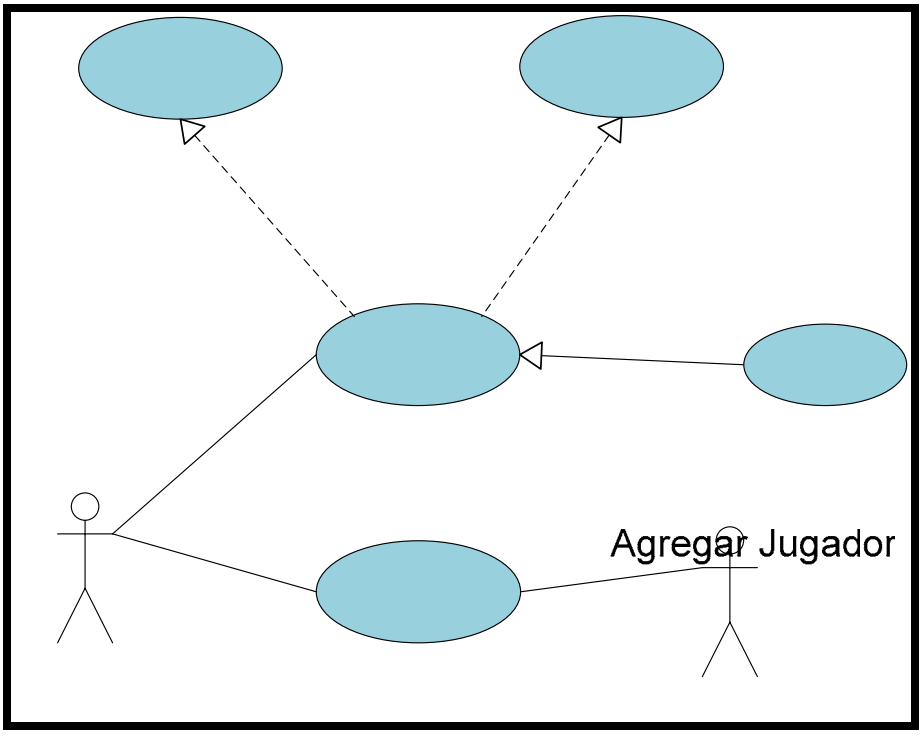

Figura 3.5 Caso de Uso Jugadores

Fuente: Propio

«extends»

Nombre del Caso de Uso: "Administrar Jugadores"

Actores: Administrador

Descripción: Permite al administrador agregar jugadores, agregar jugadores a los equipos.

Pre-Condición: Este proceso es invocado por el usuario desde el proceso "Menú", al pulsar la opción.

Flujo de Eventos:

Flujo Principal:

1. El usuario invoca al caso de uso.

2. Se muestran las fichas para agregar jugadores, añadir jugadores a equipos sólo si anteriormente el usuario inició sesión como administrador.
3. El usuario visualiza la información y selecciona la operación que desea realizar.
4. Finaliza el caso de uso.

Flujo Alterno:

El usuario tiene la opción de seleccionar otra operación o puede salir del sistema.

Nombre del Caso de Uso: "Agregar Jugador"

Actores: Administrador

Descripción: Permite al administrador agregar jugadores.

Pre-Condición: El usuario realiza la selección desde el proceso "Administrar Jugadores"

Flujo de Eventos:

Flujo Principal:

1. El usuario invoca al caso de uso.
2. Se llenan los distintos formularios con la información del jugador, para posteriormente añadirlo.
3. Finaliza el caso de uso.

Flujo Alterno:

El usuario tiene la opción de seleccionar otra operación o puede salir del sistema.

Nombre del Caso de Uso: "Agregar Jugadores a Equipo"

Actores: Administrador

Descripción: Permite al administrador agregar jugadores a los equipos.

Pre-Condición: El usuario realiza la selección desde el proceso "Administrar Jugadores"

Flujo de Eventos:

Flujo Principal:

1. El usuario invoca al caso de uso.
2. Se selecciona el jugador a agregar y se pulsa el botón para hacerlo.
3. Finaliza el caso de uso.

Flujo Alterno:

El usuario tiene la opción de seleccionar otra operación o puede salir del sistema.

Nombre del Caso de Uso: "Ver Datos Jugadores"

Actores: Administrador, Público

Descripción: Permite al administrador y al público ver los datos de los jugadores.

Pre-Condición: El usuario realiza la selección desde el proceso "Administrar Jugadores"

Flujo de Eventos:

Flujo Principal:

1. El usuario invoca al caso de uso.
2. Se selecciona el jugador del que se desean ver los datos.
3. Finaliza el caso de uso.

Flujo Alterno:

El usuario tiene la opción de seleccionar otra operación o puede salir del sistema.

3.4.6 Descripción del flujo de sucesos del Caso de Uso de Entrenadores

Nombre del Caso de Uso: "Administrar Entrenadores"

Actores: Administrador

Descripción: Permite al administrador agregar entrenadores, agregar entrenadores a los equipos.

Pre-Condición: Este proceso es invocado por el usuario desde el proceso “Menú”, al pulsar la opción.

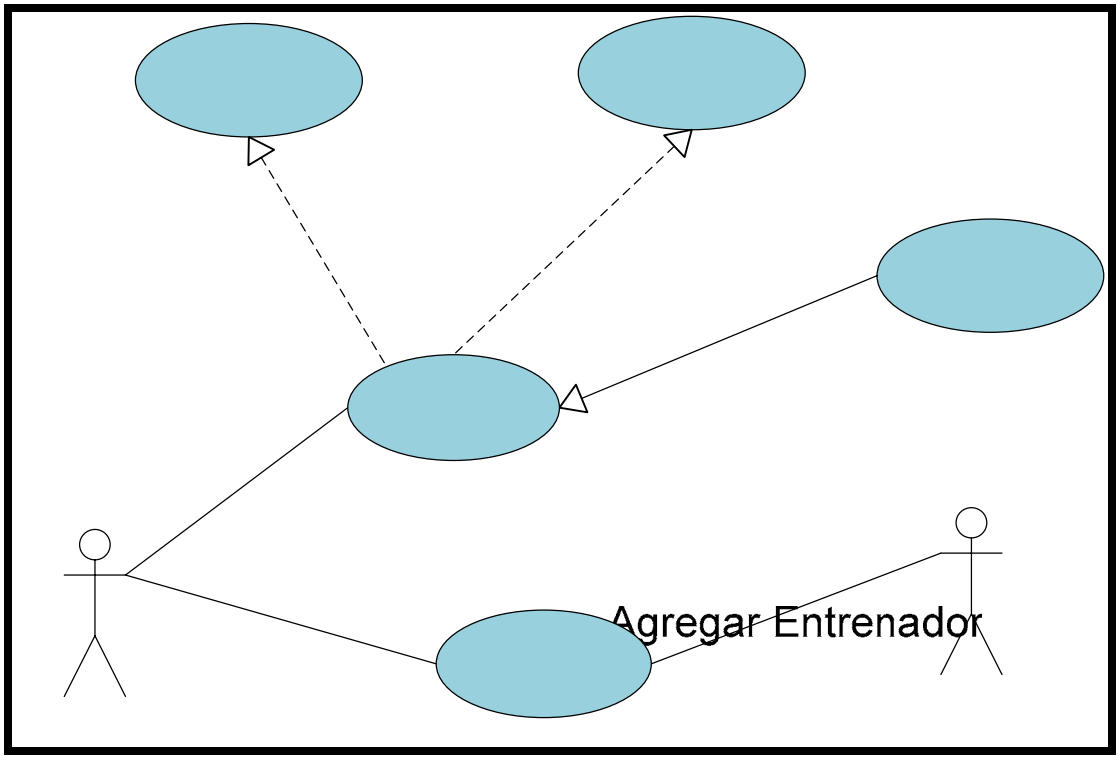

Figura 3.6 Caso de Uso Entrenadores

Fuente: Propia

Flujo de Eventos:

Flujo Principal:

1. El usuario invoca al caso de uso.
2. Se muestran las fichas para agregar entrenadores, añadir entrenadores a equipos sólo si anteriormente el usuario inició sesión como administrador.
3. El usuario visualiza la información y selecciona la operación que desea realizar.
4. Finaliza el caso de uso.

Flujo Alterno:

El usuario tiene la opción de seleccionar otra operación o puede salir del sistema.

Nombre del Caso de Uso: "Agregar Entrenador"

Actores: Administrador

Descripción: Permite al administrador agregar entrenadores.

Pre-Condición: El usuario realiza la selección desde el proceso "Administrar Entrenadores"

Flujo de Eventos:

Flujo Principal:

1. El usuario invoca al caso de uso.
2. Se llenan los distintos formularios con la información del entrenador, para posteriormente añadirlo.
3. Finaliza el caso de uso.

Flujo Alterno:

El usuario tiene la opción de seleccionar otra operación o puede salir del sistema.

Nombre del Caso de Uso: "Agregar Entrenadores a Equipo"

Actores: Administrador

Descripción: Permite al administrador agregar entrenadores a los equipos.

Pre-Condición: El usuario realiza la selección desde el proceso "Administrar Entrenadores"

Flujo de Eventos:

Flujo Principal:

1. El usuario invoca al caso de uso.
2. Se selecciona el entrenador a agregar y se pulsa el botón para hacerlo.

3. Finaliza el caso de uso.

Flujo Alterno:
El usuario tiene la opción de seleccionar otra operación o puede salir del sistema.

Nombre del Caso de Uso: "Ver Datos Entrenadores"
Actores: Administrador, Público
Descripción: Permite al administrador y al público ver los datos de los entrenadores.
Pre-Condición: El usuario realiza la selección desde el proceso "Administrar Entrenadores"

Flujo de Eventos:
Flujo Principal:

1. El usuario invoca al caso de uso.
2. Se selecciona el entrenador del que se desean ver los datos.
3. Finaliza el caso de uso.

Flujo Alterno:
El usuario tiene la opción de seleccionar otra operación o puede salir del sistema.

3.5 Diagramas de Clases de Análisis del Sistema

Una vez definidos los casos de uso del sistema, se procedió a la realización de un estudio más detallado de los mismos empleando diagramas de clases de análisis. De esta forma se representó la estructura de la aplicación, luego de un estudio independiente de cada caso de uso.

La ejecución de los casos de uso ya definidos se inicia con la entrada a la aplicación, pudiendo luego realizar una validación como administrador. Esta entrada se lleva a cabo por medio de una clase de interfaz así como también la entrada a los módulos siguientes a la validación.

Las clases de control son las encargadas de la ejecución y cumplimiento de las solicitudes hechas por los usuarios, tomando la información que se necesite en cada caso de las clases entidad; cuya tarea es modelar la información que se debe almacenar y proporcionar a las clases de control.

3.5.1 Diagrama de Clase de Análisis del Sistema

En la figura 3.7 se observa el diagrama de clase de análisis del sistema.

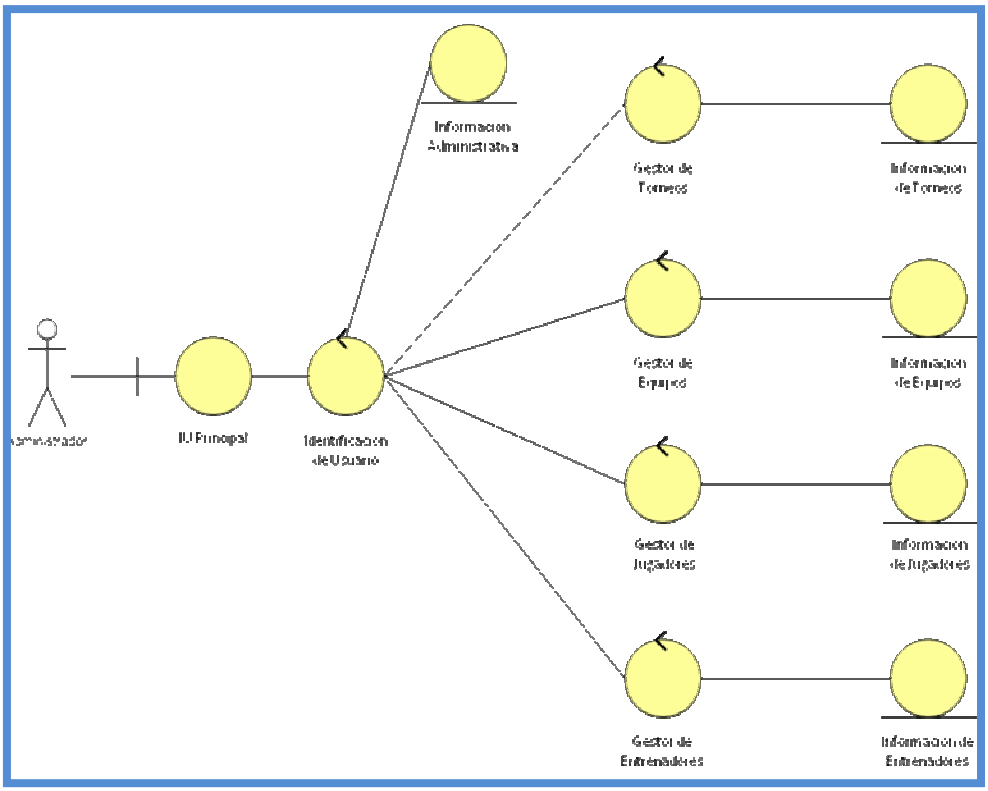

Figura 3.7. Diagrama de Clase de Análisis del Sistema

Fuente: Propia

El diagrama general de clases de análisis de la aplicación muestra una visión general de las interfaces, gestores y entidades que intervienen en el trabajo de la aplicación.

Se muestran los caminos entre procesos que debe poder seguir el usuario en su interacción con la aplicación y los gestores, entidades e interfaces que intervienen. Luego de la interacción *actor-interfaz*, actúa el gestor de autentificación para validar al usuario y desplegar las opciones posibles a seguir.

3.5.2 Diagrama de Clase de Análisis de Torneos

En la figura 3.8 se observa el diagrama de clase de análisis para los torneos.

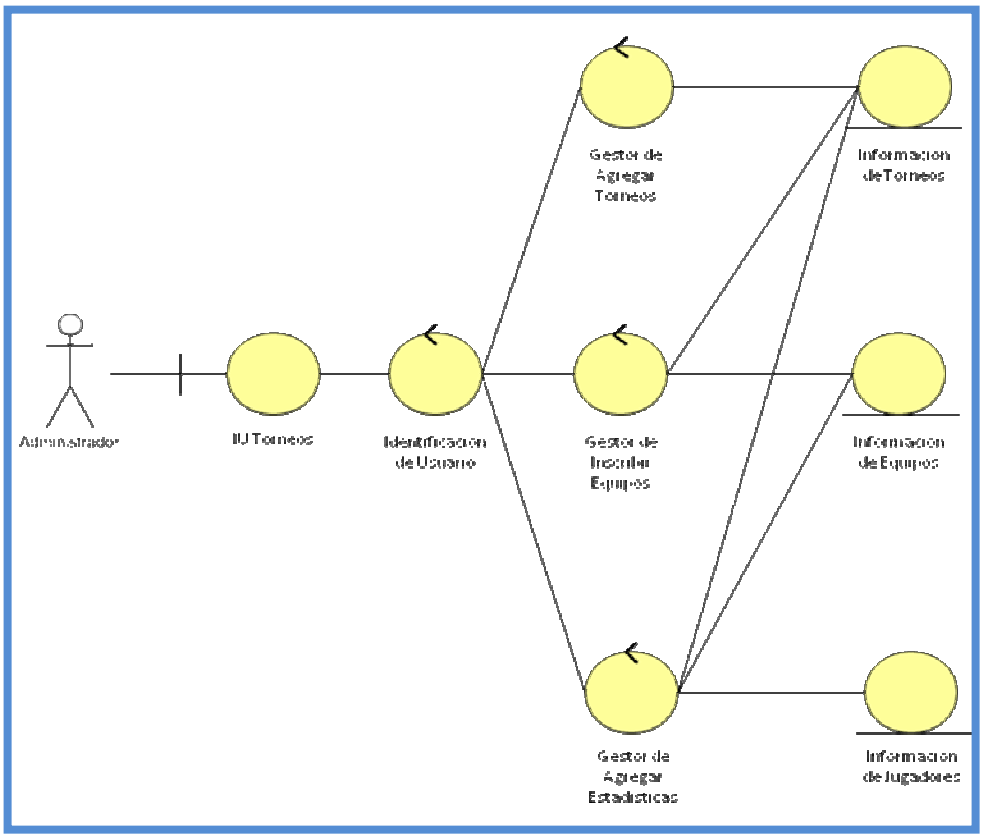

Figura 3.8 Diagrama de Clase de Análisis de Administrar Torneos

Fuente: Propia

En este diagrama de clases de análisis, se muestra de forma más detallada cómo interactúan las interfaces, los gestores y las entidades en el módulo de administración de torneos. La interfaz de entrada a la administración de torneos inicia el proceso de validación de los usuarios pidiendo un nombre de usuario y contraseña.

El gestor de agregar torneos, permite el ingreso de datos al sistema, el gestor de inscribir equipos se encarga del manejo de los equipos en el torneo y el gestor de agregar estadísticas, nos permite añadir las estadísticas a cada partido del torneo.

3.5.3 Diagrama de Clase de Análisis de Equipos

En la figura 3.9 se observa el diagrama de clase de análisis para los equipos.

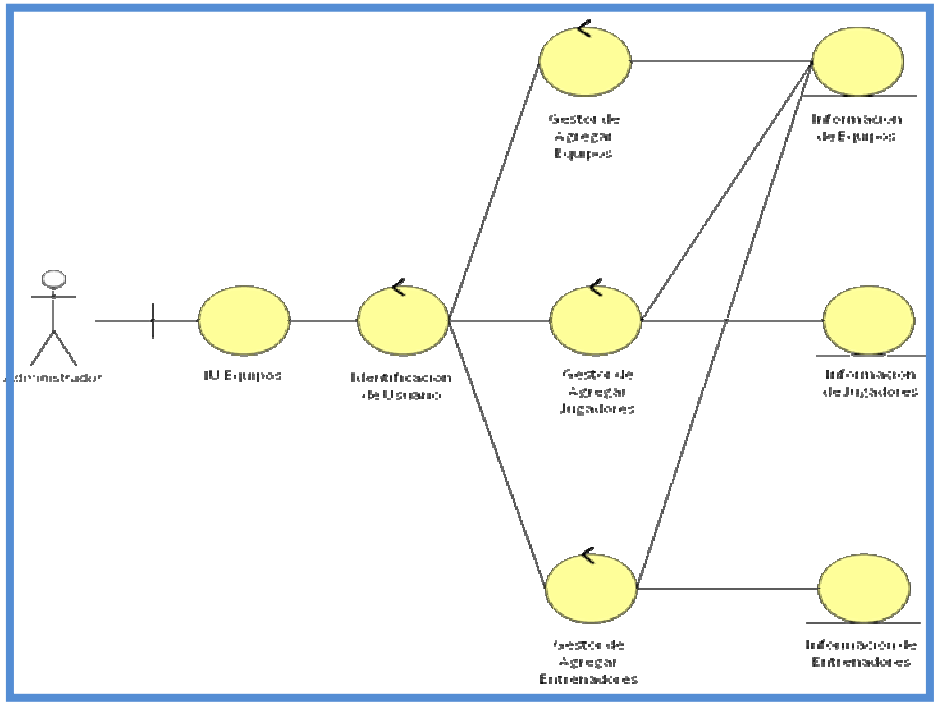

Figura 3.9 Diagrama de Clase de Análisis de Administrar Equipos

Fuente: Propia

En este diagrama de clases de análisis, se muestra de forma más detallada cómo interactúan las interfaces, los gestores y las entidades en el módulo de administración de equipos. La interfaz de entrada a la administración de equipos inicia el proceso de validación de los usuarios pidiendo un nombre de usuario y contraseña.

El gestor de agregar equipos, permite ingresar datos al sistema, el gestor de alterar jugadores se encarga del manejo de los jugadores en los equipos y el gestor de alterar entrenadores, permite manejar los entrenadores para cada equipo.

3.5.4 Diagrama de Clase de Análisis de Jugadores

En la figura 3.10 se observa el diagrama de clase de análisis para los jugadores.

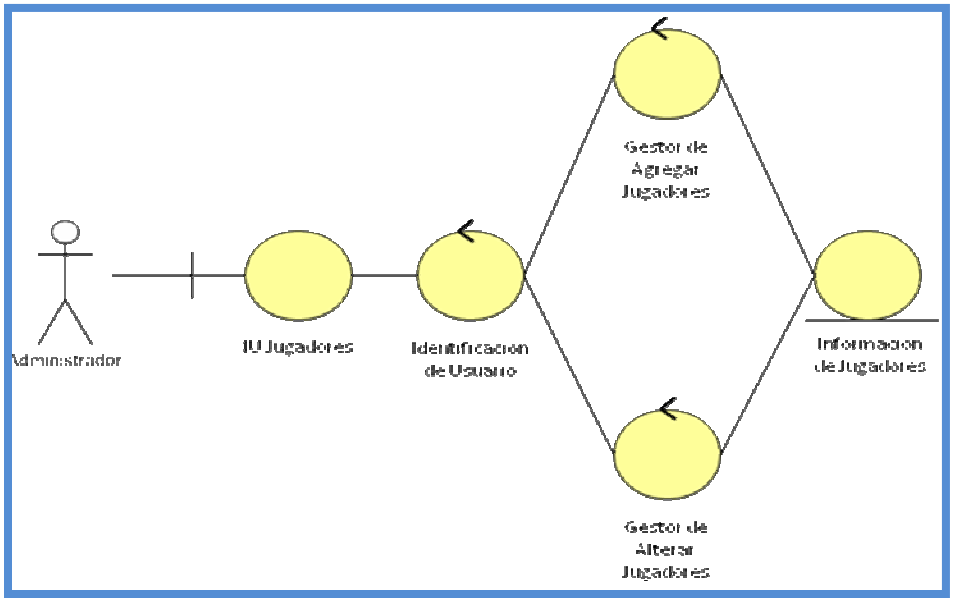

Figura 3.10 Diagrama de Clase de Análisis de Administrar Jugadores

Fuente: Propia

En este diagrama de clases de análisis, se muestra de forma más detallada cómo interactúan las interfaces, los gestores y las entidades en el módulo de administración de jugadores. La interfaz de entrada a la administración de jugadores inicia el proceso de validación de los usuarios pidiendo un nombre de usuario y contraseña.

El gestor de agregar jugadores, permite ingresar datos al sistema, el gestor de alterar jugadores se encarga de la modificación de información de los jugadores.

3.5.5 Diagrama de Clase de Análisis de Entrenadores

En la figura 3.11 se observa el diagrama de clase de análisis para los entrenadores.

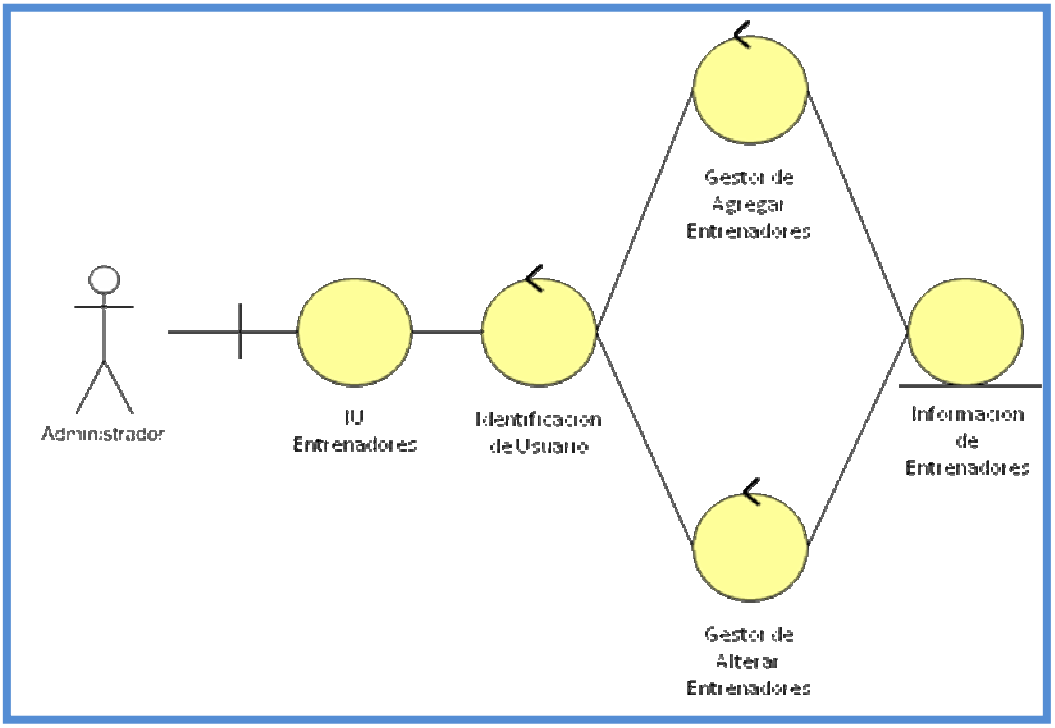

Figura 3.11 Diagrama de Clase de Análisis de Administrar Entrenadores

Fuente: Propia

En este diagrama de clases de análisis, se muestra de forma más detallada cómo interactúan las interfaces, los gestores y las entidades en el módulo de administración de entrenadores. La interfaz de entrada a la administración de entrenadores inicia el proceso de validación de los usuarios pidiendo un nombre de usuario y contraseña.

El gestor de agregar entrenadores, permite ingresar datos al sistema, el gestor de alterar entrenadores se encarga de la modificación de información de los entrenadores.

3.6 Diagramas de Colaboración del Sistema

El diagrama de colaboración muestra los diferentes objetos, las relaciones e interacciones que pueden darse entre ellos, permitiendo expresar el contexto de un grupo de objetos, a través de enlaces, y la interacción entre estos objetos, a través de mensajes.

El diagrama de colaboración se centra en estudiar todos los efectos de un objeto dado durante un escenario. Los objetos se conectan por medio de enlaces, cada enlace representa una instancia de una asociación entre las clases implicadas.

A continuación se muestras los diagramas de colaboración del sistema, el diagrama general y el de los módulos de **“Administrar Torneos”**, **“Administrar Equipos”, “Administrar Jugadores”** y el de **“Administrar Entrenadores”**, cuyo trabajo en conjunto es fundamental para que la aplicación cumpla con los requerimientos, por esto son los módulos sometidos a la mayor cantidad de pruebas.

3.6.1 Diagrama de Colaboración General del Sistema

En la figura 3.12 se presenta el diagrama de colaboración general del sistema.

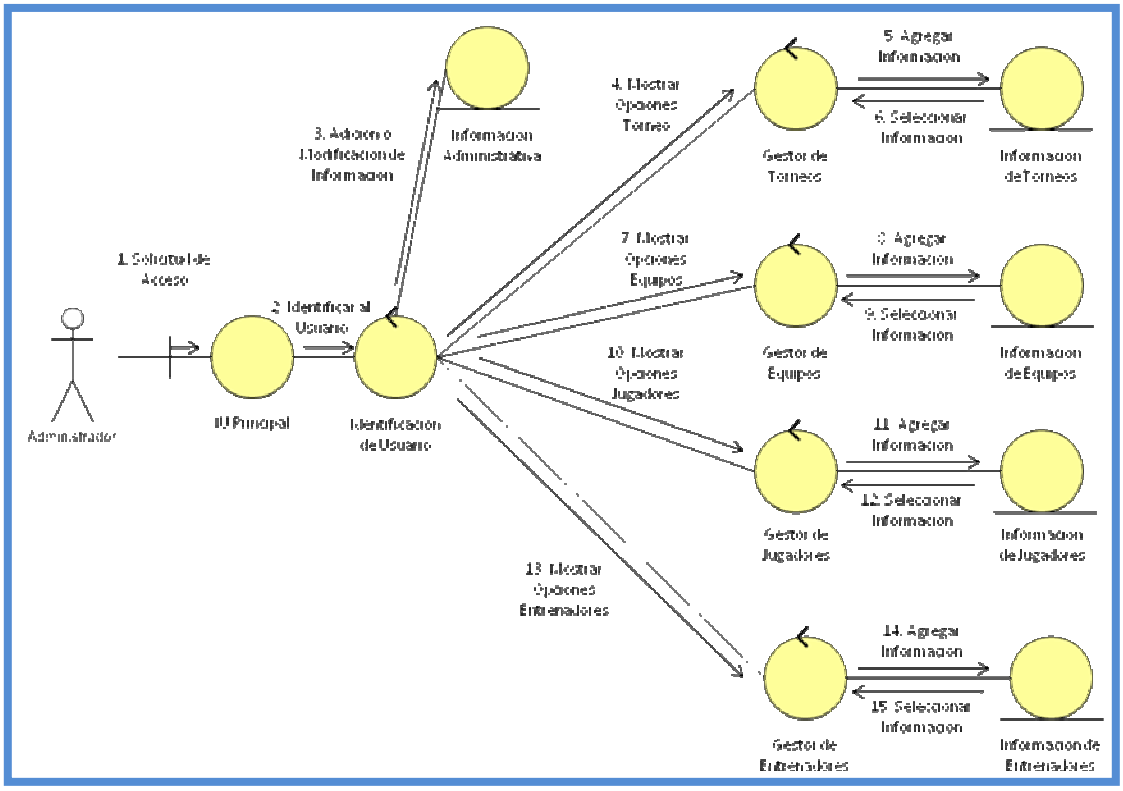

Figura 3.12 Diagrama de Colaboración General del Sistema

Fuente: Propia

3.6.2 Diagrama de Colaboración “Administrar Torneos”

En la figura 3.13 se presenta el diagrama de colaboración “Administrar Torneos”.

3.6.3 Diagrama de Colaboración “Administrar Equipos”

En la figura 3.14 se presenta el diagrama de colaboración “Administrar Equipos”.

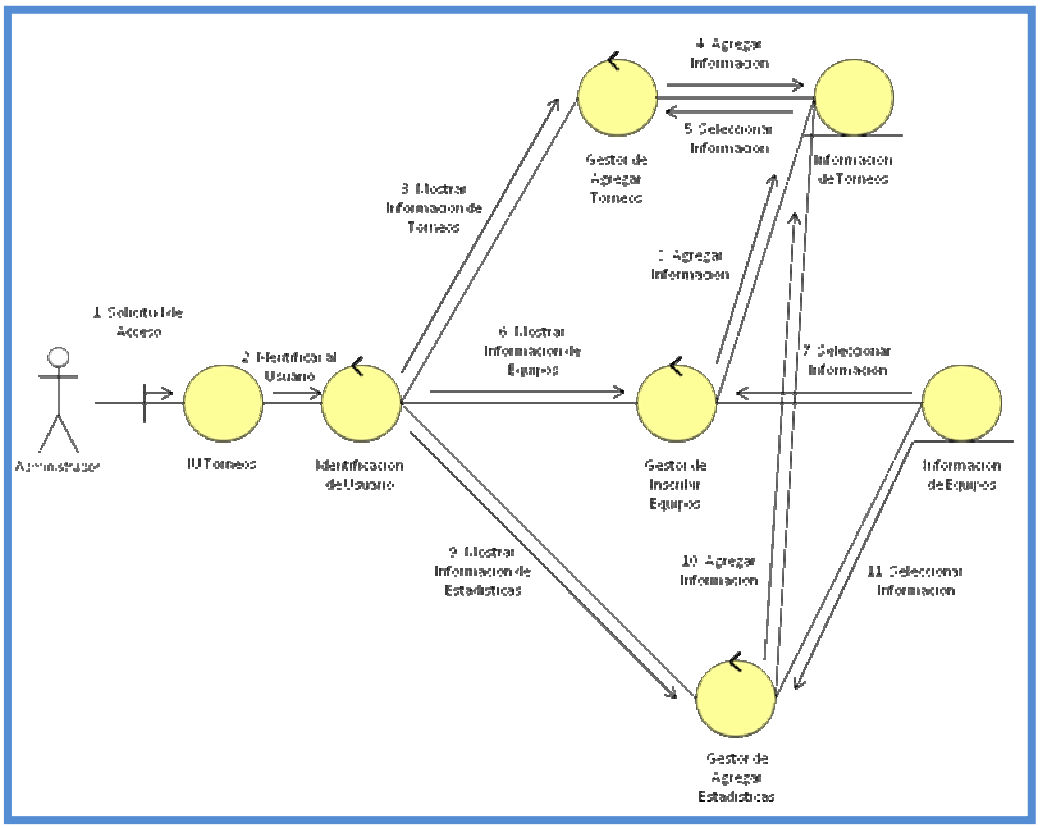

Figura 3.13 Diagrama de Colaboración "Administrar Torneos"

Fuente: Propia

3.6.4 Diagrama de Colaboración "Administrar Jugadores"

En la figura 3.15 se presenta el diagrama de colaboración "Administrar Jugadores"

3.6.5 Diagrama de Colaboración "Administrar Entrenadores"

En la figura 3.16 se presenta el diagrama de colaboración "Administrar Entrenadores".

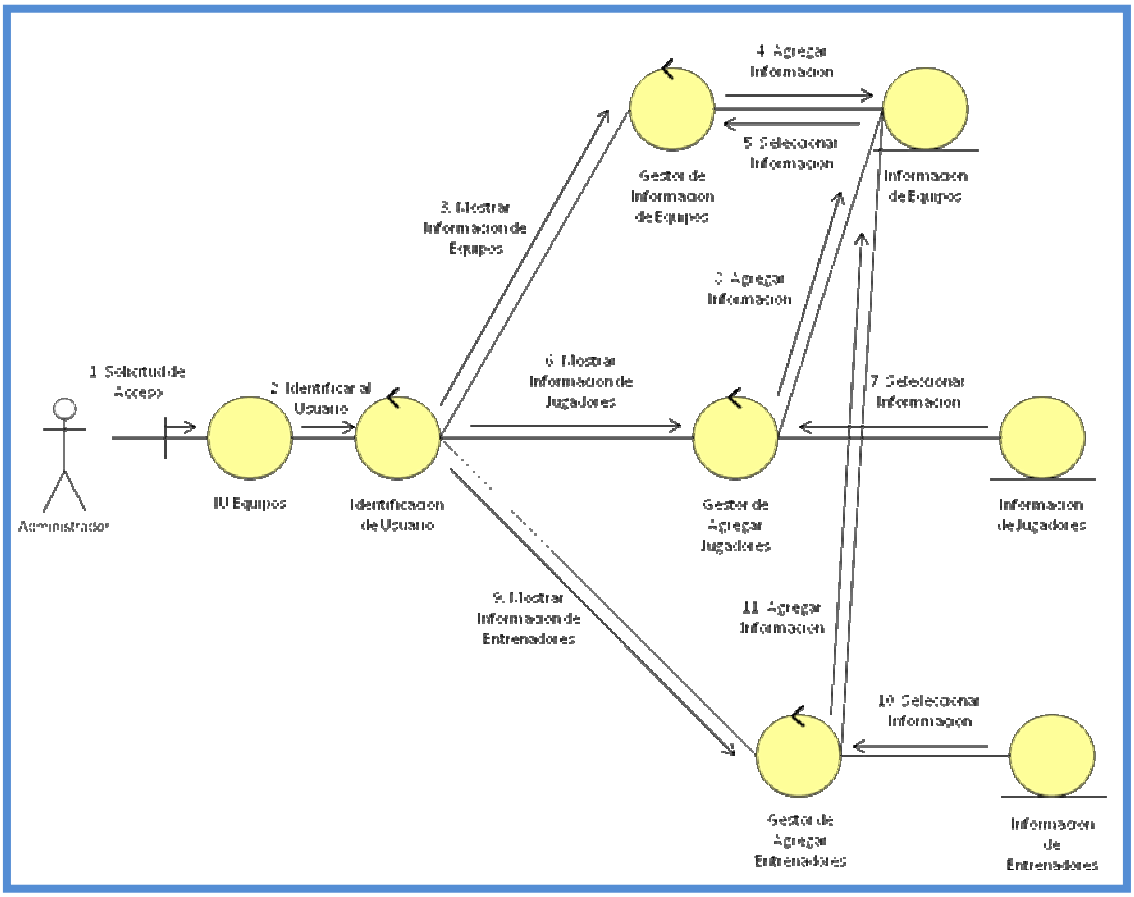

Figura 3.14 Diagrama de Colaboración "Administrar Equipos"

Fuente: Propia

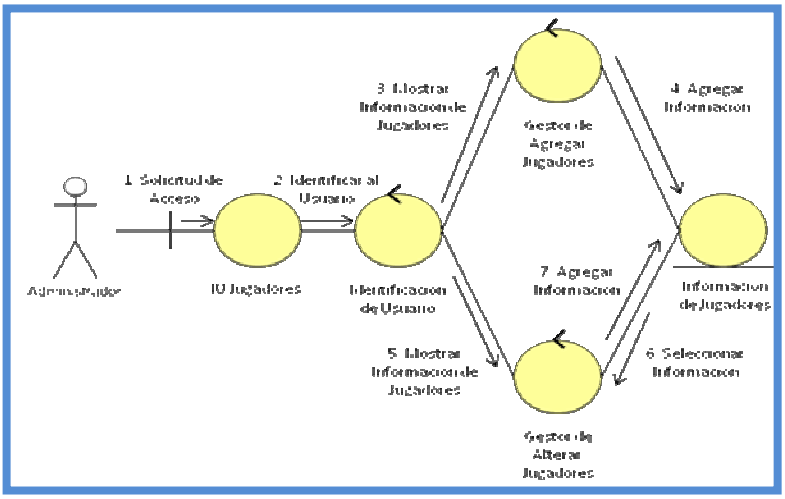

Figura 3.15 Diagrama de Colaboración "Administrar Jugadores"

Fuente: Propia

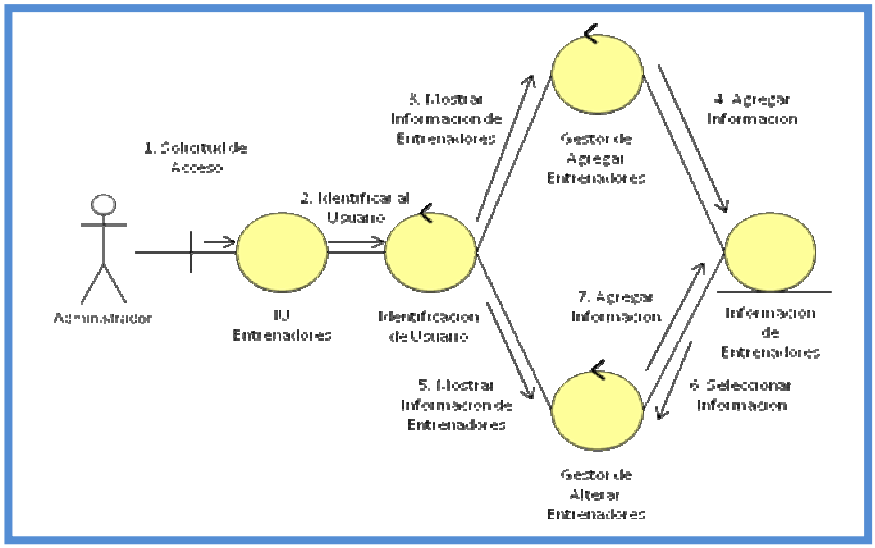

Figura 3.16 Diagrama de Colaboración "Administrar Entrenadores"

Fuente: Propia

3.7 Diagramas de Paquetes de Análisis

Representa un medio de organizar el modelo de análisis en piezas más pequeñas basándose en requisitos funcionales y dominio del sistema.

Define un espacio de nombre de modo que dos elementos diferentes, contenidos en dos paquetes diferentes, pueden tener el mismo nombre, pueda contener otros paquetes, sin límite de nivel de anidamiento. Un nivel dado puede contener una mezcla de paquetes y de otros elementos de modelado, de la misma manera que un directorio puede contener directorios y archivos. Puede constar de clases de análisis, de casos de uso y de otros paquetes. Poseen trazas con subsistemas en el diseño.

3.7.1 Diagrama de Paquetes de Análisis

El paquete representado en la figura 3.17 contiene los paquetes específicos del sistema.

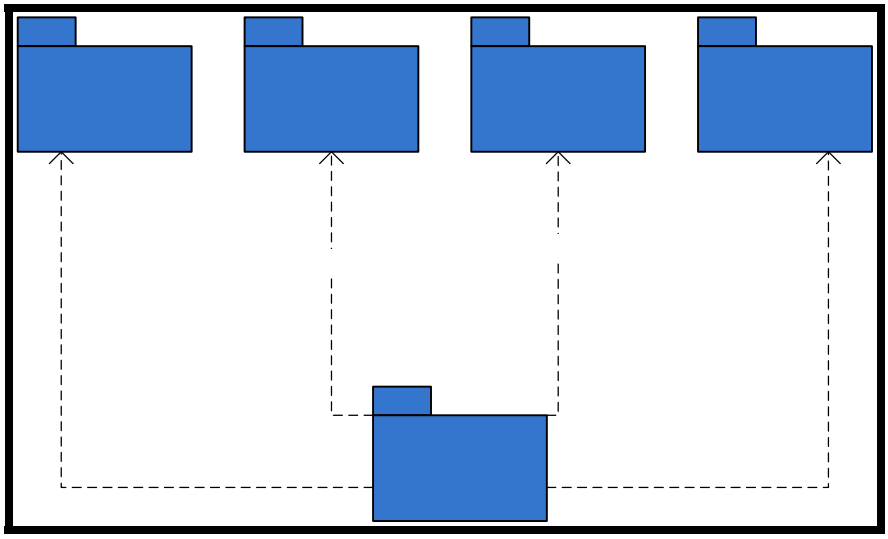

Figura 3.17 Diagrama de Paquetes de Análisis

Fuente: Propia

3.7.2 Diagrama de Paquetes de Análisis Torneos

Torneos Equipos

El paquete Torneos, representado en la figura 3.18 está asociado al caso de uso Torneos.

«traces»

Figura 3.18 Diagrama de Paquetes de Torneos

Fuente: Propia

«traces»

3.7.3 Diagrama de Paquetes de Análisis Equipos

El paquete Equipos, representado en la figura 3.19 está asociado al caso de uso Administrar Equipos.

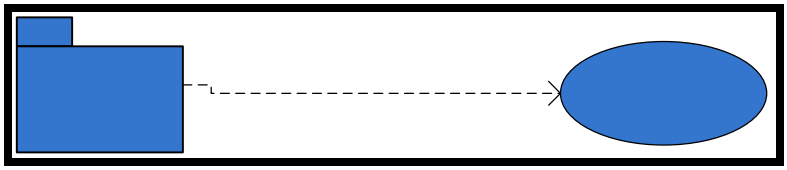

Figura 3.19 Diagrama de Paquetes de Equipos

Fuente: Propia

3.7.4 Diagrama de Paquetes de Análisis Jugadores

El paquete Jugadores, representado en la figura 3.20 está asociado al caso de uso Administrar Jugadores.

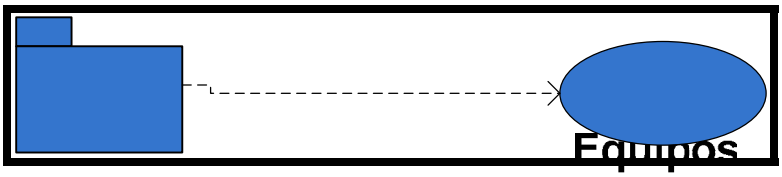

Figura 3.20 Diagrama de Paquetes de Jugadores

Fuente: Propia

3.7.5 Diagrama de Paquetes de Análisis Entrenadores

El paquete Administrar Entrenadores, representado en la figura 3.21 está asociado al caso de uso Administrar Entrenadores.

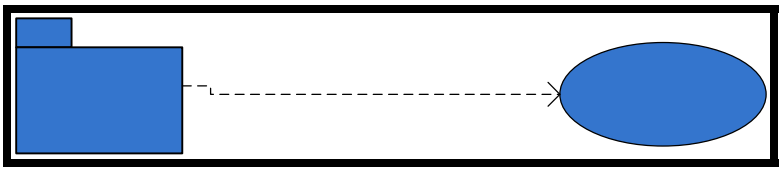

Figura 3.21 Diagrama de Paquetes de Entrenadores

Fuente: Propia

3.8 Conclusión de la Fase de Inicio

Los objetivos principales de la fase de inicio son: establecer un análisis del sistema propuesto, describir el contexto del sistema, capturar los requerimientos funcionales, identificar los riesgos críticos que pondrían en peligro el desarrollo del proyecto y proponer una arquitectura candidata factible. Dichos objetivos se cumplieron de una forma satisfactoria, ya que se obtuvo una primera versión del modelo que describe el contexto del sistema, una lista inicial de riesgos y un esbozo de los modelos que representan una primera versión el modelo de casos de uso y de modelo de análisis, los cuales describen una arquitectura candidata factible. En esta fase se obtuvo una buena comprensión del proyecto y la factibilidad de culminarlo. Los resultados alcanzados en esta fase se refinarán en la fase de elaboración.

CAPÍTULO IV. FASE DE ELABORACIÓN

4.1 Introducción

En esta fase se trabajará con la segunda fase del proceso unificado de desarrollo de software, conocida como fase de elaboración. Uno de los objetivos principales de esta fase es el obtener un entendimiento más detallado de los requerimientos de la aplicación debido a que estos son descritos brevemente en la fase de inicio. El otro objetivo prioritario de esta fase es el diseñar, implementar y validar la arquitectura del sistema. La funcionalidad a nivel de la aplicación no será completada, aunque se podrá compilar y probar la arquitectura durante esta fase. Se establecerá aquí la arquitectura de la aplicación, así como también el diseño de la Base de Datos y la interfaz de usuario. Durante esta fase se especifican la mayoría de los casos de usos, finalizando con las distintas vistas de los diferentes modelos del sistema.

En la fase de elaboración el objetivo del diseño es la obtención de la vista de la arquitectura del modelo de diseño, el cual está formado por los subsistemas e interfaces, las clases de diseño de la arquitectura, los diagramas de secuencia y el diseño de la base de datos, tomando como punto de partida los modelos previamente diseñados en la fase de inicio.

4.2 Análisis

En este flujo se debe proceder a realizar un nuevo estudio más detallado de los requerimientos del sistema para que estos puedan ser estudiados de manera más profunda.

4.2.1 Casos de Uso

Realizando un nuevo análisis para obtener mayor detalle de los requerimientos anteriormente planteados fue descubierto un nuevo requisito, el cual afecta el caso de uso Torneos del sistema representado en el capítulo anterior.

4.2.1.1 Descripción del flujo de sucesos del Caso de Uso Torneos

Debido a la introducción de este nuevo requisito se debe modificar el caso de uso "Torneos" de modo tal que éste especifique dicho requerimiento. El nuevo diagrama es planteado en la figura 4.1.

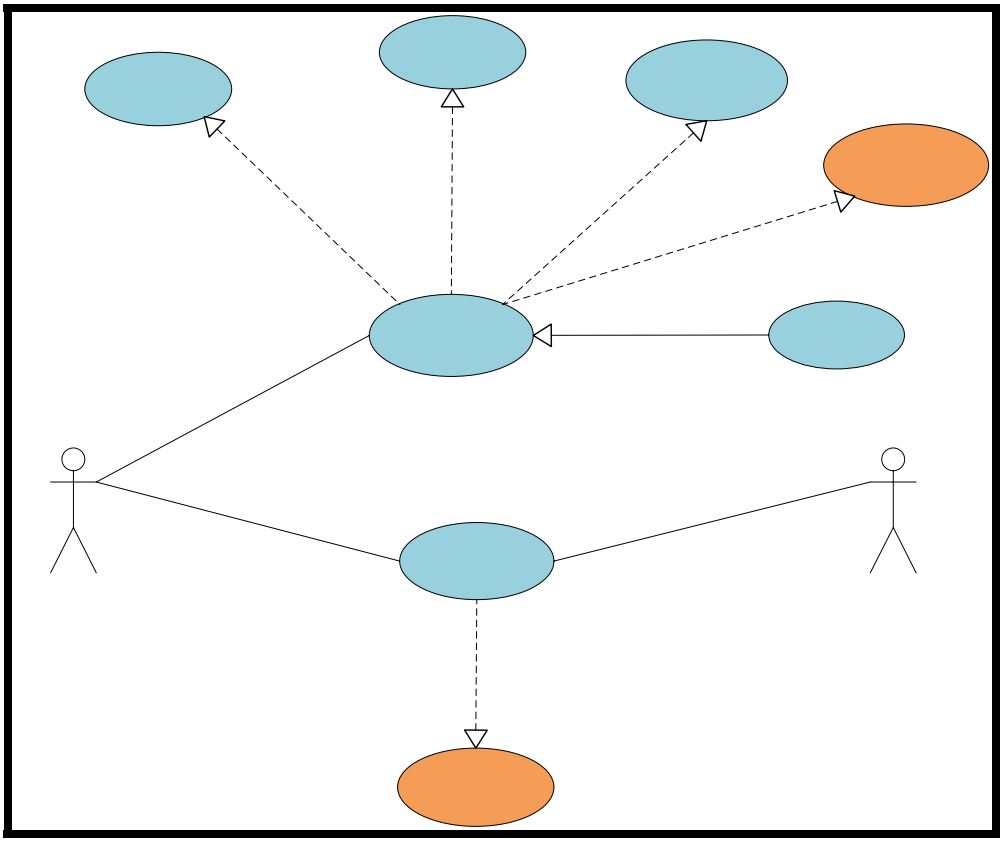

Figura 4.1. Caso de Uso Administrar Torneos

Fuente: Propia

Agregar E
Torne

Agregar Torneo

Nombre del Caso de Uso: "Generar Reporte De Torneo"

Actores: Administrador, Público

Descripción: Permite al administrador y público en general generar reportes de los torneos.

Pre-Condición: El usuario realiza la selección desde el proceso "Administrar Torneos"

Flujo de Eventos:

Flujo Principal:

1. El usuario invoca al caso de uso.
2. Se presiona el botón de generación de reportes.
3. Finaliza el caso de uso.

Flujo Alterno:

El usuario tiene la opción de seleccionar otra operación o puede salir del sistema.

Nombre del Caso de Uso: "Ver Reporte De Torneo"

Actores: Administrador, Público

Descripción: Permite al administrador y público en general ver reportes de los torneos.

Pre-Condición: El usuario realiza la selección desde el proceso "Administrar Torneos"

Flujo de Eventos:

Flujo Principal:

1. El usuario invoca al caso de uso.
2. Se presiona el botón de ver reportes.
3. Finaliza el caso de uso.

Flujo Alterno:

El usuario tiene la opción de seleccionar otra operación o puede salir del sistema.

4.2.2 Diagrama de Clases de Análisis

Se realiza una evaluación general del caso de uso agregado al sistema, sin llegar a detalles muy profundos de estos, obteniendo como resultado el Modelo de Análisis (diagramas de colaboración y clases de análisis), el cual permite tener una especificación un poco más precisa de los requisitos obtenidos y recogidos, logrando de esta manera una guía para estructurar el sistema.

4.2.2.1 Diagrama de Clase de Análisis de Administrar Torneos

El nuevo diagrama de clase de análisis para los torneos se presenta en la figura 4.2 añadiéndole un gestor para la generación de reportes.

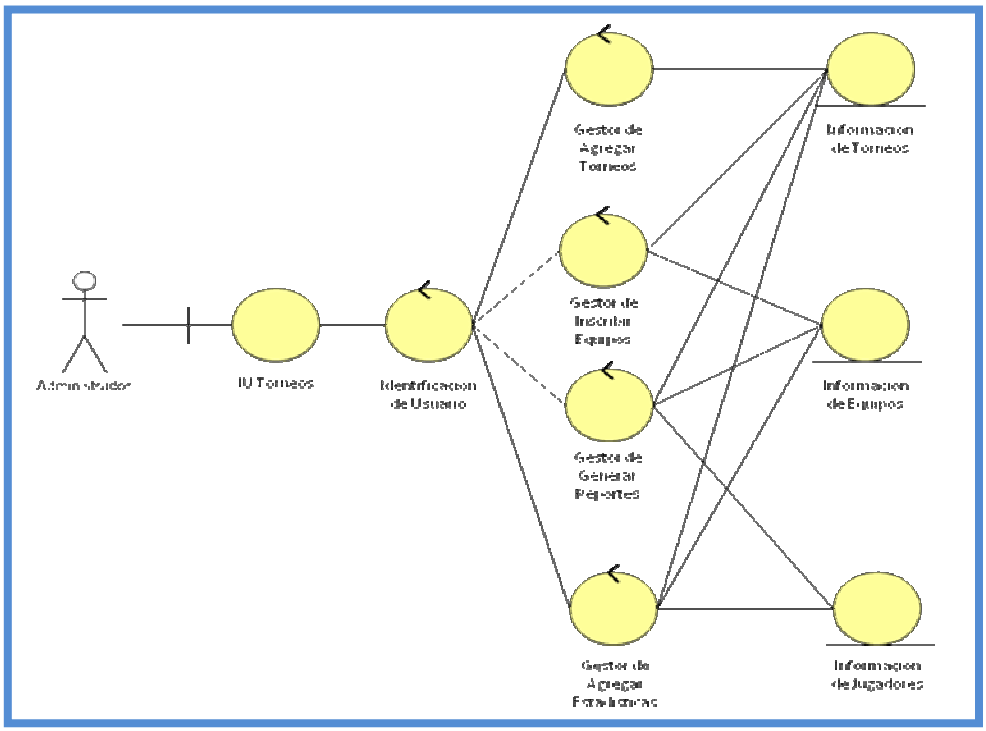

Figura 4.2. Diagrama de Clase de Análisis de Administrar Torneos

Fuente: Propia

4.2.3 Diagrama de Colaboración

Analizando el nuevo caso de uso presente, se debe realizar nuevamente un diagrama de colaboración para las partes afectadas por este nuevo caso de uso.

4.2.3.1 Diagrama de Colaboración "Administrar Torneos"

Debido a que anteriormente se contaba con un gestor para la información, este diagrama de colaboración no debe ser alterado, ya que dicho gestor se utiliza para la generación de reportes, sin embargo, se presenta nuevamente en la figura 4.3.

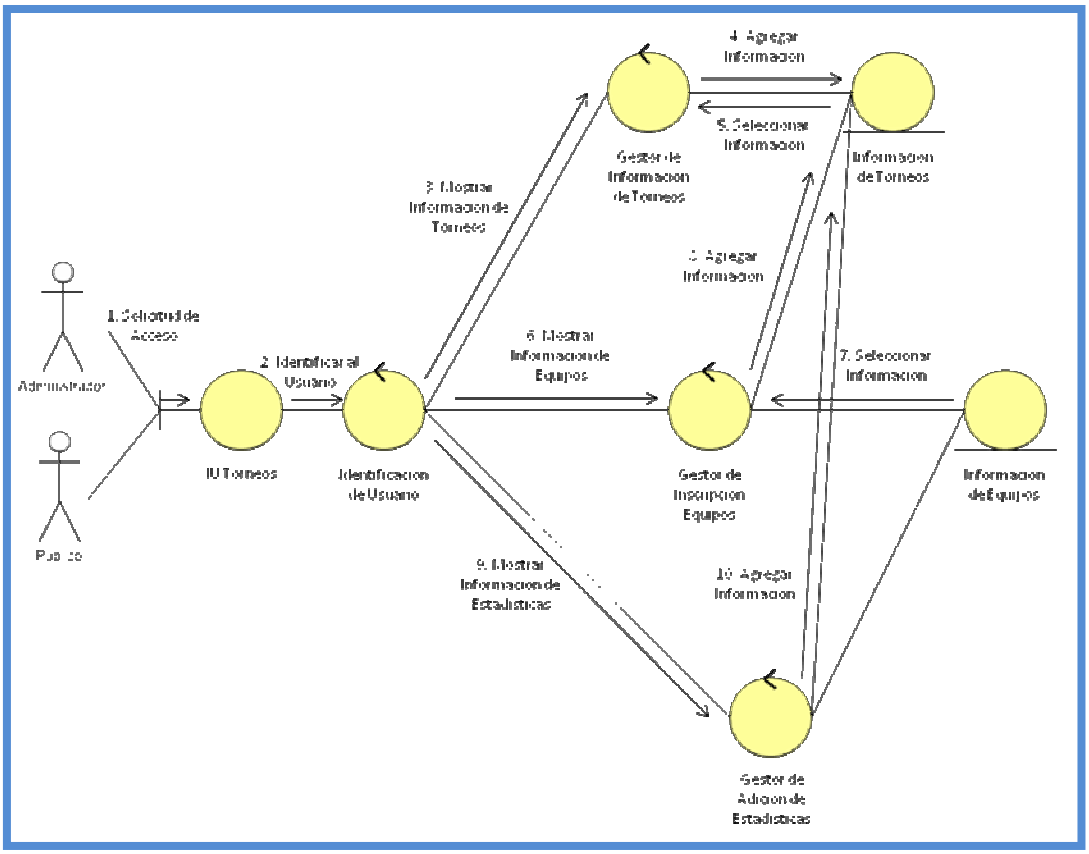

Figura 4.3. Diagrama de Colaboración "Administrar Torneos"

Fuente: Propia

4.2.4 Identificación de los Paquetes de Análisis

Los paquetes explicados en el capítulo anterior permanecen intactos, debido a esto, sólo debe ser modificada la parte interna del paquete Torneos, que se presenta en la figura 4.4.

Figura 4.4 Diagrama de Paquetes de Torneos

Fuente: Propia

4.3 Diseño de la Arquitectura del Sistema

La arquitectura del software consiste en un conjunto de abstracciones coherentes que proporcionan el marco de referencia necesario para guiar la construcción del software. La arquitectura de software define, de manera abstracta, los componentes que llevan a cabo alguna tarea de computación, sus interfaces y la comunicación ente ellos.

«tra

Torneos

En este apartado se deben definir los componentes que van a formar parte de la base del sistema en desarrollo de una forma robusta y óptima para poder luego desarrollar los módulos y realizar la integración de los mismos.

4.3.1 Identificación de las Clases de Diseño

En este apartado se desea iniciar con el diseño de las clases que compondrán la arquitectura del sistema y para esto se utilizan los gestores anteriormente definidos en

las clases de análisis. Estos gestores definen las funciones principales del sistema y cada uno de ellos debe ser asignado a una clase particular para que realice su función.

4.3.1.1 Clase de Diseño Torneos

Esta clase se compone de aquellos gestores que se encargan de manejar toda la información relacionada con los torneos y se presenta en la figura 4.5.

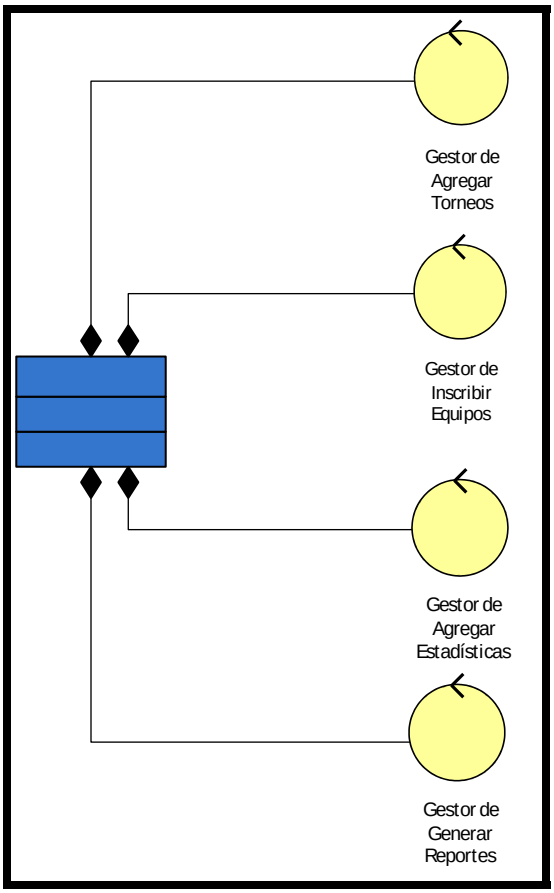

Figura 4.5 Clase de Diseño Torneos

4.3.1.2 Clase de Diseño Equipos

Esta clase se compone de aquellos gestores que se encargan de manejar toda la información relacionada con los equipos y se presenta en la figura 4.6.

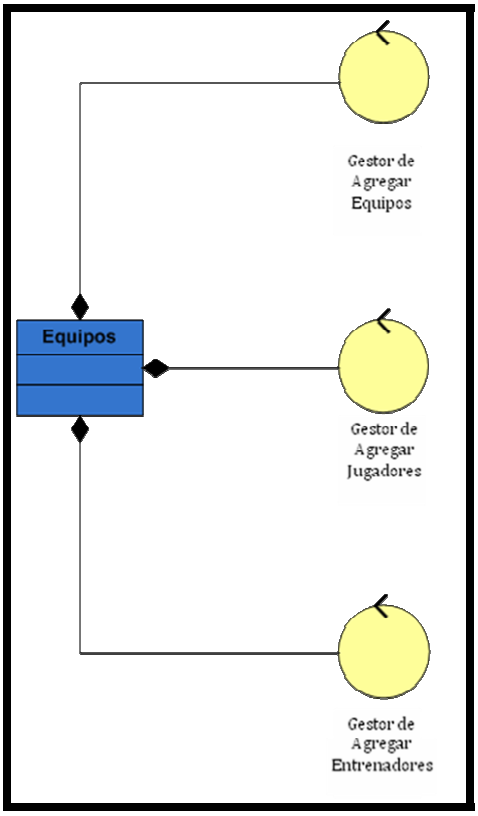

Figura 4.6 Clase de Diseño Equipos

Fuente: Propia

4.3.1.3 Clase de Diseño Jugadores

Esta clase se compone de aquellos gestores que se encargan de manejar toda la información relacionada con los jugadores y se presenta en la figura 4.7.

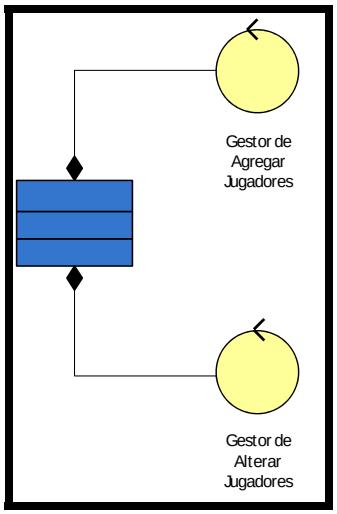

Figura 4.7 Clase de Diseño Jugadores

Fuente: Propia

Jugadores

4.3.1.4 Clase de Diseño Entrenadores

Esta clase se compone de aquellos gestores que se encargan de manejar toda la información relacionada con los entrenadores y se presenta en la figura 4.8.

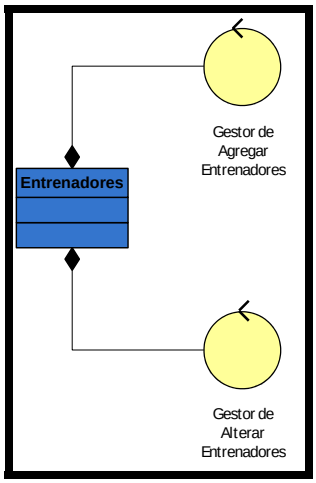

Figura 4.8 Clase de Diseño Entrenadores

Fuente: Propia

4.3.2 Identificación de los Paquetes de Diseño

En este apartado se deben asignar las clases de diseño anteriormente identificadas a los paquetes de diseño de la aplicación los cuales serán representados posteriormente en el diagrama de capas del sistema.

4.3.2.1 Diagrama de Paquetes de Diseño Torneos

El paquete de diseño de Torneos representado en la figura 4.9 está asociado a la clase de diseño Torneos.

Figura 4.9 Diagrama de Paquetes de Diseño Torneos

Fuente: Propia

4.3.2.2 Diagrama de Paquetes de Diseño Equipos

El paquete de diseño de Equipos representado en la figura 4.10 está asociado a la clase de diseño Equipos.

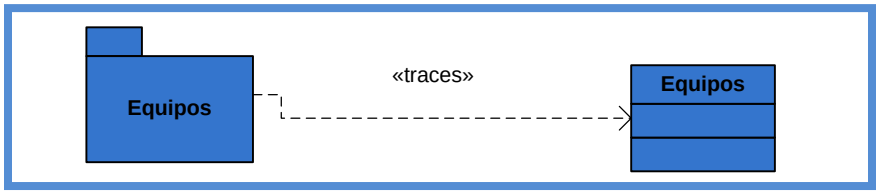

Figura 4.10 Diagrama de Paquetes de Diseño Equipos

Fuente: Propia

«trace

Torneos

4.3.2.3 Diagrama de Paquetes de Diseño Jugadores

El paquete de diseño de Jugadores representado en la figura 4.11 está asociado a la clase de diseño Jugadores.

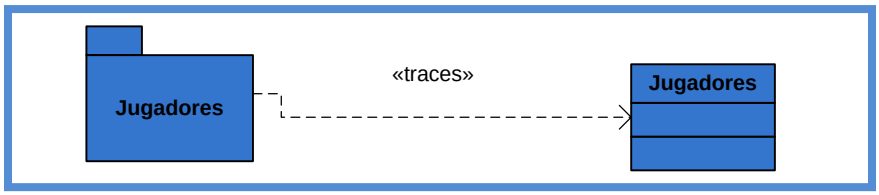

Figura 4.11 Diagrama de Paquetes de Diseño Jugadores

Fuente: Propia

4.3.2.4 Diagrama de Paquetes de Diseño Entrenadores

El paquete de diseño de Entrenadores representado en la figura 4.12 está asociado a la clase de diseño Entrenadores.

Figura 4.12 Diagrama de Paquetes de Diseño Entrenadores

Fuente: Propia

4.3.3 Diagrama de Clases

Este diagrama muestra la estructura estática del sistema reflejando las relaciones entre las clases, lo que permitirá visualizar lo que el sistema puede hacer y de cómo puede ser construido.

4.3.3.1 Diagrama de Clases del Sistema

El diagrama representado en la figura 4.13 representa tanto a las clases de la capa específica como también a las clases de la capa general de la aplicación.

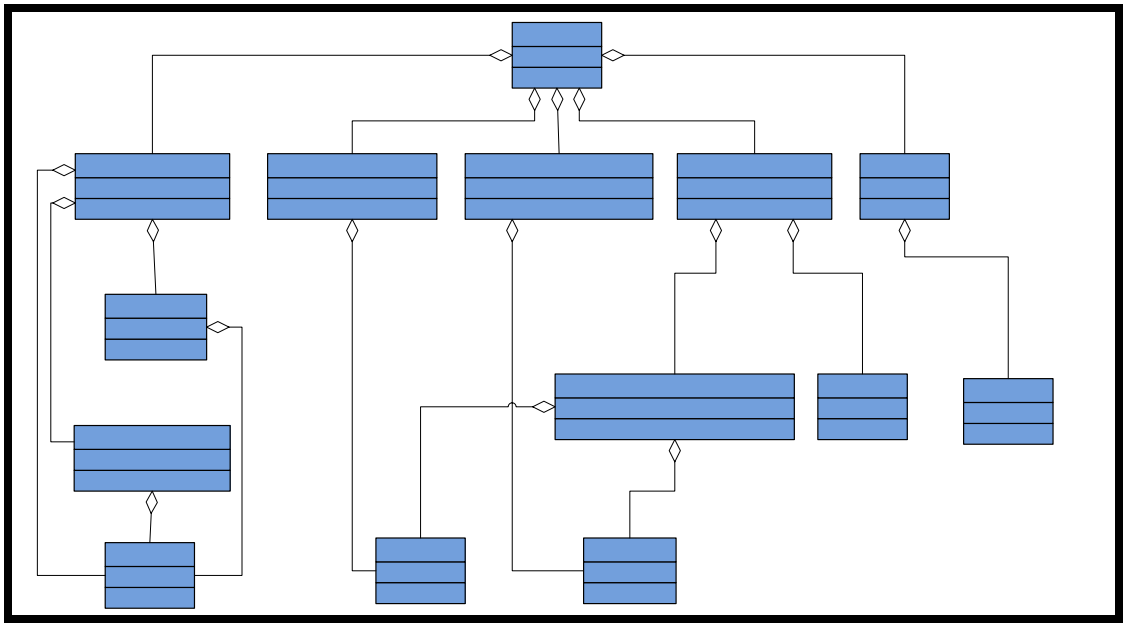

Figura 4.13. Diagrama de Clases del Sistema General

Fuente: Propia

4.3.3.1.1 Clase Torneos

IU Administrar Torneos IU Administrar Jugadores IU Admi

Esta clase gestiona toda la información y los procesos relacionados con todos los tipos de torneos que pueden ser almacenados en el sistema. Debe poseer atributos que permitan conocer el nombre, la fecha de inicio, la fecha de finalización, los días de partido, la cantidad de equipos, y la cantidad de partidos diarios de cada torneo. Esta clase también contendrá todos los métodos que permitan la inserción, modificación, eliminación y consulta de datos en la base de datos referentes a los torneos en sí,

IU Ver Torneos

IU Agregar Estadísticas

además de las funciones que generen los partidos de los torneos y la adición y obtención de estadísticas de los equipos.

4.3.3.1.2 Clase Jugadores

Esta clase se encargará de manejar todos los datos y métodos referentes a los jugadores. En ella se encuentran los métodos para insertar, modificar y eliminar registros de jugadores de la base de datos, así como también los métodos que retornan las características de los jugadores. Otro de los métodos importantes que posee esta clase es el que nos permite conocer los jugadores que no poseen ningún equipo así como también los de añadir y obtener las estadísticas para los jugadores.

4.3.3.1.3 Clase Equipos

El objetivo de esta clase es el manejo de la información referente a los equipos que serán almacenados en el sistema. Esta clase poseerá funciones que permitan almacenar u obtener equipos de la base de datos, así como también permitirá el modificar la plantilla o el conjunto de entrenadores de los equipos.

4.3.3.1.4 Clase Entrenadores

Esta clase se encargará de manejar todos los datos y métodos referentes a los entrenadores. En ella se encuentran los métodos para insertar, modificar y eliminar registros de entrenadores de la base de datos, así como también los métodos que retornan los datos personales de los entrenadores. Otro de los métodos importantes que posee esta clase es el que nos permite conocer los entrenadores que no poseen ningún equipo.

4.3.3.1.5 Clase Admin

Esta clase será únicamente utilizada por administradores del sistema, y mediante ella se puede eliminar por completo la base de datos del sistema, así como también la creación de la misma. Permite a su vez añadir nuevos administradores al sistema y el inicio de sesión de los mismos, para que estos puedan realizar actividades que requieran del permiso de un administrador.

4.3.4 Diagrama de Capas

El diseño de la arquitectura del sistema involucra cuatros capas: la capa general de la aplicación, la capa específica de la aplicación, la capa intermedia y finalmente la capa de software del sistema. El diagrama de capas mostrado en la figura 4.14 presenta todas las capas del software en las que se distribuye la funcionalidad del sistema que se requieren para llevar a cabo la aplicación y en un futuro ser implementado. Como se puede visualizar en la capa específica de la aplicación se presentan un conjunto de paquetes que encapsulan las diferentes clases del sistema. En la capa general se representa el sistema en su totalidad y la interacción de cada uno de los paquetes necesarios para la realización de la gestión del sistema y las relaciones entre los elementos de cada capa se representan mediante trazas.

4.3.5 Diagramas de Secuencia

Los diagramas de secuencia permiten especificar el sentido o recorrido del usuario dentro de la aplicación, es decir, muestran los procesos que se van ejecutando a medida que el mismo se mueve dentro de ella.

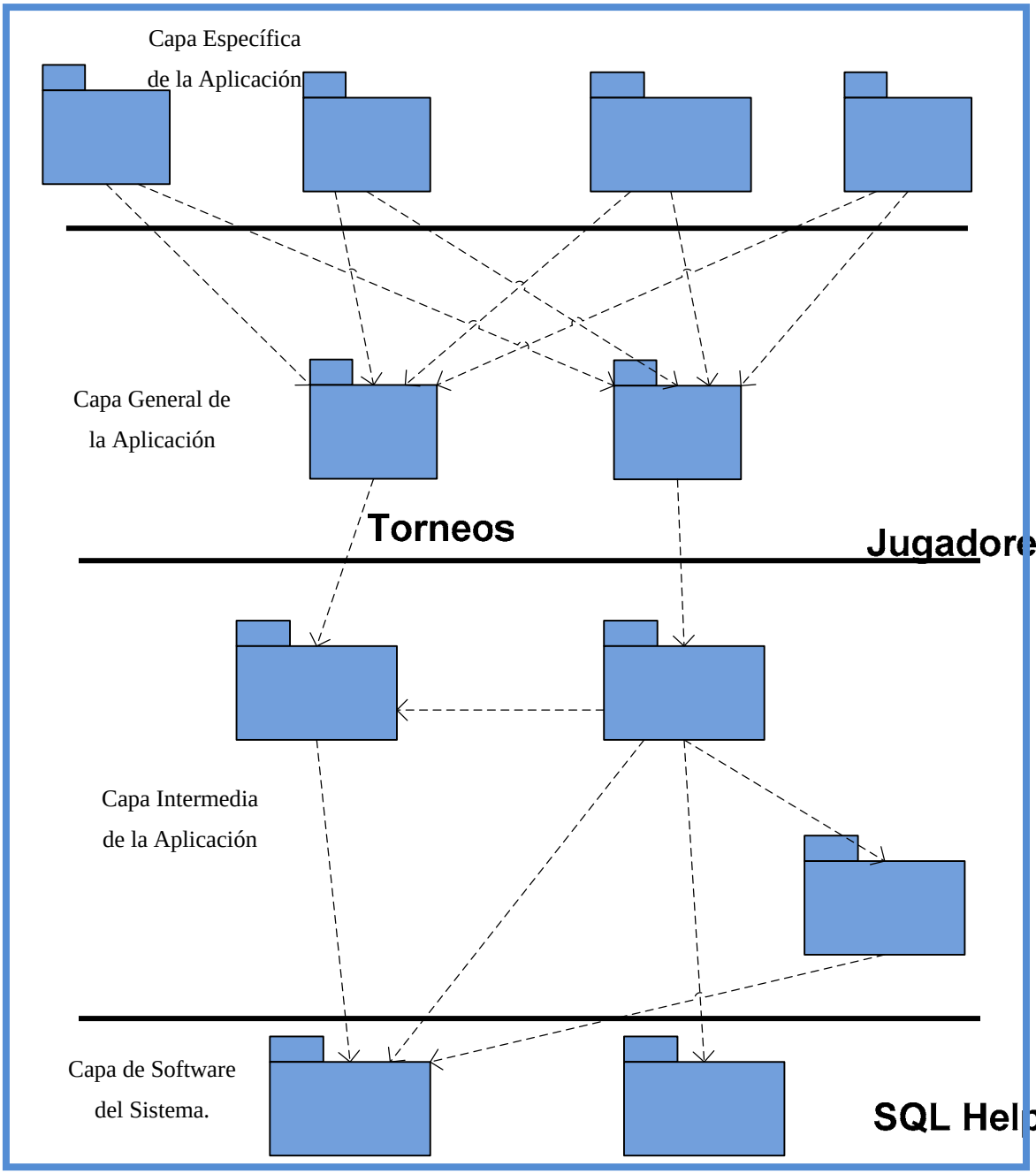

Figura 4.14 Diagrama de Capas de la Aplicación

Fuente: Propia

En estos se representan los mensajes que son enviados durante el camino de un módulo a otro a través de líneas con punta de flecha y la duración de cada uno de los módulos por los que pasa la ejecución, por medio de rectángulos cuya extensión vertical representa esa duración.

Del estudio de los requerimientos para el desarrollo de la aplicación, se derivaron los casos de uso ya especificados y entre ellos los de mayor importancia y cuya operatividad en conjunto determina o permite la realización de las operaciones de mayor valor en el manejo de eventos. Estos casos de uso son:

- Agregar Estadísticas a Torneo
- Agregar Jugadores

4.3.5.1 Diagrama de Secuencia “Agregar Estadísticas A Torneo”

El diagrama mostrado en la figura 4.15 define la secuencia de pasos y los módulos que se ejecutan durante la utilización de la aplicación por parte del usuario, para la adición de estadísticas de los partidos, en el módulo de administración de torneos. También se muestran los mensajes descriptivos del tipo de información que se va transmitiendo de un módulo a otro.

El proceso de adición de estadísticas se inicia después del usuario hacer log in como administrador del sistema, procediendo al menú de los torneos para buscar los torneos que se encuentran en proceso actualmente. Luego de seleccionar el torneo, el usuario debe proceder a seleccionar el partido al que desea añadir las estadísticas para posteriormente introducir los valores para los goles, tarjetas amarillas, tarjetas rojas y abridores de los equipos participantes en dicho partido. Tras realizar exitosamente estas operaciones, el usuario estará en la posibilidad de añadir estadísticas a otro partido o podrá volver al menú principal de la aplicación.

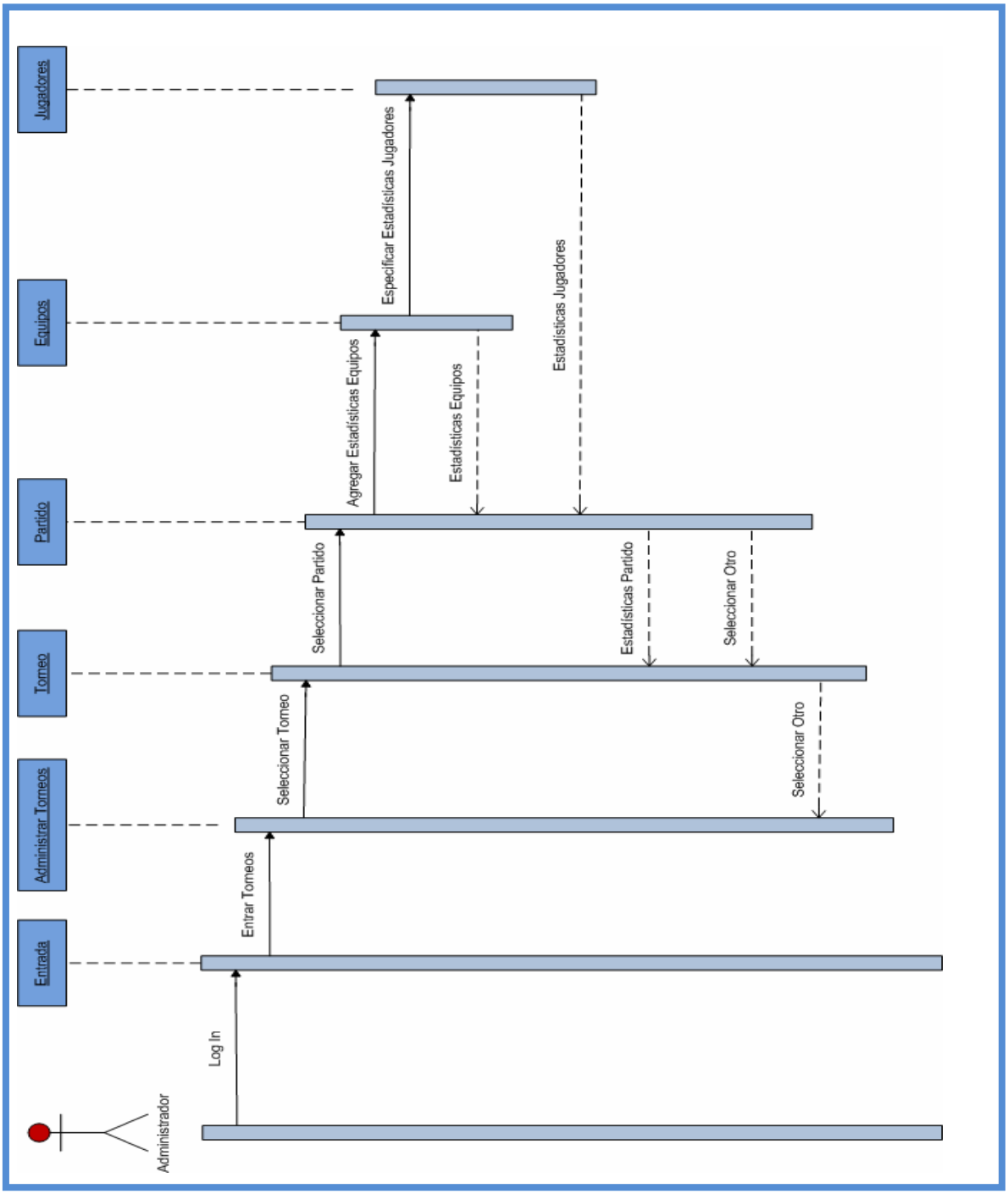

Figura 4.15 Diagrama de Secuencia "Agregar Estadísticas A Torneo"

Fuente: Propia

4.3.5.2 Diagrama de Secuencia “Agregar Jugador a Equipo”

El diagrama mostrado en la figura 4.16 define la secuencia de pasos y los módulos que se ejecutan durante la utilización de la aplicación por parte del usuario, para la adición de jugadores a los equipos, en el módulo de administración de equipos. También se muestran los mensajes descriptivos del tipo de información que se va transmitiendo de un módulo a otro.

El proceso de modificación de jugadores en los equipos se inicia después del usuario hacer log in como administrador del sistema, procediendo al menú de los equipos para buscar los equipos que se encuentran almacenados actualmente. Luego de seleccionar el equipo, el usuario debe seleccionar la opción para alterar los jugadores de dicho equipo, donde podrá agregar jugadores a la plantilla o quitarlos de la misma. Tras realizar exitosamente estas operaciones, el usuario estará en la posibilidad de realizar modificaciones a la plantilla de otro equipo o podrá volver al menú principal de la aplicación.

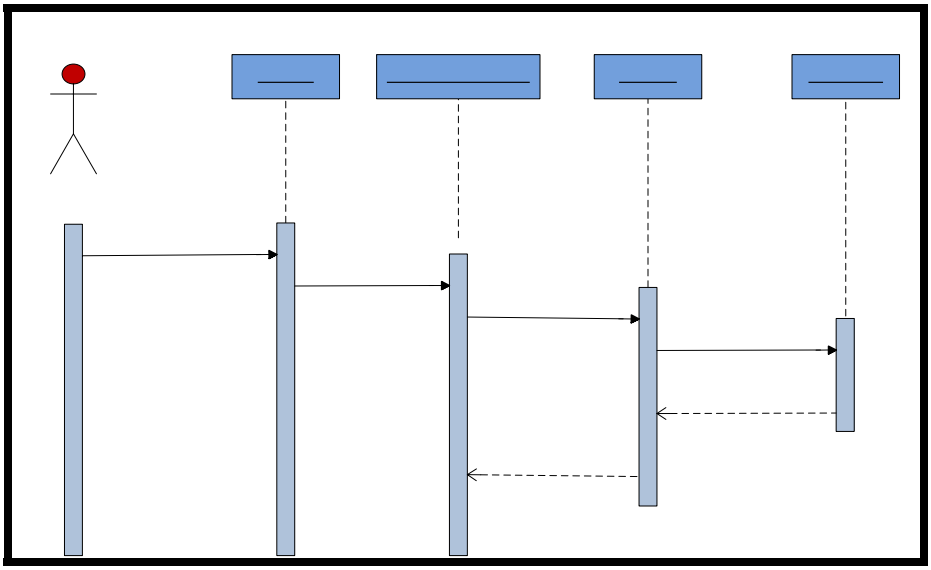

Figura 4.16 Diagrama de Secuencia “Agregar Jugador a Equipo”

Fuente: Propia

4.3.6 Diseño de la Base de Datos

Como se observó en los diferentes diagramas elaborados durante las dos fases del desarrollo del sistema, muchas de las tareas que se realizarán de acuerdo a los casos de uso tendrán acceso a la base de datos. En esta fase del proyecto se pretende el desarrollo de la estructura interna de la aplicación, por lo que se hace indispensable el diseño de la base de datos, como una de las partes más importantes de la estructura, ya que esta es la que permite almacenar los datos de manera que puedan ser manejados de una forma óptima.

El modelo de datos que se empleó sigue con todos los lineamientos y pautas del modelo relacional, donde se presentan los datos y las relaciones entre estos mediante el uso de tablas. Siendo este el elegido por la robustez que ofrece a la hora de manejar los datos y por ser el de mayor uso actualmente.

4.3.6.1 Tablas de la Base de Datos

Se presentarán a continuación las tablas diseñadas siguiendo el modelo relacional, las cuales son la base de la aplicación, resaltando en cada una el campo que represente la clave principal con sus respectivas descripciones.

4.3.6.1.1 Tabla Admin

En esta tabla se almacenan todos los nombres de usuarios y contraseñas de los administradores del sistema. (Ver Tabla 4.1)

Tabla 4.1 Tabla Admin

Fuente: Propia

Atributo	Dominio	Descripción
Login	Texto	Nombre de usuario de los administradores
Pass	Texto	Contraseña de los administradores.

4.3.6.1.2 Tabla CantEquiGrupos

En esta tabla se almacenan la cantidad de equipos que hay por cada grupo. Esta tabla únicamente se utiliza si el torneo es una liga nacional. (Ver Tabla 4.2)

Tabla 4.2 Tabla CantEquiGrupos

Fuente: Propia

Atributo	Dominio	Descripción
idTorneo	Número	Identificador del Torneo
grupoa	Número	Cantidad de equipos en el grupo a.
grupob	Número	Cantidad de equipos en el grupo b.
grupoc	Número	Cantidad de equipos en el grupo c.

4.3.6.1.3 Tabla Clasificados

En esta tabla se almacenan los equipos que clasifican a una segunda ronda de liga nacional, en conjunto con su tabla de posiciones en esa segunda ronda. (Ver Tabla 4.3)

Tabla 4.3 Tabla Clasificados

Fuente: Propia

Atributo	Dominio	Descripción
idTorneo	Número	Identificador del Torneo
idEquipo	Número	Identificador del Equipo
Grupo	Caracter	Indica al grupo al que pertenecen los equipos.
PJ	Número	Cantidad de partidos jugados en la segunda ronda.
PG	Número	Cantidad de partidos ganados en la segunda ronda.
PE	Número	Cantidad de partidos empatados en la segunda ronda.
PP	Número	Cantidad de partidos perdidos en la segunda ronda.
GF	Número	Cantidad de goles a favor en la segunda ronda.
GC	Número	Cantidad de goles en contra en la segunda ronda.
DG	Número	Diferencia de goles en la segunda ronda.
Puntos	Número	Cantidad de puntos en la segunda ronda.

4.3.6.1.4 Tabla Entrenadores

En esta tabla se almacenan los entrenadores que se ingresan en el sistema. (Ver Tabla 4.4)

Tabla 4.4 Tabla Entrenadores (1/2)

Fuente: Propia

Atributo	Dominio	Descripción
Nombre	Texto	Nombre del entrenador.
Apellido	Texto	Apellido del entrenador.
Cédula	Texto	Cédula del entrenador.
FN	Fecha	Fecha de nacimiento del entrenador.
Telefono	Texto	Número telefónico del entrenador.
Foto	Binario	Foto del entrenador.

Tabla 4.4 Tabla Entrenadores (2/2)

Fuente: Propia

Atributo	Dominio	Descripción
Especialidad	Texto	Especialidad del entrenador.
Direccion	Texto	Dirección del entrenador.
Estado	Texto	Estado donde reside el entrenador.
Ciudad	Texto	Ciudad donde reside el entrenador.
Email	Texto	Correo electrónico del entrenador.

4.3.6.1.5 Tabla EntrenadoresEquipos

En esta tabla se almacena la relación que existe cuando un entrenador es asignado a un equipo. (Ver Tabla 4.5)

Tabla 4.5 Tabla EntrenadoresEquipos

Fuente: Propia

Atributo	Dominio	Descripción
idEquipo	Número	Identificador del equipo
Cédula	Texto	Cédula del entrenador.

4.3.6.1.6 Tabla Equipos

En esta tabla se almacenan los equipos que se ingresan en el sistema. (Ver Tabla 4.6)

Tabla 4.6 Tabla Equipos (1/2)

Fuente: Propia

Atributo	Dominio	Descripción
idEquipo	Número	Identificador del equipo

Tabla 4.6 Tabla Equipos (2/2)

Fuente: Propia

Atributo	Dominio	Descripción
Nombre	Texto	Nombre del equipo.
Logo	Binario	Logotipo del equipo.
Sede	Texto	Sede del equipo.

4.3.6.1.7 Tabla EquiposParticipantes

En esta tabla se almacena la relación existente entre los torneos y sus respectivos equipos participantes. (Ver Tabla 4.7)

Tabla 4.7 Tabla Equipos

Fuente: Propia

Atributo	Dominio	Descripción
idTorneo	Número	Identificador del torneo.
idEquipo	Número	Identificador del equipo.

4.3.6.1.8 Tabla EstadisticasEquipo

En esta tabla se almacenan las estadísticas por equipo para cada partido de un torneo en específico. (Ver Tabla 4.8)

Tabla 4.8 Tabla EstadisticasEquipo (1/2)

Fuente: Propia

Atributo	Dominio	Descripción
idPartido	Número	Identificador del partido.
idTorneo	Número	Identificador del torneo.

Tabla 4.8 Tabla EstadisticasEquipo (2/2)

Fuente: Propia

Atributo	Dominio	Descripción
idEquipo	Número	Identificador del equipo.
GolesFavor	Número	Cantidad de goles a favor en el partido.
GolesContra	Número	Cantidad de goles en contra en el partido.
TA	Número	Cantidad de tarjetas amarillas en el partido.
TR	Número	Cantidad de tarjetas rojas en el partido.

4.3.6.1.9 Tabla EstadisticasJugador

En esta tabla se almacenan las estadísticas por jugador para cada partido de un torneo en específico. (Ver Tabla 4.9)

Tabla 4.9 Tabla EstadisticasJugador

Fuente: Propia

Atributo	Dominio	Descripción
idPartido	Número	Identificador del partido.
idTorneo	Número	Identificador del torneo.
idEquipo	Número	Identificador del equipo.
idJugador	Texto	Cédula del jugador.
Goles	Número	Cantidad de goles anotados en el partido.
Abridor	Texto	Indica si el jugador fue abridor en el partido.
TA	Número	Cantidad de tarjetas amarillas en el partido.
TR	Número	Cantidad de tarjetas rojas en el partido.

4.3.6.1.10 Tabla Jugadores

En esta tabla se almacenan los jugadores que se ingresan en el sistema. (Ver Tabla 4.10)

Tabla 4.10 Tabla Jugadores

Fuente: Propia

Atributo	Dominio	Descripción
Nombre	Texto	Nombre del jugador.
Apellido	Texto	Apellido del jugador.
Cédula	Texto	Cédula del jugador.
FN	Fecha	Fecha de nacimiento del jugador.
Telefono	Texto	Número telefónico del jugador.
Foto	Binario	Foto del jugador.
CopiaCedula	Binario	Fotocopia de la cédula del jugador.
Pasaporte	Texto	Número de pasaporte del jugador.
Direccion	Texto	Dirección del jugador.
Estado	Texto	Estado donde reside el jugador.
Ciudad	Texto	Ciudad donde reside el jugador.
Email	Texto	Correo electrónico del jugador.
Estatura	Número	Estatura del jugador.
Peso	Número	Peso del jugador.
Pierna	Texto	Pierna hábil del jugador
Posición	Texto	Demarcación en el campo del jugador.

4.3.6.1.11 Tabla JugadoresEquipos

En esta tabla se almacena la relación que existe cuando un jugador es asignado a un equipo. (Ver Tabla 4.11)

Tabla 4.11 Tabla JugadoresEquipos

Fuente: Propia

Atributo	Dominio	Descripción
idEquipo	Número	Identificador del equipo.
Dorsal	Número	Número de camiseta que portará el jugador.
Cédula	Texto	Cédula del jugador.

4.3.6.1.12 Tabla Partidos

En esta tabla se almacenan los partidos de los torneos en conjunto con su resultado (si ya el partido fue jugado), la fecha del partido y la jornada en el campeonato. (Ver Tabla 4.12)

Tabla 4.12 Tabla Partidos

Fuente: Propia

Atributo	Dominio	Descripción
idPartido	Número	Identificador del partido.
idTorneo	Número	Identificador del torneo.
idEquipoA	Número	Identificador del equipo de casa.
idEquipoB	Número	Identificador del equipo visitante.
Fecha	Fecha	Fecha del partido.
Resultado	Texto	Resultado del partido.
Jornada	Número	Número de jornada en el campeonato

4.3.6.1.13 Tabla PosicionesLiga

En esta tabla se almacena la tabla de posiciones en caso de que el torneo disputado sea una liguilla. (Ver Tabla 4.13)

Tabla 4.13 Tabla PosicionesLiga

Fuente: Propia

Atributo	Dominio	Descripción
idTorneo	Número	Identificador del Torneo
idEquipo	Número	Identificador del Equipo
PJ	Número	Cantidad de partidos jugados.
PG	Número	Cantidad de partidos ganados.
PE	Número	Cantidad de partidos empatados.
PP	Número	Cantidad de partidos perdidos.
GF	Número	Cantidad de goles a favor.
GC	Número	Cantidad de goles en contra.
DG	Número	Diferencia de goles.
Puntos	Número	Cantidad de puntos.

4.3.6.1.14 Tabla PosicionesLigaNacional

En esta tabla se almacena la tabla de posiciones de la primera ronda en caso de que el torneo disputado sea una liga nacional. (Ver Tabla 4.14)

Tabla 4.14 Tabla PosicionesLigaNacional (1/2)

Fuente: Propia

Atributo	Dominio	Descripción
idTorneo	Número	Identificador del Torneo
idEquipo	Número	Identificador del Equipo
Grupo	Caracter	Indica al grupo al que pertenecen los equipos.
PJ	Número	Cantidad de partidos jugados.
PG	Número	Cantidad de partidos ganados.
PE	Número	Cantidad de partidos empatados.
PP	Número	Cantidad de partidos perdidos.

Tabla 4.14 Tabla PosicionesLigaNacional (2/2)

Fuente: Propia

Atributo	Dominio	Descripción
GF	Número	Cantidad de goles a favor.
GC	Número	Cantidad de goles en contra.
DG	Número	Diferencia de goles.
Puntos	Número	Cantidad de puntos.

4.3.6.1.15 Tabla Sancionados

En esta tabla se almacenan los jugadores que se encuentren actualmente sancionados para alguno de los torneos. (Ver Tabla 4.15)

Tabla 4.15 Tabla Sancionados

Fuente: Propia

Atributo	Dominio	Descripción
idTorneo	Número	Identificador del torneo.
Cédula	Texto	Cédula del jugador.
cantJornadas	Número	Número de jornadas que faltan para cumplir la sanción.

4.3.6.1.16 Tabla PosicionesLigaNacional

En esta tabla se almacenan los datos de todos los torneos ingresados en el sistema. (Ver Tabla 4.16)

Tabla 4.16 Tabla PosicionesLigaNacional

Fuente: Propia

Atributo	Dominio	Descripción
idTorneo	Número	Identificador del Torneo
Nombre	Texto	Nombre del torneo.
Tipo	Texto	Tipo de torneo.
CantEquipos	Texto	Cantidad de equipos participantes en el torneo.
FechaInicio	Fecha	Fecha en que se inicia el torneo.
FechaFinal	Fecha	Fecha en la que finaliza el torneo.
Logo	Binario	Logotipo del torneo.
TipoCruce	Texto	Tipos de cruce entre los equipos en el torneo.
DiasJuego	Texto	Días en los que habrá partidos en el torneo.
PartidosDiarios	Número	Cantidad de partidos diarios del torneo.

4.3.7 Diseño de la Interfaz de Usuario

Otro de los requerimientos importantes para la aplicación es que ésta sea de fácil utilización e intuitiva. Para esto se requiere elaborar un diseño de interfaz gráfica, que ayude al usuario y le permite una cómoda interacción con el sistema que está utilizando. Para este fin, fue diseñada una ventana que puede incluir toda la información relevante para el usuario, pero a la vez es lo suficientemente simple y fácil de utilizar como para que cualquier usuario que esté haciendo uso de la aplicación, sea la primera vez que la utiliza o no, tenga control total y sepa a donde debe dirigirse cuando desee realizar cualquier tipo de acción.

4.3.7.1 Entrada

La ventana principal de la aplicación se presenta en la figura 4.17.

Figura 4.17. Ventana Principal

Fuente: Propia

Como se observa, la interfaz presenta un control de tabletas en el lado derecho de la misma, que permite un fácil acceso del usuario a todos los módulos principales del sistema. Al lado izquierdo del control de tabletas, se encuentra el espacio de trabajo de la aplicación, en el cual se mostrará toda la información relevante y formularios para ser llenados, conforme el usuario vaya navegando por las opciones del programa.

4.3.7.2 Torneos

Este módulo se presenta en la figura 4.18, y detalla cuando el usuario hace click en la opción de torneos en el control de tabletas ubicado a la derecha de la pantalla. En esta

interfaz se ofrece directamente la opción al usuario de crear un nuevo torneo, mediante el llenado de todos los formularios que aparecen en pantalla y luego haciendo click en el botón de almacenar. También se puede acceder a la pantalla que nos muestra todos los torneos existentes en el sistema, haciendo click en el botón Buscar Torneo.

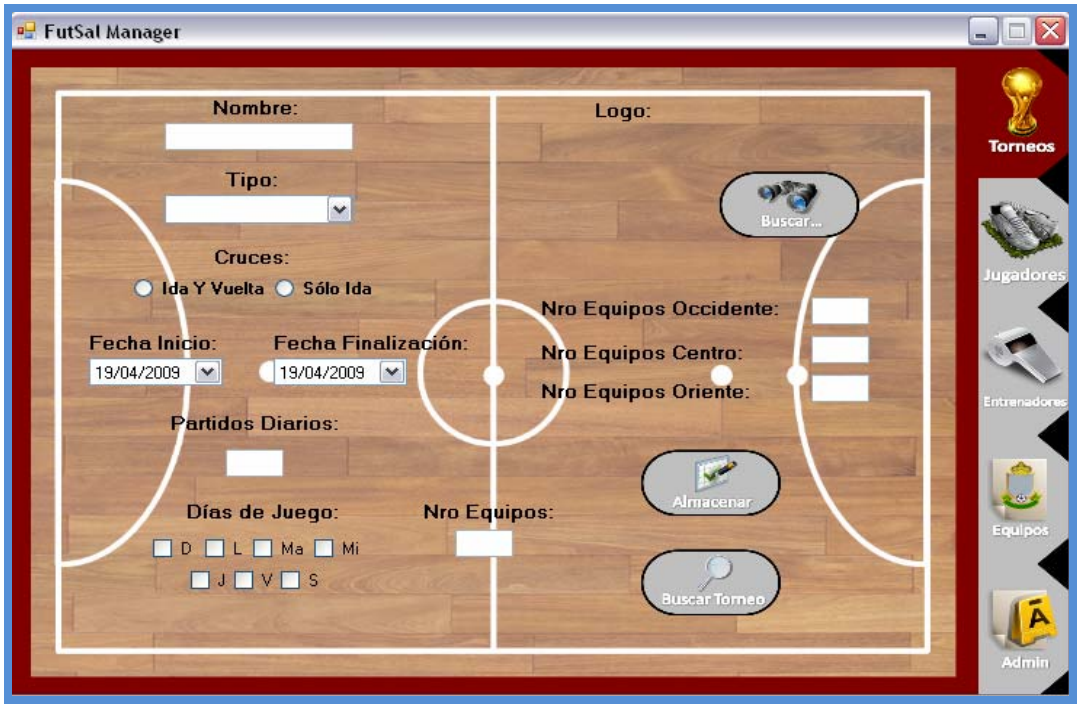

Figura 4.18 Interfaz Torneos

Fuente: Propia

4.3.7.3 Jugadores

Esta interfaz se presenta al usuario cuando este hace click en la opción de Jugadores en el control de tabletas vertical. Inmediatamente, al ser la opción más utilizada la de agregar jugadores, se muestran los formularios para el ingreso de datos de los jugadores a almacenar. En caso de que el usuario sólo desee ver los datos de un

jugador ya existente en el sistema, también está en la posibilidad, al poseer un recuadro de búsqueda por cédula en la parte inferior derecha de la ventana. La interfaz de los jugadores se presenta en la figura 4.19.

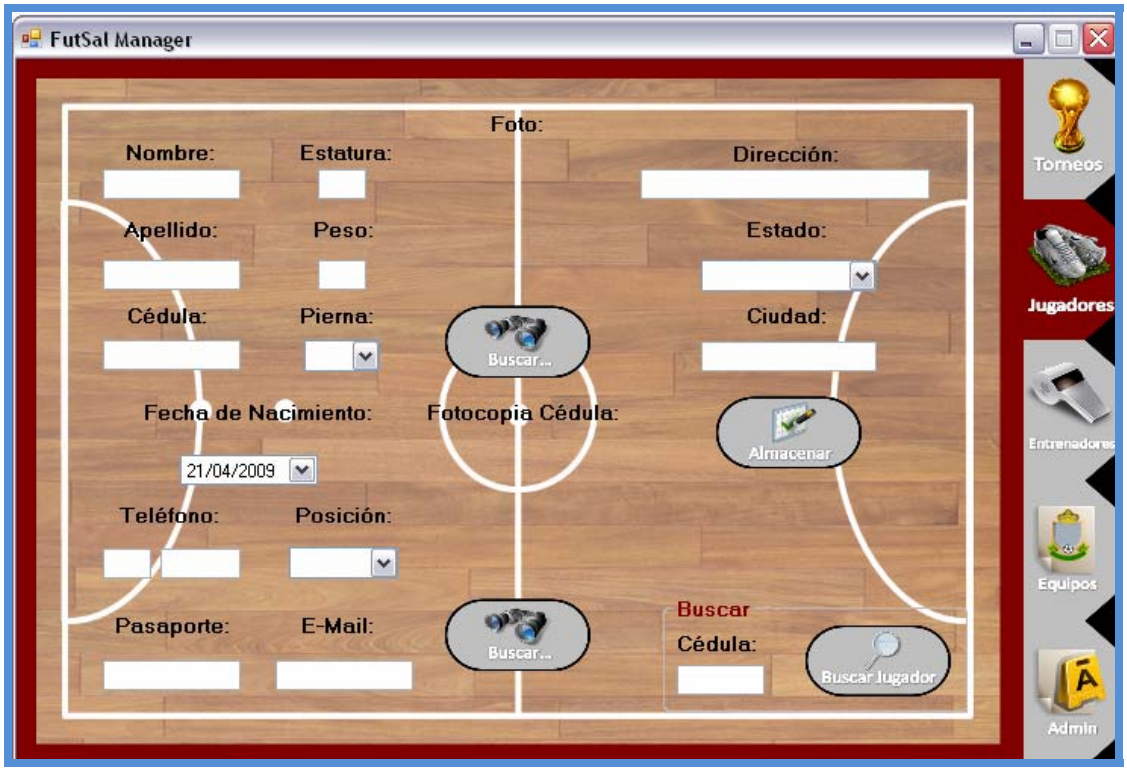

Figura 4.19 Interfaz Jugadores

Fuente: Propia

4.3.7.4 Entrenadores

La interfaz de entrenadores se presenta en la figura 4.20. Esta está compuesta de formularios que deben ser llenados para la adición de entrenadores al sistema, debido a que esta es la opción más común al momento de entrar a la tableta de los entrenadores. En caso de que el usuario desee buscar un entrenador que ya está

inscrito en el sistema, lo puede hacer mediante el recuadro de buscar que se encuentra en la parte inferior derecha de la interfaz.

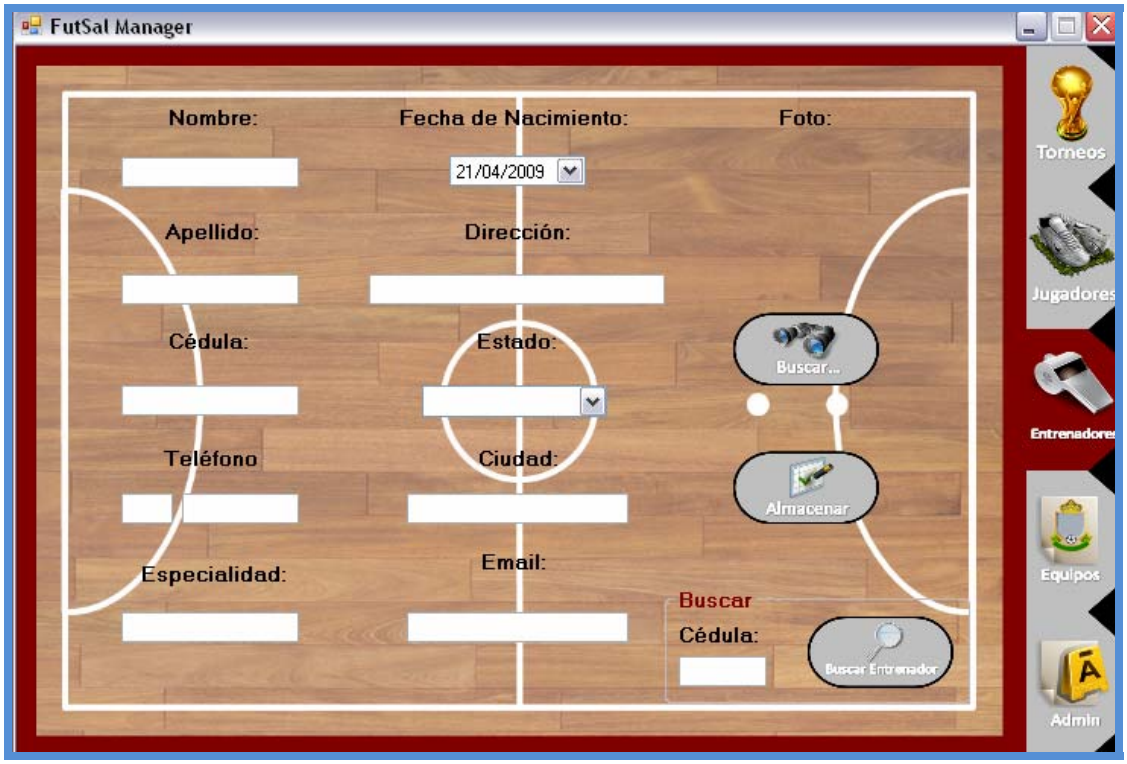

Figura 4.20 Interfaz Entrenadores

Fuente: Propia

4.3.7.5 Equipos

En esta interfaz se muestran los formularios para ingresar nuevos equipos, así como también los equipos que ya existen en el sistema, para ofrecer la posibilidad de añadirles jugadores y entrenadores a dichos equipos. La interfaz se muestra en la figura 4.21.

Figura 4.21 Interfaz Equipos

Fuente: Propia

4.3.7.6 Admin

Esta interfaz será utilizada únicamente por algún administrador del sistema, y será la que permita realizar todas las actividades propias de los administradores. La interfaz de administrador se presenta en la figura 4.22.

4.3.8 Diseño del Entorno WEB

La aplicación tendrá también un entorno WEB con el fin de que los resultados de las ligas y los torneos más importantes del país puedan ser vistos a través de la página de la comisión nacional de fútbol sala.

Figura 4.22 Interfaz Admin

Fuente: Propia

4.3.8.1 Interfaz del Entorno WEB

Este entorno web deberá ser amigable también y de fácil uso y acceso. En la figura 4.23 se presenta el formato de la página WEB.

4.4 Implementación

En este flujo de trabajo se debe empezar a codificar la arquitectura del sistema en lenguajes de programación que sean adecuados para la finalidad de cada uno de los bloques.

Figura 4.23 Formato de la página WEB

Fuente: Propia

4.4.1 Procedimientos Almacenados

Luego del diseño de las tablas debemos establecer los procedimientos almacenados para inserción, extracción y modificación de los datos en la base de datos. A

continuación se presentan las consultas fundamentales que permiten cumplir con los requisitos del sistema.

4.4.1.1 Obtener Torneos

Esta consulta retorna los torneos que se encuentren en el sistema.

Create Procedure ObTodosTorneos As Select Nombre, FechaInicio, FechaFinal From Torneos

4.4.1.2 Obtener Jugadores

Esta consulta retorna los jugadores que se encuentren en el sistema.

Create Procedure ObTodosJugadores As Select Nombre, Apellido, Cedula From Jugadores

4.4.1.3 Obtener Equipos

Esta consulta retorna los equipos que se encuentren en el sistema.

Create Procedure ObTodosEquipos As Select Nombre, Sede From Equipos

4.4.1.4 Obtener Agentes Libres

Esta consulta retorna los jugadores que no pertenezcan a ningún equipo. Se utilizará para saber que jugadores están libres para ser agregados a cualquier equipo en un momento dado.

Create Procedure ObAgentesLibres As Select Nombre, Apellido, Cedula From Jugadores Where Cedula NOT IN (Select Cedula From JugadoresEquipos)

4.4.1.5 Obtener Entrenadores Libres

Esta consulta retorna los entrenadores que no pertenezcan a ningún equipo. Se utilizará para saber que entrenadores están libres para ser agregados a cualquier equipo en un momento dado.

Create Procedure ObEntrenadoresLibres As Select Nombre, Apellido, Cedula, Especialidad From Entrenadores Where Cedula NOT IN (Select Cedula From EntrenadoresEquipos)

4.4.1.6 Obtener Plantilla

Esta consulta retorna los jugadores que pertenezcan a un equipo dado. La consulta recibirá por parámetros el identificador del equipo.
Create Procedure ObPlantilla(@idEquipo int) As Select J.Apellido, J.Nombre, J.Cedula, JE.Dorsal From Jugadores J, JugadoresEquipos JE Where @idEquipo = JE.idEquipo AND JE.Cedula = J.Cedula

4.4.1.7 Obtener Partidos de una Jornada

Esta consulta retorna los partidos que pertenezcan a la jornada especificada por parámetros a la consulta.

Create Procedure ObPartidoJornada (@idTorneo int, @Jornada int) As Select P.idTorneo, P.idPartido, E1.Nombre, P.Resultado, E2.Nombre, P.Fecha From

Partidos P, Equipos E1, Equipos E2 Where idTorneo = @idTorneo AND Jornada = @Jornada AND E1.idEquipo = P.idEquipoA AND E2.idEquipo = P.idEquipoB

4.4.1.8 Obtener Estadísticas de Partido

Esta consulta retorna las estadísticas de un partido, desde el punto de vista de un equipo, sin detallar la información de sus jugadores.

*Create Procedure ObEstadisticaPartido (@idTorneo int, @idPartido int, @idEquipo int) As Select * From EstadisticasEquipo Where @idTorneo = idTorneo AND @idPartido = idPartido AND @idEquipo = idEquipo*

4.4.1.9 Obtener Estadísticas de Jugadores en Partido

Esta consulta retorna las estadísticas de los jugadores pertenecientes a un equipo en un partido especificado.

*Create Procedure ObEstadisticaPartidoJ (@idTorneo int, @idPartido int, @idEquipo int) As Select * From EstadisticasJugador Where @idTorneo = idTorneo AND @idPartido = idPartido AND @idEquipo = idEquipo*

4.4.1.10 Obtener Tabla

Esta consulta retorna la tabla de posiciones para los torneos de tipo Liguilla.

Create Procedure ObTabla (@idTorneo int) As Select E.Nombre, PL.PJ, PL.PG, PL.PE, PL.PP, PL.GF, PL.GC, PL.DG, PL.Puntos From Equipos E, PosicionesLiga PL Where idTorneo = @idTorneo AND E.idEquipo = PL.idEquipo Order by \"Puntos\" DESC, \"DG\" DESC

4.4.1.11 Obtener Tabla Liga Nacional

Esta consulta retorna la tabla de posiciones para la primera ronda de los torneos de tipo Liga Nacional.

Create Procedure ObTablaLigaNacional (@idTorneo int, @Grupo varchar(1)) As Select E.Nombre, PL.PJ, PL.PG, PL.PE, PL.PP, PL.GF, PL.GC, PL.DG, PL.Puntos From Equipos E, PosicionesLigaNacional PL Where idTorneo = @idTorneo AND @Grupo = Grupo AND E.idEquipo = PL.idEquipo Order by \"Puntos\" DESC, \"DG\" DESC

4.4.1.12 Obtener Tabla Clasificados

Esta consulta retorna la tabla de posiciones para la segunda ronda de los torneos de tipo Liga Nacional.

Create Procedure ObTablaClas (@idTorneo int, @Grupo varchar(1)) As Select E.Nombre, PL.PJ, PL.PG, PL.PE, PL.PP, PL.GF, PL.GC, PL.DG, PL.Puntos From Equipos E, Clasificados PL Where idTorneo = @idTorneo AND @Grupo = Grupo AND E.idEquipo = PL.idEquipo Order by \"Puntos\" DESC, \"DG\" DESC

4.4.1.13 Obtener Equipos No Participantes

Esta consulta retorna los equipos que no han sido inscritos en un torneo, permitiendo así la posibilidad de inscribirlos en el mismo.

Create Procedure ObEquiposNoParticipantes (@idTorneo int) As Select Nombre From Equipos Where idEquipo NOT IN (Select idEquipo From EquiposParticipantes Where @idTorneo = idTorneo)

4.4.2 Implementación del Entorno Web

4.4.2.1 Implementación de la Interfaz Gráfica

La página web está dividida en módulos con el fin de mejorar el entendimiento de la misma así como también otorgarle una mayor portabilidad a la misma.

```
<TextBlock Name="TituloUltimos" TextAlignment="Center" Text="Últimos Resultados" Foreground="DarkRed" FontSize="18" FontFamily="Calibri"/>

<Border x:Name="BordeUltimos" BorderBrush="Black" BorderThickness="2">

  <ListBox x:Name="ListaResultados" Background="Transparent">
    <ListBox.ItemTemplate>
      <DataTemplate>
        <Grid HorizontalAlignment="Center">

          <Grid.RowDefinitions>
            <RowDefinition/>
          </Grid.RowDefinitions>

          <Grid.ColumnDefinitions>
            <ColumnDefinition MinWidth="80"/>
            <ColumnDefinition MinWidth="80"/>
            <ColumnDefinition MinWidth="80"/>
          </Grid.ColumnDefinitions>

          <TextBlock Grid.Column="0" Grid.Row="0" Text="{Binding local}" Foreground="Black" FontSize="12" VerticalAlignment="Center" TextWrapping="Wrap" TextAlignment="Left" HorizontalAlignment="Stretch"/>
          <StackPanel Grid.Column="1" Grid.Row="0" Orientation="Vertical" HorizontalAlignment="Stretch">
```

```
                        <TextBlock Text="{Binding fecha}" Foreground="DarkRed"
TextAlignment="Center"/>
                        <TextBlock Text="{Binding resultado}"
Foreground="DarkRed" TextAlignment="Center"/>
                    </StackPanel>
                    <TextBlock Grid.Column="2" Grid.Row="0" Text="{Binding
visitante}" Foreground="Black" FontSize="12" VerticalAlignment="Center"
TextWrapping="Wrap" TextAlignment="Right" HorizontalAlignment="Stretch"/>

                </Grid>
            </DataTemplate>
        </ListBox.ItemTemplate>
    </ListBox>

</Border>
```

Este fragmento de código genera un panel que contiene los últimos resultados de la liga nacional masculina. Se define una tabla en este módulo, que permita añadir los resultados en una lista y ser mostrados al usuario de una manera ordenada y comprensible. También se definen plantillas para los objetos que formarán la lista automatizando así la tarea de asignación de los objetos a ser mostrados en la lista teniendo únicamente que extraer los datos de la base de datos.

```
<UserControl x:Class="PaginaFutSal.Galeria"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:my="clr-namespace:PaginaFutSal">

  <Canvas x:Name="LayoutRoot">

    <TextBlock Name="Titulo" Text="Galería" Foreground="DarkRed"
FontSize="18" FontFamily="Calibri" Margin="5"/>

    <Border Name="BordeGaleria" BorderBrush="Black" BorderThickness="2"
Canvas.Left="5">

      <Image x:Name="BigImage" Stretch="Fill"/>

    </Border>
```

```
<Grid x:Name="GridGaleria" Visibility="Collapsed">

    <Grid.ColumnDefinitions>
      <ColumnDefinition/>
      <ColumnDefinition/>
    </Grid.ColumnDefinitions>
    <Grid.RowDefinitions>
      <RowDefinition/>
    </Grid.RowDefinitions>

    <Border Grid.Column="0" Grid.Row="0" Grid.ColumnSpan="2"
Name="BordeGaleria2" BorderBrush="Black" BorderThickness="2"
Canvas.Left="5">

      <MediaElement x:Name="BigVideo" Stretch="Fill"
MouseEnter="BigVideo_MouseEnter" MouseLeave="BigVideo_MouseLeave"/>

    </Border>

    <Button x:Name="PlayPause" Content="Pause" Margin="0,5,0,0"
Grid.Column="0" Grid.Row="0" VerticalAlignment="Top"
HorizontalAlignment="Center" Visibility="Collapsed"
MouseEnter="BigVideo_MouseEnter" MouseLeave="BigVideo_MouseLeave"
Click="PlayPause_Click"/>

    <Button x:Name="Stop" Content="Stop" Margin="0,5,0,0"
Grid.Column="1" Grid.Row="0" VerticalAlignment="Top"
HorizontalAlignment="Center" Visibility="Collapsed"
MouseEnter="BigVideo_MouseEnter" MouseLeave="BigVideo_MouseLeave"
Click="Stop_Click"/>

  </Grid>

  <Rectangle x:Name="Rectangulo" Opacity="0.8" Fill="Black"/>

  <Button x:Name="Siguiente" Content=">" Click="Siguiente_Click"/>

  <Button x:Name="Anterior" Content="&lt;" Click="Anterior_Click"/>

  <Border x:Name="Borde1">
```

```
        <Image x:Name="Image1" Stretch="Fill" Visibility="Collapsed"
MouseLeftButtonDown="Image1_MouseLeftButtonDown"/>

    </Border>

    <Border x:Name="Borde2">

        <Image x:Name="Image2" Stretch="Fill" Visibility="Collapsed"
MouseLeftButtonDown="Image2_MouseLeftButtonDown"/>

    </Border>

    <Border x:Name="Borde3">

        <Image x:Name="Image3" Stretch="Fill" Visibility="Collapsed"
MouseLeftButtonDown="Image3_MouseLeftButtonDown"/>

    </Border>

    <Border x:Name="Borde1Video">

        <MediaElement x:Name="Video1" Stretch="Fill" Visibility="Collapsed"
AutoPlay="False" MouseLeftButtonDown="Video1_MouseLeftButtonDown"/>

    </Border>

    <Border x:Name="Borde2Video">

        <MediaElement x:Name="Video2" Stretch="Fill" Visibility="Collapsed"
AutoPlay="False" MouseLeftButtonDown="Video2_MouseLeftButtonDown"/>

    </Border>

    <Border x:Name="Borde3Video">

        <MediaElement x:Name="Video3" Stretch="Fill" Visibility="Collapsed"
AutoPlay="False" MouseLeftButtonDown="Video3_MouseLeftButtonDown"/>

    </Border>

  </Canvas>

</UserControl>
```

El fragmento anteriormente presentado funciona como una galería multimedia en la cual se van desplegando fotos y videos. En la parte baja hay un carrusel de imágenes que al ser seleccionadas se muestran en la parte de arriba a un mayor tamaño, y que en el caso de ser un video, se reproducen.

```
<UserControl x:Class="PaginaFutSal.Eventos"
   xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
   xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">
   <Canvas x:Name="LayoutRoot" Background="Transparent">

   <TextBlock Name="Titulo" Text="Eventos" Grid.Column="0" Grid.Row="0"
Foreground="DarkRed" FontSize="18" FontFamily="Calibri" Margin="5"/>

   <Border Name="BordeEventos" BorderBrush="Black" BorderThickness="2"
Canvas.Left="5">

         <ScrollViewer Background="Transparent"
VerticalScrollBarVisibility="Auto" HorizontalScrollBarVisibility="Auto">

            <ListBox x:Name="ListaEventos" Background="Transparent">

               <ListBox.ItemTemplate>

                  <DataTemplate>

                     <Grid Name="GridEventos" >

                        <Grid.ColumnDefinitions>

                           <ColumnDefinition Width="50"/>
                           <ColumnDefinition Width="200"/>

                        </Grid.ColumnDefinitions>

                        <Grid.RowDefinitions>

                           <RowDefinition Height="40"/>

                        </Grid.RowDefinitions>
```

```
                    <Border Name="BordeImagen" Grid.Column="0"
Grid.Row="0" BorderBrush="DarkGray" BorderThickness="1" Margin="5">

                        <Image Name="ImagenEvento" Source="{Binding
imagen}" Stretch="Fill"/>

                    </Border>

                    <TextBlock Name="TituloEvento" Grid.Column="1"
Grid.Row="0" Text="{Binding nombre}" TextWrapping="Wrap"
Foreground="DarkRed" FontSize="12" FontFamily="Calibri"/>

                </Grid>

            </DataTemplate>

        </ListBox.ItemTemplate>

    </ListBox>

</ScrollViewer>

</Border>

</Canvas>

</UserControl>
```

Este último fragmento genera un panel que contiene los próximos eventos que tendrán lugar en el país. Se define una tabla en este módulo, que permita añadir los eventos en una lista y ser mostrados al usuario de una manera ordenada y comprensible. También se definen plantillas para los objetos que formarán la lista automatizando así la tarea de asignación de los objetos a ser mostrados en la lista teniendo únicamente que extraer los datos de la base de datos.

4.4.2.2 Implementación de los Servicios Web

Es un componente del marco de trabajo .NET de Microsoft, que añade capacidades nativas de realizar consultas a los lenguajes de la plataforma .NET. LINQ ofrece la ventaja de que manipula los datos en las tablas como una clase, de forma tal que al realizar una extracción, modificación o inserción, se pueden utilizar las clases creadas ya en el sistema sin necesidad de realizar asignaciones de atributos.

Para el manejo de servicios web que realizan consultas en una base de datos al nivel del servidor para luego llevar la información al cliente, LINQ es la mejor herramienta que existe actualmente para la plataforma .NET y por ende se le da un amplio uso en éste ámbito de la aplicación. A continuación se presentan las consultas de mayor relevancia descritas en el lenguaje LINQ.

4.4.2.2.1 Obtener Jugadores de Equipo

Esta consulta retorna todos los jugadores que forman parte de un equipo dado. Permite mostrar al usuario de la página WEB la plantilla completa de cada uno de los equipos que se encuentren en el sistema.

```
public List<Jugadore> obJugadoresEquipo(int s)
{
   PaginaWebDataContext pw = new PaginaWebDataContext();
   var tbljugadores = from tbljug in pw.Jugadores
                      from tbljugequi in pw.JugadoresEquipos
                      from tblequi in pw.Equipos
                      where tbljug.Cedula == tbljugequi.Cedula &&
tbljugequi.idEquipo == tblequi.idEquipo && tblequi.idEquipo == s
                      select tbljug;
```

```
    return tbljugadores.ToList();
}
```

4.4.2.2.2 Obtener Próximos Partidos de Torneo

Esta consulta retorna los próximos partidos a disputar de un torneo en específico, permitiendo a los internautas conocer todo acerca de los próximos encuentros.

```
public List<Partido> obProxPartidosTorneo(int s)
{
    PaginaWebDataContext pw = new PaginaWebDataContext();
    var tblultpartidos = from tblult in pw.Partidos
                         from tblequi in pw.Equipos
                         from tblequi2 in pw.Equipos
                         where tblult.idTorneo == s && tblult.Resultado == null &&
tblult.idLocal == tblequi.idEquipo && tblult.idVisitante == tblequi2.idEquipo
                         orderby tblult.Fecha ascending
                         select tblult;
    List<Partido> lp = tblultpartidos.ToList();
    List<Partido> aux = new List<Partido>();
    for (int i = 0; i < 8 && i < lp.Count; i++)
    {
        aux.Add(lp[i]);
    }
    return aux;
}
```

4.4.2.2.3 Obtener Últimos Partidos Disputados de Torneo

Muchas veces el internauta desea conocer el resultado de los últimos partidos de un torneo en específico y esta consulta tiene como fin proporcionarle esa información.

```
public List<Partido> obUltPartidosTorneo(int s)
{
    PaginaWebDataContext pw = new PaginaWebDataContext();
    var tblultpartidos = from tblult in pw.Partidos
                         from tblequi in pw.Equipos
                         from tblequi2 in pw.Equipos
                         where tblult.idTorneo == s && tblult.Resultado != null &&
tblult.idLocal == tblequi.idEquipo && tblult.idVisitante == tblequi2.idEquipo
                         orderby tblult.Fecha descending
                         select tblult;
    List<Partido> lp = tblultpartidos.ToList();
    List<Partido> aux = new List<Partido>();
    for (int i = 0; i < 8 && i < lp.Count; i++)
    {
        aux.Add(lp[i]);
    }
    return aux;
}
```

4.5 Pruebas

El objetivo principal de llevar a cabo este flujo de trabajo es asegurarse de que todos los subsistemas de todos los niveles hasta las capas específicas de la aplicación funcionen y no presenten ningún problema.

4.5.1 Partición Equivalente

Una partición equivalente es un método de prueba de caja negra que divide el dominio de entrada de un programa en clases de datos. El diseño de casos de prueba para la partición equivalente se basa en la evaluación de las clases de equivalencia.

4.5.1.1 Identificación de las Clases de Equivalencia

1. Sólo números.
2. Sólo caracteres.
3. Caracteres y números.

4.5.1.2 Grupo de Tipos de Entrada de Datos Almacenar Jugador

1. Este grupo se conforma por los textbox que sólo deben permitir números como entrada. (Estatura, peso, teléfono y cédula)
2. Este grupo está conformado por los textbox que sólo deben permitir caracteres como entrada. (Nombre, apellido, email, dirección y ciudad)
3. Este grupo se compone de los textbox que permiten entrada tanto de caracteres como números. (Pasaporte)

4.5.1.2.1 Aplicación de Casos de Prueba

La aplicación de los casos de prueba para almacenar jugador se presentan en la tabla 4.17.

Tabla 4.17. Casos de Prueba Almacenar Jugador

Fuente: Propia

Grupo	Casos De Prueba	Válida	No Válida	Clases De Equivalencia
1	553356	X		1
1	2	X		1
1	ABBdefg		X	2
1	45ggh46		X	3
2	553356		X	1
2	ABBdefg	X		2
2	45ggh46		X	3
3	553356		X	1
3	ABBdefg		X	2
3	45ggh46	X		3

4.5.1.3 Grupo de Tipos de Entrada de Datos Almacenar Entrenador

1. Este grupo se conforma por los textbox que sólo deben permitir números como entrada. (Teléfono y cédula)
2. Este grupo está conformado por los textbox que sólo deben permitir caracteres como entrada. (Nombre, apellido, email, especialidad, dirección y ciudad)

4.5.1.3.1 Aplicación de Casos de Prueba Almacenar Entrenador

La aplicación de los casos de prueba para almacenar entrenador se presentan en la tabla 4.18.

Tabla 4.18. Casos de Prueba Almacenar Entrenador

Fuente: Propia

Grupo	Casos De Prueba	Válida	No Válida	Clases De Equivalencia
1	553356	X		1
1	2	X		1
1	ABBdefg		X	2
1	45ggh46		X	3
2	553356		X	1
2	ABBdefg	X		2
2	45ggh46		X	3

4.5.1.4 Grupo de Tipos de Entrada de Datos Almacenar Torneo

1. Este grupo se conforma por los textbox que sólo deben permitir números como entrada. (Partidos diarios, Nro. Equipos, Nro. Equipos Regionales)
2. Este grupo se compone de los textbox que permiten entrada tanto de caracteres como números. (Nombre)

4.5.1.4.1 Aplicación de Casos de Prueba Almacenar Torneo

La aplicación de los casos de prueba para almacenar torneo se presentan en la tabla 4.19.

Tabla 4.19. Casos de Prueba Almacenar Torneo (1/2)

Fuente: Propia

Grupo	Casos De Prueba	Válida	No Válida	Clases De Equivalencia
1	553356		X	1

Tabla 4.19. Casos de Prueba Almacenar Torneo (2/2)

Fuente: Propia

Grupo	Casos De Prueba	Válida	No Válida	Clases De Equivalencia
1	2	X		1
1	ABBdefg		X	2
1	45ggh46		X	3
2	553356		X	1
2	ABBdefg	X		2
2	45ggh46	X		3

4.5.1.5 Grupo de Tipos de Entrada de Datos Almacenar Equipo

1. Este grupo está conformado por los textbox que sólo deben permitir caracteres como entrada. (Nombre y sede)

4.5.1.5.1 Aplicación de Casos de Prueba Almacenar Equipo

La aplicación de los casos de prueba para almacenar entrenador se presentan en la tabla 4.20.

Tabla 4.20. Casos de Prueba Almacenar Equipo

Fuente: Propia

Grupo	Casos De Prueba	Válida	No Válida	Clases De Equivalencia
1	553356		X	1
1	2		X	1
1	ABBdefg	X		2
1	45ggh46		X	3

4.5.1.6 Grupo de Tipos de Entrada de Datos Almacenar Estadísticas de Equipo

1. Este grupo está conformado por los textbox que sólo deben permitir números como entrada. (Goles, tarjetas amarillas y tarjetas rojas)

4.5.1.6.1 Aplicación de Casos de Prueba Almacenar Estadísticas de Equipo

La aplicación de los casos de prueba para almacenar entrenador se presentan en la tabla 4.21.

Tabla 4.21. Casos de Prueba Almacenar Estadísticas de Equipo

Fuente: Propia

Grupo	Casos De Prueba	Válida	No Válida	Clases De Equivalencia
1	553356		X	1
1	2	X		1
1	ABBdefg		X	2
1	45ggh46		X	3

4.5.1.7 Grupo de Tipos de Entrada de Datos Administrador

1. Este grupo está conformado por los textbox que debieran permitir entrada tanto de números como de caracteres. (Login, Password)

4.5.1.7.1 Aplicación de Casos de Prueba Almacenar Estadísticas de Equipo

La aplicación de los casos de prueba para almacenar entrenador se presentan en la tabla 4.22.

Tabla 4.22. Casos de Prueba Almacenar Estadísticas de Equipo

Fuente: Propia

Grupo	Casos De Prueba	Válida	No Válida	Clases De Equivalencia
1	553356		X	1
1	2		X	1
1	ABBdefg	X		2
1	45ggh46	X		3

4.5.2 Consistencia de Datos

Luego de hacer pruebas para la validación de datos a ser introducidos en el sistema, es necesaria la verificación de que los datos introducidos sean consistentes con los datos almacenados en la base de datos. Mediante un riguroso proceso de observación, se determinó que los procedimientos para la inserción de datos funcionan de manera adecuada debido a que los datos introducidos a nivel de la aplicación son almacenados correctamente en la base de datos.

En este apartado también se verificó la completa funcionalidad de los procedimientos almacenados detallados en el flujo de implementación obteniendo resultados totalmente satisfactorios.

4.6 Conclusión de la Fase de Elaboración

En el flujo de trabajo de requisitos, se identificaron los nuevos casos de uso y se desarrollaron sus diagramas de análisis y colaboración. Siguiendo la cronología del flujo de trabajo para la fase de elaboración, se identificaron los paquetes de análisis de los nuevos casos de uso. Seguidamente se entró en el diseño, donde se identificó la organización en capas y también se desarrolló un diagrama de clase de diseño general del sistema, donde se reflejan las diferentes relaciones impuestas entre clases, tales como las de agregación y composición.

Para complementar lo anterior, se presentó el diseño de las clases más importantes del sistema, identificando sus atributos y métodos, luego se continúo con los diagramas de secuencia para los casos de uso más relevantes, donde se mostraron las instancias de los objetos del diseño y las transmisiones de mensajes entre estos. Finalmente se elaboraron las tablas que van a conformar la base de datos del sistema, dando una breve explicación de ellas. Los objetivos de la fase fueron cumplidos, ya que se acumuló la información requerida para planificar la fase siguiente denominada fase de construcción. Al final de la fase de elaboración, se ha podido analizar y diseñar todos los requerimientos funcionales y técnicos del software, empezando por aquellos que son críticos para establecer la arquitectura.

CAPÍTULO V. FASE DE CONSTRUCCIÓN

5.1 Introducción

En la fase de construcción se hace énfasis en el desarrollo de una versión operativa del sistema mediante la finalización de la implementación y la realización de pruebas al software. También en caso de ser necesario se debe reforzar la arquitectura del sistema para que esta pueda soportar las nuevas características añadidas. Se debe construir un producto completo que esté listo para realizar la transición a la comunidad de usuarios.

En esta fase se completa la arquitectura y se llega a un punto estable del proyecto, para finalizar la codificación y construcción de los subsistemas. Estos son sometidos a rigurosas pruebas para garantizar el buen funcionamiento del sistema.

5.2 Análisis

En esta fase el análisis pasa a un segundo plano debido al nivel de avance del proyecto. Los casos de uso fueron identificados y desarrollados completamente en la fase de elaboración, por lo que ya no será necesario modificarlos ni identificar nuevos casos de uso en esta fase.

5.3 Diseño

El diseño de esta fase se concentra en la finalización de la arquitectura de la aplicación con el fin de terminar la construcción. En esta fase se pueden agregar nuevas columnas a las tablas de la base de datos de ser necesario, pero nunca debe ocurrir una gran reestructuración de las tablas, ya que esto demostraría que la

arquitectura no fue estabilizada de manera correcta y por lo tanto la fase de construcción fue iniciada de manera prematura.

Los diseños preliminares de las interfaces fueron realizados en la fase de elaboración y luego mostrados a distintos usuarios finales del sistema FutSal Manager, los cuales manifestaron su conformidad y por lo tanto no hubo necesidad de hacer una modificación al diseño.

En esta fase sólo se deben terminar las otras interfaces de la aplicación no expuestas en el capítulo anterior así como también la culminación del entorno WEB.

5.3.1 Diagrama de Clases

Este diagrama muestra la estructura estática del sistema reflejando las relaciones entre las clases, lo que permitirá visualizar lo que el sistema puede hacer y de cómo puede ser construido. En este diagrama se añadirán las clases que no forman parte de la arquitectura del sistema que no se incluyeron en el capítulo pasado.

5.3.1.1 Diagrama de Clases del Sistema

El diagrama representado en la figura 5.1 representa tanto a las clases de la capa específica como también a las clases de la capa general de la aplicación.

5.3.1.1.1 Clase GenerarPDF

Esta clase se encarga de manejar todas las funciones o métodos que se utilizan para la generación de los reportes de los torneos en el sistema. Ella extrae la información de la base de datos y la ubica en hojas tamaño carta en un documento de acrobat. Para

cada jornada se genera un reporte diferente que contiene todas las estadísticas de la misma.

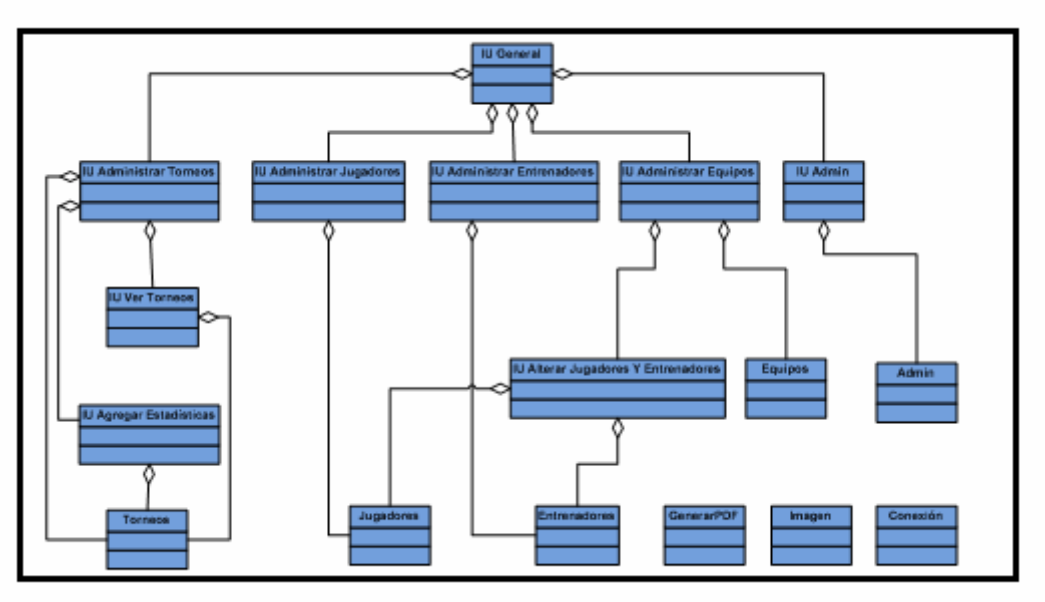

Figura 5.1. Diagrama de Clases del Sistema General

Fuente: Propia

5.3.1.1.2 Clase Imagen

Esta clase se utiliza para gestionar todas las imágenes del sistema. Posee métodos para llevar de un archivo de imagen a una matriz de bytes y viceversa. Esto permite el almacenamiento de imágenes en la base de datos, y su posterior extracción

5.3.1.1.3 Clase Conexión

Esta clase gestiona y administra las diferentes conexiones con la base de datos del sistema. Indica de qué manera se van a conectar los métodos a la base de datos.

5.3.2 Diagrama de Secuencia

Debido a su gran importancia se decidió incluir el diagrama de secuencia para el caso de uso Generar Reporte de Torneo.

5.3.2.1 Diagrama de Secuencia “Generar Reporte de Torneo”

En la figura 5.2 se detalla la secuencia de pasos que sigue la aplicación para la generación de reportes de los torneos. El usuario debe hacer log in, para luego seleccionar el torneo al que desea generar un reporte. Luego de esto debe seleccionar la opción y el programa automáticamente buscará los datos para la jornada actual, las estadísticas de todos los partidos por equipo y por jugadores y las plasmará en un documento PDF.

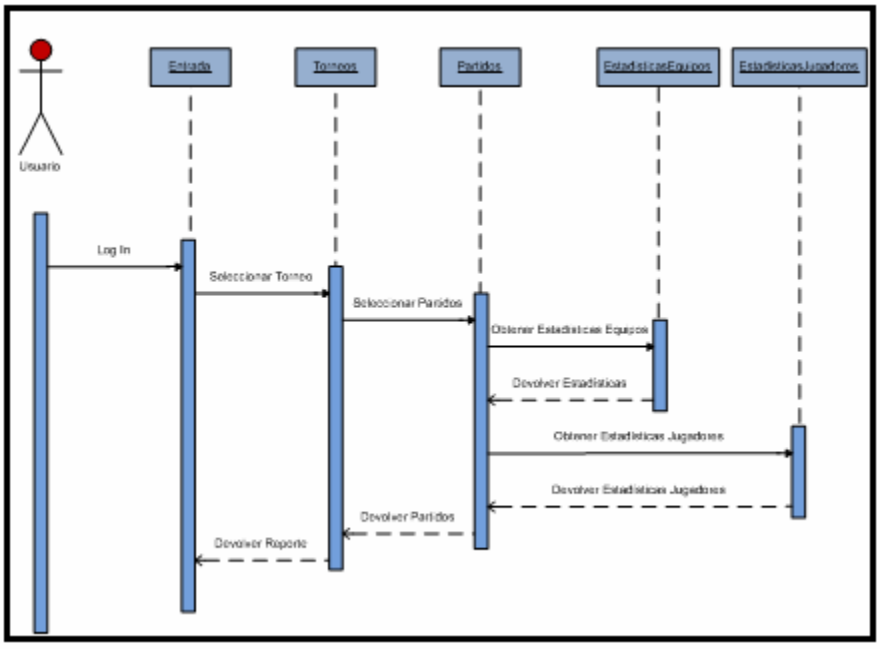

Figura 5.2 Diagrama de Secuencia “Generar Reporte de Torneo”

Fuente: Propia

5.3.2 Diseño de la Interfaz de Usuario

En este apartado, se culmina con el diseño de las interfaces de usuario, asignándole los eventos a éstas para su interacción con las personas que utilicen la aplicación. Se utilizará el mismo diseño empleado en el capítulo anterior debido a su buena acogida y fácil utilización por los usuarios.

5.3.2.1 Ver Torneos

La interfaz mostrada en la figura 5.3 permite ver los torneos que están actualmente en curso, permitiendo así, añadir estadísticas a los partidos, generar reportes de las jornadas, añadir equipos participantes a un torneo e inclusive dar inicio a fases del torneo.

Figura 5.3 Interfaz Ver Torneos

Fuente: Propia

5.3.2.2 Ver Partidos Disponibles de Torneo

En esta interfaz se muestran los partidos que no se han jugado, permitiéndole al usuario proceder a una nueva interfaz para cambiar la fecha de dicho partido o añadirle los resultados y estadísticas. Esta interfaz se puede observar en la figura 5.4.

Figura 5.4 Ver Partidos

Fuente: Propia

5.3.2.3 Agregar Estadísticas de Equipos en Partido

La interfaz mostrada en la figura 5.5 permite la adición de estadísticas al sistema referentes a un partido en específico, mediante el llenado de los formularios que se presentan en pantalla.

Figura 5.5 Agregar Estadísticas de Equipos en Partidos

Fuente: Propia

5.3.2.4 Agregar Estadísticas de Jugadores en Partido

La interfaz presentada en la figura 5.6 permite añadir estadísticas específicas para cada jugador en los partidos que su equipo jugó mediante el llenado de los formularios en pantalla.

Figura 5.6 Agregar Estadísticas de Jugadores en Partido

Fuente: Propia

5.3.3 Diseño del Entorno WEB

El diseño preliminar del entorno WEB presentado en el capítulo anterior fue presentado a diferentes personas con el fin de obtener una retroalimentación de los usuarios finales de la página. La retroalimentación de dichos usuarios fue buena, el diseño tuvo una buena acogida y por lo tanto se decidió proseguir con el mismo formato para el resto de las páginas que conforman el sistema.

5.3.3.1 Agenda

La agenda permite visualizar todos los eventos que se encuentran pautados a nivel nacional. En ella el usuario selecciona el mes y el año, y a continuación se muestra en el navegador el calendario de dicho mes y año, con los respectivos eventos resaltados

en azul, donde el usuario puede hacer click para observar información más detallada de dicho evento. La página de la agenda se muestra en la figura 5.7.

Figura 5.7 Agenda

Fuente: Propia

5.3.3.2 Selecciones

En esta página (Figura 5.8), se muestran las selecciones nacionales de fútbol sala, en conjunto con sus jugadores, sus últimos resultados, su cuerpo técnico, y sus próximos partidos a disputar.

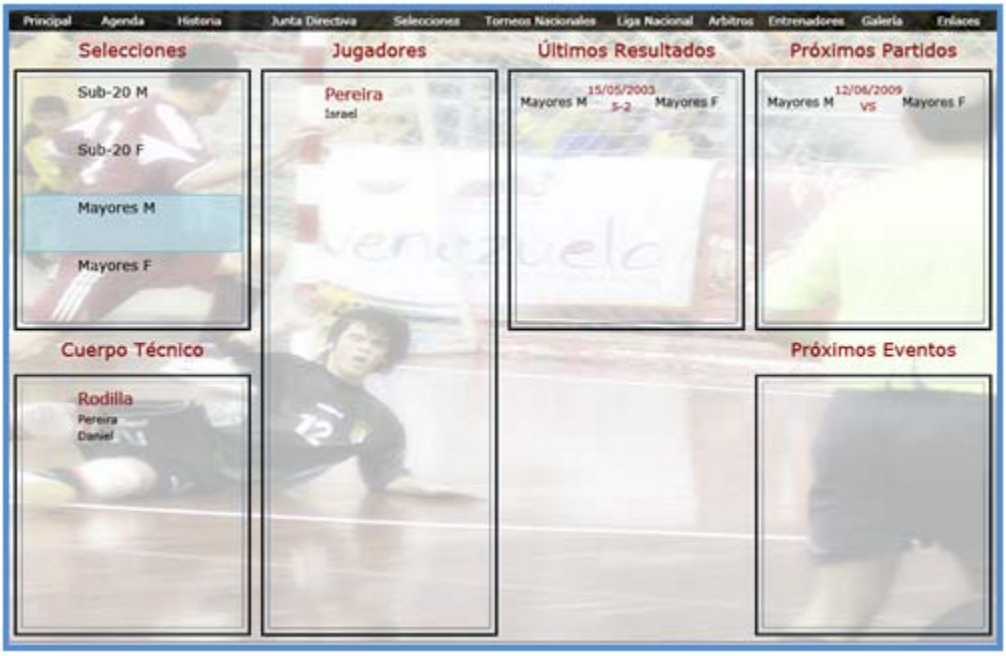

Figura 5.8. Selecciones

Fuente: Propia

5.4 Implementación

En este flujo se realiza la culminación de la implementación de la arquitectura del sistema, en conjunto con la codificación del resto de los componentes de la aplicación, con el fin de llegar a una versión del software que pueda ser transmitida a la comunidad de usuarios. Aquí se debe finalizar la construcción del software para que este cumpla con todos los requisitos estipulados anteriormente.

5.4.1 Procedimientos Almacenados

En este apartado se presentarán los procedimientos almacenados que no forman parte de la arquitectura principal de la aplicación debido a su menor relevancia en comparación con los expuestos en el capítulo anterior. Una vez realizadas, las

consultas fueron probadas con distintos casos de prueba obteniendo resultados satisfactorios.

5.4.1.1 Obtener Equipo

Esta consulta retorna un equipo que está almacenado y recibe por parámetros el identificador de dicho equipo.

*Create Procedure ObEquipoPorId (@idEquipo int) As Select * From Equipos E Where @idEquipo = E.idEquipo*

5.4.1.2 Obtener Plantilla No Sancionada

Este procedimiento nos permite obtener la plantilla de un equipo compuesta por los jugadores que no están sancionados para la próxima jornada.

Create Procedure ObPlantillaNoSancionada(@idEquipo int, @idTorneo int) As Select J.Apellido, J.Nombre, J.Cedula, JE.Dorsal From Jugadores J, JugadoresEquipos JE Where @idEquipo = JE.idEquipo AND JE.Cedula = J.Cedula AND J.Cedula NOT IN (Select Cedula From Sancionados Where @idTorneo = idTorneo)

5.4.1.3 Obtener Partidos Jugados

A partir de esta consulta se obtienen todos los partidos que se han jugado en un torneo en específico.

Create Procedure ObPartidosJugados (@idTorneo int) As Select P.idTorneo, P.idPartido, E1.Nombre, P.Resultado, E2.Nombre, P.Fecha, P.Jornada From

Partidos P, Equipos E1, Equipos E2 Where Resultado IS NOT NULL AND idTorneo = @idTorneo AND E1.idEquipo = P.idEquipoA AND E2.idEquipo = P.idEquipoB

5.4.1.4 Obtener Partidos No Jugados

Este procedimiento almacenado nos permite conocer todos los partidos ya programados para un torneo, que no han sido jugados aún.

Create Procedure ObPartidosNoJugados (@idTorneo int)As Select P.idTorneo, P.idPartido, E1.Nombre, E2.Nombre, P.Fecha, P.Jornada From Partidos P, Equipos E1, Equipos E2 Where idTorneo = @idTorneo AND Resultado IS NULL AND E1.idEquipo = P.idEquipoA AND E2.idEquipo = P.idEquipoB

5.4.1.5 Obtener Sancionados

Esta consulta retorna los jugadores que se encuentran sancionados actualmente para algún torneo en específico.

*Create Procedure ObSancionados (@idTorneo int) As Select * From Sancionados Where @idTorneo = idTorneo*

5.4.2 Implementación de la Interfaz de Usuario

En este renglón se especifica la implementación a código en el lenguaje de programación C# de la interfaz de usuario para el sistema FutSal Manager.

5.4.2.1 Ver Partidos Disponibles de Torneo

El código de esta interfaz se muestra a continuación:

```
public partial class VerPartidos : Form
  {

    MenuPrincipal X; //Atributo donde se guarda una referencia de la interfaz principal
    Tesis.Clases.Torneo T; // El torneo actual, del que mostraremos los partidos

    DataSet data = null;

    public VerPartidos(MenuPrincipal X, Tesis.Clases.Torneo T)
    {
      InitializeComponent(); //Método que inicializa los componentes de la interfaz gráfica.
      this.X = X;
      this.T = T;
      X.Visible = false;

      data = this.T.obPartidosNoJugados(); //Extraemos los partidos no jugados de la BDD
      data.Tables[0].Columns.Remove("idTorneo");
      data.Tables[0].Columns.Remove("idPartido");
      dataGridViewPartidosDisp.DataSource = data.Tables[0]; //Colocamos los datos en un datagridview
    }

    private void pictureBoxAddResults_MouseEnter(object sender, EventArgs e)
    {
      pictureBoxAddResults.Image = Tesis.Properties.Resources.AgregarResultadosFocused;
```

```
        }

        private void pictureBoxAddResults_MouseLeave(object sender, EventArgs e)
        {
            pictureBoxAddResults.Image = Tesis.Properties.Resources.AgregarResultadosUnfocused;
        }

// Método que se ejecuta al hacer click al botón de añadir resultados, que nos lleva a otra interfaz gráfica.
        private void pictureBoxAddResults_Click(object sender, EventArgs e)
        {
            if (dataGridViewPartidosDisp.SelectedCells.Count != 0)
            {
                int x = dataGridViewPartidosDisp.SelectedCells[0].RowIndex;
                Tesis.Clases.Equipo E1 = Tesis.Clases.Equipo.getEquipo(data.Tables[0].Rows[x]["Nombre"].ToString());
                Tesis.Clases.Equipo E2 = Tesis.Clases.Equipo.getEquipo(data.Tables[0].Rows[x]["Nombre1"].ToString());
                int idPartido = Int32.Parse(data.Tables[0].Rows[x]["idPartido"].ToString());
                AgregarEstadisticasP AEP = new AgregarEstadisticasP(this, idPartido, E1, E2, T);
                AEP.Visible = true;
            }
        }
// Método que actualiza el datagridview en caso de cambios al dataset
        public void actDatagrid()
        {
```

```
            data = this.T.obPartidosNoJugados();
            data.Tables[0].Columns.Remove("idTorneo");
            data.Tables[0].Columns.Remove("idPartido");
            dataGridViewPartidosDisp.DataSource = data.Tables[0];
        }
// Método que se ejecuta al hacer click en el botón para cambiar fecha, que nos lleva a otra interfaz.
        private void pictureBox1_Click(object sender, EventArgs e)
        {
            int x = dataGridViewPartidosDisp.SelectedCells[0].RowIndex;
            SelecFecha SF = new SelecFecha(T, Int32.Parse(data.Tables[0].Rows[x]["idPartido"].ToString()), this);
            SF.Visible = true;
        }

        private void VerPartidos_FormClosed(object sender, FormClosedEventArgs e)
        {
            X.Visible = true;
        }

        private void pictureBox1_MouseEnter(object sender, EventArgs e)
        {
            pictureBox1.Image = Tesis.Properties.Resources.CambiarFechaFocused;
        }

        private void pictureBox1_MouseLeave(object sender, EventArgs e)
        {
            pictureBox1.Image = Tesis.Properties.Resources.CambiarFechaUnfocused;
        }
```

5.4.2.2 Agregar Estadísticas de Equipos en Partido

El código de esta interfaz gráfica se presenta a continuación:

```
public partial class AgregarEstadisticasP : Form
{

    int idPartido;
    Tesis.Clases.Torneo T; //Torneo Actual
    Tesis.Clases.Equipo A; // Equipo de Casa
    Tesis.Clases.Equipo B; // Equipo Visitante
    Form F; // Interfaz de la que esta deriva

    public AgregarEstadisticasP(Form F, int idPartido, Tesis.Clases.Equipo A,
Tesis.Clases.Equipo B, Tesis.Clases.Torneo T)
    {
        InitializeComponent();
        this.A = A;
        this.B = B;
        this.T = T;
        this.F = F;
        this.F.Visible = false;
        this.idPartido = idPartido;
        pictureBox1.Image = A.getLogo();
        pictureBox2.Image = B.getLogo();
        label1.Text = A.getNombre();
        label6.Text = B.getNombre();
    }
```

```
        private void AgregarEstadisticasP_FormClosed(object sender, FormClosedEventArgs e)
        {
            this.F.Visible = true;
        }

        private void pictureBox3_MouseEnter(object sender, EventArgs e)
        {
            pictureBox3.Image = Tesis.Properties.Resources.AddResultsFocused;
        }

        private void pictureBox3_MouseLeave(object sender, EventArgs e)
        {
            pictureBox3.Image = Tesis.Properties.Resources.AddResultsUnfocused;
        }

//Método que toma los datos incluidos en los formularios y los almacena en la base de datos, y nos lleva a la pantalla de especificación de estadísticas por jugador
        private void pictureBox3_Click(object sender, EventArgs e)
        {
            T.AgregarStats(idPartido, A, B, Int32.Parse(textBox1.Text), Int32.Parse(textBox5.Text), Int32.Parse(textBox3.Text), Int32.Parse(textBox7.Text), Int32.Parse(textBox4.Text), Int32.Parse(textBox8.Text));
            T.AgregarResultado(idPartido, textBox1.Text, textBox5.Text);
            int pga = 0, pea = 0, ppa = 0, puntosa = 0;
            int pgb = 0, peb = 0, ppb = 0, puntosb = 0;
            if (Int32.Parse(textBox1.Text) > Int32.Parse(textBox5.Text))
            {
                pga = 1;
```

```
            puntosa = 3;
            ppb = 1;
        }
        else if (Int32.Parse(textBox1.Text) < Int32.Parse(textBox5.Text))
        {
            ppa = 1;
            puntosb = 3;
            pgb = 1;
        }
        else
        {
            pea = 1;
            peb = 1;
            puntosa = 1;
            puntosb = 1;
        }
        DataSet data = T.obTablaClas("A");
        if (data.Tables[0].Rows.Count == 0)
        {
            T.agregarResulLiga(A.getId(), pga, pea, ppa, Int32.Parse(textBox1.Text),
Int32.Parse(textBox5.Text), puntosa);
            T.agregarResulLiga(B.getId(), pgb, peb, ppb, Int32.Parse(textBox5.Text),
Int32.Parse(textBox1.Text), puntosb);
        }
        else if (data.Tables[0].Rows.Count != 0)
        {
            bool b = false;
            DataSet data2 = T.obTablaClas("A");
            for (int i = 0; i < data2.Tables[0].Rows.Count; i++)
```

```
            {
                if (Int32.Parse(data2.Tables[0].Rows[i]["PJ"].ToString()) <
data2.Tables[0].Rows.Count - 1)
                {
                    b = true;
                }
            }
            data2 = T.obTablaClas("B");
            for (int i = 0; i < data2.Tables[0].Rows.Count; i++)
            {
                if (Int32.Parse(data2.Tables[0].Rows[i]["PJ"].ToString()) <
data2.Tables[0].Rows.Count - 1)
                {
                    b = true;
                }
            }
            if (b)
            {
                T.agregarResulClas(A.getId(), pga, pea, ppa,
Int32.Parse(textBox1.Text), Int32.Parse(textBox5.Text), puntosa);
                T.agregarResulClas(B.getId(), pgb, peb, ppb,
Int32.Parse(textBox5.Text), Int32.Parse(textBox1.Text), puntosb);
            }
        }
        this.Visible = false;
        Tesis.Forms.AgregarEstadisticasJ AEJ = new
Tesis.Forms.AgregarEstadisticasJ(this.F, this.idPartido, this.A, this.B, this.T,
Int32.Parse(textBox1.Text), Int32.Parse(textBox5.Text), Int32.Parse(textBox3.Text),
Int32.Parse(textBox7.Text), Int32.Parse(textBox4.Text), Int32.Parse(textBox8.Text));
```

```
            AEJ.Visible = true;
        }
    }
```

5.4.2.3 Agregar Estadísticas de Jugadores en Partido

El código de esta interfaz se muestra a continuación:

```
    public partial class AgregarEstadisticasJ : Form
    {

        Form F; //Interfaz de la que esta deriva
        int idPartido;
        Tesis.Clases.Equipo A; //Equipo de casa
        Tesis.Clases.Equipo B; //Equipo Visitante
        Tesis.Clases.Torneo T; //Torneo Actual
        List<string> goles = new List<string>();
        List<string> ta = new List<string>();
        List<string> tr = new List<string>();
        List<string> plantilla = new List<string>();
        List<string> abridores = new List<string>();
        int GolesA, GolesB, TAA, TAB, TRA, TRB;
        bool b = false;

        public AgregarEstadisticasJ(Form F, int idPartido, Tesis.Clases.Equipo A,
Tesis.Clases.Equipo B, Tesis.Clases.Torneo T, int GolesA, int GolesB, int TAA, int
TAB, int TRA, int TRB)
        {
            InitializeComponent();
```

```
            this.F = F;
            this.F.Visible = false;
            this.idPartido = idPartido;
            this.A = A;
            this.B = B;
            this.T = T;
            this.GolesA = GolesA;
            this.GolesB = GolesB;
            this.TAA = TAA;
            this.TAB = TAB;
            this.TRA = TRA;
            this.TRB = TRB;
            this.loadData();
        }

    //Método que carga los diferentes listbox presentados en la interfaz
        public void loadData()
        {
            listBox1.DataSource = null;
            listBox2.DataSource = null;
            listBox3.DataSource = null;
            listBox4.DataSource = null;
            listBox5.DataSource = null;
            plantilla = new List<string>();
            abridores = new List<string>();
            goles = new List<string>();
            ta = new List<string>();
            tr = new List<string>();
            if (!b)
```

```
        {
            pictureBox1.Image = A.getLogo();
            label1.Text = A.getNombre();
            DataSet D = A.getPlantillaNoSancionada(T.getId());
            for (int i = 0; i < D.Tables[0].Rows.Count; i++)
            {
                plantilla.Add(D.Tables[0].Rows[i]["Nombre"].ToString() + " " +
D.Tables[0].Rows[i]["Apellido"].ToString());
            }
            listBox1.DataSource = plantilla;
        }
        else
        {
            pictureBox1.Image = B.getLogo();
            label1.Text = B.getNombre();
            DataSet D = B.getPlantillaNoSancionada(T.getId());
            for (int i = 0; i < D.Tables[0].Rows.Count; i++)
            {
                plantilla.Add(D.Tables[0].Rows[i]["Nombre"].ToString() + " " +
D.Tables[0].Rows[i]["Apellido"].ToString());
            }
            listBox1.DataSource = plantilla;
        }
        listBox1.SelectedIndex = -1;
    }

    private void AgregarEstadisticasJ_FormClosed(object sender,
FormClosedEventArgs e)
    {
```

```
            ((VerTodos)(this.F)).loadData();
            this.F.Visible = true;
        }
    //Método para añadir un goleador
        private void pictureBox2_Click(object sender, EventArgs e)
        {
            if (!b)
            {
                if (goles.Count < GolesA)
                {
                    goles.Add(listBox1.SelectedValue.ToString());
                    listBox3.DataSource = null;
                    listBox3.DataSource = goles;
                }
            }
            else
            {
                if (goles.Count < GolesB)
                {
                    goles.Add(listBox1.SelectedValue.ToString());
                    listBox3.DataSource = null;
                    listBox3.DataSource = goles;
                }
            }
            listBox1.SelectedIndex = -1;
            listBox3.SelectedIndex = -1;
        }
    // Método para añadir un jugador que recibió tarjeta amarilla
        private void pictureBox3_Click(object sender, EventArgs e)
```

```
        {
            if (!b)
            {
                if (ta.Count < this.TAA)
                {
                    ta.Add(listBox1.SelectedValue.ToString());
                    listBox4.DataSource = null;
                    listBox4.DataSource = ta;
                }
            }
            else
            {
                if (ta.Count < this.TAB)
                {
                    ta.Add(listBox1.SelectedValue.ToString());
                    listBox4.DataSource = null;
                    listBox4.DataSource = ta;
                }
            }
            listBox1.SelectedIndex = -1;
            listBox4.SelectedIndex = -1;
        }
    // Método para añadir un abridor
        private void pictureBox4_Click(object sender, EventArgs e)
        {
            if (abridores.Count < 5)
            {
                if (!abridores.Contains(listBox1.SelectedValue.ToString()))
                {
```

```
                abridores.Add(listBox1.SelectedValue.ToString());
                listBox2.DataSource = null;
                listBox2.DataSource = abridores;
            }
        }
        listBox2.SelectedIndex = -1;
    }
// Método para añadir un jugador que recibió tarjeta roja
    private void pictureBox5_Click(object sender, EventArgs e)
    {
        if (!b)
        {
            if (tr.Count < this.TRA)
            {
                tr.Add(listBox1.SelectedValue.ToString());
                listBox5.DataSource = null;
                listBox5.DataSource = tr;
            }
        }
        else
        {
            if (tr.Count < this.TRB)
            {
                tr.Add(listBox1.SelectedValue.ToString());
                listBox5.DataSource = null;
                listBox5.DataSource = tr;
            }
        }
        listBox1.SelectedIndex = -1;
```

```
            listBox5.SelectedIndex = -1;
        }
// Método para eliminar un goleador
        private void pictureBox6_Click(object sender, EventArgs e)
        {
            string x = listBox3.SelectedValue.ToString();
            goles.Remove(x);
            listBox3.DataSource = null;
            listBox3.DataSource = goles;
        }
// Método para eliminar un jugador que recibió tarjeta amarilla

        private void pictureBox7_Click(object sender, EventArgs e)
        {
            string x = listBox4.SelectedValue.ToString();
            ta.Remove(x);
            listBox4.DataSource = null;
            listBox4.DataSource = ta;
        }
// Método para eliminar un jugador que recibió tarjeta roja

        private void pictureBox8_Click(object sender, EventArgs e)
        {
            string x = listBox5.SelectedValue.ToString();
            tr.Remove(x);
            listBox5.DataSource = null;
            listBox5.DataSource = tr;
        }
// Método para eliminar a un abridor
```

```
        private void pictureBox9_Click(object sender, EventArgs e)
        {
            string x = listBox2.SelectedValue.ToString();
            abridores.Remove(x);
            listBox2.DataSource = null;
            listBox2.DataSource = abridores;
        }
// Método que almacena todas las estadísticas recogidas en la pantalla en la base de datos del sistema
        private void pictureBox10_Click(object sender, EventArgs e)
        {
            if (!b)
            {
                DataSet D = A.getPlantillaNoSancionada(T.getId());
                for (int i = 0; i < D.Tables[0].Rows.Count; i++)
                {
                    int tara;
                    int tarr;
                    int goal = 0;
                    string abr;
                    if (abridores.Contains(D.Tables[0].Rows[i]["Nombre"].ToString() + " " + D.Tables[0].Rows[i]["Apellido"].ToString()))
                    {
                        abr = "Si";
                    }
                    else
                    {
                        abr = "No";
                    }
```

```
            if (ta.Contains(D.Tables[0].Rows[i]["Nombre"].ToString() + " " +
D.Tables[0].Rows[i]["Apellido"].ToString()))
            {
               tara = 1;
            }
            else
            {
               tara = 0;
            }
            if (tr.Contains(D.Tables[0].Rows[i]["Nombre"].ToString() + " " +
D.Tables[0].Rows[i]["Apellido"].ToString()))
            {
               tarr = 1;
            }
            else
            {
               tarr = 0;
            }
            if (goles.Contains(D.Tables[0].Rows[i]["Nombre"].ToString() + " " +
D.Tables[0].Rows[i]["Apellido"].ToString()))
            {
               string[] s = goles.ToArray();
               for (int j = 0; j < s.Length; j++)
               {
                  if (s[j].Equals(D.Tables[0].Rows[i]["Nombre"].ToString() + " " +
D.Tables[0].Rows[i]["Apellido"].ToString()))
                  {
                     goal++;
                  }
```

```
                }
            }
            else
            {
                goal = 0;
            }
            Jugador J = Jugador.getJugador(D.Tables[0].Rows[i]["Cedula"].ToString());
            J.agregarEstadisticas(this.idPartido, this.T.getId(), A.getId(), goal, tara, tarr, abr);
            if (tarr == 1)
            {
                DataSet data = J.obEstadisticas();
                int cantam = 0;
                for (int j = 0; j < data.Tables[0].Rows.Count; j++)
                {
                    if (Int32.Parse(data.Tables[0].Rows[i]["TA"].ToString()) > 0)
                    {
                        cantam++;
                    }
                }
                if (cantam % 2 == 0)
                {
                    T.addSancionado(J.getCedula(), 2);
                }
            }
        }
    }
    else
```

```
        {
            DataSet D = B.getPlantillaNoSancionada(T.getId());
            for (int i = 0; i < D.Tables[0].Rows.Count; i++)
            {
                int tara;
                int tarr;
                int goal = 0;
                string abr;
                if (abridores.Contains(D.Tables[0].Rows[i]["Nombre"].ToString() + " "
+ D.Tables[0].Rows[i]["Apellido"].ToString()))
                {
                    abr = "Si";
                }
                else
                {
                    abr = "No";
                }
                if  (ta.Contains(D.Tables[0].Rows[i]["Nombre"].ToString()  +  "  "  +
D.Tables[0].Rows[i]["Apellido"].ToString()))
                {
                    tara = 1;
                }
                else
                {
                    tara = 0;
                }
                if  (tr.Contains(D.Tables[0].Rows[i]["Nombre"].ToString()  +  "  "  +
D.Tables[0].Rows[i]["Apellido"].ToString()))
                {
```

```
                tarr = 1;
            }
            else
            {
                tarr = 0;
            }
            if (goles.Contains(D.Tables[0].Rows[i]["Nombre"].ToString() + " " +
D.Tables[0].Rows[i]["Apellido"].ToString()))
            {
                string[] s = goles.ToArray();
                for (int j = 0; j < s.Length; j++)
                {
                    if (s[j].Equals(D.Tables[0].Rows[i]["Nombre"].ToString() + " " +
D.Tables[0].Rows[i]["Apellido"].ToString()))
                    {
                        goal++;
                    }
                }
            }
            else
            {
                goal = 0;
            }
            Jugador J = Jugador.getJugador(D.Tables[0].Rows[i]["Cedula"].ToString());
            J.agregarEstadisticas(this.idPartido, this.T.getId(), B.getId(), goal, tara,
tarr, abr);
            if (tarr == 1)
            {
```

```
                DataSet data = J.obEstadisticas();
                int cantam = 0;
                for (int j = 0; j < data.Tables[0].Rows.Count; j++)
                {
                    if (Int32.Parse(data.Tables[0].Rows[i]["TA"].ToString()) > 0)
                    {
                        cantam++;
                    }
                }
                if (cantam % 2 == 0)
                {
                    T.addSancionado(J.getCedula(), 2);
                }
            }
        }
        DataSet d1 = T.obPartidosJugados();
        int x = Int32.Parse(d1.Tables[0].Rows[d1.Tables[0].Rows.Count -
1]["Jornada"].ToString());
        DataSet d2 = T.obPartidosNoJugados();
        if (d2.Tables[0].Rows.Count > 0)
        {
            int y = Int32.Parse(d2.Tables[0].Rows[0]["Jornada"].ToString());
            if (y > x)
            {
                T.alterarSancionado();
                T.delSancionados();
                Tesis.Clases.GenerarPDF GP = new
Tesis.Clases.GenerarPDF(this.T);
            }
```

```
            }
            else
            {
                T.alterarSancionado();
                T.delSancionados();
                Tesis.Clases.GenerarPDF GP = new Tesis.Clases.GenerarPDF(this.T);
            }
            this.Close();
        }
        b = true;
        this.loadData();
    }

    private void pictureBox10_MouseEnter(object sender, EventArgs e)
    {
        pictureBox10.Image = Tesis.Properties.Resources.AlmacenarFocused;
    }

    private void pictureBox10_MouseLeave(object sender, EventArgs e)
    {
        pictureBox10.Image = Tesis.Properties.Resources.AlmacenarUnfocused;
    }
}
```

5.4.3 Implementación del Entorno WEB

En este renglón se especifica la implementación a código en el lenguaje de programación Silverlight del entorno Web para el sistema FutSal Manager.

5.4.3.1 Agenda

El código en XAML para el módulo de la agenda, se presenta a continuación:

```
<Canvas x:Name="LayoutRectangulo">

<Rectangle x:Name="RectanguloFondo" Margin="0" Fill="White"
Opacity="0.7"/>

<Border x:Name="BordeCalendario" BorderBrush="Black"
BorderThickness="2" Grid.Row="1" Grid.Column="0" Grid.ColumnSpan="7">

<Grid x:Name="Calendario">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="*"/>
<ColumnDefinition Width="*"/>
<ColumnDefinition Width="*"/>
<ColumnDefinition Width="*"/>
<ColumnDefinition Width="*"/>
<ColumnDefinition Width="*"/>
<ColumnDefinition Width="*"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="0.3*"/>
<RowDefinition Height="0.3*"/>
<RowDefinition Height="*"/>
<RowDefinition Height="*"/>
<RowDefinition Height="*"/>
<RowDefinition Height="*"/>
<RowDefinition Height="*"/>
<RowDefinition Height="*"/>

</Grid.RowDefinitions>

<Border x:Name="BordeDias" BorderBrush="Gray"
BorderThickness="2" Grid.Row="1" Grid.Column="0" Grid.ColumnSpan="7"/>
```

```
            <ComboBox x:Name="ComboMes" Grid.Row="0" Grid.Column="3"
HorizontalAlignment="Center" VerticalAlignment="Center"/>

            <ComboBox x:Name="ComboAno" Grid.Row="0" Grid.Column="6"
HorizontalAlignment="Center" VerticalAlignment="Center"/>

            <TextBlock Text="Domingo" Foreground="DarkRed" FontSize="16"
Grid.Row="1" Grid.Column="0" HorizontalAlignment="Center"
VerticalAlignment="Center"/>
            <TextBlock Text="Lunes" Foreground="DarkRed" FontSize="16"
Grid.Row="1" Grid.Column="1" HorizontalAlignment="Center"
VerticalAlignment="Center"/>
            <TextBlock Text="Martes" Foreground="DarkRed" FontSize="16"
Grid.Row="1" Grid.Column="2" HorizontalAlignment="Center"
VerticalAlignment="Center"/>
            <TextBlock Text="Miercoles" Foreground="DarkRed" FontSize="16"
Grid.Row="1" Grid.Column="3" HorizontalAlignment="Center"
VerticalAlignment="Center"/>
            <TextBlock Text="Jueves" Foreground="DarkRed" FontSize="16"
Grid.Row="1" Grid.Column="4" HorizontalAlignment="Center"
VerticalAlignment="Center"/>
            <TextBlock Text="Viernes" Foreground="DarkRed" FontSize="16"
Grid.Row="1" Grid.Column="5" HorizontalAlignment="Center"
VerticalAlignment="Center"/>
            <TextBlock Text="Sábado" Foreground="DarkRed" FontSize="16"
Grid.Row="1" Grid.Column="6" HorizontalAlignment="Center"
VerticalAlignment="Center"/>

        </Grid>

    </Border>

</Canvas>
```

5.4.3.2 Selecciones

El código para este módulo se presenta a continuación:

```
<Canvas x:Name="LayoutRectangulo">
```

```
        <Rectangle x:Name="RectanguloFondo" Margin="0" Fill="White"
Opacity="0.7"/>

        <TextBlock Name="TituloSelecciones" TextAlignment="Center"
Text="Selecciones" Foreground="DarkRed" FontSize="18"
FontFamily="Calibri"/>

        <Border x:Name="BordeSelecciones" BorderBrush="Black"
BorderThickness="2">
            <ScrollViewer Background="Transparent"
VerticalScrollBarVisibility="Auto" HorizontalScrollBarVisibility="Auto">
                <ListBox x:Name="ListaSelecciones" Background="Transparent"
SelectionChanged="ListaSelecciones_SelectionChanged">
                    <ListBox.ItemTemplate>
                        <DataTemplate>
                            <StackPanel Orientation="Horizontal" Height="50">
                                <Image x:Name="LogoEquipo" Width="50"
Source="{Binding logo}"/>
                                <TextBlock Text="{Binding nombre}" Margin="0"
Foreground="Black" FontSize="14"/>
                            </StackPanel>
                        </DataTemplate>
                    </ListBox.ItemTemplate>
                </ListBox>
            </ScrollViewer>
        </Border>

        <TextBlock Name="TituloJugadores" TextAlignment="Center"
Text="Jugadores" Foreground="DarkRed" FontSize="18" FontFamily="Calibri"/>

        <Border x:Name="BordeJugadores" BorderBrush="Black"
BorderThickness="2">
            <ScrollViewer Background="Transparent"
VerticalScrollBarVisibility="Auto" HorizontalScrollBarVisibility="Auto">
                <ListBox x:Name="ListaJugadores" Background="Transparent">
                    <ListBox.ItemTemplate>
                        <DataTemplate>
                            <StackPanel Orientation="Horizontal" Height="50">
                                <Image x:Name="FotoJugador" Width="50"
Source="{Binding imagen}"/>
                                <StackPanel Orientation="Vertical" Margin="0">
                                    <TextBlock Text="{Binding apellido}" Margin="0"
Foreground="DarkRed" FontSize="16"/>
```

```
                        <TextBlock Text="{Binding nombre}" Margin="0"
Foreground="Black"/>
                    </StackPanel>
                </StackPanel>
            </DataTemplate>
        </ListBox.ItemTemplate>
    </ListBox>
</ScrollViewer>
</Border>

<TextBlock Name="TituloEntrenadores" TextAlignment="Center"
Text="Cuerpo Técnico" Foreground="DarkRed" FontSize="18"
FontFamily="Calibri"/>

<Border x:Name="BordeEntrenadores" BorderBrush="Black"
BorderThickness="2">
    <ScrollViewer Background="Transparent"
VerticalScrollBarVisibility="Auto" HorizontalScrollBarVisibility="Auto">
        <ListBox x:Name="ListaEntrenadores" Background="Transparent">
            <ListBox.ItemTemplate>
                <DataTemplate>
                    <StackPanel Orientation="Horizontal" Height="60">
                        <Image x:Name="FotoEntrenador" Width="50"
Source="{Binding imagen}"/>
                        <StackPanel Orientation="Vertical" Margin="0">
                            <TextBlock Text="{Binding especialidad}" Margin="0"
Foreground="DarkRed" FontSize="16"/>
                            <TextBlock Text="{Binding apellido}" Margin="0"
Foreground="Black"/>
                            <TextBlock Text="{Binding nombre}" Margin="0"
Foreground="Black"/>
                        </StackPanel>
                    </StackPanel>
                </DataTemplate>
            </ListBox.ItemTemplate>
        </ListBox>
    </ScrollViewer>
</Border>

<TextBlock Name="TituloUltimos" TextAlignment="Center" Text="Últimos
Resultados" Foreground="DarkRed" FontSize="18" FontFamily="Calibri"/>
```

```
        <Border x:Name="BordeUltimos" BorderBrush="Black"
BorderThickness="2">

            <ScrollViewer Background="Transparent"
VerticalScrollBarVisibility="Auto" HorizontalScrollBarVisibility="Auto">

                <ListBox x:Name="ListaResultados" Background="Transparent">
                    <ListBox.ItemTemplate>
                        <DataTemplate>
                            <StackPanel Orientation="Horizontal" Height="40">
                                <TextBlock Text="{Binding local}" Margin="0"
Foreground="Black" FontSize="12" VerticalAlignment="Center"
TextWrapping="Wrap" TextAlignment="Left"/>
                                <StackPanel Orientation="Vertical" Margin="0">
                                    <TextBlock Text="{Binding fecha}" Margin="0"
Foreground="DarkRed" TextAlignment="Center"/>
                                    <TextBlock Text="{Binding resultado}" Margin="0"
Foreground="DarkRed" TextAlignment="Center"/>
                                </StackPanel>
                                <TextBlock Text="{Binding visitante}" Margin="0"
Foreground="Black" FontSize="12" VerticalAlignment="Center"
TextWrapping="Wrap" HorizontalAlignment="Right" TextAlignment="Right"/>
                            </StackPanel>
                        </DataTemplate>
                    </ListBox.ItemTemplate>
                </ListBox>

            </ScrollViewer>

        </Border>

        <TextBlock Name="TituloProximos" TextAlignment="Center"
Text="Próximos Partidos" Foreground="DarkRed" FontSize="18"
FontFamily="Calibri"/>

        <Border x:Name="BordeProximos" BorderBrush="Black"
BorderThickness="2">

            <ScrollViewer Background="Transparent"
VerticalScrollBarVisibility="Auto" HorizontalScrollBarVisibility="Auto">

                <ListBox x:Name="ListaProximos" Background="Transparent">
                    <ListBox.ItemTemplate>
```

```
                    <DataTemplate>
                        <StackPanel Orientation="Horizontal" Height="40">
                            <TextBlock Text="{Binding local}" Margin="0"
Foreground="Black" FontSize="12" VerticalAlignment="Center"
TextWrapping="Wrap" TextAlignment="Left"/>
                            <StackPanel Orientation="Vertical" Margin="0">
                                <TextBlock Text="{Binding fecha}" Margin="0"
Foreground="DarkRed" TextAlignment="Center"/>
                                <TextBlock Text="VS" Margin="0"
Foreground="DarkRed" TextAlignment="Center"/>
                            </StackPanel>
                            <TextBlock Text="{Binding visitante}" Margin="0"
Foreground="Black" FontSize="12" VerticalAlignment="Center"
TextWrapping="Wrap" HorizontalAlignment="Right" TextAlignment="Right"/>
                        </StackPanel>
                    </DataTemplate>
                </ListBox.ItemTemplate>
            </ListBox>

        </ScrollViewer>

    </Border>

    <TextBlock Name="TituloProximos2" TextAlignment="Center"
Text="Próximos Eventos" Foreground="DarkRed" FontSize="18"
FontFamily="Calibri"/>

    <Border x:Name="BordeProximos2" BorderBrush="Black"
BorderThickness="2">

        <ScrollViewer Background="Transparent"
VerticalScrollBarVisibility="Auto" HorizontalScrollBarVisibility="Auto">

            <ListBox x:Name="ListaEventos" Background="Transparent">
                <ListBox.ItemTemplate>
                    <DataTemplate>
                        <StackPanel Orientation="Horizontal" Height="50">
                            <Image x:Name="LogoEvento" Width="50"
Source="{Binding Logo}"/>
                            <StackPanel Orientation="Vertical" Margin="0">
                                <TextBlock Text="{Binding nombre}" Margin="0"
Foreground="DarkRed" FontSize="14"/>
                                <StackPanel Orientation="Horizontal">
```

```
                                <TextBlock Text="Inicio: " Foreground="Black"
FontSize="10"/>
                                <TextBlock Text="{Binding fi}" Margin="0"
Foreground="Black" FontSize="10"/>
                            </StackPanel>
                            <StackPanel Orientation="Horizontal">
                                <TextBlock Text="Final: " Foreground="Black"
FontSize="10"/>
                                <TextBlock Text="{Binding ff}" Margin="0"
Foreground="Black" FontSize="10"/>
                            </StackPanel>
                        </StackPanel>
                    </StackPanel>
                </DataTemplate>
            </ListBox.ItemTemplate>
        </ListBox>

    </ScrollViewer>

</Border>

</Canvas>
```

5.5 Pruebas

En este flujo de trabajo es donde se llevan a cabo las pruebas de unidad de todos los componentes del sistema. Este flujo de trabajo es decisivo para la culminación del desarrollo del sistema FutSal Manager, ya que es aquí donde se lleva a cabo la integración y prueba de todos los módulos que conforman la aplicación, obteniendo de esta manera una versión confiable del sistema.

5.5.1 Pruebas de Unidad

Para las pruebas de unidad se tomaron en cuenta los módulos críticos o de mayor importancia dentro de la aplicación. Entre los módulos que componen la aplicación se destacan como los más importantes por sus funciones, los siguientes:

Entre los módulos que componen la aplicación se destacan como los más importantes por sus funciones, los siguientes:

5.5.1.1 Administrar Torneos

En este módulo se configuran y modifican todos los factores relacionados con los torneos en el sistema. Entre estos factores tenemos la adición y configuración de torneos, el registro de equipos en torneos, la generación de partidos y fases, la adición de estadísticas a los partidos y la generación de reportes. Las pruebas se muestran en la figura 5.9.

Figura 5.9 Prueba Administrar Torneos

Fuente: Propia

5.5.1.2 Administrar Equipos

En este módulo se configuran y modifican todos los factores relacionados con los equipos en el sistema. Entre estos factores tenemos la adición y configuración de equipos, la gestión de los jugadores en los equipos y la gestión de los entrenadores en los equipos. Las pruebas se muestran en la figura 5.10.

Figura 5.10. Prueba Administrar Equipos

Fuente: Propia

5.6 Conclusión de la Fase

Uno de los objetivos de la fase de construcción es desarrollar el software a partir de una línea base de la arquitectura ejecutable, hasta el punto que esté listo, para ser trasmitido a la comunidad de usuarios. Aquí se detallan y analizan los requisitos restantes, la mayor parte de este flujo de trabajo fue realizada en las dos fases anteriores. El diseño juega un papel importante, y es en esta fase en la que tiene lugar la mayor parte del trabajo de los flujos de implementación y pruebas. Al final de la fase de construcción se ha llevado el producto software a su versión operativa.

CAPÍTULO VI. FASE DE TRANSICIÓN

6.1 Introducción

El objetivo de esta fase es asegurarse que el software cumple completamente las necesidades de sus usuarios. En esta fase se realizan ajustes menores que se basan en la retroalimentación de los usuarios finales del producto. No deben ocurrir cambios abruptos en esta fase debido a que al llegar a la misma, todos los problemas y complicaciones estructurales ya deben haber sido solventadas.

6.2 Análisis

En este flujo de trabajo, se tomó la retroalimentación de los usuarios basada en la utilización del software y se recogieron pequeños ajustes que los usuarios consideraron necesarios para la perfecta ejecución del sistema. No ocurrieron ningún tipos de fallas de rendimiento por lo tanto no es necesario tomar en consideración este aspecto para los ajustes. Los usuarios mostraron la necesidad de modificar las estadísticas de un partido, debido a que al momento de introducirlas, puede ocurrir algún tipo de error y una vez almacenadas estas no se podían modificar. También fue manifestada la necesidad de modificar las sanciones de los jugadores en los torneos.

6.3 Diseño

Las nuevas necesidades manifestadas por los usuarios no requieren de cambios de diseño en la aplicación, por lo tanto no se requiere de la ejecución de este flujo de trabajo en esta fase.

6.4 Implementación

En este flujo de trabajo se deben realizar los pequeños ajustes que permitan cumplir con las nuevas necesidades y ajustes planteados por los usuarios al inicio de esta fase. Para ello es necesaria la creación de nuevos procedimientos almacenados que permitan la alteración de estadísticas en los partidos así como también las sanciones de los jugadores.

6.4.1 Procedimientos Almacenados

En este apartado se presentan los procedimientos almacenados que permiten cumplir con los requisitos recientemente planteados en esta fase que llevarán a la aplicación a su versión final.

6.4.1.1 Modificar Estadísticas de Equipos

Este procedimiento almacenado permite la modificación de las estadísticas en un partido para los equipos participantes en el mismo.

Create Procedure ModEstEqui (@idTorneo int, @idPartido int, @idEquipo int, @GolesFavor int, @GolesContra int, @TA int, @TR int) As Update EstadisticasEquipo Set GolesFavor = @GolesFavor, GolesContra = @GolesContra, TA = @TA, TR = @TR Where idTorneo = @idTorneo AND idPartido = @idPartido AND idEquipo = @idEquipo

6.4.1.2 Modificar Estadísticas de Jugadores

Este procedimiento almacenado permite la modificación de las estadísticas en un partido para los jugadores participantes en el mismo.

Create Procedure ModEstJug (@idTorneo int, @idPartido int, @idEquipo int, @idJugador int, @Goles int, @Abridor varchar(8), @TA int, @TR int) As Update EstadisticasJugador Set Goles = @Goles, Abridor = @Abridor, TA = @TA, TR = @TR Where idTorneo = @idTorneo AND idPartido = @idPartido AND idEquipo = @idEquipo AND idJugador = @idJugador

6.4.1.3 Modificar Sanción

Este procedimiento almacenado permite la alteración de las sanciones para los jugadores en un torneo en específico.

Create Procedure AltSan (@idTorneo int, @idJugador int, @Sancion int) As Update Sancionados Set cantJornadas = @Sancion Where idTorneo = @idTorneo AND idJugador = @idJugador

6.5 Pruebas

En este flujo de trabajo sólo se realizaron pruebas para la verificación del correcto funcionamiento de los procedimientos almacenados recién creados y su integración al sistema. Estos cumplieron con su función, culminando así con los ajustes planteados por los usuarios finales del sistema FutSal Manager.

6.6 Conclusión de la Fase de Transición

Los objetivos de esta fase se cumplieron en su totalidad, llevando el software de su versión BETA a su versión final la cual contó con la total aprobación por parte de los usuarios finales. Se realizaron pequeños ajustes de acuerdo con la retroalimentación recibida por parte de los usuarios, y se realizaron las últimas pruebas para revisar el correcto funcionamiento del sistema en conjunto con una evaluación de rendimiento. Al cumplir con las pruebas de forma satisfactoria, el sistema ya puede ser instalado y utilizado por todos sus usuarios finales.

CONCLUSIONES

- Mediante la utilización de la herramienta del RUP, fue posible realizar el análisis y la especificación de todos los requerimientos del sistema con un nivel de detalle variable dependiente de la fase del RUP en ejecución, lo que permite la creación de una muy buena base para el desarrollo de la aplicación.

- El modelo de datos que sustenta a la base de datos del sistema fue diseñado mediante el modelo entidad-relación el cual es el más utilizado en la actualidad y que a su vez ofrece el diseño más robusto y eficiente entre todos los modelos de diseño de bases de datos.

- Los entornos gráficos de la aplicación son un componente de mucha importancia ya que son los que permiten la interacción entre los usuarios y el sistema. Estos se realizaron de manera iterativa con la participación de los usuarios finales de forma que se llegó a un diseño muy agradable y de fácil utilización pero que a su vez permite cumplir con todos los requerimientos de la aplicación.

- Los subsistemas se diseñaron de acuerdo a las fases proceso unificado, permitiendo así la fácil y robusta construcción del software, comenzando por la arquitectura que soporta la aplicación, para así tener una buena base la cual permitió la introducción de los nuevos módulos y la realización de ajustes al sistema sin necesidad de realizar cambios fundamentales en el mismo.

- Los subsistemas que componen la aplicación fueron construidos en diferentes lenguajes de programación, C# para la aplicación de escritorio y Silverlight para la aplicación WEB. La herramienta de C# en conjunto con el Visual

Studio ofrece en las aplicaciones un gran rendimiento y robustez, y a su vez cumple a cabalidad con el paradigma de la programación orientada a objetos. Silverlight a pesar de ser un lenguaje nuevo, posee un gran nivel de soporte en la WEB, y permite codificar páginas WEB de una forma sencilla y ordenada, convirtiéndolo en un lenguaje que se introducirá en gran medida en el campo de los lenguajes de programación para Internet.

- Los subsistemas se integraron y fueron probados de acuerdo al proceso unificado luego de la construcción de la arquitectura y de los diferentes subsistemas. Debido a que la arquitectura y los subsistemas fueron diseñados y construidos siguiendo los pasos del RUP, la integración no tuvo ningún tipo de complicaciones lo que indica que esta herramienta es muy importante para la construcción e integración de software.

- La documentación técnica es una parte muy importante para el desarrollo de aplicaciones de cualquier tipo, ya que es la que permite a los usuarios conocer el software con el que se disponen a trabajar, y a su vez les ayuda con cualquier duda o inconveniente que tengan con la aplicación.

RECOMENDACIONES

- Se recomienda realizar mantenimiento o mejoras del sistema periódicamente, a fin de garantizar resguardo de la información y el buen funcionamiento del mismo.

- Se recomienda realizar un respaldo periódico de la base de datos del sistema, guardándola en otro archivo, para luego borrarla del sistema y crearla nuevamente. De esta forma se evita la sobrecarga de la base de datos y aún así en caso de ser necesario, se puede volver a utilizar la base de datos respaldada.

- Es importante ejecutar auditorías al sistema, para el evaluar la eficiencia del mismo y así alargar su vida útil. Esto se realiza mediante la agregación de nuevos procedimientos qué adaptan el sistema a las necesidades y plataformas del momento.

BIBLIOGRAFÍA CITADA

[1] Moya A. **"Desarrollo de un Sistema para la Automatización del proceso de Admisión y Control de Estudios en un Instituto Universitario"** Venezuela. (2004)

[2] Simoni M **"Desarrollo de un Sistema de Información para la Automatización de los procesos realizados en el Departamento de Ciencias de la Unidad de Estudios Básicos de la Universidad de Oriente, Núcleo Anzoátegui"** Venezuela (2004)

[3] Díaz A. y Guerra Z. **"Actualización del Software Educativo Introducción a la Ingeniería en Computación, creado como herramienta didáctica para el proceso Enseñanza-Aprendizaje de la unidad curricular Introducción a la Ingeniería en Computación"** Venezuela. (2005)

[4] Núñez L. **"Desarrollo de una aplicación WEB para la visualización en tiempo real de los parámetros operacionales de producción de la empresa PDVSA"** Venezuela. (2007)

[5] Federación Venezolana de Fútbol "**Historia del Fútbol Sala En Venezuela"** http://www.futsala.com.ve/Historia/Historia.asp (2008)

[6] Purdum J. **"Introducción a la Programacion Orientada a Objetos"** Wrox. USA. (2008)

[7] Nagelet C. "**Professional C# 2005**" Wrox. USA. (2005).

[8] Elmasri R y Navathe S. "**Sistemas de Bases de Datos"** Addison-Wesley. Argentina. (1997).

[9] Vieira R. **"Beginning SQL Server 2005 Programming**" Wrox. USA. (2005).

[10] Elmasri R y Navathe S. "**Fundamentals of Database Systems**" Addison-Wesley. USA. (2006)

[11] Evjen B, Hanselman S y Rader D. "**Professional ASP.NET 2.0**" Wrox. USA. (2006).

[12] Kruchten P. "**The Rational Unified Process".** Addison-Wesley. USA. (2000).

[13] Larman C. **"UML y Patrones"** Prentice-Hall. México. (2006)

BIBLIOGRAFÍA ADICIONAL

Scanlon J. **"Accelerated Silverlight 2"**. Apress. USA. (2008)

Bellinaso M. "**ASP.NET 2.0 Website Programming: Problem - Design - Solution".** Wrox. USA. (2006).

Vieira R. **"Professional SQL Server 2005 Programming".** Wrox. USA. (2006).

Watsonet K. "**Beginning Visual C# 2005".** Wrox. USA. (2005)

Kroll P. y Kruchten P. **"The Rational Unified Process Made Easy: A Practitioner's Guide to the RUP".** Addison-Wesley USA. (2003)

METADATOS PARA TRABAJOS DE GRADO, TESIS Y ASCENSO:

TÍTULO	**DESARROLLO DE UN SISTEMA DE GESTIÓN PARA LOS PROCESOS ADMINISTRATIVOS DE LA COMISIÓN NACIONAL DE FÚTBOL SALA DE LA FEDERACIÓN VENEZOLANA DE FÚTBOL**
SUBTÍTULO	

AUTOR (ES):

APELLIDOS Y NOMBRES	**CÓDIGO CVLAC / E MAIL**
Pereira Ginestra, Israel	**CVLAC: 17.900.211** **E MAIL: israelpereira@gmail.com**
	CVLAC: **E MAIL:**
	CVLAC: **E MAIL:**
	CVLAC: **E MAIL:**

PALABRAS O FRASES CLAVES:

BASES DE DATOS

APLICACIONES WEB

SQL SERVER

PROCESO UNIFICADO DE RATIONAL

PROGRAMACIÓN ORIENTADA A OBJETOS

METADATOS PARA TRABAJOS DE GRADO, TESIS Y ASCENSO:

ÁREA	SUBÁREA
Ingeniería y Ciencias Aplicadas	Ingeniería en Computación

RESUMEN (ABSTRACT):

En el presente trabajo se desarrolla una aplicación que permite gestionar todos los procesos administrativos de la Comisión Nacional de Fútbol Sala. El software desarrollado brinda al usuario la posibilidad de administrar torneos, equipos, jugadores y entrenadores, y a su vez permite la generación de reportes que faciliten la visualización de estadísticas para el público en general. La aplicación fue realizada utilizando el lenguaje de programación orientado a objetos C# y el sistema manejador de bases de datos SQLServer pertenecientes al entorno de desarrollo integrado Microsoft Visual Studio 2008. Para el desarrollo del proyecto se utilizó la metodología del Proceso Unificado de Rational (RUP), utilizando el análisis y diseño iterativo por medio del Lenguaje Unificado de Modelado (UML).

METADATOS PARA TRABAJOS DE GRADO, TESIS Y ASCENSO:

CONTRIBUIDORES:

APELLIDOS Y NOMBRES	ROL / CÓDIGO CVLAC / E_MAIL				
Jose Guevara	ROL	CA	AS X	TU	JU
	CVLAC:				
	E_MAIL				
	E_MAIL				
Gabriela Veracierta	ROL	CA	AS	TU	JU X
	CVLAC:				
	E_MAIL				
	E_MAIL				
Víctor Mujica	ROL	CA	AS	TU	JU X
	CVLAC:				
	E_MAIL				
	E_MAIL				
	ROL	CA	AS	TU	JU
	CVLAC:				
	E_MAIL				
	E_MAIL				

FECHA DE DISCUSIÓN Y APROBACIÓN:

2009	06	09
AÑO	MES	DÍA

LENGUAJE. SPA

METADATOS PARA TRABAJOS DE GRADO, TESIS Y ASCENSO:

ARCHIVO (S):

NOMBRE DE ARCHIVO	TIPO MIME
TESIS. Desarrollo de un Sistema de Gestion.doc	Application/MSWord

CARACTERES EN LOS NOMBRES DE LOS ARCHIVOS: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z. a b c d e f g h i j k l m n o p q r s t u v w x y z. 0 1 2 3 4 5 6 7 8 9.

ALCANCE

ESPACIAL: _____ (**OPCIONAL**)

TEMPORAL: **Intemporal** (**OPCIONAL**)

TÍTULO O GRADO ASOCIADO CON EL TRABAJO:
Ingeniero en Computación

NIVEL ASOCIADO CON EL TRABAJO:
Pregrado

ÁREA DE ESTUDIO:
Departamento de Computación y Sistemas

INSTITUCIÓN:
Universidad de Oriente. Núcleo de Anzoátegui

METADATOS PARA TRABAJOS DE GRADO, TESIS Y ASCENSO:

DERECHOS

ARTÍCULO 44: "LOS TRABAJOS DE GRADO SON DE EXCLUSIVA PROPIEDAD DE LA UNIVERSIDAD DE ORIENTE Y SOLO PODRÁN SER UTILIZADOS A OTROS FINES CON EL CONSENTIMIENTO DEL CONSEJO DE NÚCLEO RESPECTIVO, QUIEN LO PARTICIPARÁ AL CONSEJO UNIVERSITARIO"

AUTOR

TUTOR **JURADO** **JURADO**

POR LA SUBCOMISIÓN DE TESIS