UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE INGENIERÍA EN SISTEMAS ELECTRÓNICA E INDUSTRIAL

CARRERA DE INGENIERÍA EN SISTEMAS COMPUTACIONALES E INFORMÁTICOS

TEMA:

"SISTEMA AUTOMATIZADO PARA EL REGISTRO Y SANCIONES DE LOS JUGADORES EN LA LIGA DEPORTIVA PARROQUIAL HUACHI GRANDE"

Trabajo de Graduación. Modalidad: TEMI. Trabajo Estructurado de Manera Independiente, presentado previo la obtención del título de Ingenieroen Sistemas Computacionales e Informáticos.

Autor:Byron Bladimir Cárdenas Villacres.

Tutor: Ing. Eduardo Chaso.

Ambato - Ecuador

Enero, 2012

APROBACIÓN DEL TUTOR

En mi calidad de tutor del trabajo de investigación sobre el tema:"Sistema automatizado para el registro y sanciones de los jugadores en la Liga Deportiva Parroquial Huachi Grande", del señor Byron Bladimir Cárdenas Villacres, egresado de la Carrera de Ingeniería en Sistemas Computacionales e Informáticos, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, de la Universidad Técnica de Ambato, considero que el informe investigativo reúne los requisitos suficientes para que continúe con los trámites y consiguiente aprobación de conformidad al Art. 16 del Capítulo II, del Reglamento de Graduación para Obtener el Título Terminal de Tercer Nivel de la Universidad técnica de Ambato.

Ambato Enero26, 2012

EL TUTOR

Ing. Eduardo Chaso

AUTORÍA

El presente trabajo de investigación titulado:"Sistema automatizado para el registro y sanciones de los jugadores en la Liga Deportiva Parroquial Huachi Grande". Es absolutamente original, auténtico y personal, en tal virtud, el contenido, efectos legales y académicos que se desprenden del mismo son de exclusiva responsabilidad del autor.

Ambato Enero26, 2012

Byron Bladimir Cárdenas Villacres
CC: 180403772-7

APROBACIÓN DE LA COMISIÓN CALIFICADORA

La Comisión Calificadora del presente trabajo conformada por los señores docentes Ing. Galo López y el Ing. Hernando Buenaño, revisó y aprobó el Informe Final del trabajo de graduación titulado: "Sistema automatizado para el registro y sanciones de los jugadores en la Liga Deportiva Parroquial Huachi Grande", presentado por el señor Byron Bladimir Cárdenas Villacres de acuerdo al Art. 18 del Reglamento de Graduación para Obtener el Título Terminal de Tercer Nivel de la Universidad Técnica de Ambato.

Ing. Oswaldo Paredes
PRESIDENTE DEL TRIBUNAL

Ing. Galo Lopez	Ing. Hernando Buenaño
DOCENTE CALIFICADOR	DOCENTE CALIFICADOR

DEDICATORIA

A mis padres, porque creyeron en mi y porque me sacaron adelante, dándome ejemplos dignos de superación y entrega, porque en gran parte gracias a ustedes, hoy puedo alcanzar mi meta, ya que siempre estuvieron impulsándome en los momentos más difíciles de mi carrera, y que por el orgullo que sienten por mí, fue lo que me hiso ir hasta el final. Va por ustedes, por lo que valen, porque admiro su fortaleza y por lo que han hecho de mí.

Byron Bladimir CardenasVillacres

AGRADECIMIENTO

A mi esposa, mis hermanos, y mis amigos. Gracias por haber fomentado en mí el deseo de superación y el anhelo de triunfo en la vida.

Mil palabras no bastarían para agradecerles su apoyo, su comprensión y sus consejos en los momentos difíciles.

A todos, espero no defraudarlos y contar siempre con su valioso apoyo, sincero e incondicional.

Byron Bladimir Cárdenas Villacres

GUION DE CONTENIDOS

CAPÍTULO I

EL PROBLEMA DE INVESTIGACIÓN

CAPÍTULO II

MARCO TEÓRICO

CAPÍTULO III

METODOLOGÍA

CAPÍTULO IV

ANÁLISIS E INTERPRETACIÓN DE RESULTADOS

CAPÍTULO V

CAPÍTULO VI

PROPUESTA

CAPITULO I

EL PROBLEMA DE INVESTIGACIÓN

1.1Tema de investigación

Sistema automatizado para el registro y sanciones de los jugadores en la Liga Deportiva Parroquial Huachi Grande.

1.2 Planteamiento del problema

1.2.1 Contextualización

La mayoría de Ligas del país no puede contar con un sistema para el control de registro y sanciones de los jugadores de cada equipo al que pertenece, la mayoría no cuenta con una red de datos puesto a que poseen los recursos para poder obtener la infraestructura necesaria para una buena comunicación de datos, con la implementación de una red muchas ligas han logrado un mejor desarrollo que son cabezas parroquiales, facilitando el flujo de información y la agilidad en los tramites que se realizan dentro de estas.

La mayoría de ligas deportivas de los cantones de la provincia de Tungurahua tiene incorporada una adecuada infraestructura de red para su intercambio de información de dependencias, esto permite mayor agilidad cuando empieza y en el transcurso del campeonato ya que se evitan una serie de inconvenientes dentro y fuera de la Liga Deportiva, ya que pueden ser controlados de manera rápida y eficaz, evitando ser sancionados o multados en los diferentes actos deportivos.

En el cantón Ambato existen diferentes ligas que poseen un sistema para el registro de jugadores más no para las sanciones que imponen dentro y fuera de la Liga Deportiva, el 70% de la información de las ligas no se encuentra correctamente estructurada, lo que limita la posibilidad de utilizarla de forma ventajosa.

1.2.2 Análisis crítico

En la Liga Deportiva Parroquial Huachi Grande no cuentan con los suficientes recursos económicos por lo que se les limita poder comprar o adquirir un sistema con las funciones que la Liga requiera, esto conlleva a que en la Liga no se lleve un registro adecuado de la información de los jugadores, las sanciones que son aplicadas a los equipos o a los jugadores dentro del campo de juego, que pueden ser sanciones económicas en caso de ser el equipo y sanciones por fechas de juego en caso de ser jugador.

El no contar con personas capacitadas para poder realizar un software específico para el beneficio de la Liga que contenga todo lo que esta requiera para su uso dentro y fuera de la institución provoca que la información necesaria no puede ser llevada de forma ágil y segura por lo que no facilita el normal funcionamiento de la liga y tal vez que su ejecución debe ser de manera inmediata tanto así que puede haber pérdidas económicas, perdida de información de los equipos, conflictos entre equipos y socios de la liga.

Este sistema realizará el registro de los jugadores de cada equipo en donde se los archiva por cada institución a la que pertenece, el control de contabilidad de las sanciones impuestas a jugadores y equipos, inscripciones de cada institución, el mismo que será desarrollado en lenguaje C con la herramienta SharpDevelop trabajando con Base de Datos la cual será creada en Postgres.

1.2.3 Prognosis

De continuar con este problema en la Liga Deportiva Parroquial Huachi Grande, seguirá existiendo un control inadecuado de la documentación, la información que brinda no será ágil ni segura, existirá un retraso en los trámites, además de pérdidas económicas y de información necesaria para el buen funcionamiento de la liga. Por lo que se hace necesario desarrollar e implementar el sistema automatizado para el registro y sanciones de los jugadores en la Liga Deportiva Parroquial Huachi Grande.

1.2.4 Formulación del problema

¿Qué incidencia tiene un sistema automatizado en el registro y sanciones de los jugadores en la Liga Deportiva Parroquial Huachi Grande?

1.2.5 Preguntas Directrices

¿Qué características deberá tener el sistema operativo para implementar un sistema automatizado?

¿Qué características debe tener el modelado de datos para el registro y sanciones de los jugadores?

¿Cuáles son los requerimientos necesarios para implementar el sistema automatizado?

1.2.6 Delimitación del problema

El presente trabajo se desarrollará en la Liga Deportiva Parroquial Huachi Grande ubicado en la panamericana Sur Km 2 en la parroquia Huachi Grande junto a la iglesia, el tiempo estimado para el desarrollo del proyecto es de seis meses a partir de que se apruebe el proyecto de graduación.

1.3 Justificación

El estudio para el diseño del sistema de registro y sanción de los jugadores se lo realiza principalmente enfocado en mantener un control mayor de los datos que actualmente se llevan en archivos planos, labor que no es eficiente para poder realizar todos los procesos que esto involucra como son los cobros de multas, registros de equipos, sanciones a jugadores y directivos de la liga, entre otros.

Por la labor desempeñada hasta la actualidad por la liga se ha podio evidenciar que el proceso actualmente dado por la misma no es el óptimo causando malestar en los usuarios finales que son los que se siente inconformes con esos servicios.

El sistema de registro y sanción tendrá como finalidad mejorar el servicio a los usuarios, y evitará pérdidas económicas, confusión en las inscripciones de los equipos que pertenecen a la Liga y el ágil control de las sanciones de los jugadores, brindando así un mejor servicio en su labor.

La Liga Deportiva Parroquial Huachi Grande a más de lo antes mencionado con la implantación de este sistema tendrá un realce institucional dentro de lo que se refiere en instituciones deportivas de la provincia de Tungurahua ya que no todas pueden contar con un sistema automatizado para el registro y sanción de los jugadores, además se

convertirá en una de las pioneras en implementar nuevas tecnologías al ámbito deportivo resolviendo así necesidades sociales.

Este convenio se lo ha realizado además para aplicar las bases teórico-práctico adquiridos en el transcurso de la vida universitaria, la misma que ayudará a fomentar la creatividad, iniciativa, una actitud positiva e innovadora y sobre todo buscando especializarse en el campo del desarrollo para cada vez llegar a ser más profesional.

1.4 Objetivos

1.4.1 Objetivo General

- Diseñar e implementar un sistema automatizado para el registro y sanciones de los jugadores en la Liga Deportiva Parroquial Huachi Grande.

1.4.2 Objetivos Específicos

- Determinar los procesos con los que funciona el registro y sanciones en la Liga Deportiva Parroquial Huachi Grande.
- Diseñar e implementar el modelado de datos para el registro y sanciones de los jugadores.
- Implantar el sistema automatizado para la Liga Deportiva Parroquial Huachi Grande.

CAPITULO II

MARCO TEÓRICO

2.1 Antecedentes Investigativos

Revisado los archivos de la biblioteca de la Universidad Técnica de Ambato, Facultad de Ingeniería en Sistemas Electrónica e Industrial no se ha realizado algún trabajo parecido.

Revisado en internet existen software parecidos al presente trabajo como:

WinCup.- El programa WinCup sirve para la gestión de ligas deportivas, por ejemplo Fútbol, balonmano, voleibol, ajedrez, tenis, dardos..., 2-24 incluyendo clubes de cantidades impares, ligas estándares de 3 o 5 sets, definición libre. Sistema de puntos (3 puntos por una victoria, 3-1-0, etc.), lista de los mejores clasificados, tabla eterna, todas las temporadas en un archivo, comparación directa de juegos de unos contra otros con la paridad punto -+gol, TOTO-Tip, Tablas de resultados fuera y en casa, todos los resultados a partir de 1.Bundesliga desde 1963.

2.2 Fundamentación Legal

Acuerdo N° 084

SECRETARÍA NACIONAL DEL DEPORTE,

EDUCACIÓN FÍSICA Y RECREACIÓN

CONSIDERANDO:

Que, la ¨LIGA DEPORTIVA PARROQUIAL HUACHI GRANDE¨, del cantón Ambato, Provincia del Tungurahua, solicita del señor Secretario Nacional del Deporte , Educación Física y Recreación, la aprobación a su estatuto, previa la presentación y análisis de la documentación respectiva;

Que, la Secretaria Nacional del Deporte, Educación Física y Recreación de conformidad con lo establecido en el artículo 26 literal ¨d¨ de la ley de Educación

Física, Deporte y Recreación; tiene la facultad para aprobar estatutos de entidades deportivas;

Que, la Federación Deportiva de Tungurahua, mediante oficio N° 0945-S-FDT del 23 de diciembre del 2002, emite informe favorable para la aprobación del estatuto de la ¨LIGA DEPORTIVA PARROQUIAL HUACHI GRANDE¨.

En uso de las atribuciones concedidas por el señor Presidente Constitucional de la República, mediante el Derecho Ejecutivo N° 66 del 27 de enero del 2003, publicado en el Registro Oficial N° 11 del 30 de enero del 2003.

ACUERDA:

Aprobar el estatuto de la ¨LIGA DEPORTIVA PARROQUIAL HUACHI GRANDE¨, para cumplir con los fines establecidos en el Titulo Tercero, Capítulo Octavo de la Ley de Educación Física, Deportes y Recreación y su Reglamento General:

CAPÍTULO II

DE LOS CLUBES

Art. 7. La ¨LIGA DEPORTIVA PARROQUIAL HUACHI GRANDE ¨, estará conformada por los clubes legalmente constituidos quienes deberán cumplir como único requisito el presentar su respectivo acuerdo otorgado por la Secretaria Nacional del Deporte, Educación Física y Recreación.

Art. 8. Son filiales de la liga los clubes que cumplan con el requisito señalado en el artículo anterior.

Art. 9. Son derechos de los clubes filiales:

- **a.** Participar con sus representantes para elegir y ser elegido;
- **b.** Participar de todos los beneficios de la entidad;
- **c.** Recibir ayuda moral y económica de la Liga como tal, y de sus miembros en caso de necesidad; y,
- **d.** Intervenir directa y activamente en la vida de Liga.

Art. 10. Son deberes de los clubes filiales:

- **a.** Cumplir con las obligaciones que se imponen al ingresar como filial de la ¨LIGA DEPORTIVA PARROQUIAL HUACHI GRANDE¨,

b. Observar y cumplir con la Ley de Educación Física, Deportes y Recreación y su Reglamento General, el Estatuto y reglamentos de la Federación Deportiva de Tungurahua, estatuto y reglamento interno de la Liga y demás leyes pertinentes;

c. Los clubes filiales deberán pagar las cuotas ordinarias y extraordinarias establecidas por la Asamblea General, en forma puntual;

d. Los representantes de los clubes deberán desempeñar los cargos o comisiones que les fuere encomendados;

e. Velar por el registro de la Institución dentro y fuera de los locales deportivos y sociales;

f. Intervenir disciplinadamente en todas las actividades sociales, culturales y deportivas de la Liga, siempre que fueren requeridos; y,

g. Todas las demás que se desprendiesen del contenido del estatuto y reglamentos internos de la Liga.

Art. 11. El carácter de filial puede suspenderse o perderse.

11.1. Puede suspenderse por las siguientes razones:

a. Deber cuotas de la Liga por tres meses consecutivos o más;

b. Por agresiones verbales entre directores técnicos y/o deportistas;

c. Por negarse a participar en eventos o programaciones organizadas por la Liga o por los organismos rectores del deporte a la cual la Institución es afiliada.

d. Faltar a los reglamentos en el desarrollo de actos, sesiones, competencias o cualquier evento deportivo en el que participe su club a través de sus dignatarios deportistas y/o socios;

e. Actos que incumplen desacato a la autoridad de la Liga, debidamente comprobados;

f. Las demás contempladas en la Ley, los estatutos, el reglamento interno y reglamentaciones internacionales.

En el caso de que el club pida suspensión en forma voluntaria, este deberá estar al día en sus obligaciones con la Liga.

11.2. El tiempo máximo que podrá durar la suspensión temporal será de un año según la gravedad de la falta. Una vez concluido el periodo de la sanción o transcurrido por lo menos el 50% del periodo de esta, el sancionado podrá pedir

su reincorporación a la Liga, mediante solicitud dirigida a la Asamblea General, previo informe favorable de Presidente de la Liga.

11.3. Si la sanción impuesta por la Liga a uno o más clubes afiliados a ésta, es la de suspensión temporal, esta sanción regirá para todas las actividades deportivas, en que pudieren o estén participando.

Los sancionados por falta de pago, podrán pedir su rehabilitación, una vez que hayan cancelado todo lo adeudado.

11.4. Se garantiza a todos los clubes el derecho a la defensa; de acuerdo con el procedimiento que para el caso se señale en los reglamentos internos de la Liga.

11.5. El carácter de filial se pierde en los siguientes casos:

a. Por descalificación notada al Directorio;
b. Por expulsión decretada por la Asamblea General, de acuerdo con lo que dicte el reglamento correspondiente;
c. Por la no participación en las competencias deportivas y sociales, de acuerdo al reglamento correspondiente, y;
d. Por disolución o liquidación del club.

CAPÍTULO V

DE LAS SANCIONES Y PENAS

Art. 45. Está prohibido a los deportistas afiliados:

a. Abandonar una competencia o evento deportivo sin el permiso del Juez o Arbitro, entrenador o dirigente;
b. Protestar por las decisiones de los jueces o árbitros;
c. Promover incidentes en el lugar de las competencias o eventos deportivos;
d. Dar muestras de indisciplina o inmoralidad en el desarrollo de las competencias o eventos deportivos en los que representen a la Liga;
e. Actuar en representación de otros clubes sin el permiso respectivo;
f. Presentarse a los eventos organizados por la Liga con uniforme distinto al oficial del mismo; y,

g. Los demás que establezcan en las leyes que rigen al deporte y los reglamentos que dicten las Federaciones Provinciales respectivas a sus filiales, según el deporte que se practique; lo señalado en el presente estatuto y en sus reglamentos y demás leyes conexas.

Art. 46. La comisión Jurídica y de Disciplina presentara un informe de las faltas en que hubieren incurrido los clubes, el Directorio, quien una vez recibida la exposición de defensa del club, resolverá sobre el caso y si fuere comprobada debidamente la falta, emitirá la sanción, la cual será comunicada inmediatamente al club y a la Entidad inmediatamente superior de la Liga.

Art. 47. Los socios estarán sujetos a la amonestación y/o multa en los siguientes casos:

a. Por inasistencia a las sesiones de Asamblea General o Directorio;
b. Por abandonar una competencia sin la debida autorización, siempre que este abandono no sea definitivo;
c. Por escándalo ocasionado por mal comportamiento;
d. Por falta de respeto a jueces, árbitros, entrenadores, dirigentes o demás autoridades;
e. Por protestar o reclamar por las decisiones tomadas por los jueces y árbitros;
f. Por utilizar vocabulario inadecuado, en los organismos de la Liga, competencias o lugares de concentración; y,
g. Por las demás que se señalen en el reglamento.

Se debe indicar que la multa será fijada por el Presidente o por la Asamblea General en su caso.

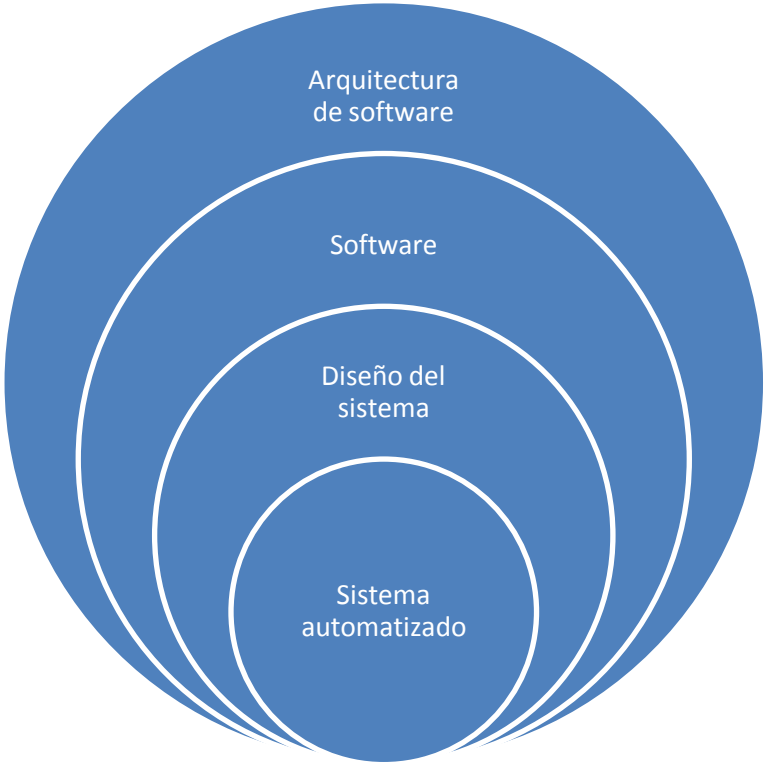
Art. 48. Los clubes podrán ser sujetos de expulsión de la Liga, en los siguientes casos:

a. Quien habiendo sido sancionado o suspendido, participe activamente en los programas o competencias dentro o fuera de la Liga;
b. Los clubes que se negaren a conformar una selección o preselección de su deporte , sin causa justificada alguna; y,
c. Por las demás que se indiquen en el reglamento interno de la Liga.

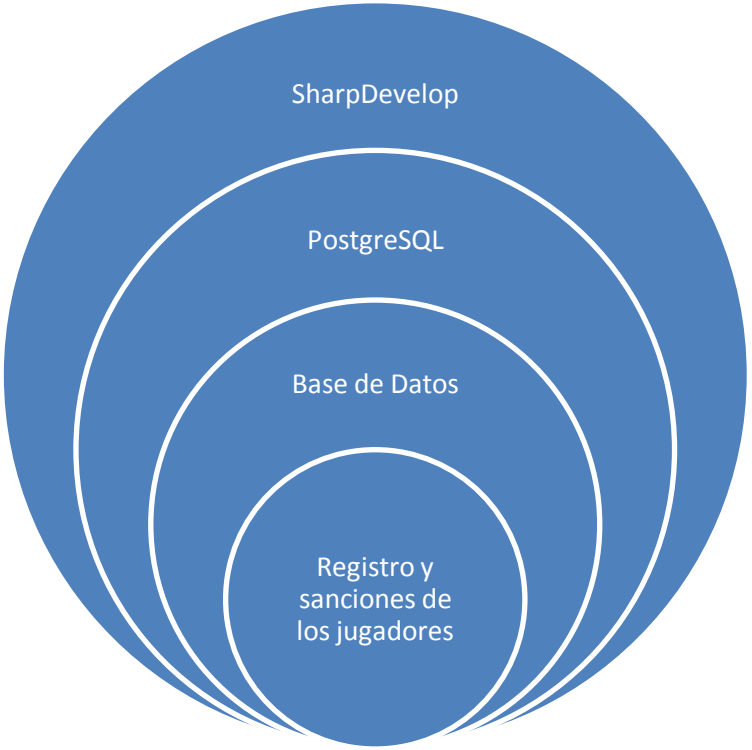
Art. 49. Los dirigentes de los clubes filiales de la Liga, podrán ser sujetos de expulsión, en los siguientes casos, que deberán ser debidamente comprobados respetando el derecho a la defensa del dirigente:

a. Por haber sido sancionado penalmente, mediante sentencia ejecutoriada;
b. Los dirigentes que realicen actos ilícitos, con el objeto de obtener beneficios económicas en desmedro de los ingresos y pertenencias de la Liga;
c. En caso de agresión física entre dirigentes;
d. Por traición a la Entidad a la que se pertenecen debidamente comprobada; y,
e. Por los demás que se indiquen en el reglamento interno de la Liga.

2.3 Categorías fundamentales



V I



V D

Categorías Fundamentales de la variable independiente.

2.3.1 Arquitectura de software

Componentes de software:

"Se distinguen tres componentes básicos de software:

- **Presentación.-** Tiene que ver con la presentación al usuario de un conjunto de objetos visuales y llevar a cabo el procesamiento de los datos producidos por el mismo y los devueltos por el servidor.
- **Lógica de aplicación.-** Esta capa es la responsable del procesamiento de la información que tiene lugar en la aplicación.
- **Base de datos.-** Esta compuesta de los archivos que contienen los datos de la aplicación."

WEB SITE, (2010). Aplicaciones distribuidas. Extraído el 07 de Noviembre del 2010 desde http://knol.google.com/k/aplicaciones-distribuidas

2.3.2 Software

"Se conoce como software al equipamiento lógico o soporte lógico de una computadora digital; comprende el conjunto de los componentes lógicos necesarios que hacen posible la realización de tareas específicas, en contraposición a los componentes físicos del sistema, llamados hardware.

Tales componentes lógicos incluyen, entre muchos otros, aplicaciones informáticas como el procesador de textos, que permite al usuario realizar todas las tareas concernientes a la edición de textos o el software de sistema tal como el sistema operativo, que, básicamente, permite al resto de los programas funcionar adecuadamente, facilitando la interacción con los componentes físicos y el resto de las aplicaciones, proporcionando también una interfaz para el usuario."

WEB SITE, (2010). Software. Extraído el 07 de Noviembre del 2010 desde http://es.wikipedia.org/wiki/Software

2.3.2.1 Software de alta calidad

"La visión trascendental, según la cual la calidad es algo que se puede reconocer pero no se puede definir.

La visión del usuario, para la cual la calidad es la adecuación al propósito.

La visión de manufactura, donde calidad es conformidad con la especificación.

La visión de producto, donde la calidad está vinculada a las características inherentes del producto.

La visión basada en valor según la cual la calidad depende de la cantidad de dinero que el usuario está dispuesto a pagar por el producto."

Autor: LAWRENCE, Shari (2002) Titulo: Ingeniería de Software Teoría y práctica

2.3.2.2 Modelos de proceso de software

"Algunos son prescripciones para la manera que debe avanzar el desarrollo del software, y otras son descripciones de la manera en que el desarrollo del software se hace en la realidad. En teoría las dos clases de modelo deberían ser similares o iguales, pero en la práctica no lo son.

Modelo en cascada.- Las etapas se presentan cayendo en cascada desde una etapa a la siguiente. Una etapa de desarrollo debe completarse antes de dar paso a la siguiente. El modelo en cascada presenta una visión muy clara de cómo suceden la etapas durante el desarrollo, y siguiere a los desarrolladores cual es la secuencia de eventos que pueden encontrar.

El modelo V.- Variación del modelo en cascada que demuestra cómo se relacionan las actividades de prueba con las de análisis y diseño. La codificación forma la punta de la V, con el análisis y diseño a la izquierda y la prueba y el mantenimiento a la derecha. Este modelo sugiere que la prueba unitaria y de integración sea utilizada para verificar el diseño del programa.

Método de prototipos.- Permite que todo el sistema, o algunas de sus partes se construyan rápidamente para comprender o aclara aspectos, tiene el mismo objetivo que un prototipo de ingeniería, donde los requerimientos o el diseño requieren la información repetida para asegurar que el desarrollador, el usuario y el cliente tengan una comprensión unificada tanto de lo que se necesita como de lo que se propone como solución.

Especificación operacional.- En este modelo los requerimientos del sistema son evaluados o ejecutados en una forma que demuestra el comportamiento del sistema. De esta manera, una vez que los requerimientos están especificados, pueden implementarse utilizando un paquete de software de modo que sus implicancias pueden ser avaluadas antes de que comience el diseño.

Modelo de transformación.- Este intenta reducir las oportunidades de error por medio de la eliminación de varios de los pasos de desarrollo. Usando un soporte automatizado, el proceso de transformación aplica una serie de transformaciones para convertir una especificación en un sistema implementable.

Desarrollo por fases: incrementos e iteraciones.- El sistema se diseña de modo que puede ser entregado en piezas, lo que permite que los usuarios dispongan de cierta funcionalidad mientras el resto del sistema está siendo desarrollado. El sistema en producción es aquel que está siendo utilizado actualmente por el cliente; el sistema en desarrollo es la siguiente versión, que está siendo preparada para reemplazar al sistema en producción actual.

El modelo en espiral.- Este combina las actividades del desarrollo con la gestión de riesgo, comenzando con los requerimientos y un plan inicial de desarrollo, el proceso inserta un paso para evaluar riesgos y construir prototipos de las alternativas, antes de escribir el concepto de las operaciones en un documento que describe el más alto nivel como debe trabajar el sistema."

Autor: LAWRENCE, Shari (2002) Titulo: Ingeniería de Software Teoría y práctica

2.3.3 Diseño del sistema

"Es el proceso creativo de transformación del problema a una solución; la descripción de una solución también se denomina diseño.

Diseño conceptual y diseño técnico

Para transformar los requerimientos en un sistema que funcione, los diseñadores deben satisfacer tanto a los clientes como a los constructores de sistemas de su equipo de desarrollo. Primero se elabora un diseño conceptual o diseño del sistema que le dice al cliente, exactamente, qué hará dicho sistema, una vez que el cliente aprueba el diseño

conceptual se vuelca este diseño en un documento mucho más detallado, el diseño técnico, que permite que los constructores comprendan el hardware y software concretos que se necesitan para resolver el problema del cliente. Ambos documentos del sistema describen el mismo sistema, pero en formas diferentes de acuerdo con la audiencia a cual está destinado cada uno de ellos. En consecuencia, el diseño conceptual se concentra en las funciones del sistema, y el diseño técnico describe la forma que tomara el sistema."

Autor: LAWRENCE, Shari (2002) Titulo: Ingeniería de Software Teoría y práctica

2.3.4 Sistema Automatizado

"La automatización es un sistema donde se trasfieren tareas de producción, realizadas habitualmente por operadores humanos a un conjunto de elementos tecnológicos.

Un sistema automatizado consta de dos partes principales:

- Parte de Mando
- Parte Operativa

La Parte Operativa es la parte que actúa directamente sobre la máquina. Son los elementos que hacen que la máquina se mueva y realice la operación deseada. Los elementos que forman la parte operativa son los accionadores de las máquinas como motores, cilindros, compresores y los captadores como fotodiodos.

La Parte de Mando suele ser un autómata programable (tecnología programada), aunque hasta hace bien poco se utilizaban relés electromagnéticos, tarjetas electrónicas o módulos lógicos neumáticos (tecnología cableada). En un sistema de fabricación automatizado el autómata programable está en el centro del sistema. Este debe ser capaz de comunicarse con todos los constituyentes de sistema automatizado.

Objetivos de la automatización

- Mejorar la productividad de la empresa, reduciendo los costes de la producción y mejorando la calidad de la misma.
- Mejorar las condiciones de trabajo del personal, suprimiendo los trabajos penosos e

incrementando la seguridad.

- Realizar las operaciones imposibles de controlar intelectual o manualmente.
- Mejorar la disponibilidad de los productos, pudiendo proveer las cantidades necesarias en el momento preciso.
- Simplificar el mantenimiento de forma que el operario no requiera grandes conocimientos para la manipulación del proceso productivo.
- Integrar la gestión y producción."

WEB SITE, (2010). Automatización. Extraído el 07 de Noviembre del 2010 desde http://www.grupo-maser.com/PAG_Cursos/Auto/auto2/auto2/PAGINA%20

PRINCIPAL/Automatizacion/Automatizacion.htm

Categorías Fundamentales de la variable dependiente.

2.3.5 SharpDevelop

"Es un entorno de desarrollo integrado libre para los lenguajes de programación C#, Visual Basic .NET y Boo.Es usado típicamente por aquellos programadores de los citados lenguajes, que no desean o no pueden usar el entorno de desarrollo de Microsoft, el Microsoft Visual Studio. Hay disponible un port para Mono/Gtk#, llamado MonoDevelop, el cual funciona en otros sistemas operativos.

Para el completado automático de código, la aplicación incorpora sus propios parsers. La versión 1.1 de la aplicación puede importar proyectos de Visual Studio .NET. La versión 2.0 ya es capaz de editarlos directamente. La versión 3.0 integra soporte para python y f#.

Características principales

- Incorpora un diseñador de Windows Forms
- Completado de código. Soporta el uso de la combinación de teclas Ctrl + Espacio
- Depurador incorporado
- Herramientas para "Ir a Definición", "Encontrar referencias" y "renombrado"

- Títulos para títulos y para depuración
- Conversor bidireccional entre C# y Visual Basic .NET, y unidireccional hacia Boo
- Escrito enteramente en C#
- Compilación de código directamente dentro del entorno de desarrollo integrado
- Complementos para ILAsm y C++
- Integración con herramientas de pruebas unitarias NUnit y MbUnit
- Analizador para ensamblado FxCop
- Previsualización de documentación XML
- Gran integración con plantillas a la hora de añadir o crear ficheros, proyectos o compiladores
- Escritura de código C#, ASP.NET, ADO.NET, XML y HTML
- Coloreado de sintaxis para los lenguajes C#, HTML, ASP, ASP.NET, VBScript, Visual Basic .NET, y XML
- Llaves inteligentes en la escritura de código
- Gestión de marcadores (favoritos)
- Soporte para plantillas de código
- Extensible mediante herramientas externas, o complementos"

WEB SITE, (2010). SharpDevelop a free .NET. Extraído el 07 de Noviembre del 2010 desde http://www.willydev.net/descargas/WillyDEV_SharpDevelop.Pdf

2.3.6 Base de Datos

"Una **base de datos** o **banco de datos** (en ocasiones abreviada BB.DD.) es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso. En este sentido, una biblioteca puede considerarse una base de datos compuesta en su mayoría por documentos y textos impresos en papel e indexados para su consulta. 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.

Existen programas denominados sistemas gestores de bases de datos, abreviados SGBD, que permiten almacenar y posteriormente acceder a los datos de forma rápida y

estructurada. Las propiedades de estos SGBD, así como su utilización y administración, se estudian dentro del ámbito de la informática.

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.

Aunque las bases de datos pueden contener muchos tipos de datos, algunos de ellos se encuentran protegidos por las leyes de varios países. Por ejemplo, en España los datos personales se encuentran protegidos por la Ley Orgánica de Protección de Datos de Carácter Personal (LOPD).”

WEB SITE, (2010). Base de Datos. Extraído el 07 de Noviembre del 2010 desde http://es.wikipedia.org/wiki/Base_de_datos

2.3.6.1 Tipos de Base de Datos

“Las bases de datos pueden clasificarse de varias maneras, de acuerdo al contexto que se esté manejando, o la utilidad de la misma:

Según la variabilidad de los datos almacenados

Bases de datos dinámicas

Éstas son bases de datos donde la información almacenada se modifica con el tiempo, permitiendo operaciones como actualización, borrado y adición de datos, además de las operaciones fundamentales de consulta. Un ejemplo de esto puede ser la base de datos utilizada en un sistema de información de una tienda de abarrotes, una farmacia, un videoclub.

WEB SITE, (2010). Base de Datos. Extraído el 07 de Noviembre del 2010 desde http://es.wikipedia.org/wiki/Base_de_datos

2.3.6.2 Modelos de Base de Datos

Un modelo de datos es básicamente una "descripción" de algo conocido como contenedor de datos (algo en donde se guarda la información), así como de los métodos para almacenar y recuperar información de esos contenedores. Los modelos de datos no son cosas físicas: son abstracciones que permiten la implementación de un sistema eficiente de base de datos; por lo general se refieren a algoritmos, y conceptos matemáticos.

Algunos modelos con frecuencia utilizados en las bases de datos:

Base 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 (posibilidad no permitida en el modelo jerárquico).

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.

Bases de datos transaccionales

Son bases de datos cuyo único fin es el envío y recepción de datos a grandes velocidades, estas bases son muy poco comunes y están dirigidas por lo general al entorno de análisis de calidad, datos de producción e industrial, es importante entender que su fin único es recolectar y recuperar los datos a la mayor velocidad posible, por lo tanto la redundancia y duplicación de información no es un problema como con las demás bases de datos, por lo general para poderlas aprovechar al máximo permiten algún tipo de conectividad a bases de datos relacionales.

Bases de datos relacionales

Éste es el modelo utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente. Tras ser postulados sus fundamentos en 1970 por Edgar Frank Codd, de los laboratorios IBM en San José (California), no tardó en consolidarse como

un nuevo paradigma en los modelos de base de datos. Su idea fundamental es el uso de "relaciones". Estas relaciones podrían considerarse en forma lógica como conjuntos de datos llamados "tuplas". Pese a que ésta es la teoría de las bases de datos relacionales creadas por Codd, la mayoría de las veces se conceptualiza de una manera más fácil de imaginar. Esto es pensando en cada relación como si fuese una tabla que está compuesta por registros (las filas de una tabla), que representarían las tuplas, y campos (las columnas de una tabla).

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, StructuredQueryLanguage o Lenguaje Estructurado de Consultas, un estándar implementado por los principales motores o sistemas de gestión de bases de datos relacionales.

Durante su diseño, una base de datos relacional pasa por un proceso al que se le conoce como normalización de una base de datos.

Durante los años 80 la aparición de dBASE produjo una revolución en los lenguajes de programación y sistemas de administración de datos. Aunque nunca debe olvidarse que dBase no utilizaba SQL como lenguaje base para su gestión.”

WEB SITE, (2010). Base de Datos. Extraído el 07 de Noviembre del 2010 desde http://es.wikipedia.org/wiki/Base_de_datos

2.3.7 PostgreSQL

“PostgreSQL es un sistema de gestión de base de datos relacional orientada a objetos y libre, publicado bajo la licencia BSD.

Como muchos otros proyectos de código abierto, el desarrollo de PostgreSQL no es manejado por una sola empresa sino que es dirigido por una comunidad de

desarrolladores y organizaciones comerciales las cuales trabajan en su desarrollo. Dicha comunidad es denominada el PGDG (*PostgreSQL Global DevelopmentGroup*).

Historia

PostgreSQL ha tenido una larga evolución, la cual se inicia en 1982 con el proyecto Ingres en la Universidad de Berkeley. Este proyecto, liderado por Michael Stonebraker, fue uno de los primeros intentos en implementar un motor de base de datos relacional. Después de haber trabajado un largo tiempo en Ingres y de haber tenido una experiencia comercial con él mismo, Michael decidió volver a la Universidad en 1985 para trabajar en un nuevo proyecto sobre la experiencia de Ingres, dicho proyecto fue llamado post-ingres o simplemente POSTGRES.

El proyecto post-ingres pretendía resolver los problemas con el modelo de base de datos relacional que habían sido aclarados a comienzos de los años 1980. El principal de estos problemas era la incapacidad del modelo relacional de comprender "tipos", es decir, combinaciones de datos simples que conforman una única unidad. Actualmente estos son llamados objetos. Se esforzaron en introducir la menor cantidad posible de funcionalidades para completar el soporte de tipos. Estas funcionalidades incluían la habilidad de definir tipos, pero también la habilidad de describir relaciones - las cuales hasta ese momento eran ampliamente utilizadas pero mantenidas completamente por el usuario. En Postgres la base de datos «comprendía» las relaciones y podía obtener información de tablas relacionadas utilizando reglas. Postgres usó muchas ideas de Ingres pero no su código.

La siguiente lista muestra los hitos más importantes en la vida del proyecto Postgres.

- 1986: se publicaron varios papers que describían las bases del sistema.
- 1988: ya se contaba con una versión utilizable.
- 1989: el grupo publicaba la versión 1 para una pequeña comunidad de usuarios.
- 1990: se publicaba la versión 2 la cual tenía prácticamente reescrito el sistema de reglas.
- 1991: publicación de la versión 3, esta añadía la capacidad de múltiples motores de almacenamiento.

- 1993: crecimiento importante de la comunidad de usuarios, la cual demandaba más características.
- 1994: después de la publicación de la versión 4, el proyecto terminó y el grupo se disolvió.

Después de que el proyecto POSTGRES' terminara, dos graduados de la universidad, Andrew Yu y JollyChen, comenzaron a trabajar sobre el código de POSTGRES, esto fue posible dado que POSTGRES estaba licenciado bajo la BSD, y lo primero que hicieron fue añadir soporte para el lenguaje SQL a POSTGRES, dado que anteriormente contaba con un intérprete del lenguaje de consultas QUEL (basado en Ingres), creando así el sistema al cual denominaron Postgres95.

Para el año 1996 se unieron al proyecto personas ajenas a la Universidad como Marc Fournier de Hub.Org NetworkingServices, Bruce Momjian y Vadim B. Mikheev quienes proporcionaron el primer servidor de desarrollo no universitario para el esfuerzo de desarrollo de código abierto y comenzaron a trabajar para estabilizar el código de Postgres95.

En el año 1996 decidieron cambiar el nombre de Postgres95 de tal modo que refleje la característica del lenguaje SQL y lo terminaron llamando PostgreSQL, cuya primera versión de código abierto fue lanzada el 1 de agosto de 1996. La primera versión formal de PostgreSQL (6.0) fue liberada en enero de 1997. Desde entonces, muchos desarrolladores entusiastas de los motores de base de datos se unieron al proyecto, coordinaron vía Internet y entre todos comenzaron a incorporar muchas características al motor.

Aunque la licencia permitía la comercialización de PostgreSQL, el código no se desarrolló en principio con fines comerciales, algo sorprendente considerando las ventajas que PostgreSQL ofrecía. La principal derivación se originó cuando Paula Hawthtorn (un miembro del equipo original de Ingres que se pasó a Postgres) y Michael Stonebraker conformaron IllustraInformation Technologies para comercializar Postgres.

En 2000, ex inversionistas de Red Hat crearon la empresa Great Bridge para comercializar PostgreSQL y competir contra proveedores comerciales de bases de datos. Great Bridge auspició a varios desarrolladores de PostgreSQL y donó recursos de

vuelta a la comunidad, pero a fines de 2001 cerró debido a la dura competencia de compañías como Red Hat y pobres condiciones del mercado.

En 2001, CommandPrompt, Inc. lanzó Mammonth PostgreSQL, la más antigua distribución comercial de PostgreSQL. Continúa brindando soporte a la comunidad PostgreSQL a través del auspicio de desarrolladores y proyectos, incluyendo PL/Perl, PL/php y el alojamiento de proyectos de comunidades como PostgreSQL Build Farm.

En enero de 2005, PostgreSQL recibió apoyo del proveedor de base de datos Pervasive Software, conocido por su producto Btrieve que se utilizaba en la plataforma Novell Netware, Pervasive anunció soporte comercial y participación comunitaria y logró algo de éxito. Sin embargo, en julio de 2006 dejó el mercado de soporte de PostgreSQL.

A mediados de 2005 otras dos compañías anunciaron planes para comercializar PostgreSQL con énfasis en nichos separados de mercados. EnterpriseDB añadió funcionalidades que le permitían a las aplicaciones escritas para trabajar con Oracle ser más fáciles de ejecutar con PostgreSQL. Greenplum contribuyó mejoras directamente orientadas a aplicaciones de Data Warehouse e Inteligencia de negocios, incluyendo el proyecto BizGres.

En octubre de 2005, John Loiacono, vicepresidente ejecutivo de software en San Microsystems comentó: "No estamos yendo tras el OEM de Microsoft pero estamos viendo a PostgreSQL ahora", aunque no se dieron especificaciones en ese momento. Para noviembre de 2005, San Solares 10 (lanzamiento 6/06) incluía PostgreSQL.

En agosto de 2007 EnterpriseDB anunció el Postgres Rezurce Center y EnterpriseDB Postgres, diseñados para ser una completamente configurada distribución de PostgreSQL incluyendo muchos módulos contribuidos y agregados. EnterpriseDB Postgres fue renombrado Postgres Plus en marzo de 2008.

El proyecto PostgreSQL continúa haciendo lanzamientos principales anualmente y lanzamientos menores de reparación de bugs, todos disponibles bajo la licencia BSD, y basados en contribuciones de proveedores comerciales, empresas aportantes y programadores de código abierto mayormente.

Características

Algunas de sus principales características son, entre otras:

Alta concurrencia

Mediante un sistema denominado MVCC (Acceso concurrente multiversión, por sus siglas en inglés) PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Cada usuario obtiene una visión consistente de lo último a lo que se le hizo commit. Esta estrategia es superior al uso de bloqueos por tabla o por filas común en otras bases, eliminando la necesidad del uso de bloqueos explícitos.

Amplia variedad de tipos nativos

PostgreSQL provee nativamente soporte para:

- Números de precisión arbitraria.
- Texto de largo ilimitado.
- Figuras geométricas (con una variedad de funciones asociadas)
- Direcciones IP (IPv4 e IPv6).
- Bloques de direcciones estilo CIDR.
- Direcciones MAC.
- Arrays.

Adicionalmente los usuarios pueden crear sus propios tipos de datos, los que pueden ser por completo indexables gracias a la infraestructura GiST de PostgreSQL. Algunos ejemplos son los tipos de datos GIS creados por el proyecto PostGIS.

Otras características

- Claves ajenas también denominadas Llaves ajenas o Claves Foráneas (foreignkeys).
- Disparadores (triggers): Un disparador o trigger se define en una acción específica basada en algo ocurrente dentro de la base de datos. En PostgreSQL esto significa la ejecución de un procedimiento almacenado basado en una

determinada acción sobre una tabla específica. Ahora todos los disparadores se definen por seis características:

- El nombre del disparador o trigger
- El momento en que el disparador debe arrancar
- El evento del disparador deberá activarse sobre...
- La tabla donde el disparador se activará
- La frecuencia de la ejecución
- La función que podría ser llamada

Entonces combinando estas seis características, PostgreSQL le permitirá crear una amplia funcionalidad a través de su sistema de activación de disparadores (triggers).

- Vistas.
- Integridad transaccional.
- Herencia de tablas.
- Tipos de datos y operaciones geométricas.
- Soporte para transacciones distribuidas. Permite a PostgreSQL integrase en un sistema distribuido formado por varios recursos (p.ej, una base de datos PostgreSQL, otra Oracle, una cola de mensajes IBM MQ JMS y un ERP SAP) gestionado por un servidor de aplicaciones donde el éxito ("commit") de la transacción goblal es el resultado del éxito de las transacciones locales.

Funciones

Bloques de código que se ejecutan en el servidor. Pueden ser escritos en varios lenguajes, con la potencia que cada uno de ellos da, desde las operaciones básicas de programación, tales como bifurcaciones y bucles, hasta las complejidades de la programación orientada a objetos o la programación funcional.Los disparadores (triggers en inglés) son funciones enlazadas a operaciones sobre los datos.

Algunos de los lenguajes que se pueden usar son los siguientes:

- Un lenguaje propio llamado PL/PgSQL (similar al PL/SQL de oracle).
- C.
- C++.
- Java PL/Java web.

- PL/Perl.
- plPHP.
- PL/Python.
- PL/Ruby.
- PL/sh.
- PL/Tcl.
- PL/Scheme.
- Lenguaje para aplicaciones estadísticas R por medio de PL/R.

PostgreSQL soporta funciones que retornan "filas", donde la salida puede tratarse como un conjunto de valores que pueden ser tratados igual a una fila retornada por una consulta (query en inglés).

Las funciones pueden ser definidas para ejecutarse con los derechos del usuario ejecutor o con los derechos de un usuario previamente definido. El concepto de funciones, en otros DBMS, son muchas veces referidas como "procedimientos almacenados" (stored procedures en inglés).”

WEB SITE, (2010). PostgreSQL. Extraído el 07 de Noviembre del 2010 desde http://es.wikipedia.org/wiki/PostgreSQL

2.3.8 Registro y Sanciones de los jugadores

“En cuanto a las sanciones a los jugadores que competen a la Comisión Disciplinaria de la Federación Ecuatoriana de Fútbol, están vienen por dos vías, las propias del fútbol, es decir las que se producen como consecuencia del juego, en el partido, dentro de las acciones u omisiones que constituyan infracción que este profesional del deporte realiza en el ejercicio de su profesión y las faltas que pueden darse fuera del campo de juego que envuelvan el comportamiento ético y deportivo del jugador, fuera de las canchas, en situaciones tales como: el cometimiento de actos que afectaren gravemente la cultura deportiva del país y sean lesivos al prestigio del deporte nacional; el doping o uso indebido de sustancias prohibidas para los deportistas; el hecho de negarse a integrar las selecciones nacionales o provinciales; el abandonar un partido de fútbol oficial, se entiende contra la voluntad de los dirigentes; el ofrecer o recibir incentivos o recompensas ilegitimas en numerario o en especie; el prestarse para suplantar a otro; el

inscribirse con documentos adulterados que induzcan al engaño sobre la edad; el que un jugador actúe sin el consentimiento de su club en otro; el que se ofenda de palabra u obra o se impute hechos falsos a los dirigentes u organismos nacionales o provinciales del fútbol algunas más."

WEB SITE, (2010). Sanciones a Jugadores. Extraído el 07 de Noviembre del 2010 desde http://www.derechoecuador.com/index.php?option=com_content&task=view&id=2897&Itemid=426

2.4 Hipótesis

La implantación de un sistema automatizado optimizara el registro y las sanciones de los jugadores de la Liga Deportiva Parroquial Huachi Grande.

2.5 Variables

2.5.1 Variable Independiente

Sistema automatizado

2.5.2 Variable Dependiente

Registro y sanciones de los jugadores de la Liga Deportiva Parroquial Huachi Grande.

CAPITULO III

METODOLOGÍA

3.1 Enfoque

El enfoque de la investigación es eminentemente cuantitativa ya que el investigador conoce, analiza y toma las decisiones más óptimas con las técnicas adecuadas para solucionar el problema y la institución nos proporciona la información necesaria; es importante conocer la información que proporcione la institución respecto al problema. La investigación dentro de la institución fue realizada por el técnico para solucionar el problema que fue proporcionado por la población interesada en solucionarlo.

3.2 Modalidad básica de la Investigación

3.2.1 Investigación de Campo

Esta investigación permite el estudio sistemático de los hechos en el lugar en que se producen los acontecimientos, el investigador toma contacto en forma directa con la realidad, para temer informes de acuerdo con los objetivos del problema.

El Lugar donde se realizo la investigación de campo en en el centro de la parroquia Huachi Grande y se enmarco a la Liga Deportiva Parroquial del Lugar.

3.2.2 Investigación Documental – Bibliográfica

Esta modalidad permite conocer, comparar, ampliar, profundizar y deducir diferentes enfoques, teorías, conceptualizaciones y criterios de diversos autores para el diseño del presente programa para dar control de carnets, basándose en documentos (fuentes primarias), libros así también como en Internet (fuentes segundarias) que se recomienda para estudios sociales, geográficos, históricos, geopolíticos, literarios, entre otros.

Cabe Recalcar que no se encontró ningún trabajo parecido en la Facultad de Ingenieria en Sistemas Electronica e Industrial o en ninguna universidad del país, pero existe un sistema similar al presente trabajo.

3.2.3 Proyecto Factible

Se realizará una propuesta en base al desarrollo de un sistema para el registro y sanciones de los jugadores, que es un modelo practico que permitirá solucionar los problemas detectados en la Liga Deportiva Parroquial Huachi Grande, previo el diagnóstico realizado con anterioridad al transcurso de la investigación y sustentación en el marco teórico.

3.3 Niveles o Tipos de Investigación

La investigación utilizará un nivel exploratorio para detectar las características del problema, determinar si es factible o no solucionarse; se procedera al nivel descriptivo para conocer con mayor profundidad las circunstancias y la realidad en la que se desarrolla el problema dentro de la institución; el nivel correlacional, facilita la comprensión, el análisis y el estudio del fenómeno dentro de un contexto determinado para la construcción de la base teórica de las variables.

3.4 Población y Muestra

3.4.1 Población

La población que será utilizada para la investigación está conformada por las personas que forman la directiva.

3.4.2 Muestra

Como la población es pequeña se tomará como muestra todo el universo.

3.5 Operacionalización de variables

Conceptualización	Categorías	Indicadores	Ítems	Tec-Inst
Variable Independiente: Sistema automatizado La automatización es un sistema donde se trasfieren			¿Sabe que es un sistema automatizado? Si ☐ No ☐ ¿Cree usted que con el sistema automatizado se	Encuestas Encuestas

tareas de producción, realizadas habitualmente por operadores humanos a un conjunto de elementos tecnológicos.			optimice tiempo? Si ☐ No ☐ ¿Con la automatización de los procesos se realizara un mejor control de los jugadores? Si ☐ No ☐ ¿Con la implantación del sistema la Liga tendrá mejores ingresos y salida de datos? Si ☐ No ☐	 Encuestas Encuestas
Conceptualización	Categorías	Indicadores	Ítems	Tec-Inst
Variable Dependiente: Registro y sanción de los jugadores en la L.D.P.H.G.			¿Con el registro de los jugadores podrán realizar un mejor control? Si ☐ No ☐ ¿Con la implantación del sistema podrán emitir las sanciones a los jugadores de manera eficaz? Si ☐ No ☐	Encuestas Encuestas

3.6 Recolección de información

3.6.1 Plan de recolección de información

La recolección de Información se realizará mediante encuestas realizadas a los miembros de la directiva de la Liga Deportiva Parroquial Huachi Grande.

3.7 Procesamiento y análisis

3.7.1 Plan de procesamiento de información

El análisis de la información se realizará mediante la interpretación de los datos recolectados, los cuales al ser procesados permitirá obtener un informe en base a sus

resultados, el cual permitirá plantear conclusiones y recomendaciones para dar solución al problema trazado.

3.7.2 Plan de análisis e interpretación de los resultados

Los resultados obtenidos a través de las encuestas realizadas a la directiva de la Liga Deportiva Parroquial Huachi Grande se los represento mediante la utilización de gráficos estadísticos, los cuales servirán para dar solución al problema planteado, por último se elaborará una síntesis general para la elaboración de las conclusiones y recomendaciones.

CAPITULO IV

ANÁLISIS E INTERPRETACIÓN DE RESULTADOS

4.1 Análisis de la necesidad

La Universidad Técnica de Ambato a través de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial en vinculación con la Liga Deportiva Parroquial Huachi Grande, se han enfocado en solucionar los problemas ocasionados por llevar en forma manual los registros de los jugadores e ingresos económicos recaudados por los vocales de mesa en los estadios, así como también la falta de un sistema de control de los valores recaudados causando inconsistencia, duplicidad y pérdida de información antes expuesta, por lo que se suscribió un convenio entre las dos instituciones para que un alumno de esta carrera aporte soluciones a través de la ejecución de un sistema informático.

La Liga Deportiva Parroquial Huachi Grande conjuntamente con el estudiante responsable han realizado reuniones para socializar con los vocales a sesión de cada institución filial a la Liga para obtener información sobre las necesidades y requerimientos de cada una de las mismas, información que servirá para desarrollar el sistema que les ayudara a administrar y controlar de una mejor manera los registros de los jugadores e ingresos recaudados.

Por tanto la Liga Deportiva Parroquial Huachi Grande requiere del desarrollo e implantación de un Sistema Automatizado para el Registro Y Sanciones de los Jugadores.

4.2 Análisis e interpretación de los resultados

Para la realización del análisis e interpretación de resultados se aplicó 50 encuestas a dirigentes de los diferentes clubes filiales y directivos de la Liga Deportiva Parroquial Huachi Grande.

4.2.1 Análisis de los resultados de las encuestas

1. ¿La información del jugador que se recopila por la Liga se encuentra almacenada en:?

Objetivo:

Investigar la forma en que se recopila la información del jugador y valores recaudados en la Liga Deportiva Parroquial Huachi Grande.

RESPUESTA	CANTIDAD	PORCENTAJE
Archivos físicos	40	80%
Archivos digitales	4	8%
De ambas formas	2	4%
Ninguna	4	8%
Total	**50**	**100%**

Tabla 4.1 Cuadro porcentual pregunta 1

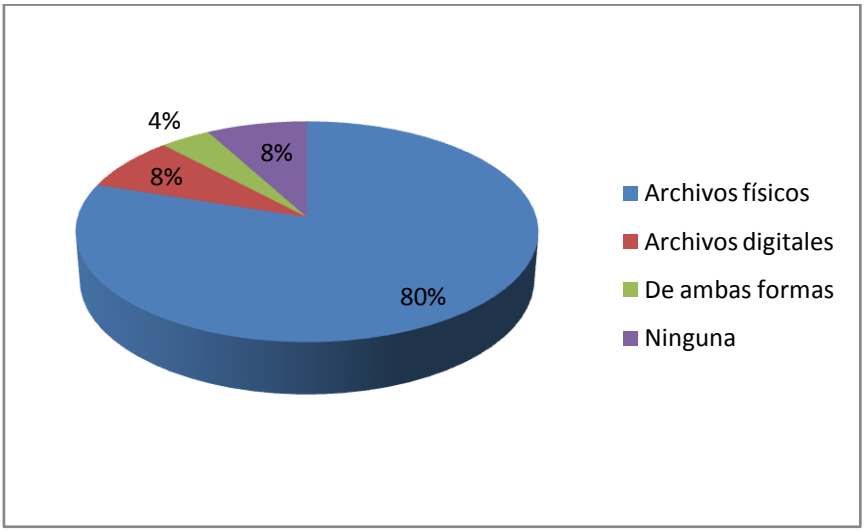


Figura 4.1 Gráfico pregunta 1

Interpretación: La gráfica indica que el 80% de los dirigentes encuestados respondieron que la información se recopila en archivos físicos, el 8% contestaron que guardan la información en archivos digitales, el 4% manifiesta que conservan la información en archivos físicos y digitales y el 8% no utiliza ninguno de estos medios para guardar la información referente a la información del jugador.

Análisis: Se demuestra que en la Liga Deportiva Parroquial Huachi Grande guardan la información del Jugador y valores recaudados en archivos físicos.

2. ¿Sabe que es un sistema automatizado?

Objetivo:

Verificar si los dirigentes de los clubes filiales a la Liga y Directivos de la Liga Deportiva Parroquial Huachi Grande tienen conocimiento de lo q significa un sistema automatizado.

RESPUESTA	CANTIDAD	PORCENTAJE
Si	9	18%
No	41	82%
Total	**50**	**100%**

Tabla 4.2 Cuadro porcentual pregunta 2

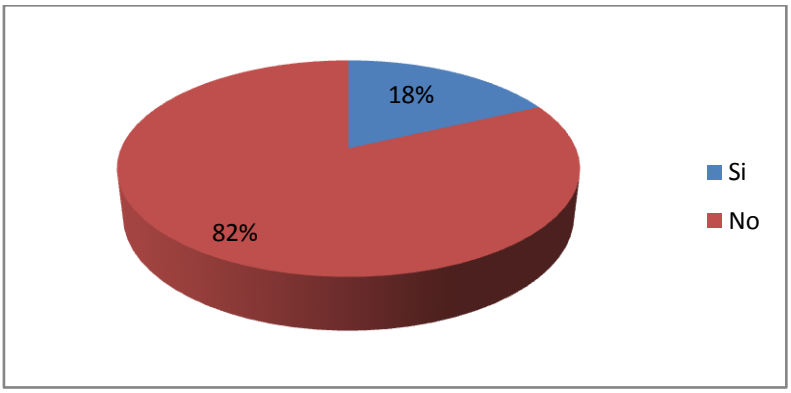


Figura 4.2 Gráfico pregunta 2

Interpretación: La gráfica refleja que el 82% de los dirigentes y directivos encuestados respondieron que no saben que es un sistema automatizado, el 18% manifiesta que si conocen el significado de un sistema automatizado.

Análisis: En la Liga deportiva Parroquial Huachi Grande desconocen el significado de lo que significa un sistema automatizado en su gran mayoría.

3. ¿Cree usted que con el sistema automatizado se optimice tiempo?

Objetivo:

Determinar si los dirigentes de los clubes filiales a la Liga y Directivos de la Liga Deportiva Parroquial Huachi Grande si implantando el sistema facilitarían su trabajo.

RESPUESTA	CANTIDAD	PORCENTAJE
Si	45	90%
No	5	10%
Total	**50**	**100%**

Tabla 4.3 Cuadro porcentual pregunta 3

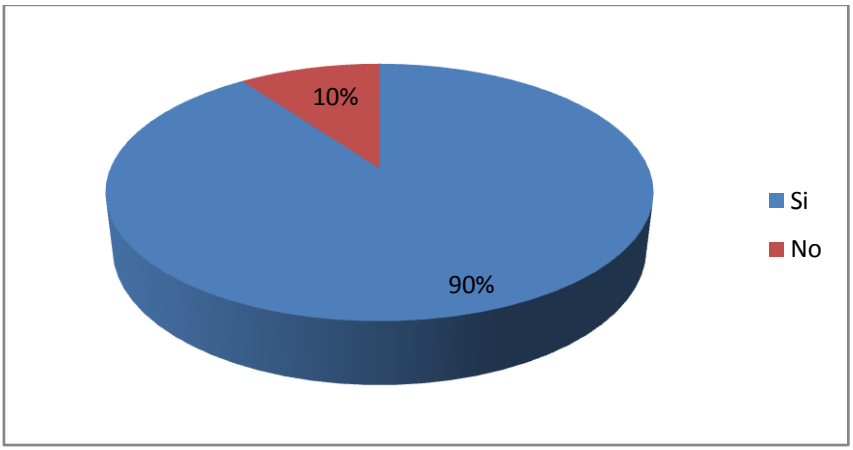


Figura 4.3 Gráfico pregunta 3

Interpretación: La gráfica refleja que el 90% de los dirigentes y directivos encuestados respondieron que si se optimizaría tiempo un sistema automatizado para el registro de los jugadores, mientras que el 10% manifiesta que no es optimizaría tiempo un sistema automatizado.

Análisis: En la Liga deportiva Parroquial Huachi Grande están convencidos que un sistema automatizado si optimizaría el tiempo.

4. ¿Cree usted que con la automatización de los procesos se realizará un mejor control de los jugadores?

Objetivo:

Determinar si la Liga Deportiva Parroquial Huachi Grande mejorara el control de los registros de los jugadores.

RESPUESTA	CANTIDAD	PORCENTAJE
Si	48	96%
No	2	4%
Total	**50**	**100%**

Tabla 4.4 Cuadro porcentual pregunta 4

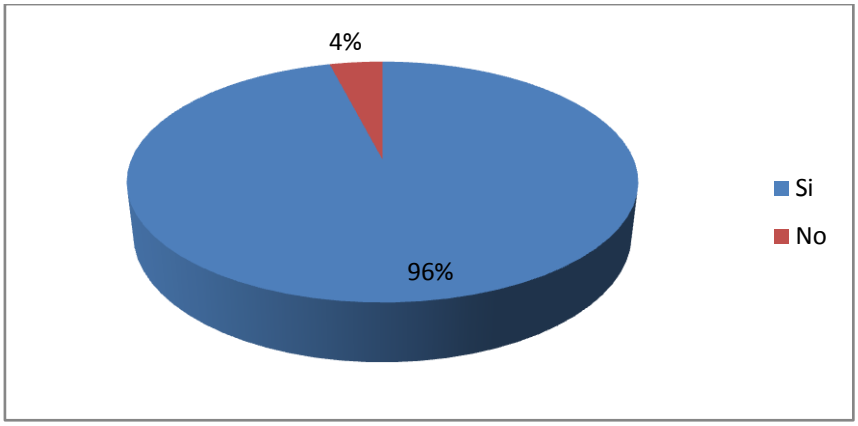


Figura 4.4 Gráfico pregunta 4

Interpretación: La gráfica refleja que el 96% de los dirigentes y directivos encuestados respondieron que si se realizara un mejor control para el registro de los jugadores, mientras que el 4% manifiesta que no se realizara un mejor control.

Análisis: En la Liga deportiva Parroquial Huachi Grande saben que es necesario llevar un mejor control de los registros de los jugadores y cobros de la Liga.

5. ¿Cree usted que con la implantación del sistema la Liga tendrá mejores ingresos y salida de datos?

Objetivo:

Investigar si la Liga Deportiva Parroquial Huachi Grande tendrá mejor fluidez y rapidez al momento de realizar una consulta de los registros.

RESPUESTA	CANTIDAD	PORCENTAJE
Si	43	86%
No	7	14%
Total	**50**	**100%**

Tabla 4.5 Cuadro porcentual pregunta 5

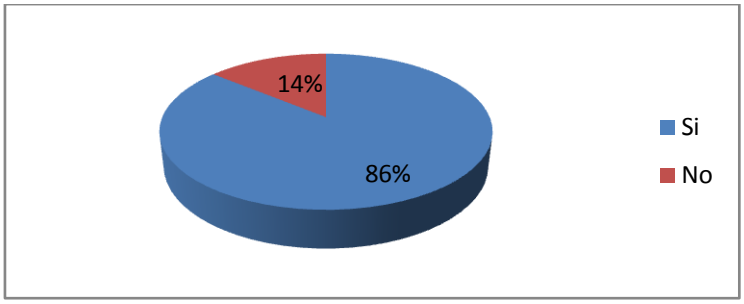


Figura 4.5 Gráfico pregunta 5

Interpretación: La gráfica refleja que el 86% de los dirigentes y directivos encuestados respondieron que si se tendrá en menor tiempo los datos consultados de los registros, mientras que el 14% manifiesta que no se tendrá en menos tiempo los datos a consultar.

Análisis: Mayoritariamente los encuestados manifiestan que si se obtendrán los datos consultados en menor tiempo real y confiable, Presumiblemente quienes manifiestan estar en contra de este tipo de implementación tecnológica sea porque no se han capacitado en el uso de un computador razón por la cual tengan resistencia a este tipo de propuestas.

6. ¿Cree usted que con el registro de los jugadores podrán realizar un mejor control?

Objetivo:

Conocer si la Liga Deportiva Parroquial Huachi Grande podrá controlar de manera eficaz los registros de las actas de juego.

RESPUESTA	CANTIDAD	PORCENTAJE
Si	47	94%
No	3	6%
Total	**50**	**100%**

Tabla 4.6 Cuadro porcentual pregunta 6

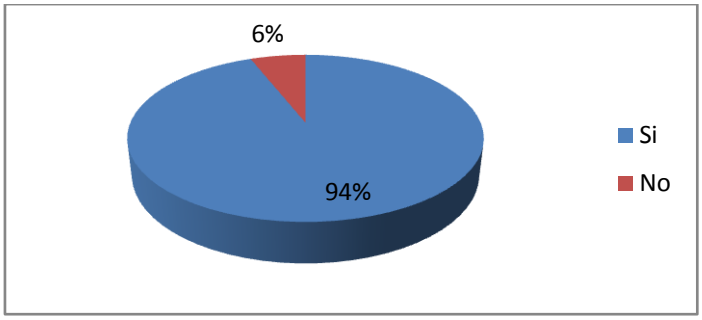


Figura 4.6 Gráfico pregunta 6

Interpretación: Se observa que el 94% de los encuestados manifiestan que si se realizara de una manera eficaz los registros de las actas, mientras que el 6% de los encuestados están de acuerdo como se lleva el registro de las actas en la actualidad.

Análisis: La mayoría de personas requieren mejorar la manera de como se llevan los archivos de las actas de juego que se emiten en los campos de juego.

7. ¿Cree usted que con la implantación del sistema podrán emitir las sanciones a los jugadores de manera eficaz?

Objetivo:

Determinar si la Liga Deportiva Parroquial Huachi Grande obtendrá un control eficiente de las sanciones impuestas a los jugadores impuestas en el partido de futbol o por la comisión de disciplina.

RESPUESTA	CANTIDAD	PORCENTAJE
Si	49	98%
No	1	2%
Total	**50**	**100%**

Tabla 4.7 Cuadro porcentual pregunta 7

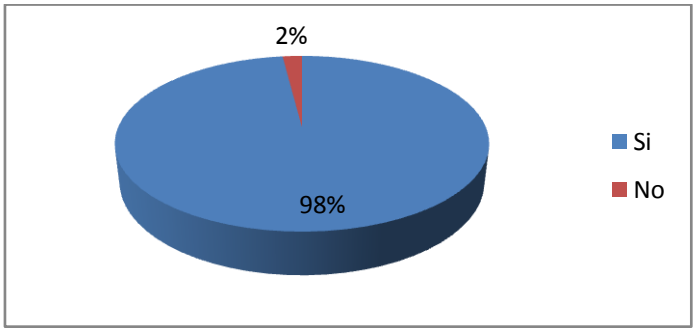


Figura 4.7 Gráfico pregunta 7

Interpretación: Se observa que el 98% de las personas manifiestan que si se realizara de una manera eficaz las sanciones impuestas a los jugadores, mientras que el 2% están de acuerdo en cómo están llevando los registros en la actualidad.

Análisis: En la gran mayoría de encuestados requieren que las sanciones impuestas a los jugadores se lo realice de una manera rápida y eficaz.

4.3 Verificación de la hipótesis

Luego de haber tabulado las encuestas se procede a la comprobación de la hipótesis, mediante el método estadístico:

Chi-cuadrado

$$x^2 = \sum \left(\frac{(O - E)^2}{E} \right)$$

En donde:

$\boldsymbol{x^2}$**=** Chi-cuadrado

$\sum$ = Sumatoria

O = Frecuencia Observada

E = Frecuencia esperada o técnica

4.3.1 Combinación de frecuencias

Nº	Pregunta	Si/Excelente/ Alto/Siempre	Muy Bueno/Medio /Ocasionalmente	Bueno	No/Nunca/ Ninguno/Malo/ Bajo	Total
2	¿Sabe que es un sistema automatizado?	9			41	50
3	¿Cree usted que con el sistema automatizado se optimice tiempo?	45			5	50
4	¿Con la automatización de los procesos se realizara un mejor control de los jugadores?	48			2	50
5	¿Con la implantación del sistema la Liga tendrá mejores ingresos y salida de datos?	43			7	50
6	¿Con el registro de los jugadores podrán realizar un mejor control?	47			3	50
7	¿Con la implantación del sistema podrán emitir las sanciones a los jugadores de manera eficaz?	49			1	50

Tabla 4.8 Combinación de frecuencias

4.3.2 Frecuencias esperadas

Nº	Pregunta	Si/Excelente/ Alto/Siempre	Muy Bueno/Medio /Ocasionalmente	Bueno	No/Nunca/ Ninguno/Malo/ Bajo	Total
2	¿Sabe que es un sistema automatizado?	22,36	3,43	2,07	22,14	50
3	¿Cree usted que con el sistema automatizado se optimice tiempo?	22,36	3,43	2,07	22,14	50
4	¿Con la automatización de los procesos se realizara un mejor control de los jugadores?	22,36	3,43	2,07	22,14	50
5	¿Con la implantación del sistema la Liga tendrá mejores ingresos y salida de datos?	22,36	3,43	2,07	22,14	50
6	¿Con el registro de los jugadores podrán realizar un mejor control?	22,36	3,43	2,07	22,14	50
7	¿Con la implantación del sistema podrán emitir las sanciones a los jugadores de manera eficaz?	22,36	3,43	2,07	22,14	50
TOTAL		134,16	20,58	12,42	132,84	300

Tabla 4.9 Frecuencias esperadas

4.3.4 Nivel de significancia y regla de decisión

4.3.4.1 Grado de libertad

GL = (c-1)*(f-1)
GL = (8-1)*(4-1)
GL = 7 * 3
GL = 21

4.3.4.2 Grado de significancia

Nivel de significación (P): Denominado nivel de confianza, se refiere a la probabilidad de que los resultados observados se deban al azar. Este valor es fijado por el investigador, usualmente es el 5% o 10%. Lo que indica que si se toma P=0.05, se está significando que solo en un 5% de las veces en que se realice la medición, el resultado obtenido podría deberse al azar. De lo contrario sería que existe un nivel de confianza del 95% que el resultado es real y no debido a la casualidad.
Nivel de confiabilidad = 95%
El grado de significancia será 0.05

Valores críticos de chi-cuadrado

Esta tabla contiene los valores x^2 que corresponden a un área específica de la extremidad de la derecha y a un número determinado de grados de libertad.

v/p	0,001	0,0025	0,005	0,01	0,025	0,05	0,1
1	10,8274	9,1404	7,8794	6,6349	5,0239	3,8415	2,7055
2	13,8150	11,9827	10,5965	9,2104	7,3778	5,9915	4,6052
3	16,2660	14,3202	12,8381	11,3449	9,3484	7,8147	6,2514
4	18,4662	16,4238	14,8602	13,2767	11,1433	9,4877	7,7794
5	20,5147	18,3854	16,7496	15,0863	12,8325	11,0705	9,2363
6	22,4575	20,2491	18,5475	16,8119	14,4494	12,5916	10,6446
7	24,3213	22,0402	20,2777	18,4753	16,0128	14,0671	12,0170
8	26,1239	23,7742	21,9549	20,0902	17,5345	15,5073	13,3616
9	27,8767	25,4625	23,5893	21,6660	19,0228	16,9190	14,6837
10	29,5879	27,1119	25,1881	23,2093	20,4832	18,3070	15,9872
11	31,2635	28,7291	26,7569	24,7250	21,9200	19,6752	17,2750
12	32,9092	30,3182	28,2997	26,2170	23,3367	21,0261	18,5493
13	34,5274	31,8830	29,8193	27,6882	24,7356	22,3620	19,8119
14	36,1239	33,4262	31,3194	29,1412	26,1189	23,6848	21,0641
15	37,6978	34,9494	32,8015	30,5780	27,4884	24,9958	22,3071
16	39,2518	36,4555	34,2671	31,9999	28,8453	26,2962	23,5418
17	40,7911	37,9462	35,7184	33,4087	30,1910	27,5871	24,7690
18	42,3119	39,4220	37,1564	34,8052	31,5264	28,8693	25,9894
19	43,8194	40,8847	38,5821	36,1908	32,8523	30,1435	27,2036
20	45,3142	42,3358	39,9969	37,5663	34,1696	[illegible]4104	28,4120
21	46,7963	43,7749	41,4009	38,9322	35,4[illegible]	32,670	29,6151

Tabla 4.10. Valores críticos de chi-cuadrado

$X_t^2(c-1)*(f-1)= 32,67$

4.3.5 Calculo del Chi-cuadrado

En donde:

O= Frecuencia observada

E= Frecuencia esperada

O-E= Frecuencias observada – frecuencias esperadas

(O-E) ^{2}= Resultado de las frecuencias observadas y esperadas al cuadrado

(O-E) ^{2}**/E** = Resultado de las frecuencias observadas y esperadas al cuadrado dividido para las frecuencias esperadas.

CALCULO DE CHI – CUADRADO

O	E	O-E	$(O-E)^2$	$(O-E)^2/E$
9	22,36	-13,36	178,4896	7,9825403
0	3,43	-3,43	11,7649	3,43
0	2,07	-2,07	4,2849	2,07
41	22,14	18,86	355,6996	16,065926
45	22,36	22,64	512,5696	22,923506
0	3,43	-3,43	11,7649	3,43
0	2,07	-2,07	4,2849	2,07
5	22,14	-17,14	293,7796	13,269178
48	22,36	25,64	657,4096	29,401145
0	3,43	-3,43	11,7649	3,43
0	2,07	-2,07	4,2849	2,07
2	22,14	-20,14	405,6196	18,320668
43	22,36	20,64	426,0096	19,052308
0	3,43	-3,43	11,7649	3,43
0	2,07	-2,07	4,2849	2,07
7	22,14	-15,14	229,2196	10,353189
47	22,36	24,64	607,1296	27,152487
0	3,43	-3,43	11,7649	3,43
0	2,07	-2,07	4,2849	2,07
3	22,14	-19,14	366,3396	16,546504
49	22,36	26,64	709,6896	31,739249
0	3,43	-3,43	11,7649	3,43
0	2,07	-2,07	4,2849	2,07
1	22,14	-21,14	446,8996	20,185167
			TOTAL:	**265,99187**

Tabla 4.11. Cálculo chi-cuadrado

$X^2 = 265,99187$

Xt2(c-1)*(f-1) = 32,67

Criterio de decisión:

X2 < Xt2(c-1)*(f-1) → Acepta Ho.

Valores de decisión:

1314,41>32,67 → Se rechaza Ho

Debido a que X2 es mayor a Xt2(c-1)*(f-1) se rechaza Ho y se acepta Ha.

Por lo tanto la realización de un Sistema Automatizado para el registro y sanciones de los jugadores en la Liga Deportiva Parroquial Huachi Grande es factible realizarlo.

CAPÍTULO V
CONCLUSIONES Y RECOMENDACIONES

5.1 Conclusiones

- En la actualidad no se realiza un control adecuado sobre los registros de las actas de juego que se realiza en el campo de juego y cobros de haberes en la Liga Deportiva Parroquial Huachi Grande.
- La implementación de un sistema de registro y sanciones buscará evitar las inconsistencias y duplicidad de la información al momento realizar consultas.
- SHARPDEVELOP es una herramienta de código abierto, que facilita el desarrollo de todos los requerimientos que necesitan en la Liga Deportiva Parroquial Huachi Grande.
- PHP es una herramienta de código abierto, que facilita con el desarrollo de todos los requerimientos que necesitan en la Liga Deportiva Parroquial Huachi Grande.
- PostgresSQL es una herramienta muy potente y gratuita que permite almacenar grandes cantidades de datos y permite su fácil recuperación.
- Debido a que el sistema se implementará en una entidad pública, requiere ser desarrollado mediante software libre por cual razón se escogió las herramientas antes mencionadas.
- Los requerimientos y necesidades recopiladas a los diferentes dirigentes de los clubes filiales y directivos de la Liga permitió que el sistema sea factible para la administración de la Liga Deportiva Parroquial Huachi Grande.
- El sistema de registro y sanciones permitirá optimizar los recursos y ayudará a la toma de decisiones para así obtener una adecuada administración de los recursos económicos y humanos.

- Al estar la información almacenada en una base de datos se facilita su recuperación, aumentará los niveles de seguridad y con información más confiable.

5.2 Recomendaciones

- Capacitar al personal sobre el uso de sistemas informáticos, para evitar así malos manejos y errores al navegar en el sitio Web y Windows.
- Se recomienda la investigación de herramientas para el desarrollo de software que sean bajo licencias libres, además de la investigación de gestores de base de datos que puedan almacenar gran cantidad de información.
- Continuar trabajando con los dirigentes de los clubes filiales y directivos de la Liga para que los diagnósticos de necesidades sean reales.
- Investigar sobre nuevas métodos de seguridad para aplicar en el sistema de registro y sanciones para que no sea vulnerable y manipulable por personas que no tengan permisos al sistema.

CAPÍTULO VI
PROPUESTA

6.1 Tema

SISTEMA AUTOMATIZADO PARA EL REGISTRO Y SANCIONES DE LOS JUGADORES EN LA LIGA DEPORTIVA PARROQUIAL HUACHI GRANDE.

6.2 Datos informativos

Institución Ejecutoriada:	Liga Deportiva Parroquial Huachi Grande.
Beneficiarios:	Liga Deportiva Parroquial Huachi Grande.
Ciudad:	Ambato.
Dirección:	Huachi Grande casa de la junta parroquial segundo piso.
Investigador:	Byron Bladimir Cárdenas Villacres.
Tiempo:	El presente proyecto será ejecutado entre el mes de Octubre del 2010 hasta octubre del 2011.
Tutor:	Ing. Eduardo Chaso.

6.3 Antecedentes

La Liga Deportiva Parroquial Huachi Grande en su gran mayoría han realizado los registros y sanciones de los jugadores manualmente sin apoyo de alguna herramienta informática que les facilite la obtención de mejores resultados para el control de los jugadores, causando así inconsistencias en la información hasta incluso llegando a crear malestar entre los usuarios. Esto se debe en gran medida a la falta de capacitación sobre las ventajas tecnológicas que existen en la actualidad, y por otra parte a la falta de visión de algunos líderes pues mucha de la responsabilidad recae en la decisión política de cambiar y mejorar las actuales administraciones de estos organismos, pues se requiere

de inversión para optimizar los recursos y dar respuestas ágiles y oportunas a las necesidades de los usuarios.

Considerando que la seguridad, fiabilidad y fácil recuperación de la información se diseñará la base de datos en el SGBD PostgresSQL, el desarrollo se lo realizará en SHARPDEVELOP y consultas web en PHP.

6.4 Justificación

En la actualidad las aplicaciones informáticas son innumerables, pues su crecimiento y desarrollo han marcado una nueva era en el mundo actual, gracias a la globalización la masificación tecnológica ha permitido la reducción de sus costos permitiendo a la gran mayoría de personas acceder a ellas.

Otro factor importante en la revolución tecnológica ha sido la masificación del internet, gracias a que los gobiernos han invertido en la infraestructura que permita a la gran mayoría acceder al servicio pues hace unos años atrás esto estaba reservado a pocas personas ya que se le consideraba como un lujo.

En definitiva un sistema informático optimizara el tiempo en los registros de los jugadores así como también al usuario, además permitirá obtener información de calidad para ejercer la responsabilidad, además, la elaboración de la presente propuesta es factible y viable para utilizarla como instrumento de cambio y mejoramiento para la entidad, a la vez que permitirá al investigador aplicar conocimientos adquiridos en el aprendizaje de toda la carrera universitaria.

6.5 Objetivos

6.5.1 Objetivo general

Desarrollar un Sistema de Registro y sanciones basados en Software Libre para el control automatizado de los jugadores en la Liga Deportiva Parroquial Huachi Grande.

6.5.2 Objetivos específicos

- Determinar las necesidades administrativas de la Liga Deportiva Parroquial Huachi Grande, para definir las características operacionales del sistema a desarrollarse.
- Diseñar una óptima base de datos para un acceso eficiente a la información.
- Desarrollar el sistema de registro y sanciones para llevar a cabo un control de los jugadores que se manejan en la Liga Deportiva Parroquial Huachi Grande.
- Proporcionar manuales de instalación y uso a los usuarios que utilizarán el sistema.

6.6 Análisis de factibilidad

6.6.1 Factibilidad operativa

Debido a que el Sistema fue desarrollado de acuerdo a las necesidades de los usuarios, este interactúa directamente con el usuario, por lo tanto no se requiere de una capacitación extensa para poder utilizarlo; además de esto cuenta con interfaces sencillas, amigables y fáciles de navegar donde podrán realizar todo el proceso de registro de los jugadores dando a conocer las sanciones impuestas a los jugadores; de esta forma a los directivos de la Liga se les hará más fácil obtener la información de registro para poder tomar decisiones rápidas y oportunas.

El sistema cuenta con los siguientes tipos de usuario:

Administrador
Secretario(a)
Usuario

Administrador.- Se implementará el acceso para la persona que será la encargada de crear y habilitar los usuarios como secretario(a) para la Liga Deportiva Parroquial Huachi Grande. Una vez logeado puede modificar su cuenta y crear más usuarios secretarios(as).

Secretario(a).- Se implementará el acceso para la persona que será la encargada de crear y habilitar los registros en la Liga Deportiva Parroquial Huachi Grande.

Usuario.- Se implementará el acceso a cualquier persona para que pueda realizar consultas básicas al sistema sobre los registros y calendarios de juego.

6.6.2 Factibilidad económica

Es factible el proyecto, ya que el sistema se lo desarrollará con herramientas de software libre, que no tienen ningún costo para la adquisición del lenguaje y SGBD seleccionados.

Con el desarrollo de este Sistema de registro y sanciones de los jugadores controlarán de una mejor manera sus recursos económicos, optimizando tiempo, recursos; teniendo así también una mejora en la atención a los clubes filiales a la Liga.

6.6.3 Factibilidad técnica

Para la creación del sistema de registro y sanciones de los jugadores se manejó SharpDevelop herramienta que nos permite configurar un entorno de desarrollo de aplicaciones windows, tiene herramientas necesarias para el desarrollo entre la que se utilizo fue el SGBD PostgreSQL, para el entorno web fue utilizado PHP con CSS.

Software

- Lenguaje de programación SharpDebelop
- Lenguaje de programación PHP.
- PostgreSQL como motor de base de datos.
- Wamp Server como Web Server.

Hardware

En la Liga Deportiva Parroquial Huachi Grande cuentan con computadores en los que actualmente se realiza el registro y sanción de los jugadores; y aunque estos equipos no son de última tecnología el sistema correrá perfectamente ya que este no cuenta con procesos pesados que sobrecarguen el uso de memoria.

6.7 Fundamentación

PHP (Hypertext Pre-Processor).

"PHP es un lenguaje de programación interpretado, diseñado para la creación de páginas web dinámicas. Es usado principalmente para la interpretación del lado del servidor (server-side scripting) pero actualmente puede ser utilizado desde una interfaz de línea de comandos o en la creación de otros tipos de programas incluyendo aplicaciones con interfaz gráfica."

WEB SITE, (2010). Php. Extraído el 01 de Septiembre del 2011 desde http://es.wikipedia.org/wiki/PHP

SharpDevelop

"SharpDevelop es un entorno de desarrollo integrado libre para los lenguajes de programación C#, Visual Basic .NET y Boo.

Es usado típicamente por aquellos programadores de los citados lenguajes, que no desean o no pueden usar el entorno de desarrollo de Microsoft, Microsoft Visual Studio. Hay disponible un port para Mono/Gtk#, llamado MonoDevelop, el cual funciona en otros sistemas operativos.

Para el completado automático de código, la aplicación incorpora sus propios analizadores sintácticos. La versión 1.1 de la aplicación puede importar proyectos de Visual Studio .NET. La versión 2.0 ya es capaz de editarlos directamente. La versión 3.0 integra soporte para los lenguajes de programación Python y F#."

WEB SITE, (2010). SharrpDelelop. Extraído el 01 de Septiembre del 2011 desde http://es.wikipedia.org/wiki/SharpDevelop

6.8 Metodología

En el presente proyecto se utilizó la metodología en cascada, mediante una secuencia de desarrollo dividiendo en fases iníciales, donde cada una debe cumplir sus objetivos y seguir en la secuencia de la otra fase; de esta forma, cualquier error de diseño detectado en la etapa de prueba conduce necesariamente al rediseño y nueva programación del código afectado.

Para el análisis del sistema se utilizó la herramienta UML (Lenguaje Unificado de Modelado) el cual es un conjunto de notificaciones y diagramas gráficos para modelar sistemas orientados a objetos; dando diferentes perspectivas a un sistema.

6.9 Modelo operativo

6.9.1 Análisis del Sistema

6.9.1.1 Análisis y Requerimientos del sistema

De los resultados obtenidos en las encuestas que se aplicaron a varios dirigentes y directivos de la Liga Deportiva Parroquial Huachi Grande en la provincia de Tungurahua se determinaron las necesidades de estos organismos.

Luego de las reuniones mantenidas entre los directivos de la Liga conjuntamente con los dirigentes de los clubes filiales a la Liga Deportiva Parroquial Huachi Grande han llegado a la conclusión de que se requiere un sistema de registro y sanciones de los jugadores para el adecuado control de los mismos.

Entre los requerimientos solicitados por los beneficiarios está la seguridad a la información, estableciendo restricciones de usuarios, delegando permisos a cada uno de ellos y se determino de acuerdo a las funciones que realiza cada funcionario.

Al momento de ingresar a la base de datos de la Liga, se establecerá los parámetros de inicio como son costo de arbitrajes.

Luego de haber analizado los requerimientos que solicita el departamento se ha determinado que el sistema deberá contar con:

- Un módulo para el ingreso, modificación, eliminación y control de: equipos, canchas, jugadores y actividades que pertenecen al proyecto.

- Un módulo de seguridad que permita el acceso a la información solo a personas autorizadas.

- Generación de reportes de: sanciones, calendario, tabla de posiciones y carnets de los jugadores.

- Además, el sistema deberá contar con interfaces sencillas y amigables para su manipulación.

6.9.1.2 Diagramas UML

Considerando que hoy en día, UML (Lenguaje unificado de modelado) está consolidado como el lenguaje estándar en el análisis y diseño de sistemas de cómputo, se ha procedido a utilizarlo con el fin de establecer requerimientos y estructuras necesarias para plasmar un sistema de software previo al proceso intensivo de escribir código.

6.9.1.2.1 Diagrama de casos de uso

Los diagramas de casos de uso documentan el comportamiento de un sistema desde el punto de vista del usuario. Por lo tanto los casos de uso determinan los requisitos funcionales del sistema, es decir, representan las funciones que un sistema debe ejecutar.

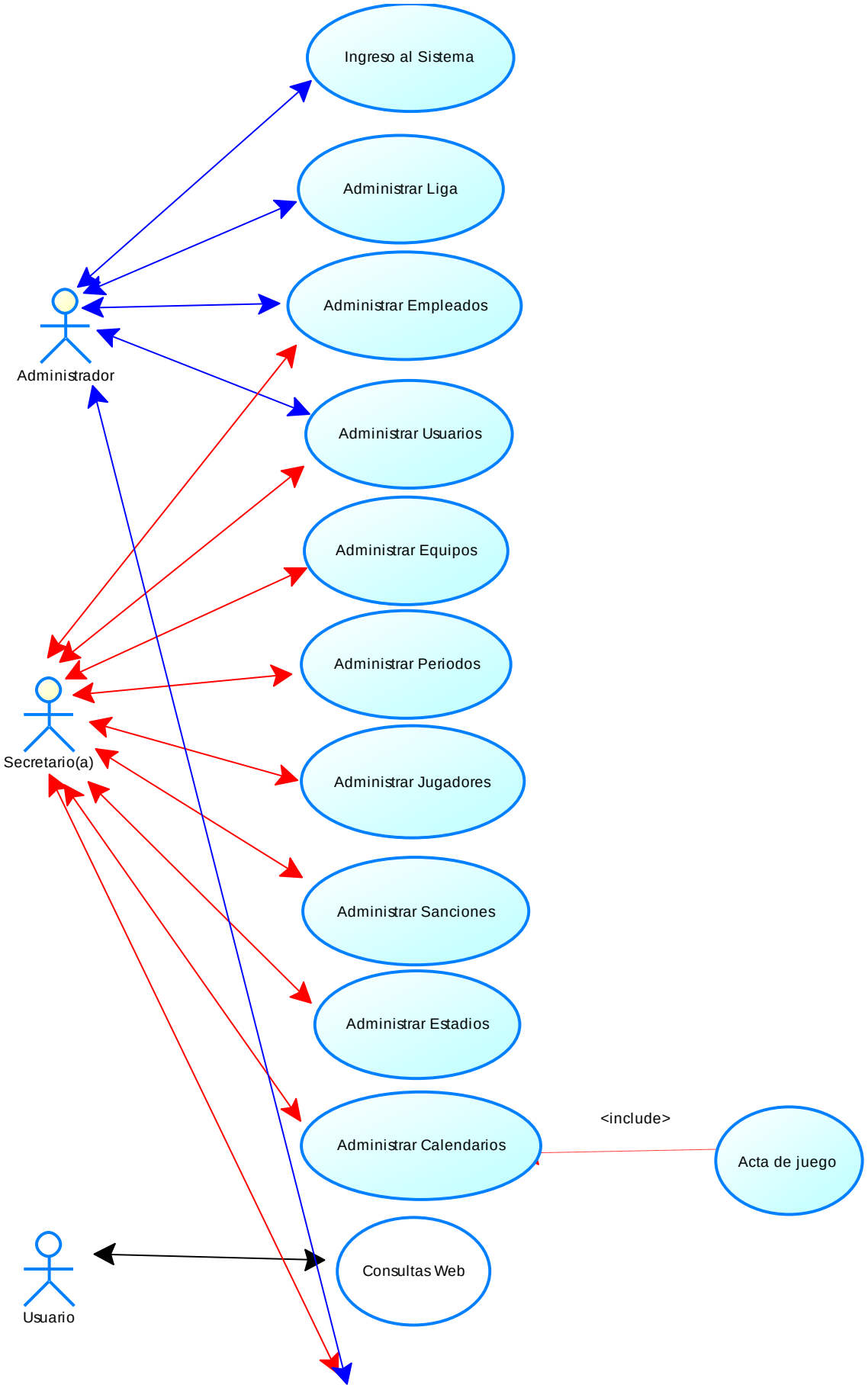


Figura 6.1. Diagrama de casos de uso Sistema de Gestión de Proyectos

Especificación de casos de uso

	Caso de uso: Ingresar al sistema
Actores:	Usuario administrador Usuario Secretario(a)
Descripción:	Validar a un usuario ya registrado para el uso del sistema.
Precondiciones:	El usuario debe estar registrado en el sistema
Pos condiciones	Validación realizada con éxito
Flujo normal de eventos:	**Validar** 1. El usuario ingresa el nombre de usuario y contraseña 2. El sistema valida los datos ingresados por el usuario. 3. Una vez validado el usuario, el sistema muestra el menú de opciones
Flujos alternos:	**usuario no registrado** 1. En el paso 2 del Flujo Normal, si el usuario no existe en el sistema se muestra un mensaje donde se indica que los datos de usuario o contraseña son erróneos.

Tabla 6.1. Especificación ingresar al sistema

	Caso de uso: Administrar Liga
Actores:	Usuario administrador
Descripción:	Añadir, modificar, eliminar Ligas del sistema.
Precondiciones:	• El usuario debe estar registrado como administrador • Datos de permisos ingresados con anterioridad
Pos condiciones	• Resultados almacenados
Flujo normal de eventos:	**Añadir** 1. El usuario escoge la opción de Administrar Liga. 2. El sistema muestra las opciones disponibles: añadir, modificar, eliminar 3. El usuario escoge la opción añadir 4. El sistema muestra los campos para ingresar la información de la Liga 5. El usuario ingresa la información 6. El sistema verifica el ingreso correcto de información
Flujos alternos:	**Modificar** 1. En el paso 3 del Flujo Normal, el usuario selecciona la opción modificar

	2. El sistema muestra los campos con información de la Liga a modificar. 3. El usuario modifica la información 4. El sistema verifica el ingreso correcto de información. **eliminar** 1. En el paso 3 del Flujo Normal, el usuario escoge la opción eliminar 2. El sistema muestra un mensaje donde se indica si el usuario está seguro de la eliminación. 3. El usuario toma una decisión 4. El sistema realiza la operación indicada.

Tabla 6.2 Especificación Administrar Liga

	Caso de uso: Administrar Empleado
Actores:	Usuario administrador Usuario secretario(a)
Descripción:	Añadir, modificar, eliminar empleados del sistema.
Precondiciones:	• El usuario debe estar registrado como administrador • Datos de permisos ingresados con anterioridad
Pos condiciones	• Resultados almacenados
Flujo normal de eventos:	**Añadir** 7. El usuario escoge la opción de Administrar empleados. 8. El sistema muestra las opciones disponibles: añadir, modificar, eliminar 9. El usuario escoge la opción añadir 10. El sistema muestra los campos para ingresar la información del empleado 11. El usuario ingresa la información 12. El sistema verifica el ingreso correcto de información
Flujos alternos:	**Modificar** 5. En el paso 3 del Flujo Normal, el usuario selecciona la opción modificar 6. El sistema muestra los campos con información del empleado a modificar. 7. El usuario modifica la información 8. El sistema verifica el ingreso correcto de información. **eliminar** 5. En el paso 3 del Flujo Normal, el usuario escoge la opción eliminar 6. El sistema muestra un mensaje donde se indica si el usuario está seguro de la eliminación.

	7. El usuario toma una decisión 8. El sistema realiza la operación indicada.

Tabla 6.3 Especificación Administrar Empleados

	Caso de uso: Administrar usuarios
Actores:	Usuario administrador Usuario secretario(a)
Descripción:	Añadir, modificar, eliminar usuarios del sistema.
Precondiciones:	• El usuario debe estar registrado como administrador • Datos de permisos ingresados con anterioridad
Pos condiciones	• Resultados almacenados
Flujo normal de eventos:	**Añadir** 13. El usuario escoge la opción de Administrar usuarios 14. El sistema muestra las opciones disponibles: añadir, modificar, eliminar 15. El usuario escoge la opción añadir 16. El sistema muestra los campos para ingresar la información de los usuarios 17. El usuario ingresa la información 18. El sistema verifica el ingreso correcto de información
Flujos alternos:	**Modificar** 9. En el paso 3 del Flujo Normal, el usuario selecciona la opción modificar 10. El sistema muestra los campos con información del usuario a modificar. 11. El usuario modifica la información 12. El sistema verifica el ingreso correcto de información. **eliminar** 9. En el paso 3 del Flujo Normal, el usuario escoge la opción eliminar 10. El sistema muestra un mensaje donde se indica si el usuario está seguro de la eliminación. 11. El usuario toma una decisión 12. El sistema realiza la operación indicada.

Tabla 6.4 Especificación Administrar Usuarios.

	Caso de uso: Administrar equipos
Actores:	Usuario secretario(a)
Descripción:	Añadir, modificar, eliminar equipos del sistema.

Precondiciones:	• El usuario debe estar registrado como secretario • Datos de permisos ingresados con anterioridad
Pos condiciones	• Resultados almacenados
Flujo normal de eventos:	**Añadir** 19. El usuario escoge la opción de Administrar equipos 20. El sistema muestra las opciones disponibles: añadir, modificar, eliminar 21. El usuario escoge la opción añadir 22. El sistema muestra los campos para ingresar la información de los equipos 23. El usuario ingresa la información 24. El sistema verifica el ingreso correcto de información
Flujos alternos:	**Modificar** 13. En el paso 3 del Flujo Normal, el usuario selecciona la opción modificar 14. El sistema muestra los campos con información del equipo a modificar. 15. El usuario modifica la información 16. El sistema verifica el ingreso correcto de información. **eliminar** 13. En el paso 3 del Flujo Normal, el usuario escoge la opción eliminar 14. El sistema muestra un mensaje donde se indica si el usuario está seguro de la eliminación. 15. El usuario toma una decisión 16. El sistema realiza la operación indicada.

Tabla 6.5 Especificación Administrar Equipos

	Caso de uso: Administrar periodo
Actores:	Usuario secretario(a)
Descripción:	Añadir, modificar, eliminar periodo del sistema.
Precondiciones:	• El usuario debe estar registrado como secretario • Datos de permisos ingresados con anterioridad
Pos condiciones	• Resultados almacenados
Flujo normal de eventos:	**Añadir** 25. El usuario escoge la opción de Administrar Periodo 26. El sistema muestra las opciones disponibles: añadir, modificar, eliminar 27. El usuario escoge la opción añadir 28. El sistema muestra los campos para ingresar la información del periodo

	29. El usuario ingresa la información 30. El sistema verifica el ingreso correcto de información
Flujos alternos:	**Modificar** 17. En el paso 3 del Flujo Normal, el usuario selecciona la opción modificar 18. El sistema muestra los campos con información del periodo a modificar. 19. El usuario modifica la información 20. El sistema verifica el ingreso correcto de información. **eliminar** 17. En el paso 3 del Flujo Normal, el usuario escoge la opción eliminar 18. El sistema muestra un mensaje donde se indica si el usuario está seguro de la eliminación. 19. El usuario toma una decisión 20. El sistema realiza la operación indicada.

Tabla 6.6 Especificación Administrar Periodos

	Caso de uso: Administrar jugadores
Actores:	Usuario secretario(a)
Descripción:	Añadir, modificar, eliminar jugadores del sistema.
Precondiciones:	• El usuario debe estar registrado como secretario • Datos de permisos ingresados con anterioridad
Pos condiciones	• Resultados almacenados
Flujo normal de eventos:	**Añadir** 31. El usuario escoge la opción de Administrar jugadores 32. El sistema muestra las opciones disponibles: añadir, modificar, eliminar 33. El usuario escoge la opción añadir 34. El sistema muestra los campos para ingresar la información de los jugadores 35. El usuario ingresa la información 36. El sistema verifica el ingreso correcto de información
Flujos alternos:	**Modificar** 21. En el paso 3 del Flujo Normal, el usuario selecciona la opción modificar 22. El sistema muestra los campos con información del jugador a modificar. 23. El usuario modifica la información 24. El sistema verifica el ingreso correcto de información.

	eliminar 21. En el paso 3 del Flujo Normal, el usuario escoge la opción eliminar 22. El sistema muestra un mensaje donde se indica si el usuario está seguro de la eliminación. 23. El usuario toma una decisión 24. El sistema realiza la operación indicada.

Tabla 6.7 Especificación Administrar Jugadores

	Caso de uso: Administrar sanciones
Actores:	Usuario secretario(a)
Descripción:	Añadir, modificar, eliminar sanciones del sistema.
Precondiciones:	• El usuario debe estar registrado como secretario • Datos de permisos ingresados con anterioridad
Pos condiciones	• Resultados almacenados
Flujo normal de eventos:	**Añadir** 37. El usuario escoge la opción de Administrar sanciones 38. El sistema muestra las opciones disponibles: añadir, modificar, eliminar 39. El usuario escoge la opción añadir 40. El sistema muestra los campos para ingresar la información de las sanciones 41. El usuario ingresa la información 42. El sistema verifica el ingreso correcto de información
Flujos alternos:	**Modificar** 25. En el paso 3 del Flujo Normal, el usuario selecciona la opción modificar 26. El sistema muestra los campos con información de la sanción a modificar. 27. El usuario modifica la información 28. El sistema verifica el ingreso correcto de información. **eliminar** 25. En el paso 3 del Flujo Normal, el usuario escoge la opción eliminar 26. El sistema muestra un mensaje donde se indica si el usuario está seguro de la eliminación. 27. El usuario toma una decisión 28. El sistema realiza la operación indicada.

Tabla 6.8 Especificación Administrar Sanciones

	Caso de uso: Administrar estadios
Actores:	Usuario secretario(a)
Descripción:	Añadir, modificar, eliminar estadios del sistema.
Precondiciones:	• El usuario debe estar registrado como secretario • Datos de permisos ingresados con anterioridad
Pos condiciones	• Resultados almacenados
Flujo normal de eventos:	**Añadir** 43. El usuario escoge la opción de Administrar estadio 44. El sistema muestra las opciones disponibles: añadir, modificar, eliminar 45. El usuario escoge la opción añadir 46. El sistema muestra los campos para ingresar la información de los estadios 47. El usuario ingresa la información 48. El sistema verifica el ingreso correcto de información
Flujos alternos:	**Modificar** 29. En el paso 3 del Flujo Normal, el usuario selecciona la opción modificar 30. El sistema muestra los campos con información del estadio a modificar. 31. El usuario modifica la información 32. El sistema verifica el ingreso correcto de información. **eliminar** 29. En el paso 3 del Flujo Normal, el usuario escoge la opción eliminar 30. El sistema muestra un mensaje donde se indica si el usuario está seguro de la eliminación. 31. El usuario toma una decisión 32. El sistema realiza la operación indicada.

Tabla 6.9 Especificación administrar estadios

	Caso de uso: Administrar calendario
Actores:	Usuario secretario(a)
Descripción:	Añadir, modificar, eliminar calendarios del sistema.
Precondiciones:	• El usuario debe estar registrado como secretario • Datos de permisos ingresados con anterioridad
Pos condiciones	• Resultados almacenados
Flujo normal de eventos:	**Añadir** 49. El usuario escoge la opción de Administrar calendario

	50. El sistema muestra las opciones disponibles: añadir, modificar, eliminar 51. El usuario escoge la opción añadir 52. El sistema muestra los campos para ingresar la información del calendario 53. El usuario ingresa la información 54. El sistema verifica el ingreso correcto de información
Flujos alternos:	**Modificar** 33. En el paso 3 del Flujo Normal, el usuario selecciona la opción modificar 34. El sistema muestra los campos con información del calendario a modificar. 35. El usuario modifica la información 36. El sistema verifica el ingreso correcto de información. **eliminar** 33. En el paso 3 del Flujo Normal, el usuario escoge la opción eliminar 34. El sistema muestra un mensaje donde se indica si el usuario está seguro de la eliminación. 35. El usuario toma una decisión 36. El sistema realiza la operación indicada.

Tabla 6.10 Especificación Administrar Calendarios

	Caso de uso: Consultas web
Actores:	Usuario
Descripción:	Consultas en la web de Sanciones, Calendario y Tabla de posiciones.
Precondiciones:	• El usuario puede realizar las consultas sin necesidad de ingresar como administrador o secretario(a) • Datos de permisos ingresados con anterioridad
Pos condiciones	• Resultados almacenados
Flujo normal de eventos:	**Añadir** 55. El usuario escoge la opción de consultar 56. La consulta web muestra las opciones disponibles: inicio, calendario, sanciones, tabla de posiciones, contactos 57. El usuario escoge la opción calendario 58. El sistema muestra la consulta de calendarios de la semana 59. El usuario escoge la opción sanciones 60. El sistema muestra la consulta Sanciones de acuerdo a los parámetros establecidos.

	61. El usuario escoge la opción tabla de posiciones 62. El sistema muestra la consulta tabla de posiciones de acuerdo a los parámetros establecidos. 63. Es usuario escoge la opción contactos 64. El sistema muestra datos de la Liga.

Tabla 6.11 Especificación consultas web

	Caso de uso: Tratar acta de juego
Actores:	Usuario secretario(a)
Descripción:	Añadir, modificar, eliminar acata de juego del sistema.
Precondiciones:	• El usuario debe estar registrado como secretario • Datos de permisos ingresados con anterioridad
Pos condiciones	• Resultados almacenados
Flujo normal de eventos:	**Añadir** 65. El usuario escoge la opción de Administrar acta de juego 66. El sistema muestra las opciones disponibles: añadir. 67. El usuario escoge la opción añadir 68. El sistema muestra los campos para ingresar la información del calendario 69. El usuario ingresa la información 70. El sistema verifica el ingreso correcto de información

Tabla 6.12 Especificación tratar acta de juego

	Caso de uso: Salir sistema
Actores:	Usuario administrador Usuario secretario(a)
Descripción:	El usuario sale del sistema
Precondiciones:	• El usuario debe estar registrado en el sistema • El usuario debe estar con la sesión iniciada en el sistema
Pos condiciones	Operación realizada con éxito
Flujo normal de eventos:	**Validar** 1. El usuario escoge la opción cerrar sesión 2. El sistema realiza la operación indicada

Tabla 6.13 Especificación Salir del Sistema

6.9.1.2.2 Diagrama de estados

Los diagramas de estado son una técnica importante en la descripción del comportamiento de un sistema. Detallan todos los estados posibles por los que puede pasar un objeto a lo largo de su vida y los estímulos que lo provocan.

Empleado

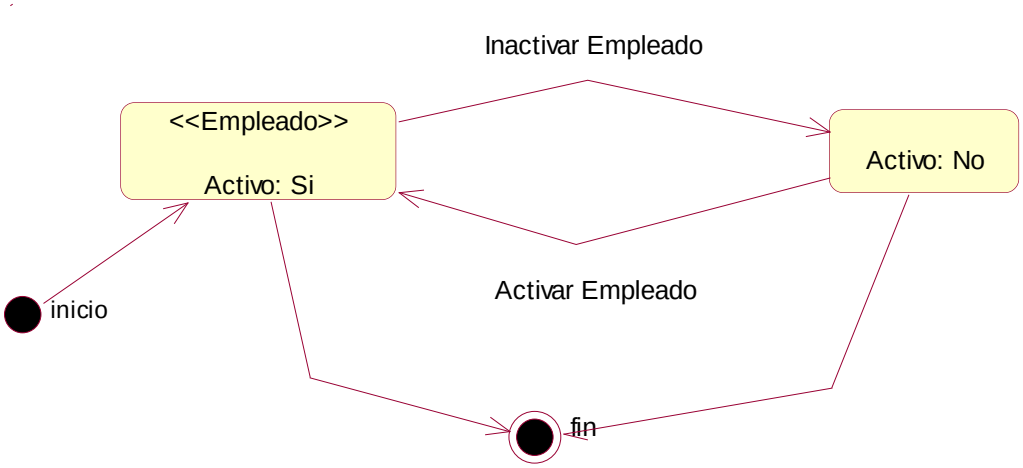


Figura 6.2 Diagrama de estados empleado

Equipo

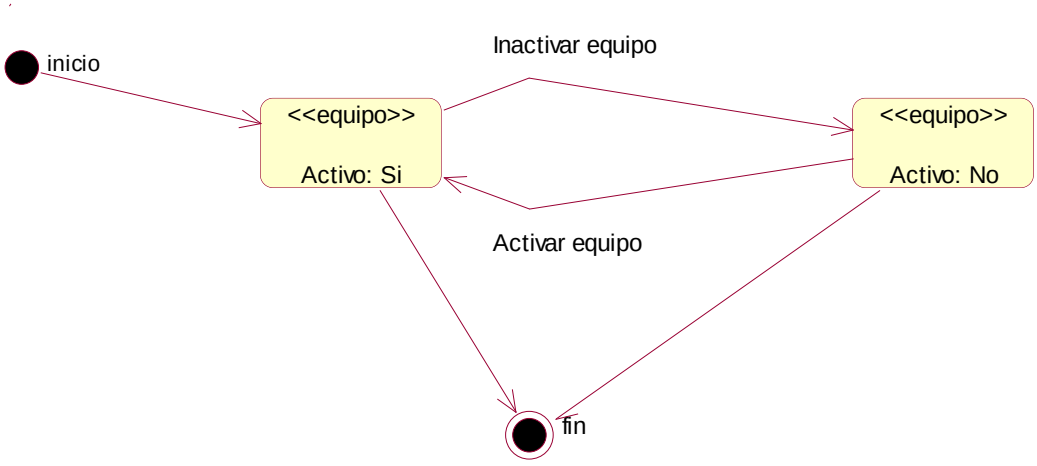


Figura 6.3 Diagrama de estado equipo

Periodo

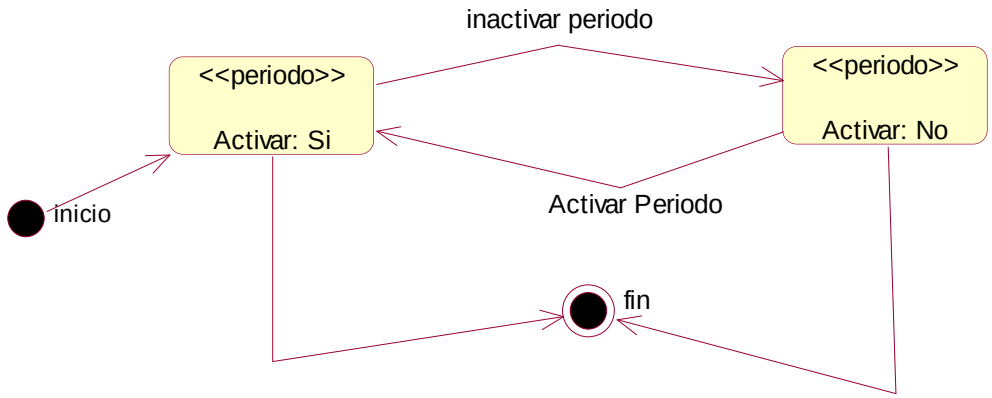


Figura 6.4 Diagrama de estado periodo

Inscripciones

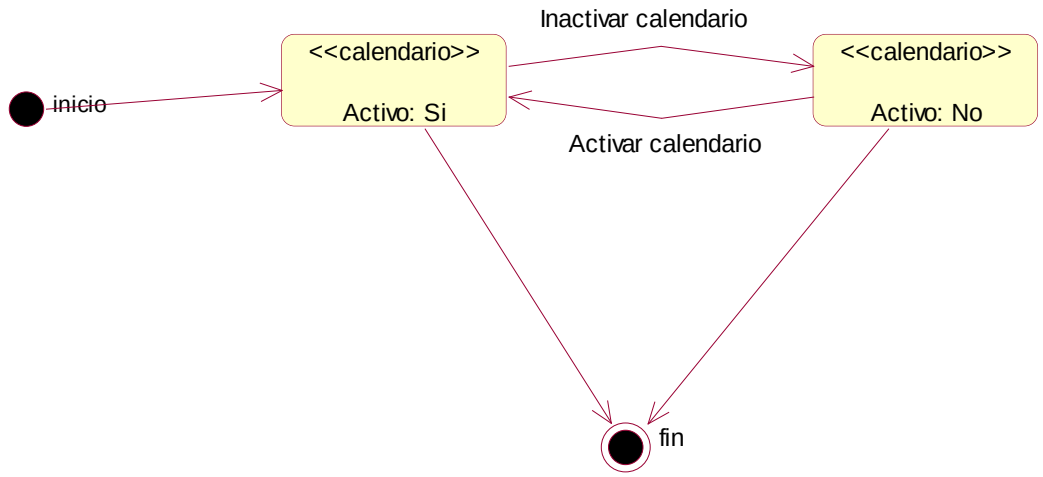


Figura 6.4 Diagrama de estado inscripciones

Jugador

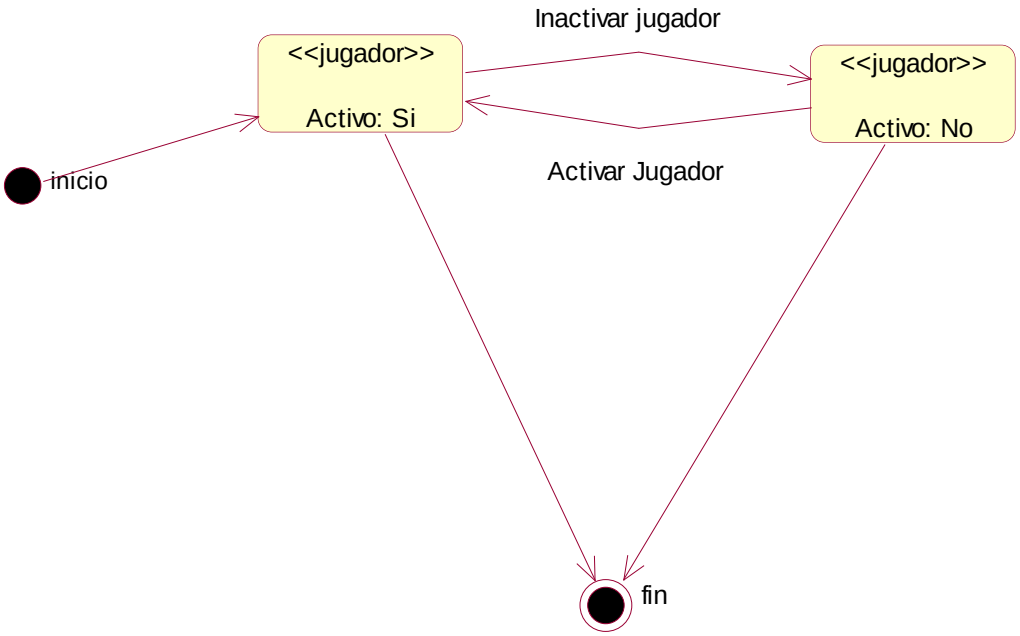


Figura 6.5 Diagrama de estado jugador

Sanciones

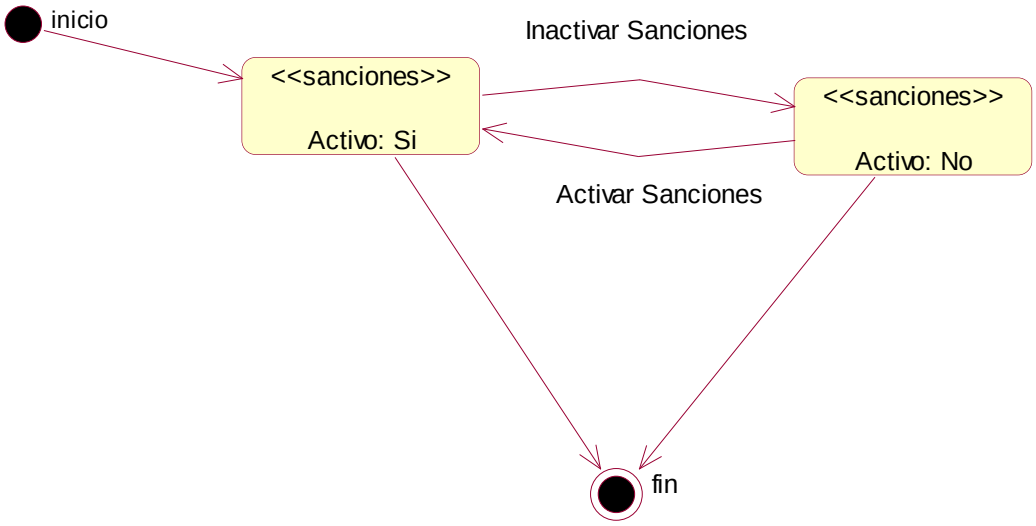


Figura 6.6 Diagrama de estado sanciones

Calendario

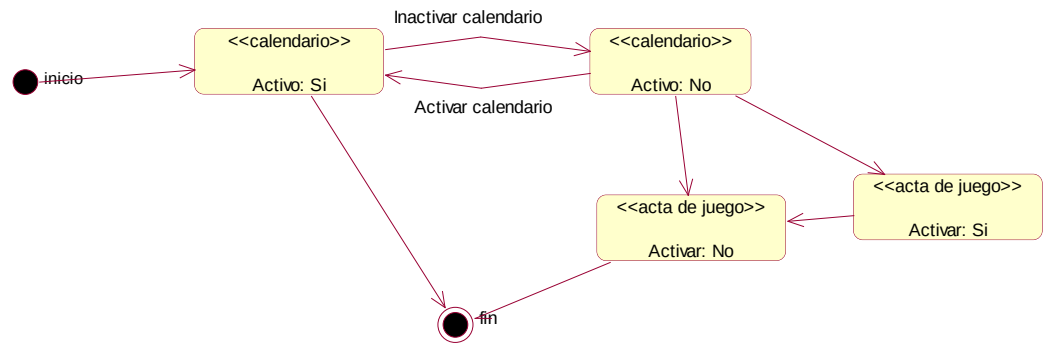


Figura 6.7 Diagrama de estado calendario

6.9.1.2.3 Diagrama de secuencias

Un diagrama de secuencia muestra la interacción de un conjunto de objetos en una aplicación a través del tiempo y se modela para cada caso de uso.

Ingresar al sistema

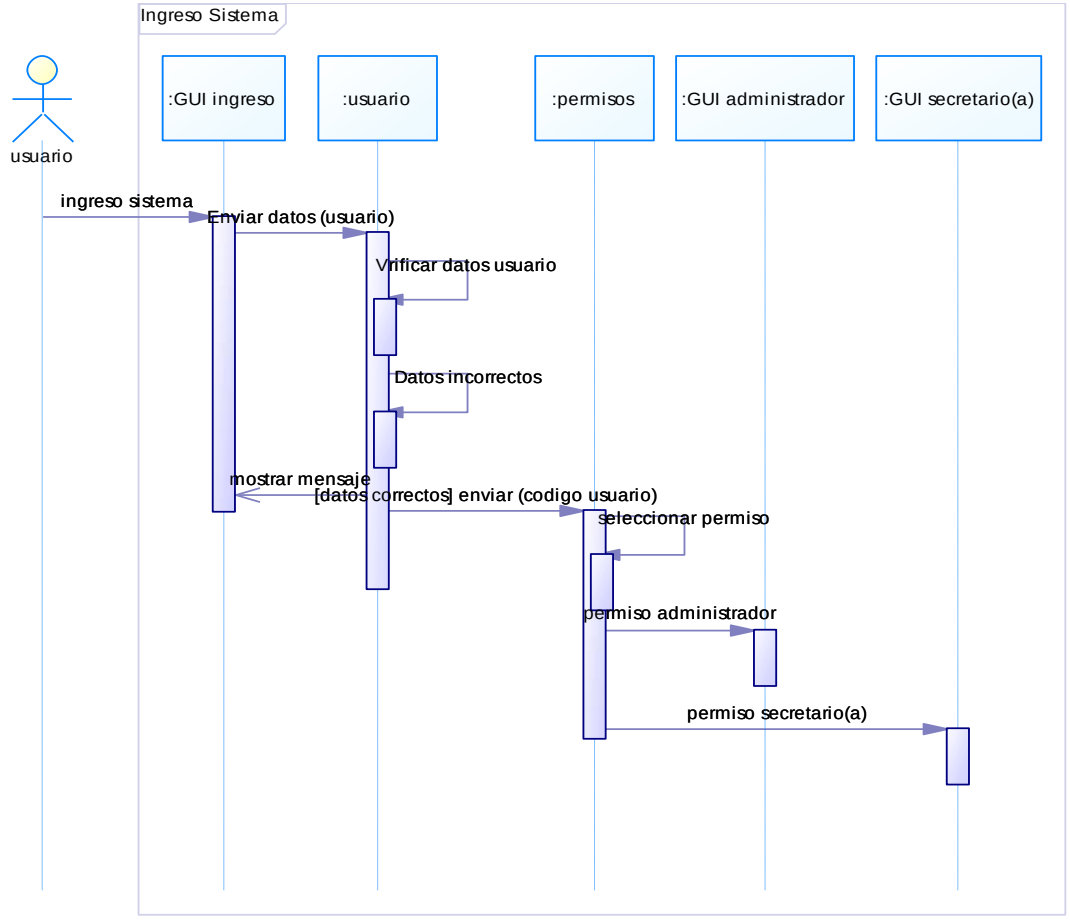


Figura 6.8 Diagrama de secuencia ingreso al sistema

Administrar Liga

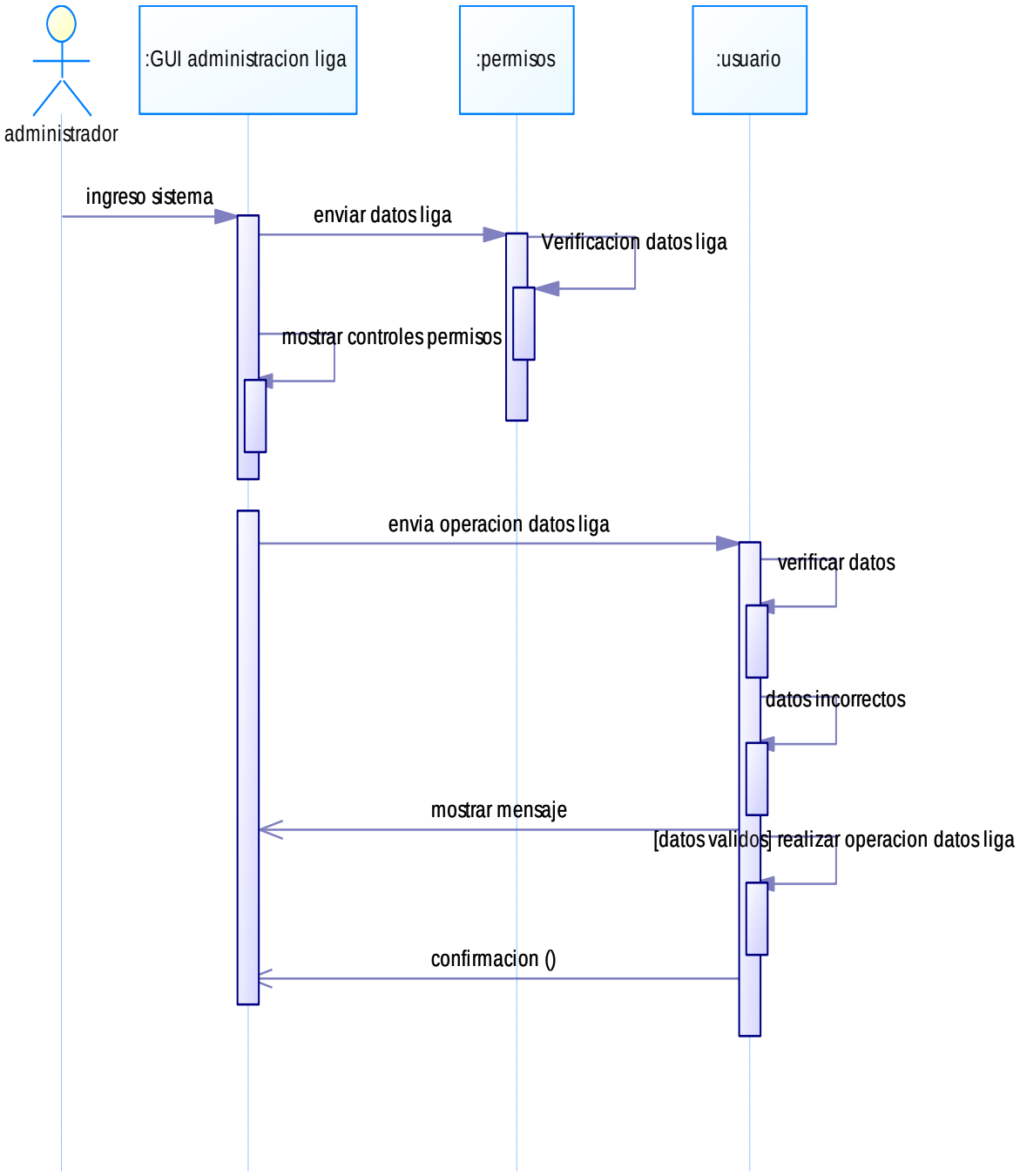


Figura 6.9 Diagrama de secuencia administrar liga

Administrar empleado

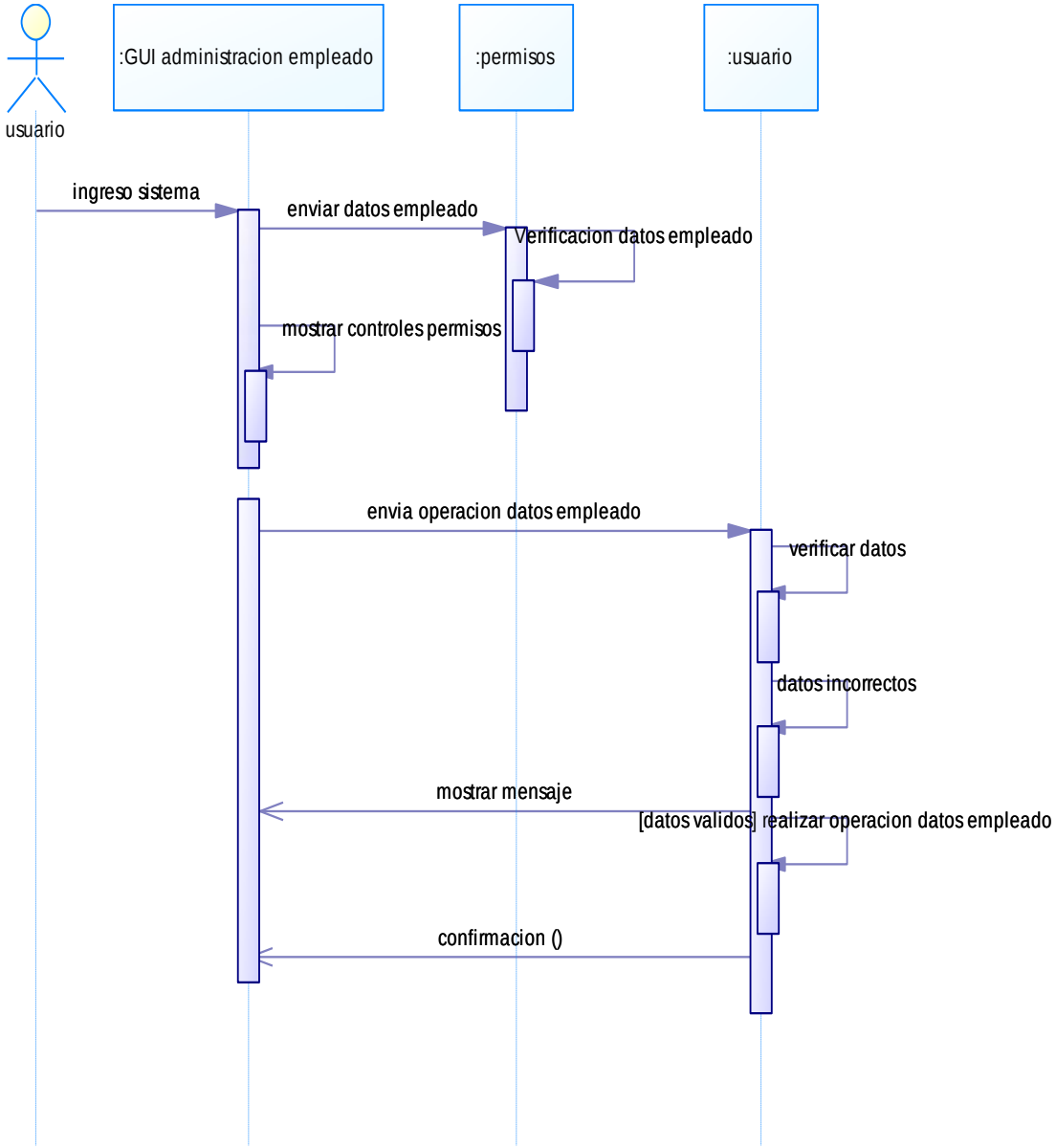


Figura 6.10 Diagrama de secuencia administrar empleado

Administrar usuarios

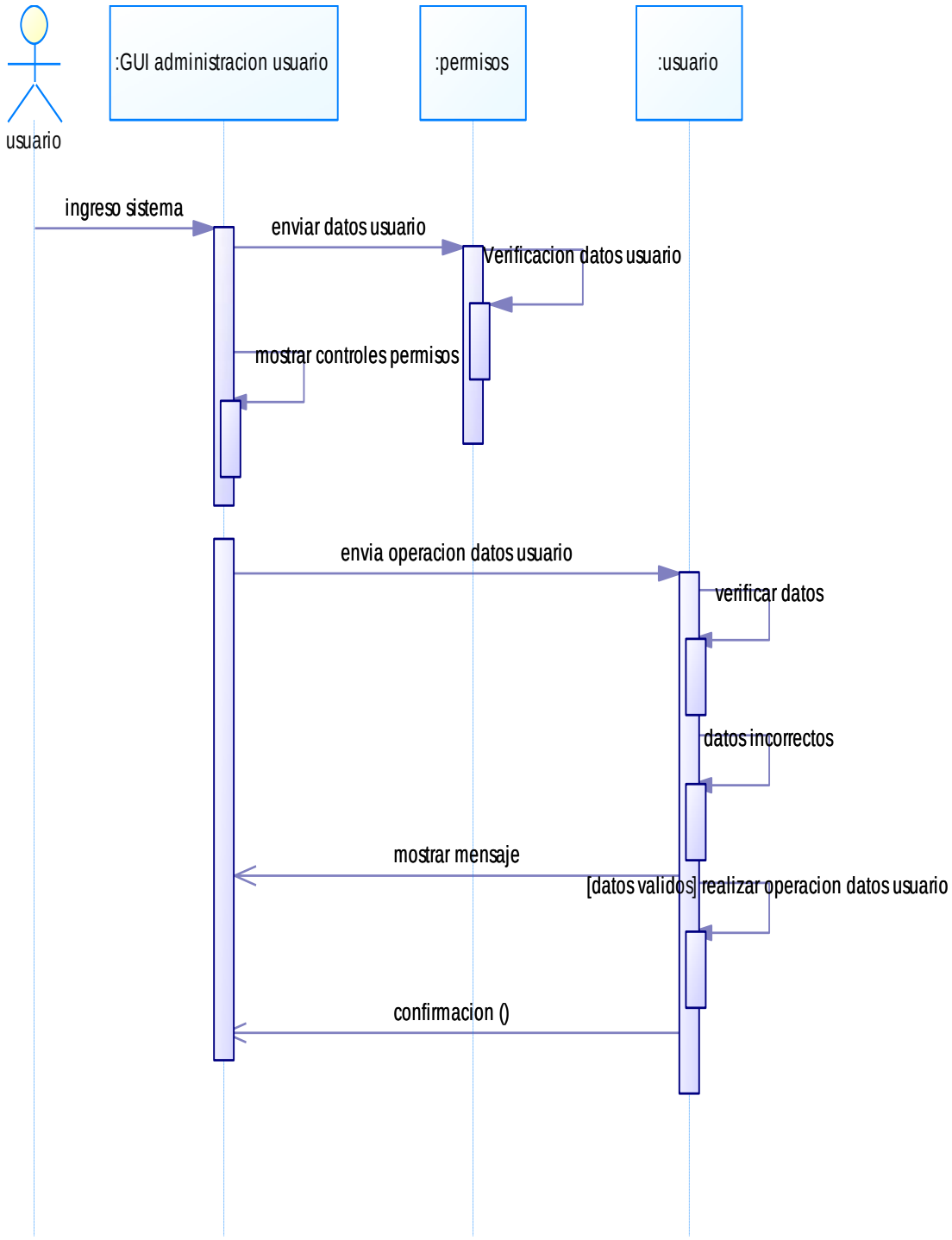


Figura 6.11 Diagrama de secuencia administrar usuario

Administrar Equipos

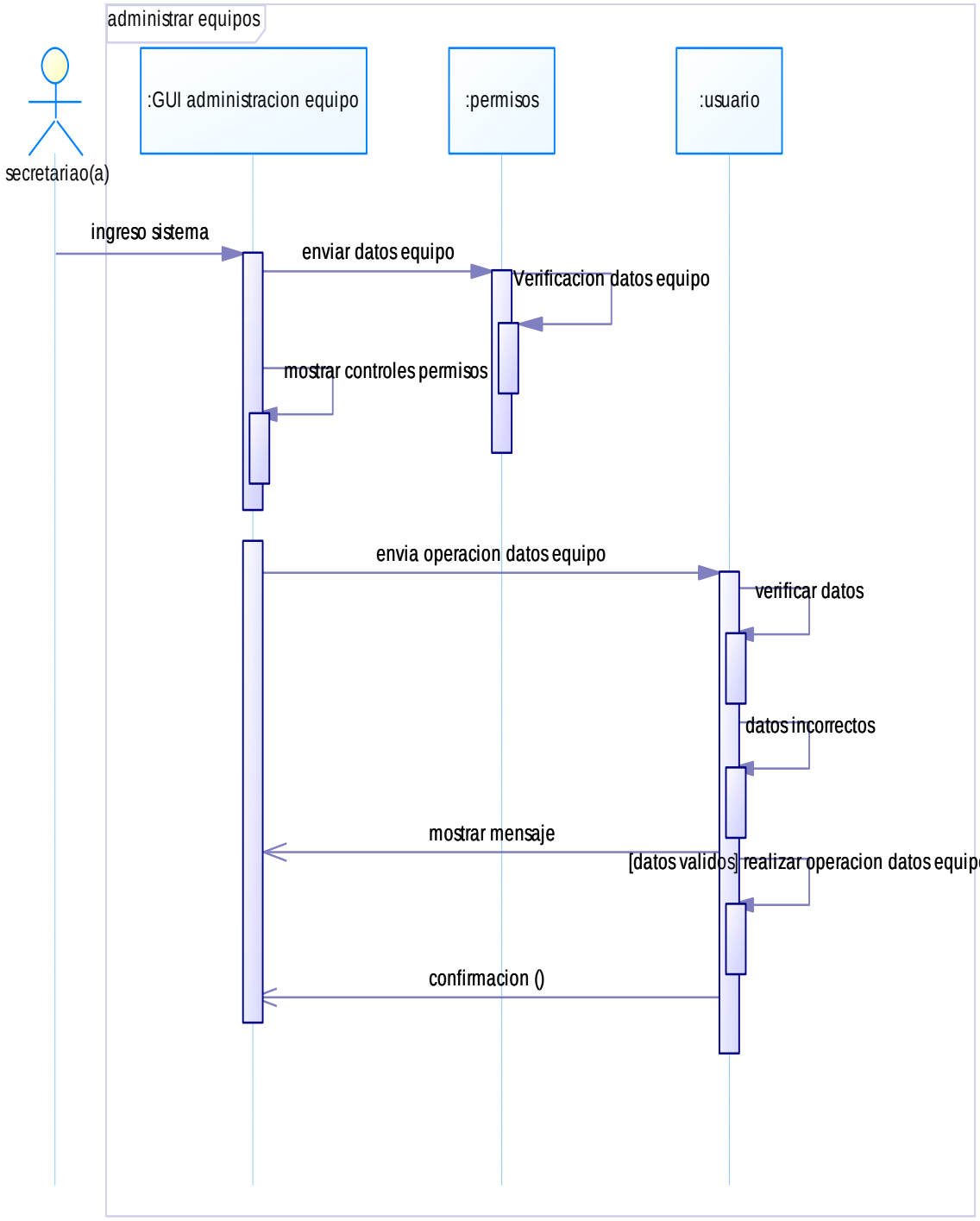


Figura 6.12 Diagrama de secuencia administrar equipo

Administrar Periodo

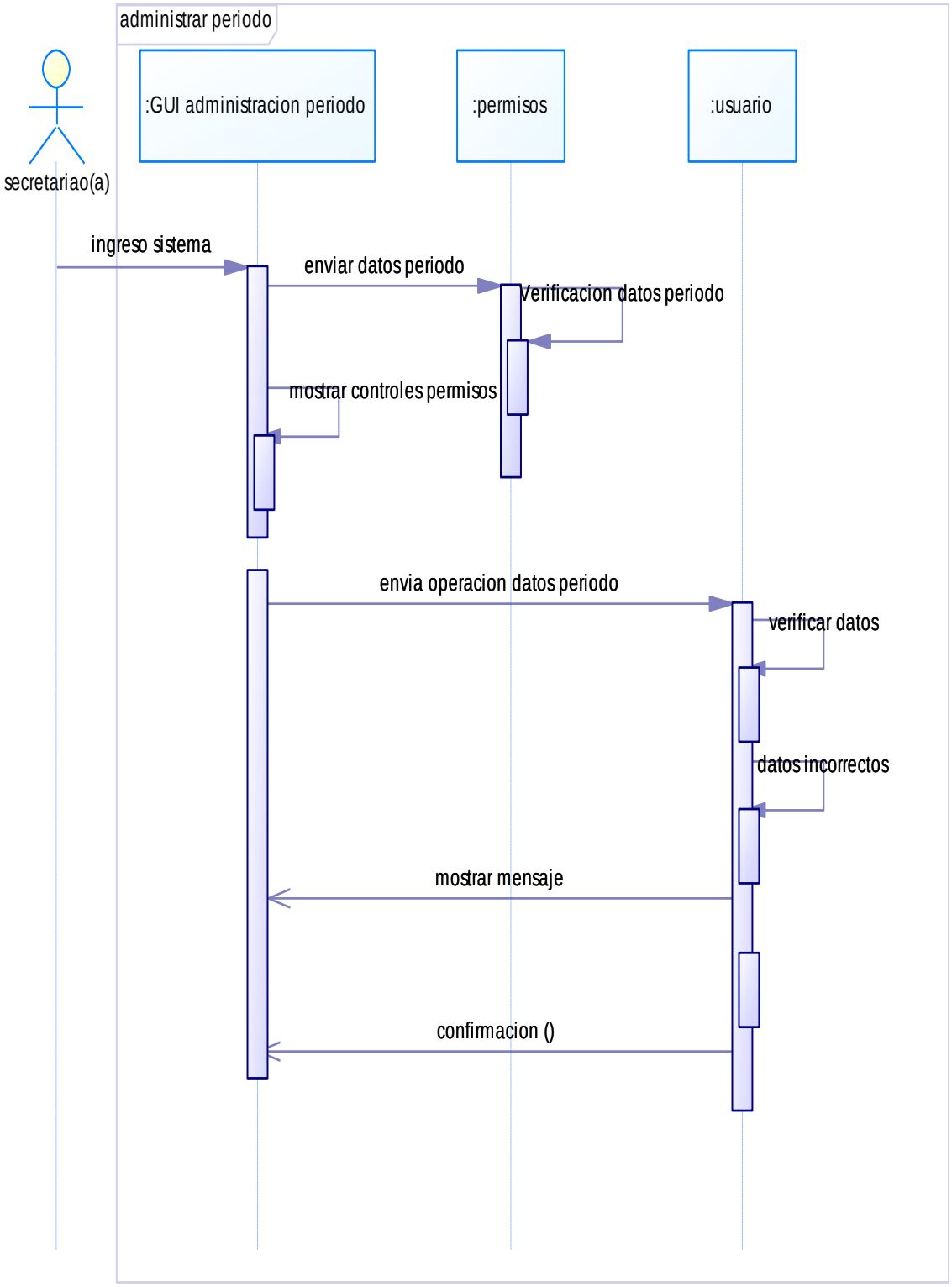


Figura 6.13 Diagrama de secuencia administrar periodo

Administrar Jugadores

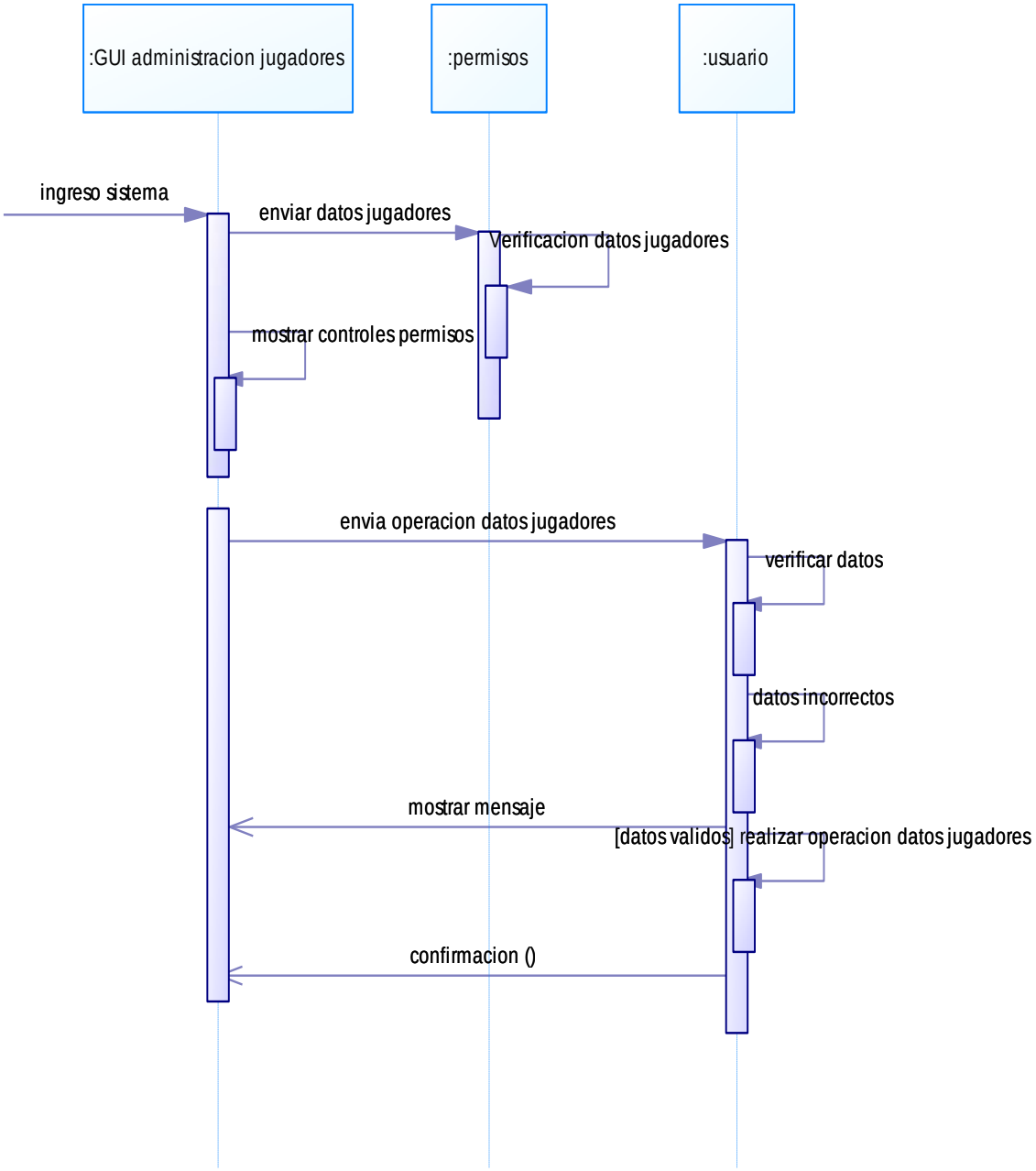


Figura 6.14 Diagrama de secuencia administrar jugadores

Administrar Sanciones

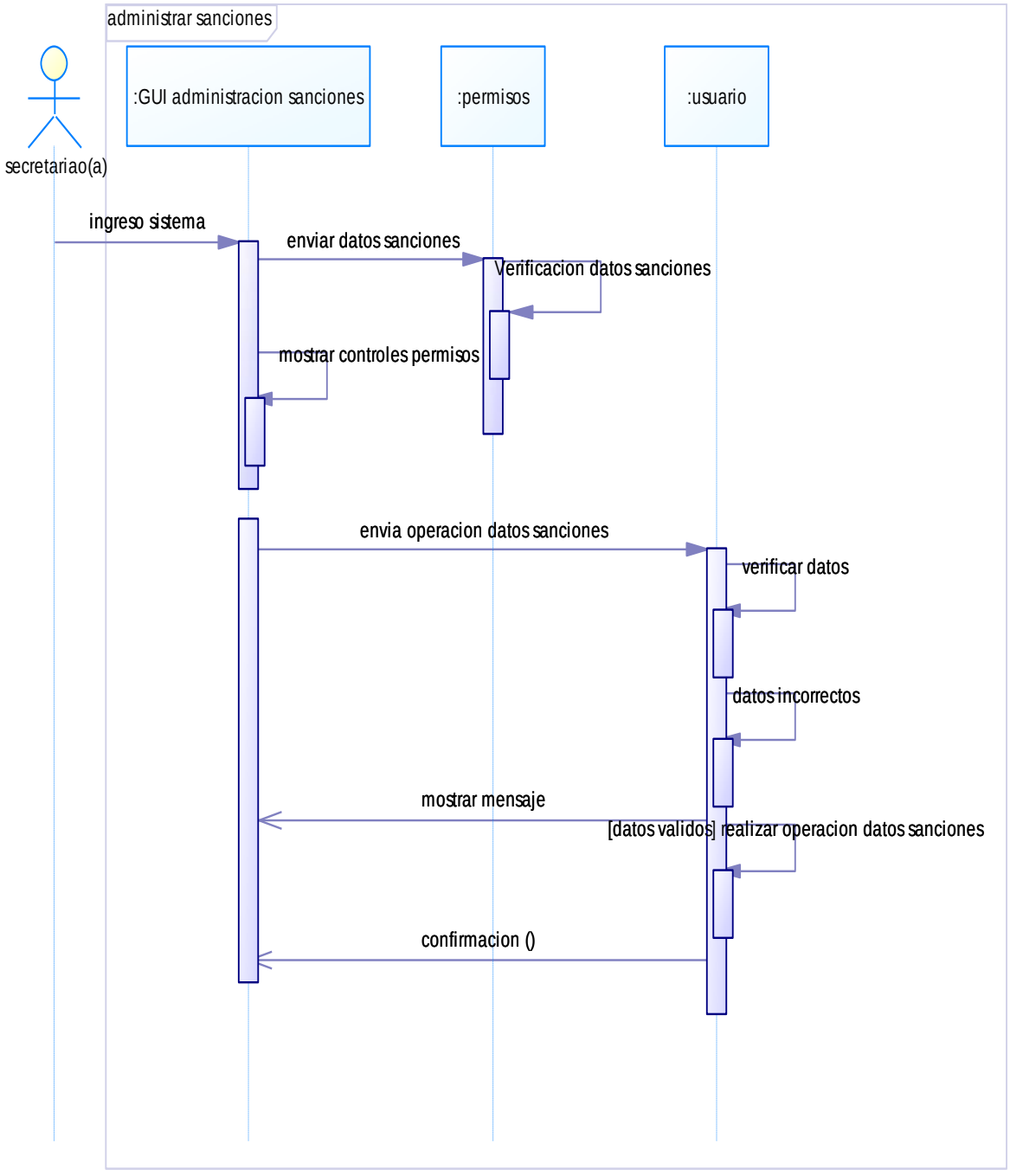


Figura 6.15 Diagrama de secuencia administrar sanciones

Administrar Estadios

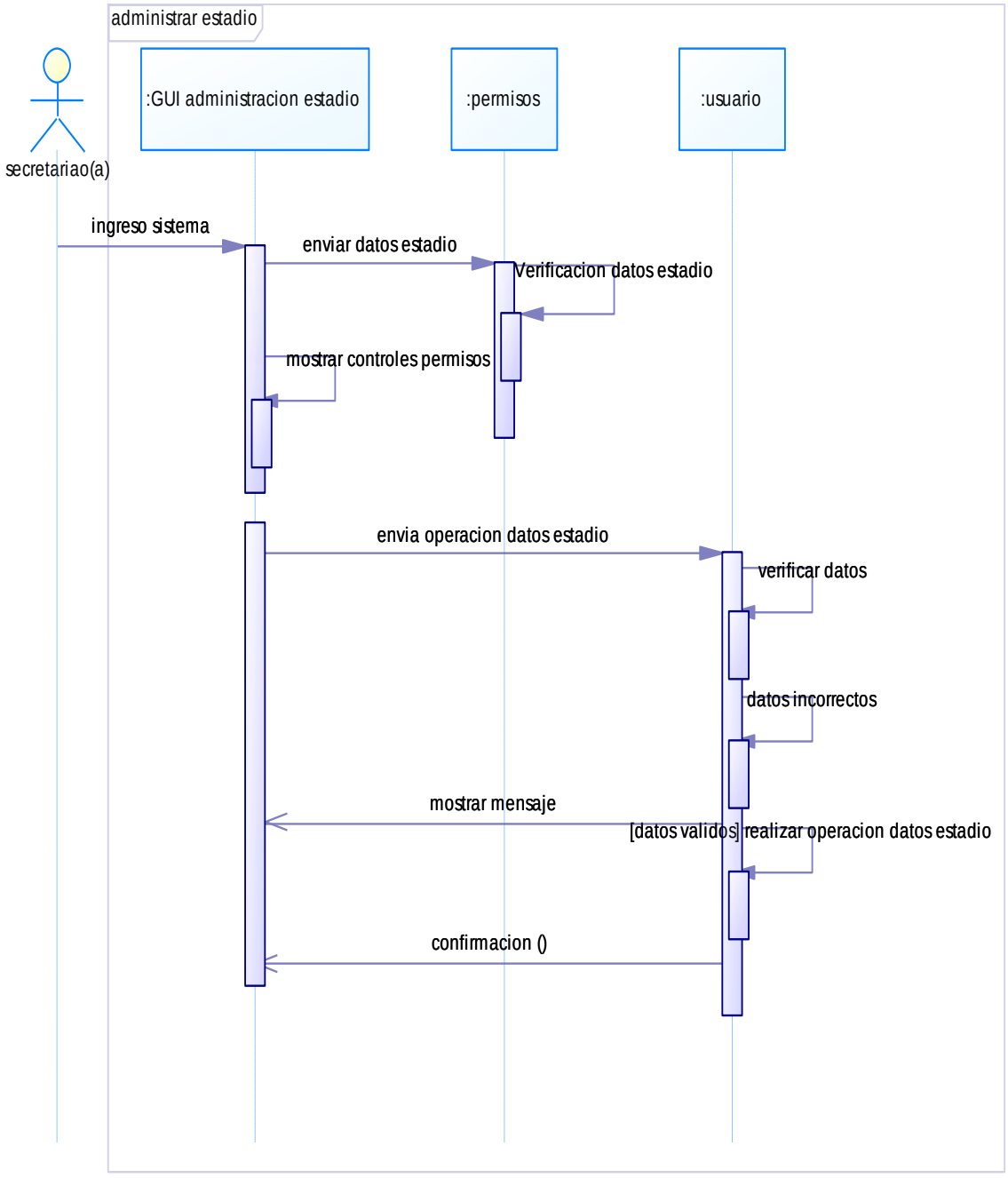


Figura 6.16 Diagrama de secuencia administrar estadios

Administrar Calendario

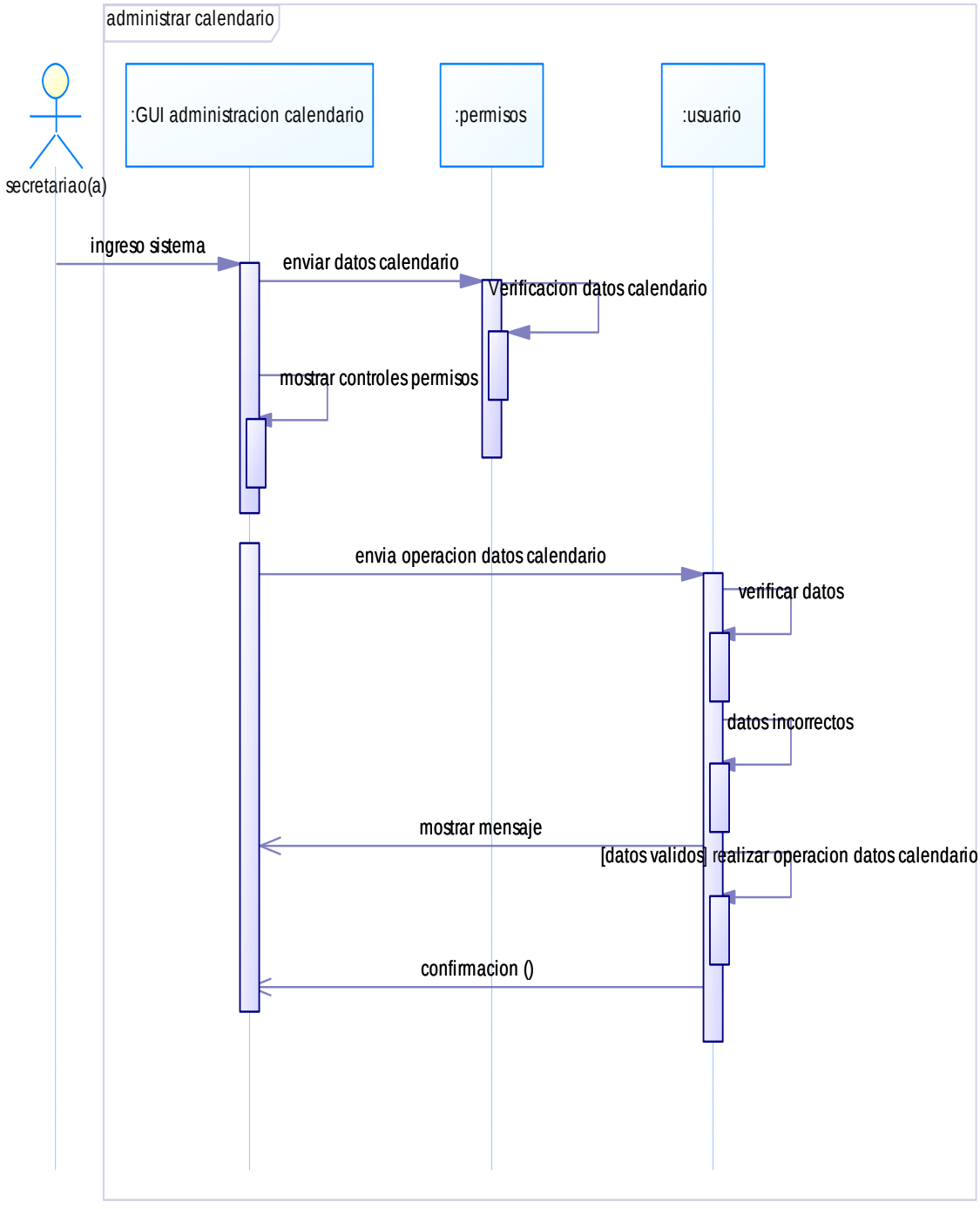


Figura 6.17 Diagrama de secuencia administrar calendario

Tratar acta de juego

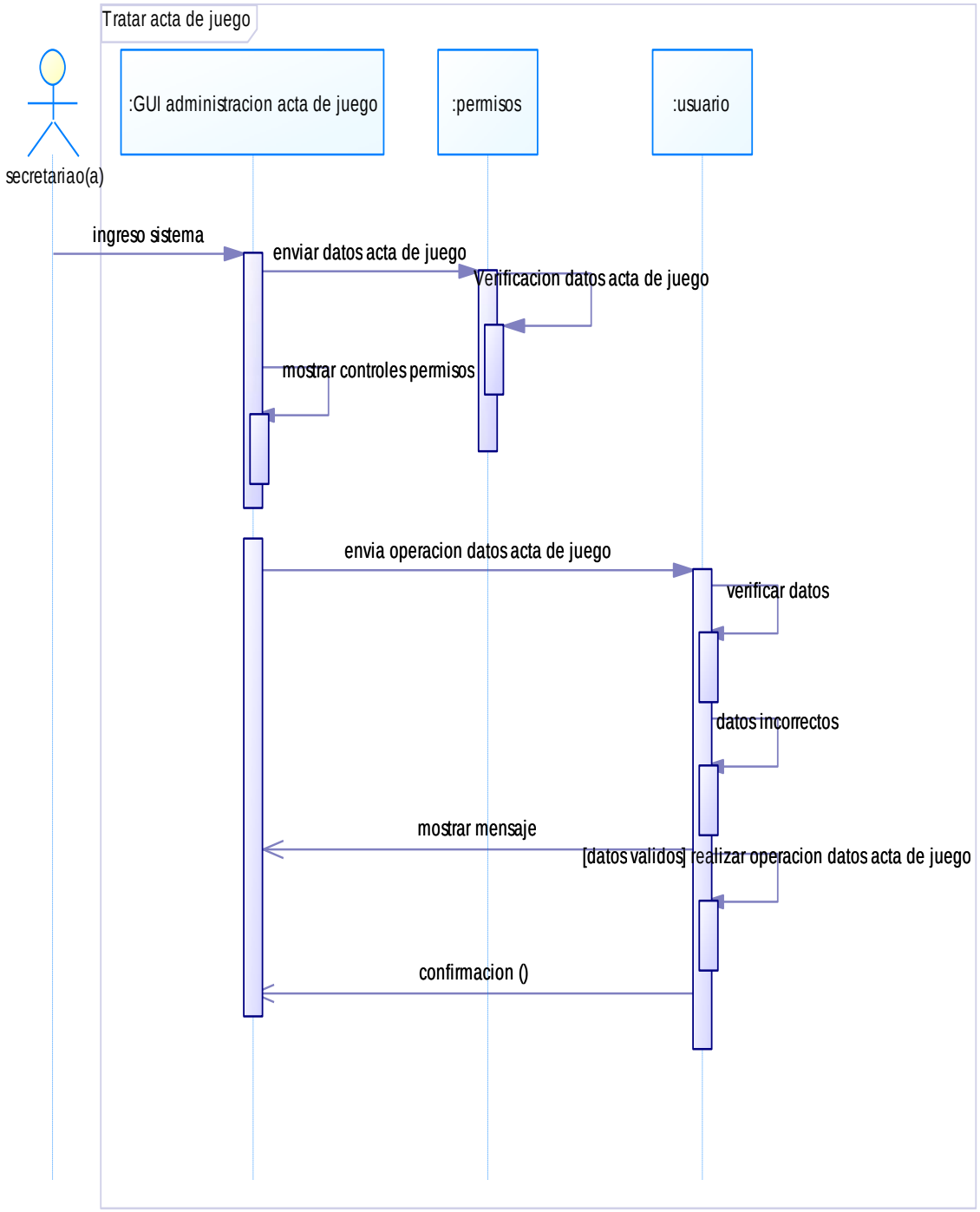


Figura 6.18 Diagrama de secuencia tratar acta de juego

Consultas web

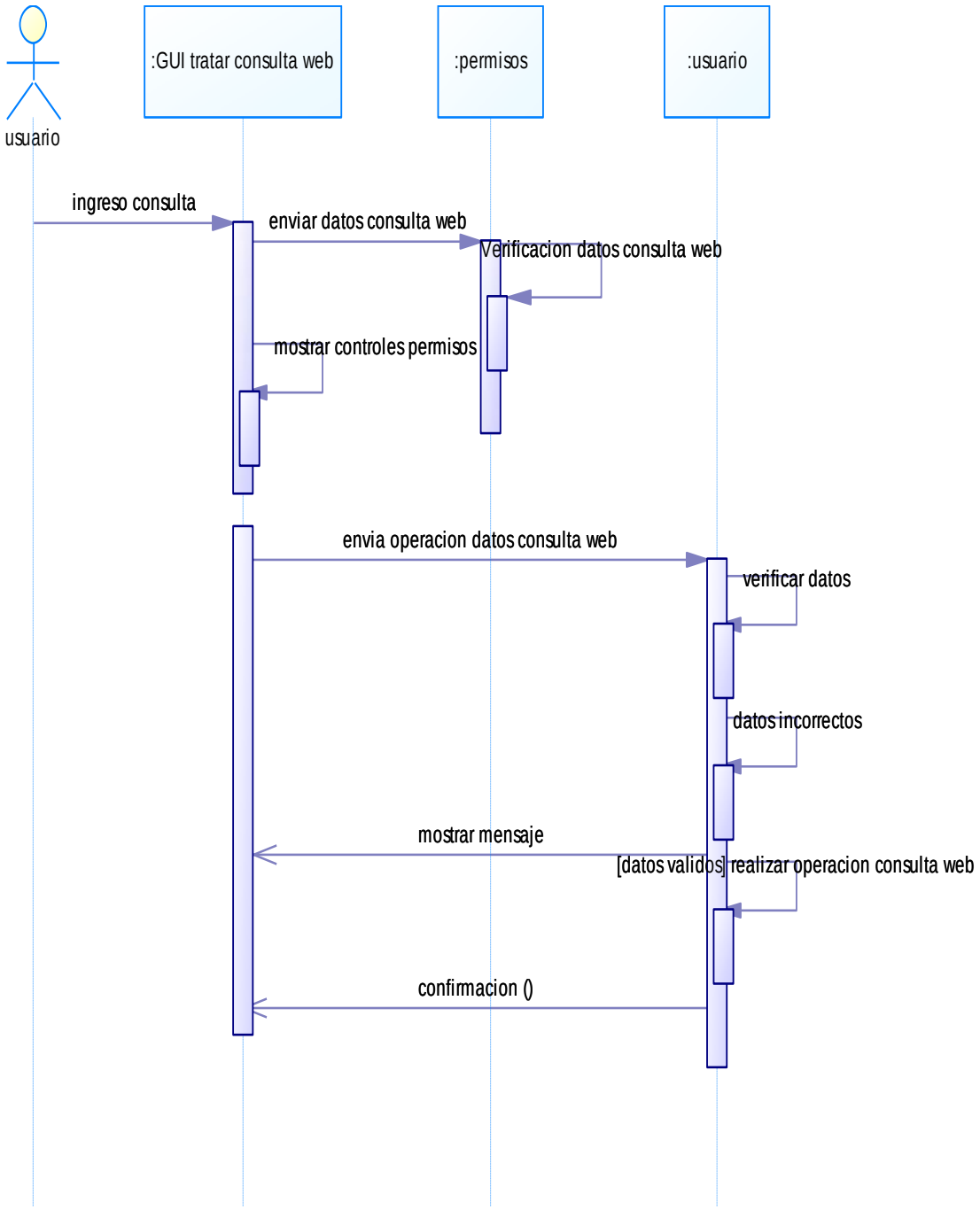


Figura 6.19 Diagrama de secuencia tratar consultas web

Salir del sistema

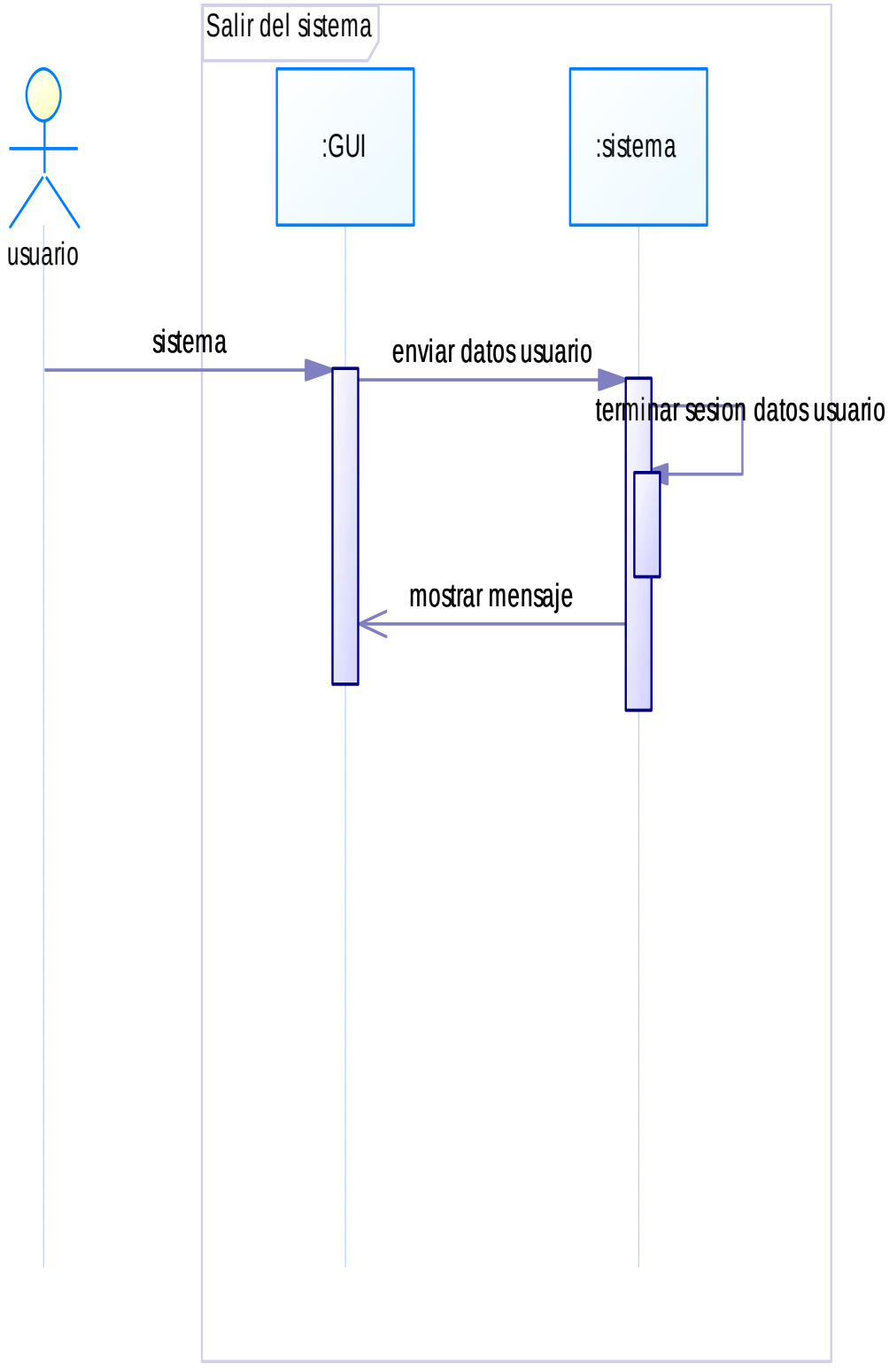


Figura 6.20 Diagrama de secuencia salir del sistema

6.9.1.2.4 Diagrama de despliegue

Los diagramas de despliegue se usan para modelar las relaciones físicas entre los componentes hardware y software en el sistema.

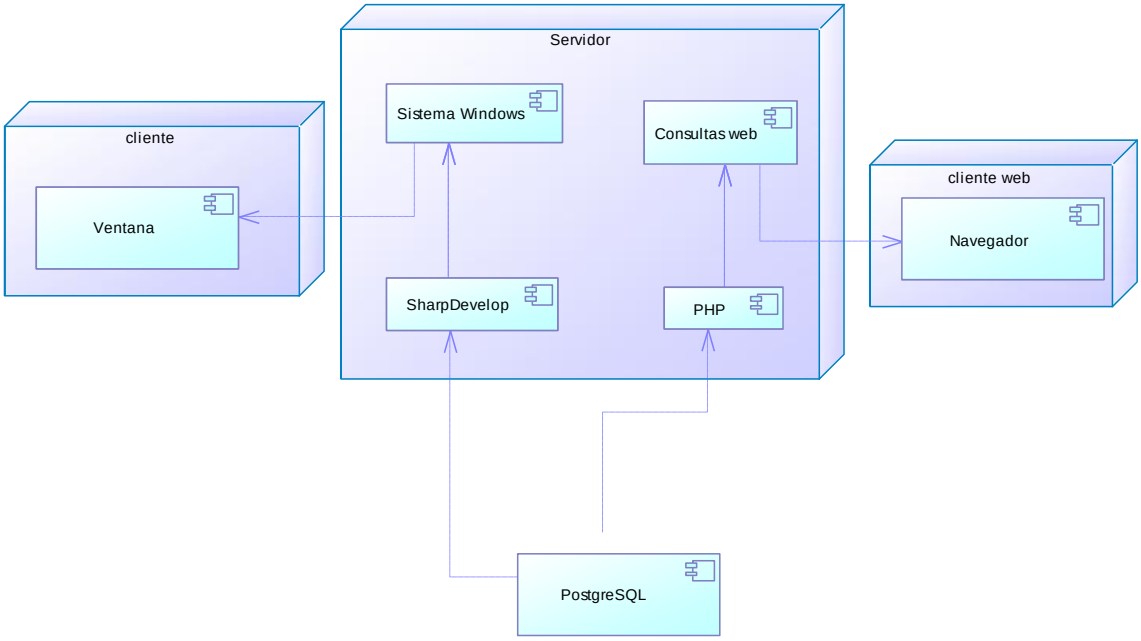


Figura 6.21 Diagrama de despliegue

6.9.2 Diseño del sistema

6.9.2.1 Diseño de la base de datos

Realizado un análisis del proceso registro y sanción de los jugadores que se realiza en la LIGA DEPORTIVA PARROQUIAL HUACHI GRANDE se ha modelado la base de datos de la siguiente manera:

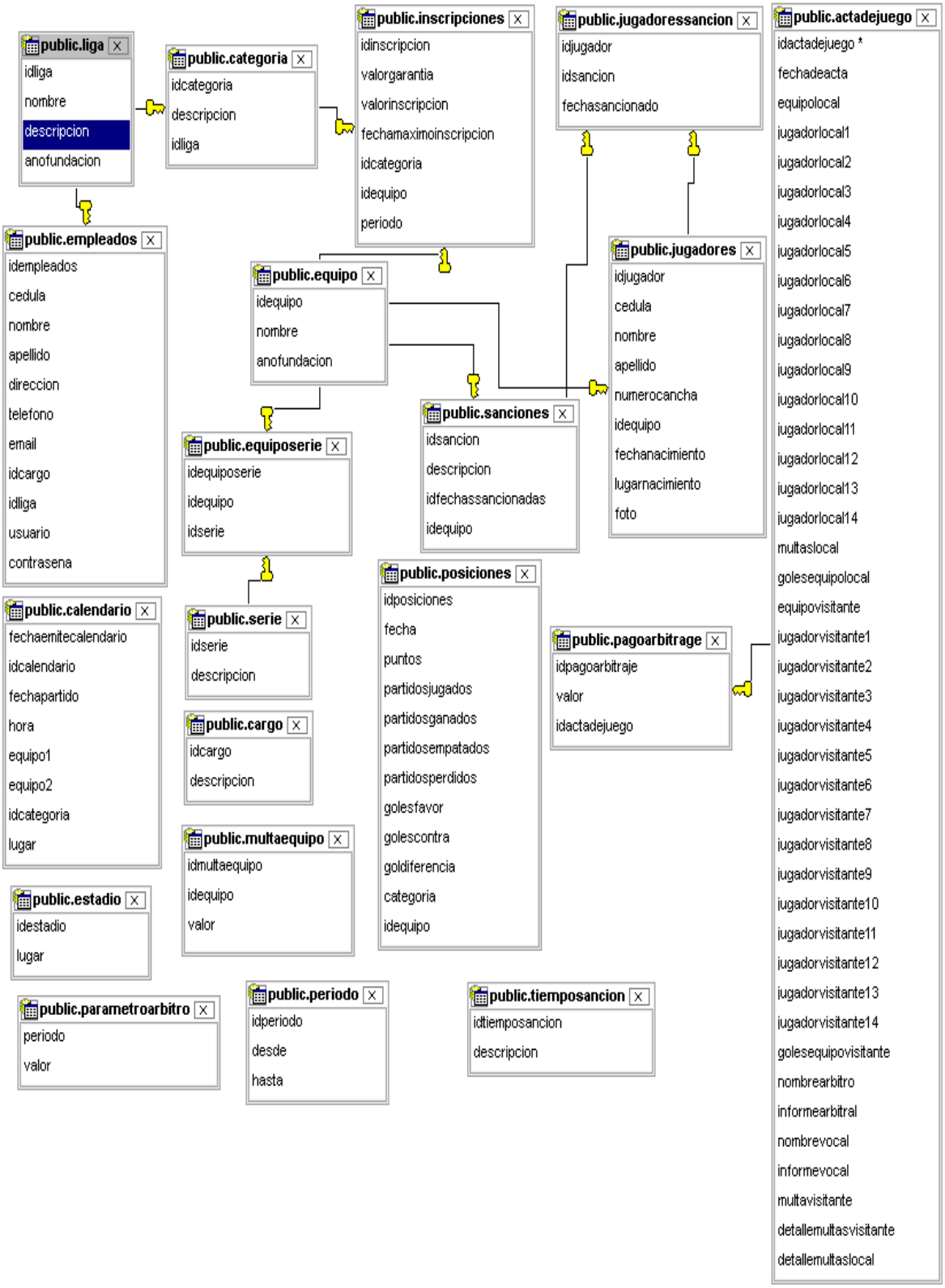


Figura 6.22 Modelo relacional lógico.

6.9.2.2 Diccionario de datos

TABLA: Liga			
CAMPO	**TIPO**	**DESCRIPCIÓN**	**RESTRICCIÓN**
idliga	Serial	Código de la liga	Primary key
nombre	Character(100)	Nombre de la liga	Not null
descripcion	Character(200)	Nombre que identifica descripción sobre la liga	Null
anofundacion	Character(4)	Nombre que identifica ano de fundación sobre la liga	Nnull

Tabla 6.14. Descripción de la tabla liga

TABLA: Empleados			
CAMPO	**TIPO**	**DESCRIPCIÓN**	**RESTRICCIÓN**
idempleados	Serial	Código de los empleados	Primary key
cedula	Character(10)	Cedula del empleado	Not null
nombre	Character(50)	Nombre que identifica el nombre sobre empleados	Not null
apellido	Character(50)	Nombre que identifica el apellido sobre empleados	Not null
direccion	Character(100)	Nombre que identifica la dirección sobre empleados	Not null
telefono	Character(12)	Nombre que identifica el teléfono sobre empleados	Null
email	Character(100)	Nombre que identifica el email sobre empleados	Null
usuario	Character(50)	Login de usuario	Not null
contraseña		Contraseña del empleado	Not null
idliga	Character(50)	Código de la liga	Not null, Foreign Key(tabla liga)

Tabla 6.15. Descripción de la tabla empleados

TABLA: Categoría			
CAMPO	**TIPO**	**DESCRIPCIÓN**	**RESTRICCIÓN**
idcategoria	Serial	Código de la categoría	Primary key
descripcion	Character(10)	descripción de la categoría	Not null
idliga	Character(50)	Código de la liga	Not null, Foreign Key(tabla liga)

Tabla 6.16. Descripción de la tabla categoría

TABLA: Equipo			
CAMPO	**TIPO**	**DESCRIPCIÓN**	**RESTRICCIÓN**
idequipo	Serial	Código del equipo	Primary key
nombre	Character(50)	Nombre del equipo	Not null
anofundacion	Character(40)	Año de fundación del equipo	Null

Tabla 6.17. Descripción de la tabla equipo

TABLA: Periodo			
CAMPO	**TIPO**	**DESCRIPCIÓN**	**RESTRICCIÓN**
idperiodo	Serial	Código del Periodo	Primary key
desde	Date	Fecha de inicio del periodo	Not null
hasta	Date	Fecha que termina el periodo	Not null

Tabla 6.18. Descripción de la tabla periodo

TABLA: Inscripciones			
CAMPO	**TIPO**	**DESCRIPCIÓN**	**RESTRICCIÓN**
idinscripcion	Serial	Código de la inscripcion	Primary key
valorgarantia	Numeric	Valor de la garantía	Not null
valorinscripcion	Numeric	Valor de la inscripción	Not null
fechamaximoinscripcion	Date	Especificacion hasta cuando pueden inscribirse	Not null
idcategoria	Integer	Código de la categoría	Not null, Foreign Key(tabla categoria)
idequipo	Integer	Código del equipo	Not null, Foreign Key(tabla equipo)
periodo	Carácter(500)	Especifica en que periodo esta actualmente	Not null

Tabla 6.19. Descripción de la tabla inscripciones

TABLA: Jugadores			
CAMPO	**TIPO**	**DESCRIPCIÓN**	**RESTRICCIÓN**
idjugadores	Serial	Código del jugador	Primary key
cedula	character(10)	Cedula que identifica al jugador	Not null
nombre	character(100)	Nombre que identifica al jugador	Not null
apellido	character(100)	Apellido que identifica al jugador	Not null
numerocancha	Integer	Numero que identifica en la cancha al jugador	Not null
idequipo	Integer	Código del equipo al que pertenece el jugador	Not null, Foreign Key(tabla jugadores)
fechanacimiento	Date	Fecha en que ha nacido el jugador	Not null
lugarnacimiento	character(100)	Lugar donde nació el jugador	Not null
foto	character(300)	Foto que identifica al jugador	Not null

Tabla 6.20. Descripción de la tabla jugadores

TABLA: Tiempo Sanción			
CAMPO	**TIPO**	**DESCRIPCIÓN**	**RESTRICCIÓN**
idtiemposancion	Serial	Código del tiempo sanción	Primary key
descripcion	Date	Descripción del tiempo que sale sancionado	Not null

Tabla 6.21. Descripción de la tabla **T**iempo Sanción

TABLA: Sanción			
CAMPO	**TIPO**	**DESCRIPCIÓN**	**RESTRICCIÓN**
idsancion	Serial	Código de la sanción	Primary key
descripcion	Date	Descripción de la sanción	Not null
idfechassancionadas	Integer	Código de tiemposancion	Not null, Foreign Key(tabla tiempo sancion)
idequipo	Integer	Código del equipo	Not null, Foreign Key(tabla equipo)

Tabla 6.22. Descripción de la tabla Sanción

TABLA: Jugadores Sanción			
CAMPO	**TIPO**	**DESCRIPCIÓN**	**RESTRICCIÓN**
idjugador	Integer	Código del jugador	Primary key, Foreign Key(tabla jugadores)
idsancion	Integer	Código de la sanción	Primary key, Foreign Key(tabla sanciones)
fechasancionado	Date	Código de tiemposancion	Primary key

Tabla 6.23. Descripción de la tabla Jugadores Sanción

TABLA: Estadio			
CAMPO	**TIPO**	**DESCRIPCIÓN**	**RESTRICCIÓN**
idestadio	Serial	Código del estadio	Primary key
lugar	character(500)	Descripción donde queda el estadio	Not null

Tabla 6.24. Descripción de la tabla Estadio

TABLA: Parámetro Arbitro			
CAMPO	**TIPO**	**DESCRIPCIÓN**	**RESTRICCIÓN**
periodo	character(100)	Código del parámetro arbitro	Primary key
valor	numeric(4,2)	Valor del costo del arbitraje	Not null

Tabla 6.25. Descripción de la tabla Parámetro Arbitro

TABLA: Calendario			
CAMPO	**TIPO**	**DESCRIPCIÓN**	**RESTRICCIÓN**
idcalendario	Integer	Código del parámetro arbitro	Primary key
fechaemitecalendario	Date	Valor del costo del arbitraje	Not null
fechapartido	Date	Código de la acta de juego	Not null, foreign key(tabla acta de juego)
hora	Interval	Hora del partido	Not null
equipo1	Integer	Código del equipo local	Not null, foreign key(tabla jugadores)
equipo2	Integer	Código del equipo visitante	Not null, foreign key(tabla jugadores)
idcategoria	Integer	Código de la categoría	Not null, foreign

			key(tabla categoria)
lugar	Integer	Código del estadio	Not null, foreign key(tabla estadio)

Tabla 6.26. Descripción de la tabla calendario

TABLA: Acta de juego			
CAMPO	**TIPO**	**DESCRIPCIÓN**	**RESTRICCIÓN**
Idactadejuego	Integer	Código del acta de juego	Primary key
Fechadeacta	Date	Fecha en que se emite al acta	Not null
Equipolocal	character(50)	Descripción del equipo local	Not null
jugadorlocal1	Integer	Código del jugador local	Not null, foreign key(tabla jugadores)
jugadorlocal2	Integer	Código del jugador local	Not null, foreign key(tabla jugadores)
Jugadorlocal3	Integer	Código del jugador local	Not null, foreign key(tabla jugadores)
Jugadorlocal4	Integer	Código del jugador local	Not null, foreign key(tabla jugadores)
Jugadorlocal5	Integer	Código del jugador local	Not null, foreign key(tabla jugadores)
Jugadorlocal6	Integer	Código del jugador local	Not null, foreign key(tabla jugadores)
Jugadorlocal7	Integer	Código del jugador local	Not null, foreign key(tabla jugadores)
Jugadorlocal8	Integer	Código del jugador local	Not null, foreign key(tabla jugadores)
Jugadorlocal9	Integer	Código del jugador local	Not null, foreign key(tabla jugadores)
Jugadorlocal10	Integer	Código del jugador local	Not null, foreign key(tabla jugadores)
Jugadorlocal11	Integer	Código del jugador local	Not null, foreign key(tabla jugadores)
Jugadorlocal12	Integer	Código del jugador local	Not null, foreign key(tabla jugadores)
Jugadorlocal13	Integer	Código del jugador local	Not null, foreign key(tabla jugadores)
Jugadorlocal14	Integer	Código del jugador local	Not null, foreign key(tabla jugadores)
multaslocal	numeric(4,2)	Multas que tiene el equipo que hace de local	Not null
golesequipolocal	Integer	Goles que ha convertido el equipo local	Not null
equipovisitante	character(50)	Descripción del equipo visitante	Not null
jugadorvisitante1	Integer	Código del jugador visitante	Not null, foreign key(tabla jugadores)
Jugadorvisitante2	Integer	Código del jugador visitante	Not null, foreign key(tabla jugadores)
Jugadorvisitante3	Integer	Código del jugador visitante	Not null, foreign key(tabla jugadores)
Jugadorvisitante4	Integer	Código del jugador visitante	Not null, foreign key(tabla jugadores)
Jugadorvisitante5	Integer	Código del jugador visitante	Not null, foreign key(tabla jugadores)

Jugadorvisitante6	Integer	Código del jugador visitante	Not null, foreign key(tabla jugadores)
Jugadorvisitante7	Integer	Código del jugador visitante	Not null, foreign key(tabla jugadores)
Jugadorvisitante8	Integer	Código del jugador visitante	Not null, foreign key(tabla jugadores)
Jugadorvisitante9	Integer	Código del jugador visitante	Not null, foreign key(tabla jugadores)
jugadorvisitante10	Integer	Código del jugador visitante	Not null, foreign key(tabla jugadores)
jugadorvisitante11	Integer	Código del jugador visitante	Not null, foreign key(tabla jugadores)
jugadorvisitante12	Integer	Código del jugador visitante	Not null, foreign key(tabla jugadores)
jugadorvisitante13	Integer	Código del jugador visitante	Not null, foreign key(tabla jugadores)
jugadorvisitante14	Integer	Código del jugador visitante	Not null, foreign key(tabla jugadores)
golesequipovisitante	Integer	Goles que ha convertido el equipo visitante	Not null
multavisitante	numeric(4,2)	Multas que tiene el equipo que hace de visitante	Not null
nombrearbitro	character(300)	Nombre y apellido del árbitro que dirige el encuentro	Not null
informearbitral	character(1000)	informe del árbitro que dirige el encuentro	Not null
nombrevocal	character(300)	Nombre y apellido del vocal que actúa en el encuentro	Not null
informevocal	character(2000)	Informe del vocal que actúa en el encuentro	Not null
detallemultasvisitante	character(5000)	Detalles de las multas del equipo visitante	Not null
detallemultaslocal	character(5000)	Detalles de las multas del equipo local	Not null

Tabla 6.27. Descripción de la tabla acta de juego

TABLA: Pago Arbitraje			
CAMPO	**TIPO**	**DESCRIPCIÓN**	**RESTRICCIÓN**
idpagoarbitraje	Integer	Código del parámetro arbitro	Primary key
valor	numeric(4,2)	Valor del costo del arbitraje	Not null
idactadejuego	Integer	Código de la acta de juego	Not null, foreign key(tabla acta de juego)

Tabla 6.28. Descripción de la tabla Pago Arbitr**aje**

TABLA: Cargo			
CAMPO	**TIPO**	**DESCRIPCIÓN**	**RESTRICCIÓN**
idcargo	Serial	Código del cargo	Primary key
descripcion	character(100)	Descripción del cargo	Not null

Tabla 6.29. Descripción de la tabla Cargo

TABLA: Serie			
CAMPO	**TIPO**	**DESCRIPCIÓN**	**RESTRICCIÓN**
idserie	Serial	Código de la serie	Primary key
descripcion	character(100)	Descripción de la serie	Not null

Tabla 6.30. Descripción de la tabla Serie

TABLA: Equipo Serie			
CAMPO	**TIPO**	**DESCRIPCIÓN**	**RESTRICCIÓN**
idequiposerie	Serial	Código del equipo serie	Primary key
idequipo	integer	Código del equipo	Not null, foreign key(tabla equipo)
idserie	Integer	Código de la serie	Not null, foreign key(tabla serie)

Tabla 6.31. Descripción de la tabla equiposerie

TABLA: Multa Equipo			
CAMPO	**TIPO**	**DESCRIPCIÓN**	**RESTRICCIÓN**
idmultaequipo	Serial	Código de la multa del equipo	Primary key
idequipo	Integer	Código del equipo	Not null, foreign key(tabla equipo)
valor	numeric(4,2)	Valor de las multas e los equipos	Not null

Tabla 6.32. Descripción de la tabla multaequipo

TABLA: Posiciones			
CAMPO	**TIPO**	**DESCRIPCIÓN**	**RESTRICCIÓN**
idposiciones	Serial	Código de la posición	Primary key
fecha	Date	Fecha de la posición	Primary key
puntos	Integer	Puntos de los equipo	Not null
partidosjugados	integer	Partidos jugados que tiene el equipo	Not null
partidosganados	Integer	Partidos ganados que tiene el equipo	Not null
partidosempatados	Integer	Partidos empatados que tiene el equipo	Not null
partidosperdidos	integer	Partidos empatados que tiene el equipo	Not null
golesfavor	integer	Goles a favor que tiene el equipo	Not null
golescontra	Integer	Goles en contra que tiene el equipo	Not null
goldiferencia	Integer	Goles diferencia que tiene el equipo	Not null
categoria	Integer	Código de la categoría	Not null

idequipo	integer	Código del equipo	Not null

Tabla 6.33. Descripción de la tabla posiciones

6.9.2.3 Diseño de la interfaz

Para el diseño de la interfaz del sistema registro y sanción a los jugadores se ha tomado como referencia los requerimientos de la LIGA DEPORTIVA PARROQUIAL HUAHI GRANDE de que ésta deba ser sencilla y amigable con el usuario.

Pantalla inicio de sesión

En este sistema automatizado la persona que desee entrar al sistema de registro y sanción de los jugadores deberá ingresar su usuario y contraseña obtenidos previamente en su registro, estos datos serán validados y permitirán su acceso con los respectivos permisos.

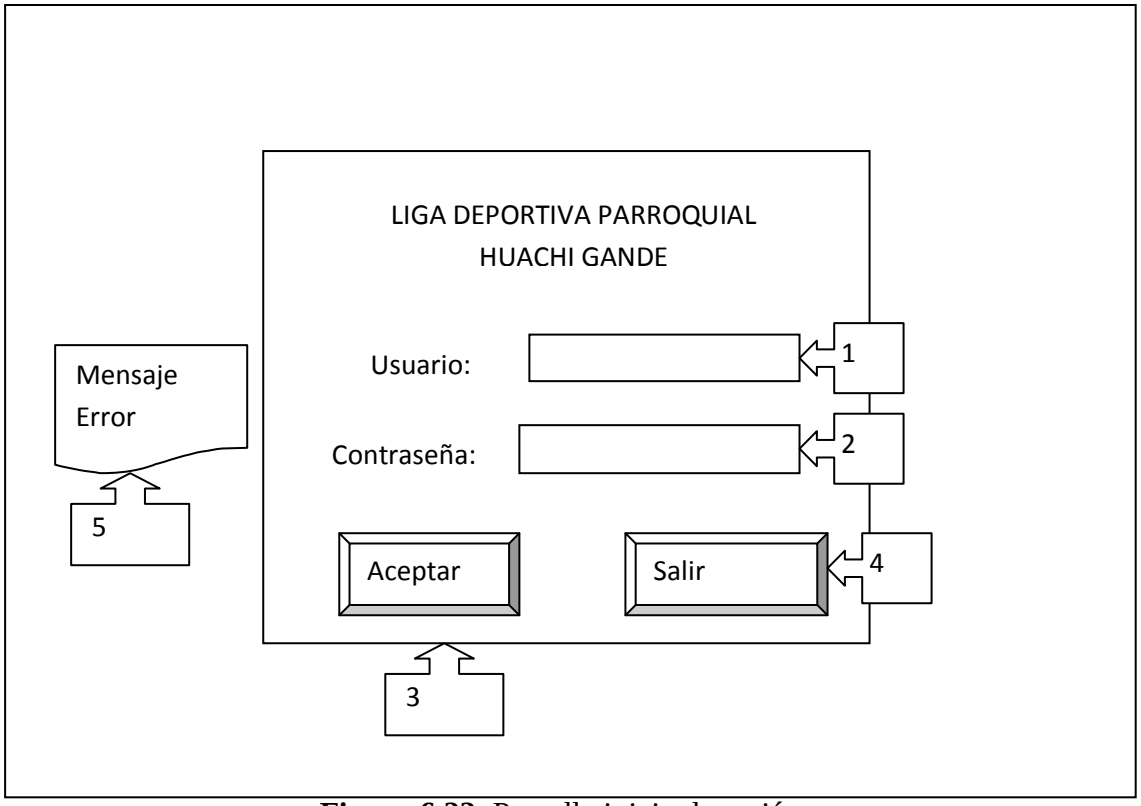


Figura 6.22. Pantalla inicio de sesión

1. Caja de texto que permite el ingreso del nombre de usuario
2. Caja de texto que permite el ingreso de la contraseña
3. Botón Aceptar: valida los parámetros ingresados en las cajas de texto y posteriormente si los datos son correctos re-direcciona al Sistema de registro y sanciones de los jugadores.
4. Botón Salir: Sale del sistema.
5. Etiqueta Mensaje Error: permite la visualización de los errores cometidos en sistema automatizado.

6.9.2.3.1 Diseño de salidas

Ventana principal del sistema de registro y sanciones

Cuando el usuario acaba de iniciar su sesión se le presenta el siguiente menú contenedor con los menús correspondientes:

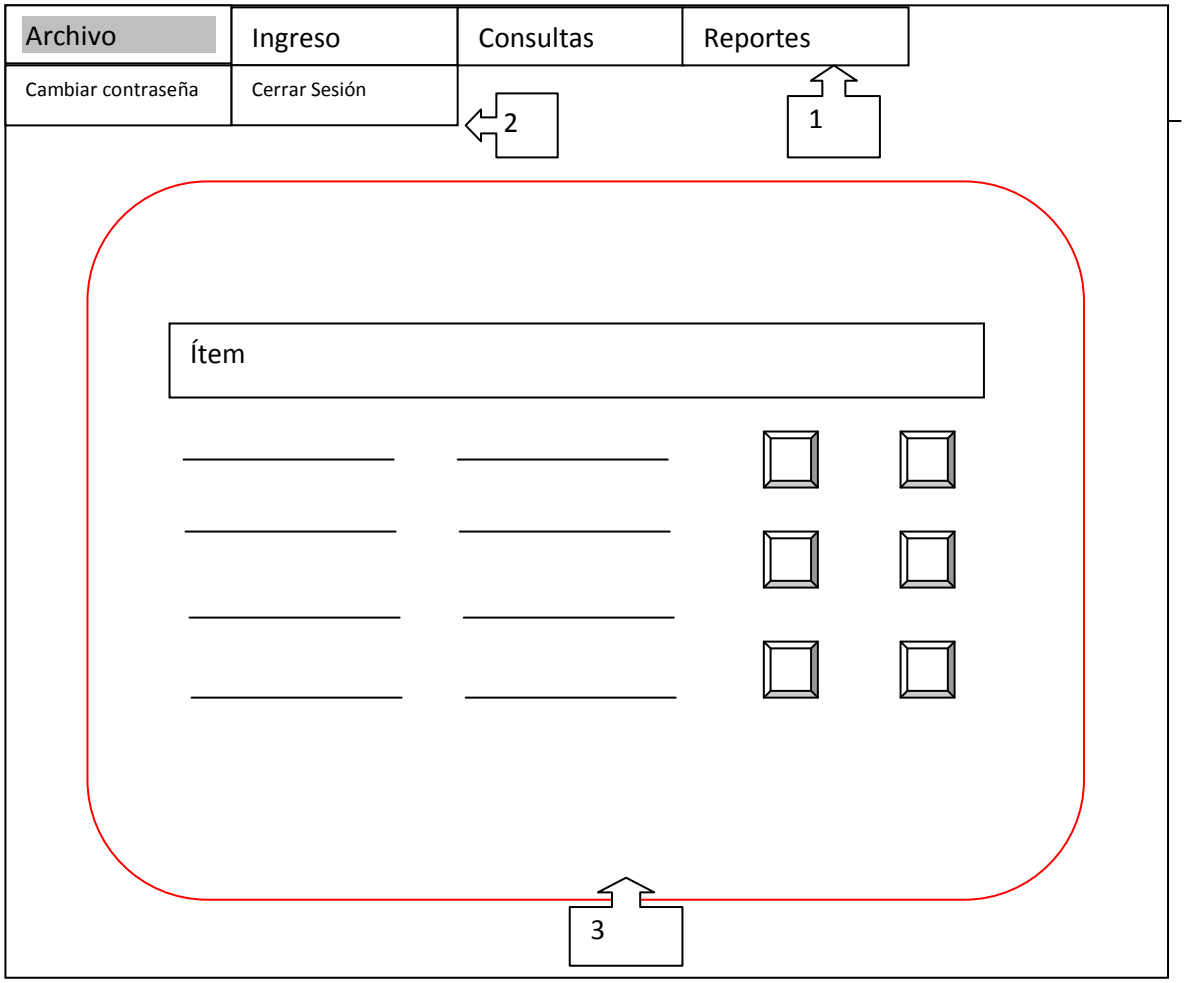


Figura 6.23. Ventana inicial

1. Menú principal. Se podrá escoger 4 opciones como: Archivo, Ingresos, Consultas, Reportes.
2. Submenú Archivo. Resultante de la selección “Archivo” del menú principal, en éste se podrá escoger 2 opciones como: Cambiar Clave, Cerrar Sesión.
    a. Cambiar Clave. Muestra información los campos necesarios para que la clave pueda ser modificada por el usuario conectado.
    b. Cerrar Sesión. Cierra la sesión del usuario que está conectado en el momento.
3. Ventana de contenido. Espacio destinado para la gestión de información. La información mostrada dependerá directamente de las opciones marcadas en los menús.

Para la administración de cada uno de los ingresos el sistema cuenta con:

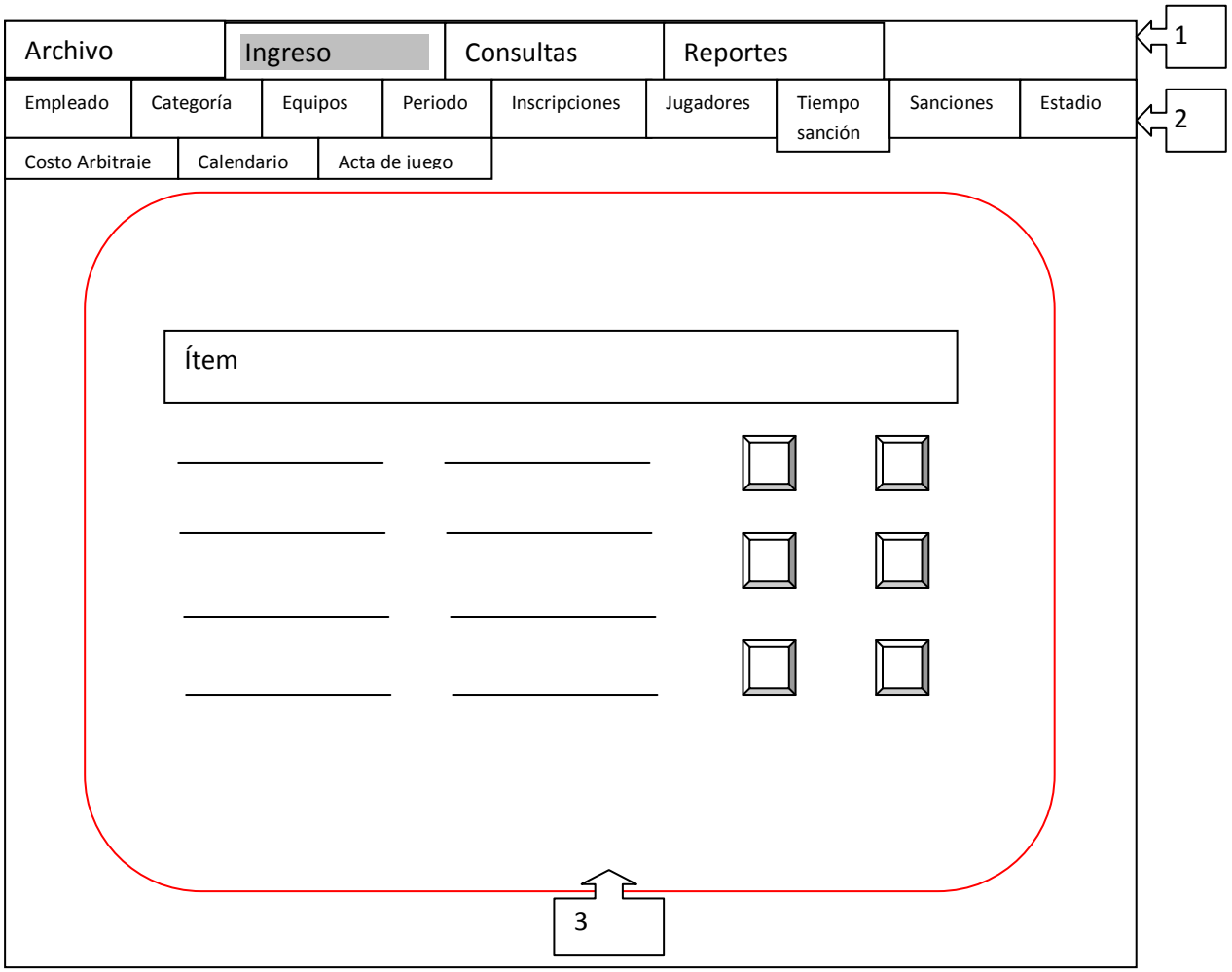


Figura 6.24. Ventana ingreso de datos

1. Menú principal. Se podrá escoger 4 opciones como: Archivo, Ingresos, Consultas, Reportes. (Seleccionado Ingresos)
2. Submenú Ingresos. Resultante de la selección “Ingresos” del menú principal, en éste se podrá escoger 12 opciones como: Empleados, categorías, equipos, periodo, inscripciones, jugadores, tiempo sanción, sanciones, estadio, costo arbitraje, calendario, acta de juego.

    a. Empleados. Muestra información del empleado seleccionado con anterioridad
    b. Categorías. Muestra información de las categorías que tiene la Liga seleccionado con anterioridad
    c. Equipos. Muestra información del equipo seleccionado con anterioridad.
    d. Periodo. Muestra información del periodo que está en vigencia en la actualidad.
    e. Inscripciones. Muestra información de la inscripción del equipo seleccionado con anterioridad.
    f. Jugadores. Muestra información del jugador que ha seleccionado con anterioridad.
    g. Tiempo Sanción. Muestra información del tiempo de sanción que ha seleccionado con anterioridad.
    h. Sanciones. Muestra información de la sanción que ha seleccionado con anterioridad.
    i. Estadio. Muestra información del estadio que ha seleccionado con anterioridad.
    j. Costo Arbitraje. Muestra información del costo del arbitraje que está vigente para el periodo actual.
    k. Calendario. Muestra información del calendario que está en la última semana vigente del periodo actual.
    l. Acta de juego. Muestra información del acta de juego que se ha registrado en el último partido.

3. Ventana de contenido. Espacio destinado para la gestión de información. La información mostrada dependerá directamente de las opciones marcadas en los menús.

Para la administración de cada uno de las consultas el sistema cuenta con:

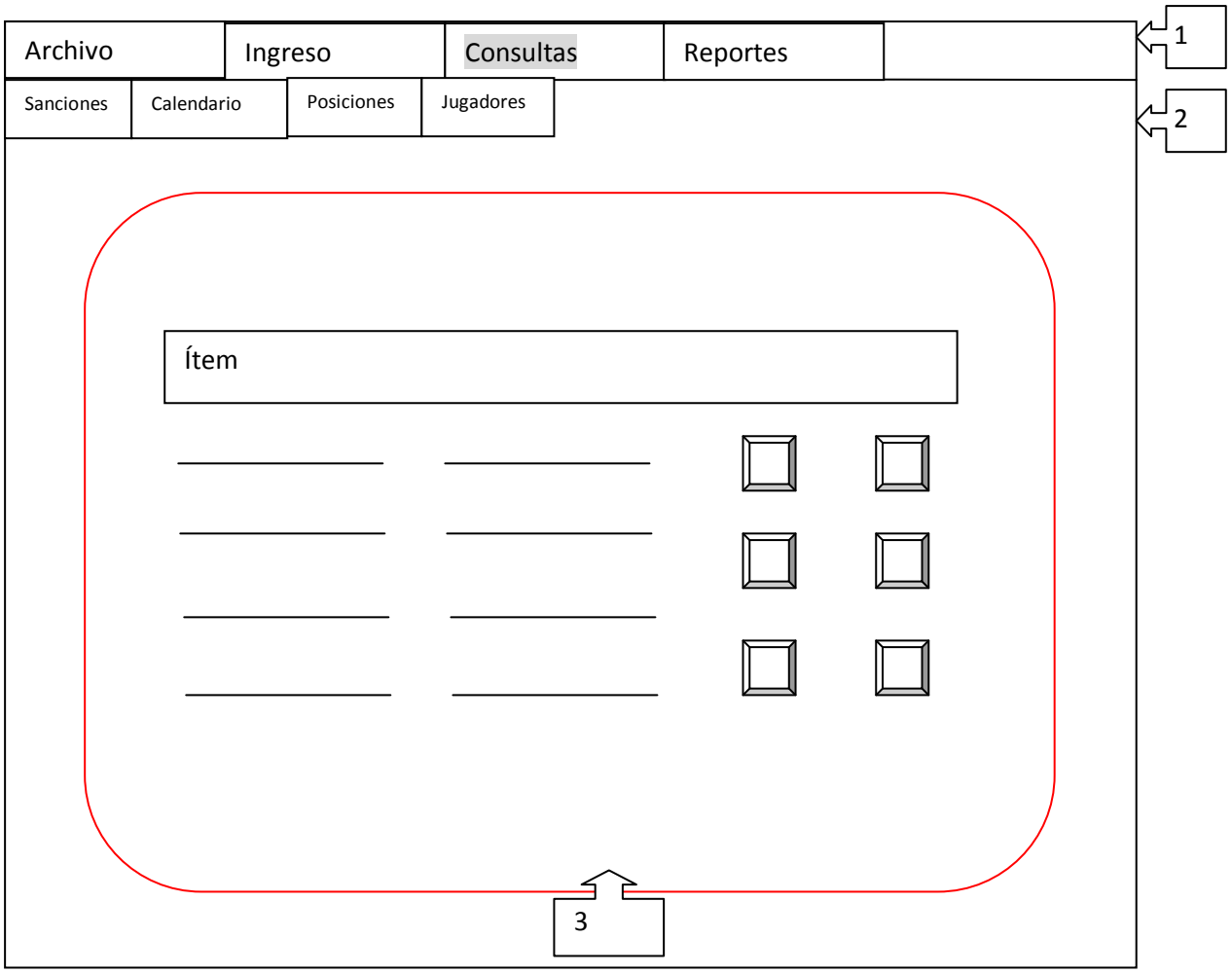


Figura 6.25. Ventana consulta de datos

1. Menú principal. Se podrá escoger 4 opciones como: Archivo, Ingresos, Consultas, Reportes. (Seleccionado Consultas)
2. Submenú Archivo. Resultante de la selección "Consultas" del menú principal, en éste se podrá escoger 4 opciones como: Sanciones, Calendario, Posiciones, Jugadores.
    a. Sanciones. Muestra información una consulta de las sanciones que ha seleccionado con anterioridad.
    b. Calendario. Muestra una consulta del último calendario que está vigente para la semana.
    c. Posiciones. Muestra una consulta de la tabla de posiciones que ha seleccionado con anterioridad.
    d. Jugadores. Muestra una consulta del jugador que ha seleccionado con anterioridad.

3. Ventana de contenido. Espacio destinado para la gestión de información. La información mostrada dependerá directamente de las opciones marcadas en los menús.

Para la administración de cada uno de los Reportes el sistema cuenta con:

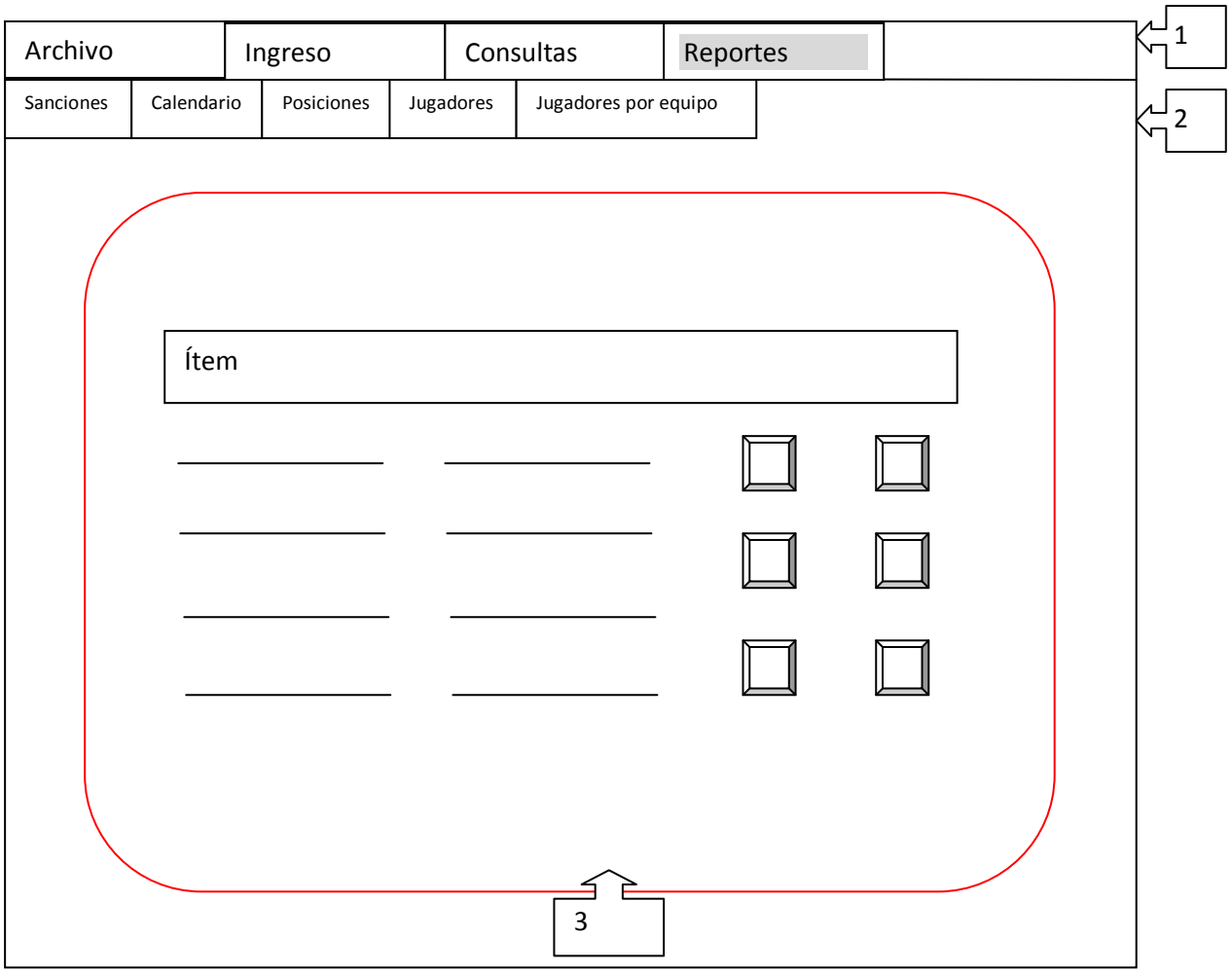


Figura 6.26. Ventana Salida de Reportes

1. Menú principal. Se podrá escoger 5 opciones como: Archivo, Ingresos, Consultas, Reportes. (Seleccionado Reportes)
2. Submenú Archivo. Resultante de la selección "Reportes" del menú principal, en éste se podrá escoger 5 opciones como: Sanciones, Calendario, Posiciones, Jugadores, Jugadores por equipo.
    a. Sanciones. Muestra un reporte de las sanciones que ha seleccionado con anterioridad.
    b. Calendario. Muestra un reporte del último calendario que está vigente para la semana.

c. Posiciones. Muestra un reporte de la tabla de posiciones que ha seleccionado con anterioridad.
d. Jugadores. Muestra un reporte del jugador que ha seleccionado con anterioridad.
e. Jugadores por equipo. Muestra un reporte de los jugadores de un equipo que ha seleccionado con anterioridad.

3. Ventana de contenido. Espacio destinado para la gestión de información. La información mostrada dependerá directamente de las opciones marcadas en los menús.

Página principal del sistema de registro y sanciones

Cuando el usuario acaba de ingresar al sitio web se le presenta el siguiente menú contenedor con los menús correspondientes:

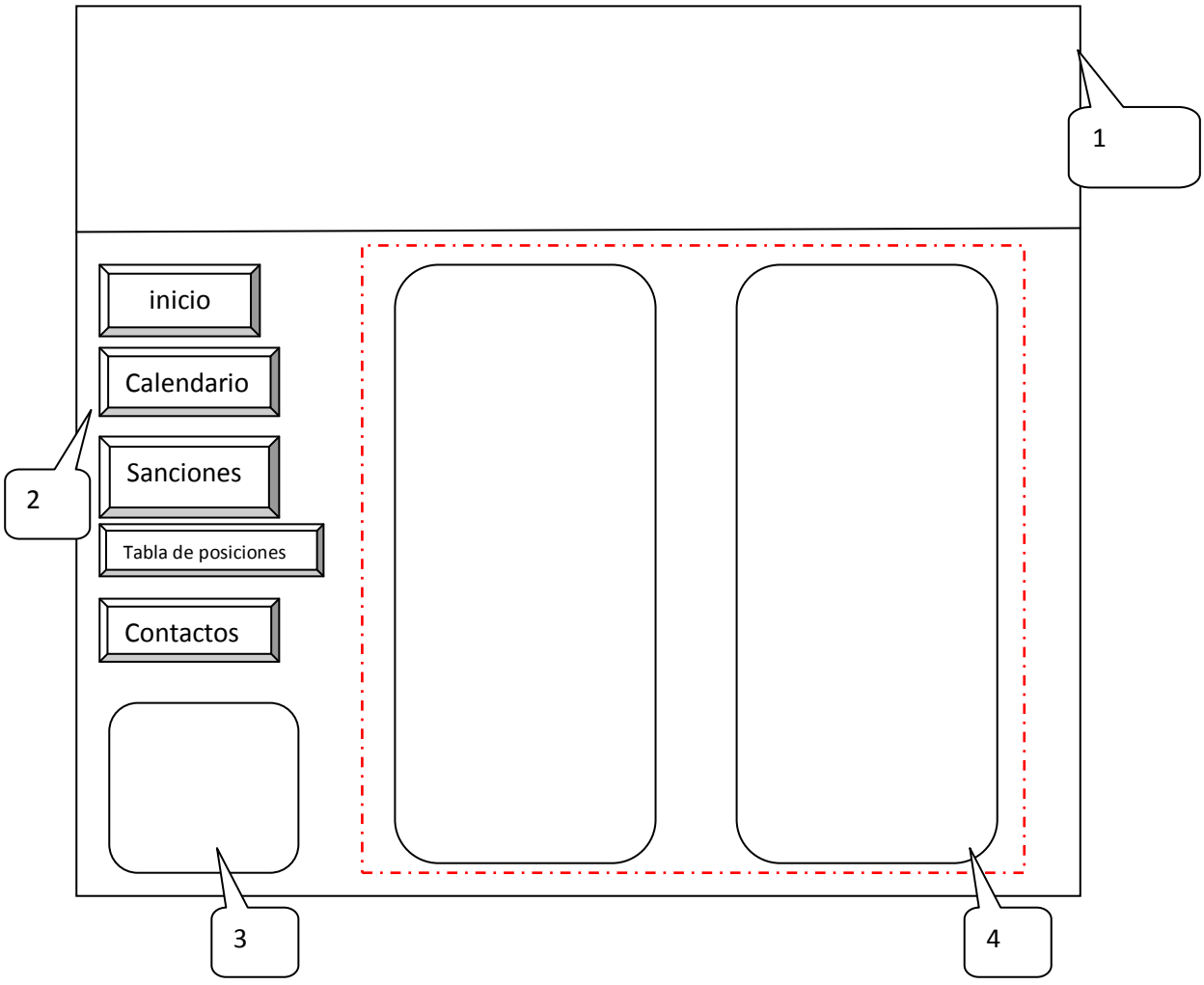


Figura 6.27. Ventana consulta de datos en la web

1. Banner animado donde se identifica el logo de la LIGA DEPORTIVA PARROQUIAL HUACHI GRANDE
2. Menú principal. Se podrá escoger 5 opciones como: Inicio, Calendario, Sanciones, Tabla de posiciones, Contactos.
    a. Inicio. Muestra información de la liga.
    b. Calendario. Muestra el último calendario vigente para la fecha a jugarse que ha seleccionado con anterioridad.
    c. Sanciones. Muestra las sanciones que ha seleccionado con anterioridad.
    d. Tabla de posiciones. Muestra la tabla de posiciones que ha seleccionado con anterioridad.
    e. Contactos. Muestra información de donde está ubicada la Liga, además podrán recibir mensajes a un correo electrónico de la Liga.
3. Contador de Visitas. Este contador de visitas nos sirve para saber cuántas personas ingresan a la página web.
4. Ventana de contenido. Espacio destinado para la gestión de información. La información mostrada dependerá directamente de las opciones marcadas en los menús.

Manipulación de información

Esta es la descripción general para las ventanas de manipulación de datos:

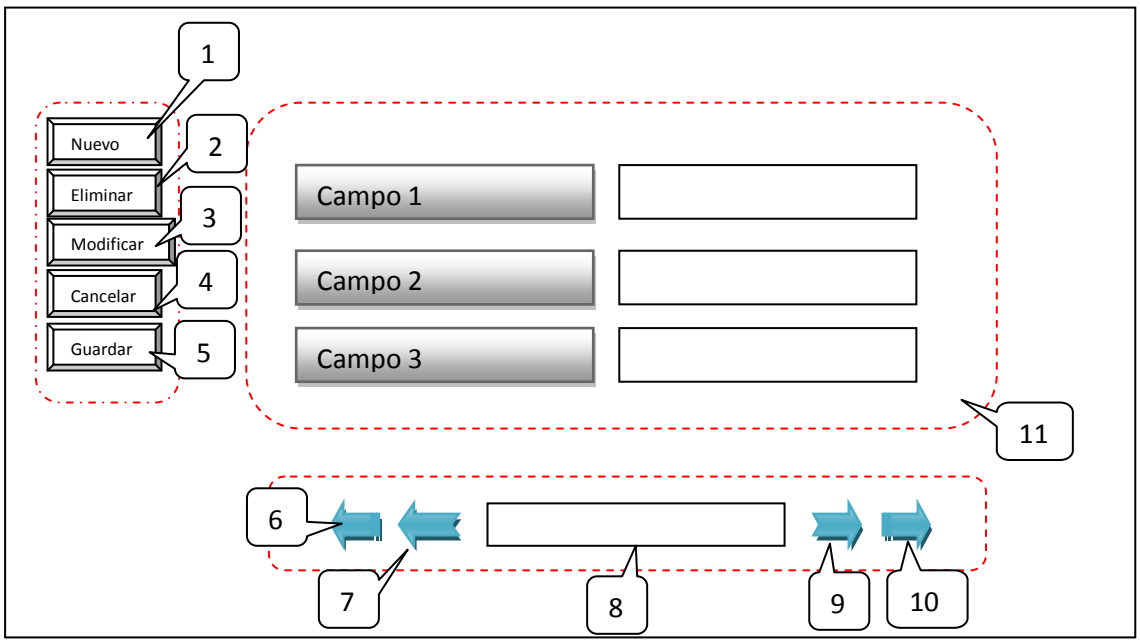


Figura 6.28. Manipulación de la información

1. Botón Nuevo. Habilita los campos para poder ingresar la información deseada.
2. Botón Eliminar. Elimina el registro que ha seleccionado con anterioridad.
3. Botón Modificar. Habilita los campos que pueden ser manipulados para actualizarlos una vez que ha seleccionado con anterioridad.
4. Botón Cancelar. Cancela cualquier operación que esté realizando bloqueando los campos.
5. Botón Guardar. Guarda los cambios que se haya realizado.
6. Botón Último. Muestra el primer registro.
7. Botón Anterior. Muestra el anterior registro.
8. Texto de Registros. Muestra el total de registros y en que registro está en la actualidad.
9. Botón Siguiente. Muestra el siguiente registro.
10. Botón Último. Muestra el último registro.

1. Espacio de manipulación. Permite ejecutar operaciones con la información traída de la Base de Datos como: Editar, Eliminar.
    a. Eliminar. Elimina el registro seleccionado
    b. Modificar. Envía la información del registro seleccionado a otra página web para su modificación.

6.9.3 Implementación

6.9.3.1 Extracto de código fuente

ConectarBD.cs

```
public static class ConectarBD
{
        #region Datos
        private static string _usuario;
        private static string _contraseña;
        private static string _cadenaDeConexion;
        private static int _idempleado;
        private static string _cargo;
        #endregion

       #region Propiedades
       public static string cargo
       {
```

```
        get
        {
            return _cargo;
        }
        set
        {
            _cargo = value;
        }
    }

public static int idempleado
{
        get
        {
            return _idempleado;
        }
        set
        {
            _idempleado = value;
        }
}
public static string Usuario
{
    get
    {
        return _usuario;
    }
    set
    {
        _usuario = value;
    }
}
public static string Contraseña
{
```

```
        get
        {
            return _contraseña;
        }
        set
        {
            _contraseña = value;
        }
    }

    public static string CadenaDeConexion
    {
        get
        {
            return _cadenaDeConexion;
        }
        set
        {
            _cadenaDeConexion = value;
        }
    }
    #endregion
}
```

Código para iniciar sesión

```
//Parametros para la conexion
ConectarBD.Usuario = txtUsuario.Text;
ConectarBD.Contraseña = txtContrasena.Text;

//Cadena de conexion
ConectarBD.CadenaDeConexion = "Server=localhost; Database=ldphg;
Port=5432;User Id=postgres;Password=sa";
//instanciar una variable para el databasehelper
```

```
DatabaseHelper seleccionarUsuarioContrasena = new
DatabaseHelper(ConectarBD.CadenaDeConexion,Providers.PostgresSQL);
//añadir parametro para el procedimiento de la clave
seleccionarUsuarioContrasena.AddParameter("@usuario",txtUsuario.Text);

DataTable dtContrasena =
(seleccionarUsuarioContrasena.ExecuteDataSet("seleccionarcontrasenaporusuario",Co
mmandType.StoredProcedure)).Tables[0];

//instanciar variable para el procedimiento del cargo
DatabaseHelper seleccionarCargoPorUsuario = new
DatabaseHelper(ConectarBD.CadenaDeConexion,Providers.PostgresSQL);
//añadir parametro para el procedimiento del cargo
seleccionarCargoPorUsuario.AddParameter("@usuario",txtUsuario.Text.Trim());
DataSet dsCargo =
(seleccionarCargoPorUsuario.ExecuteDataSet("seleccionarcargoporusuario",Command
Type.StoredProcedure));
int totalRegistros =  dtContrasena.Rows.Count;
if (totalRegistros > 0)
{
        if (txtContrasena.Text.Trim() ==
dtContrasena.Rows[0]["contrasena"].ToString().Trim())
        {
                //Compara que cargo ocupa
                if (dsCargo.Tables[0].Rows[0]["descripcion"].ToString().Trim() ==
"ADMINISTRADOR")
                {
                        frmMenu formaMenu = new frmMenu();
                        formaMenu.Show();
                        this.Hide();
                }

         }

}
```

Código para los empleados

```
//Instanciar una variable de tipo databasehelper para
//poder ejecutar un procedimiento de la BD
 DatabaseHelper recuperar = new DatabaseHelper(ConectarBD.CadenaDeConexion,
Providers.PostgresSQL);
 //Seleccionar datos de los empleados
dsDatos = recuperar.ExecuteDataSet("seleccionarempleado",
CommandType.StoredProcedure);
 //Instanciar una variable para la liga
 DatabaseHelper recuperarLiga = new
DatabaseHelper(ConectarBD.CadenaDeConexion,Providers.PostgresSQL);
 dsLiga =
recuperarLiga.ExecuteDataSet("seleccionarliga",CommandType.StoredProcedure);
 MostrarDatosEnControles();
 MostrarDatosComboBox();
 MoverComboBox();

private void MostrarDatosEnControles()
{
        //Operación para saber en qué registró nos encontramos
        registroActual = this.BindingContext[dsDatos.Tables[0]].Position;
        //Operación para saber cuantos registros se encuentran en esta tabla de la BD
        totalRegistros = this.BindingContext[dsDatos.Tables[0]].Count;
        //Condicion para saber si hay datos y poderlos mostrar
        if (totalRegistros > 0)
        {
                //Mostrar los datos en los controles con los respectivos parametros
                txtIdEmpleado.Text =
dsDatos.Tables[0].Rows[registroActual]["idempleados"].ToString();
                txtApellido.Text =
dsDatos.Tables[0].Rows[registroActual]["apellido"].ToString();
```

```
                txtCedula.Text =
dsDatos.Tables[0].Rows[registroActual]["cedula"].ToString();
                txtContrasena.Text =
dsDatos.Tables[0].Rows[registroActual]["contrasena"].ToString();
                txtDireccion.Text =
dsDatos.Tables[0].Rows[registroActual]["direccion"].ToString();
                txtEmail.Text =
dsDatos.Tables[0].Rows[registroActual]["email"].ToString();
                txtNombre.Text =
dsDatos.Tables[0].Rows[registroActual]["nombre"].ToString();
                txtTelefono.Text =
dsDatos.Tables[0].Rows[registroActual]["telefono"].ToString();
                txtUsuario.Text =
dsDatos.Tables[0].Rows[registroActual]["usuario"].ToString();
                txtLiga.Text =
dsDatos.Tables[0].Rows[registroActual]["idliga"].ToString();
                txtIdCargo.Text =
dsDatos.Tables[0].Rows[registroActual]["idcargo"].ToString();
                txtNombreLiga.Text = dsLiga.Tables[0].Rows[0]["nombre"].ToString();
        }
        //Operacion para asignar en pantalla en que registro nos
        //encontramos de un total de registros
        txtTotalRegistros.Text = "Registro " + (registroActual + 1).ToString() + " de "
+ totalRegistros.ToString();
}

private void MostrarDatosComboBox()
{
        DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion,Providers.PostgresSQL);
        DataSet dsCargo =
recuperar.ExecuteDataSet("seleccionarcargo",CommandType.StoredProcedure);
        cboCargo.DataSource = dsCargo.Tables[0];
        cboCargo.DisplayMember = "descripcion";
```

```
        cboCargo.ValueMember = "idcargo";
}

private void MoverComboBox()
{
        //Instanciamos una variable para saber si tiene mas de un
        //registro cuando ejecutamos el procedimiento
        int numeroDeRegistros = 0;
        //Instanciamos una variable para ejecutar el procedimiento de la BD
        DatabaseHelper recuperarCargo = new
DatabaseHelper(ConectarBD.CadenaDeConexion,Providers.PostgresSQL);
        //Instanciamos variable para la tabla cargo
        DataTable dtCargo = new DataTable();
        int registros = this.BindingContext[dsDatos.Tables[0]].Count;
        if (registros>0)
        {
        recuperarCargo.AddParameter("@idcargo",Convert.ToInt32(txtIdCargo.Text.
Trim()));
        dtCargo =
recuperarCargo.ExecuteDataSet("seleccionarcargoporidcargoempleado",CommandTyp
e.StoredProcedure).Tables[0];
        numeroDeRegistros = this.BindingContext[dtCargo].Count;
        if (numeroDeRegistros>0)
        {
                int idcargo = Convert.ToInt32(dtCargo.Rows[0][0]);
                cboCargo.SelectedValue = idcargo;
        }
        }
}
```

Código para ingresar datos de los empleados

Nuevo registro.

```
//Instanciar una variable para insertar un registro
```

```
DatabaseHelper a = new DatabaseHelper(ConectarBD.CadenaDeConexion,
Providers.PostgresSQL);
//Añadir los parametros necesarios para el ingreso
a.AddParameter("@cedula",txtCedula.Text.Trim());
a.AddParameter("@nombre",txtNombre.Text.Trim());
a.AddParameter("@apellido",txtApellido.Text.Trim());
a.AddParameter("@direccion",txtDireccion.Text.Trim());
a.AddParameter("@telefono",txtTelefono.Text.Trim());
a.AddParameter("@email",txtEmail.Text.Trim());
a.AddParameter("@idcargo",txtIdCargo.Text.Trim());
a.AddParameter("@idliga",txtLiga.Text.Trim());
a.AddParameter("@usuario",txtUsuario.Text.Trim());
a.AddParameter("@contrasena",txtContrasena.Text.Trim());
//Variable para saber si se inserto el registro
int registrosAfectados = a.ExecuteNonQuery("insertarempleado",
CommandType.StoredProcedure);
```

Actualización de registro

```
//Instanciar una variable para actualizar un registro
DatabaseHelper a = new DatabaseHelper(ConectarBD.CadenaDeConexion,
Providers.PostgresSQL);
//Añadir los parametros necesarios para la actualizacion
a.AddParameter("@cedula",txtCedula.Text.Trim());
a.AddParameter("@nombre",txtNombre.Text.Trim());
a.AddParameter("@apellido",txtApellido.Text.Trim());
a.AddParameter("@direccion",txtDireccion.Text.Trim());
a.AddParameter("@telefono",txtTelefono.Text.Trim());
a.AddParameter("@email",txtEmail.Text.Trim());
a.AddParameter("@idcargo", Convert.ToInt32(cboCargo.SelectedValue));
a.AddParameter("@usuario",txtUsuario.Text.Trim());
a.AddParameter("@contrasena",txtContrasena.Text.Trim());
a.AddParameter("idempleados",Convert.ToInt32(txtIdEmpleado.Text.Trim()));
```

```
//Variable para saber si se actualizo el registro
int registrosAfectados = a.ExecuteNonQuery("Updateempleado",
CommandType.StoredProcedure);
```

6.9.4 Pruebas

Una de las últimas fases del ciclo de vida antes de entregar un programa para su explotación, es la fase de pruebas.

La fase de pruebas añade valor al producto que se maneja: todos los programas tienen errores y la fase de pruebas los descubre; ese es el valor que añade. El objetivo específico de la fase de pruebas es encontrar cuantos más errores, mejor.
Una vez concluido con el desarrollo del módulo se procedió a realizar las pruebas con la intención de garantizar la seguridad, confiabilidad, integridad de la información que se va a manejar en el sistema de facturación.

6.9.4.1 Pruebas de caja negra

En este tipo de pruebas no se considera la codificación dentro de sus parámetros a evaluar, es decir, no está basado en el conocimiento interno del sistema. Estas pruebas se enfocan en la funcionalidad y requerimientos especificados del software.

Lo que se demostró con estas pruebas es:

- El sistema es de fácil navegación e interacción con el usuario.
- Los ingresos de datos a la base de datos se lo realiza de una manera sencilla sin entrar a tantas pantallas.
- El sistema es seguro ya que cada usuario tiene su función al momento de ingresar al sistema.
- La información se almacena de una manera confiable y fiable, la recuperación de la misma es inmediata.
- Se evitó la duplicidad y redundancia de datos.

- Las funciones que realiza el sistema son claras y precisas.
- El ingreso al sistema es fácil y sencillo
- Se mantiene la integridad del sistema a lo largo de su utilización.

Para obtener estas pruebas lo que se hizo es probar reiteradamente las entradas y salidas de los datos; esto quiere decir que se valoró la información que es presentada al usuario.

Condiciones de parámetros de entrada			
Iniciar sesión es poseer una sesión activa	verdadero	verdadero	Falso
Cerrar sesión es desactivar una sesión	verdadero	falso	Falso
Resultado esperado			
Iniciar sesión sea verdadero	X		
Cerrar sesión sea falso		X	X

Tabla 6.34. Tabla de Decisión

El plan de pruebas unitarias de caja negra, creado a partir de la tabla de decisión, es una tabla que tiene casos de prueba, la combinación de condiciones de valores verdadero y falso para los parámetros de entrada y los resultados esperados.

#	CASOS DE PRUEBA	OBJETIVO
1	Se ha iniciado sesión y se puede visualizar los estados de las sanciones a los jugadores	Verdadero
2	Tiempo de inactividad es menor que 5 minutos y no se puede visualizar los estados de las sanciones a los jugadores	Falso

Tabla 6.35. Plan de Pruebas de Caja Negra

6.9.4.2 Pruebas de caja blanca

Las pruebas de caja blanca se centran en los detalles procedimentales del software, por lo que su diseño está fuertemente ligado al código fuente.

Mediante estas pruebas se garantizó qué:

- Se ejecutan al menos una vez todos los caminos independientes de cada módulo
- Se utilizan las decisiones en su parte verdadera y en su parte falsa
- Se ejecutan todos los bucles en sus límites
- Se utilizan todas las estructuras de datos internas

```
private bool ValidarCedula()
    {
        bool bandera = true;
        string cadena;
        int suma = 0;
        int digitoVerficador;
        int valor = 0;
        cadena = txtCedula.Text.Trim();
        if (cadena.Length == 10)
        {
            for (int i = 0; i <= 8; i++)
            {
                if (i % 2 == 0)
                {
                        valor = Convert.ToInt32(cadena.Substring(i, 1)) * 2;
                    if (valor > 9)
                        valor -= 9;
                    suma += valor;
                }
                else
                {
                        suma += Convert.ToInt32(cadena.Substring(i, 1));
                }
            }
            digitoVerficador = 10 - (suma % 10);
                if (digitoVerficador != Convert.ToInt32(cadena.Substring(9, 1)))
            {
                bandera = false;
```

```
            }
                }
        else
        {
            bandera = false;
        }
        return bandera;
    }
```

#	CASOS DE PRUEBA	OBJETIVO
1	La cédula del usuario es correcta (la condición es verdadera)	Verdadero
2	La cédula del usuario no es correcta (la condición es falsa)	Falso

Como resultado de la ejecución de estas pruebas en el sistema se verificó y se aseguró que su estructura interna esté de acuerdo con las especificaciones.

6.9.4.3 Pruebas de Verificación y Validación

La verificación es un aspecto muy importante dentro de las pruebas ya que nos permite conocer si el sistema cumple con las especificaciones planteadas y si ejecuta la tarea para la cual fue creado, en cuanto a la validación es el proceso de comprobar que lo que se ha especificado es lo que el usuario realmente quería.

Verificación.- Este proceso determinó que el sistema satisface las condiciones impuestas al comienzo de este proyecto, este concuerda y cumple con las especificaciones planteadas.

Validación.- Esta prueba verificó si se cumple con las expectativas del cliente.

- Pruebas de aceptación que fueron desarrolladas por el administrador.
- Pruebas alfa que fueron realizadas por los usuarios que manipularán la información del sistema con el desarrollador como observador.

Pruebas beta que fueron realizadas por los usuarios que manipularan la información del sistema; sin observadores en el entorno.

6.9.5 Implantación

Luego de que el sistema de registro y sanción de jugadores pasó satisfactoriamente la fase de pruebas se procedió a su implantación, como en la LIGA DEPORTIVA PARROQUIAL HUACHI GRANDE si existen computadoras no fue necesario la adquisición de una pero si se requirió de la instalación y configuración del software que a continuación se lo describe.

La máquina de la Liga trabaja con el sistema operativo Windows xp, en el que se encuentra instalado y configurado.

6.9.5.2 Instalación del motor de base de datos

El sistema de registro y sanciones de los jugadores trabaja con PostgreSQL para la parte de Base de datos motivo por el cual se procedió a instalarlo en la computadora de la LIGA DEPORTIVA PARROQUIAL HUACHI GRANDE.

Como primer paso se efectuó la descarga del paquete llamado:

postgresql-8.4.2-1-windows.exe

Luego se procedió a realizar su ejecución en el sistema.

Una vez ejecutado el paquete se generó un asistente grafico para seguir con la instalación.

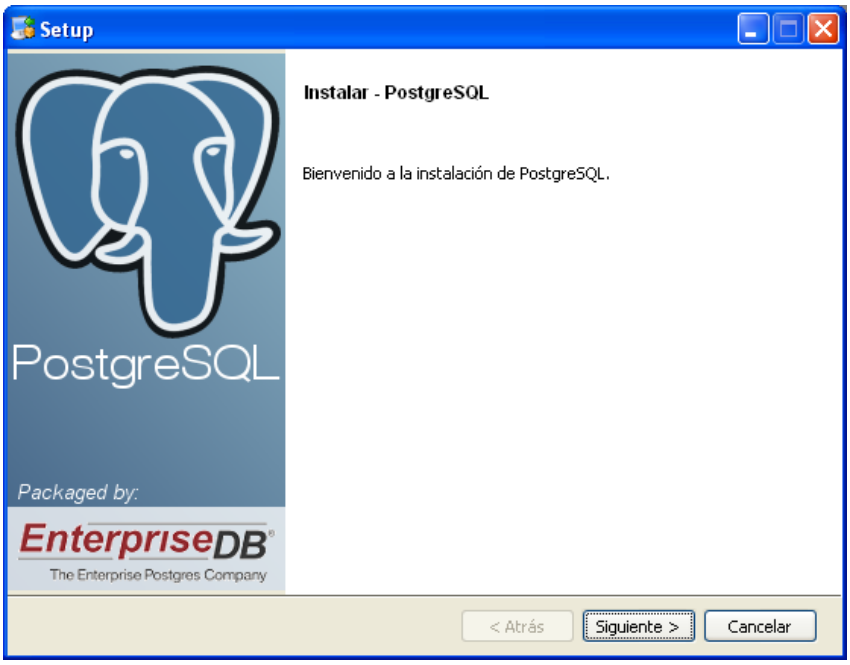


Figura 6.29. Instalar Postgres

En este paso seleccionamos la dirección de instalación de PostgreSQL

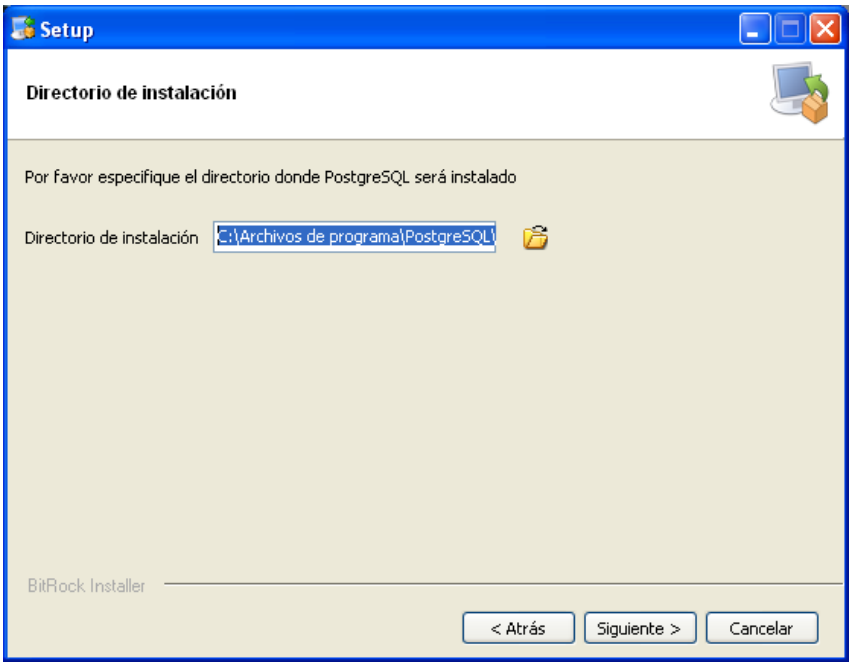


Figura 6.30. Directorio de instalación

A continuación seleccionamos el directorio dentro del cual se almacenarán los datos

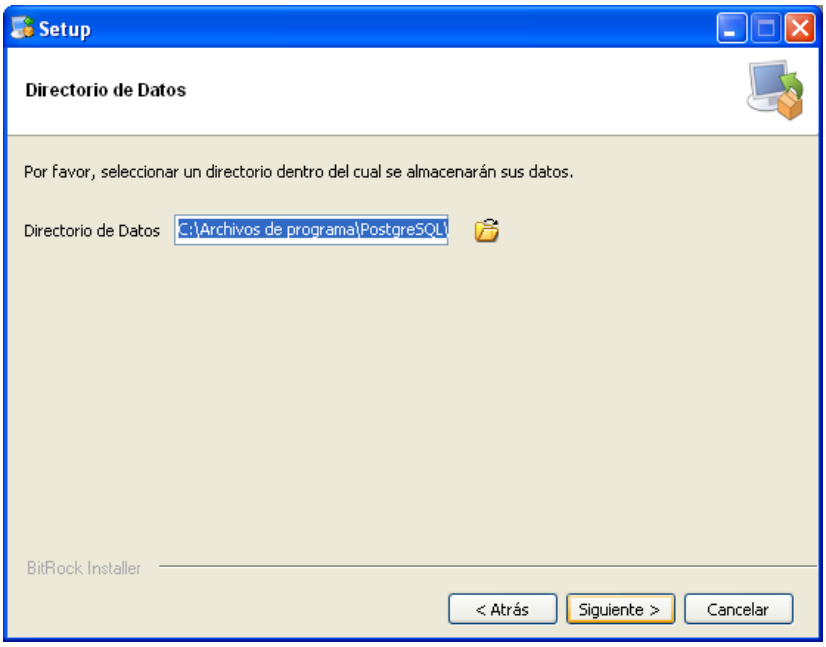


Figura 6.31. Directorio de datos

Como paso siguiente se procedió a ingresar la contraseña del usuario postgres

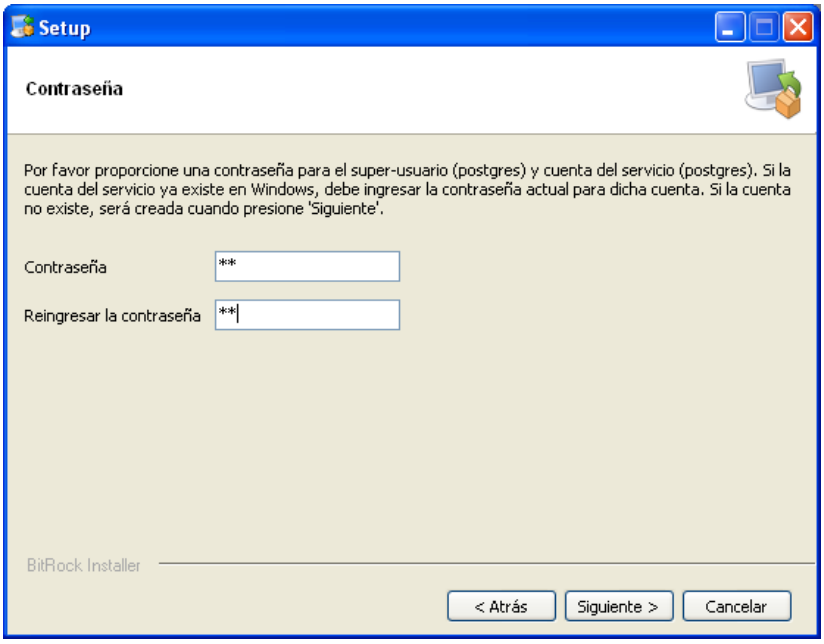


Figura 6.32. Contraseña del usuario postgres

En este paso ingresamos el número de puerto en el que el servidor debería escuchar

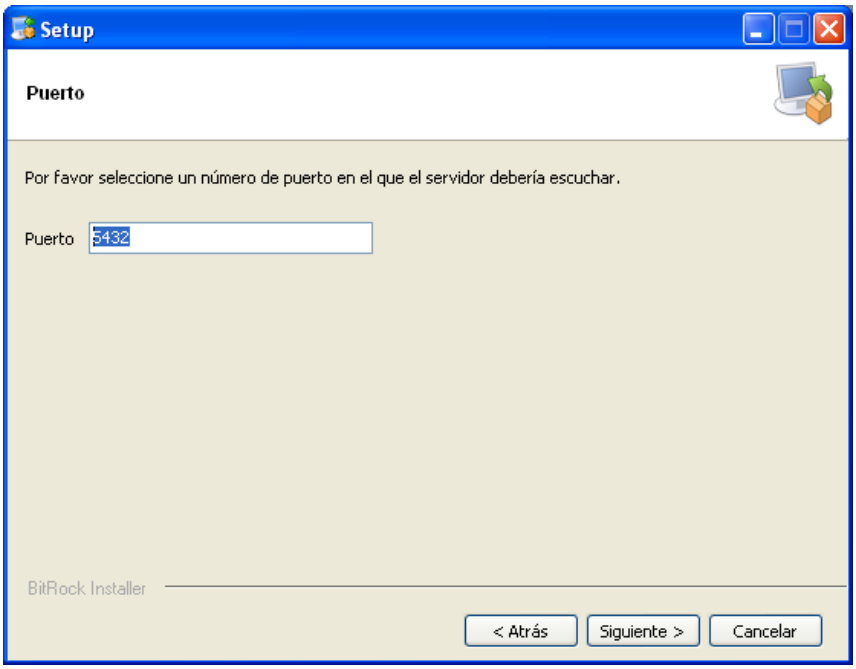


Figura 6.33. Contraseña del usuario postgres.

Configuramos el idioma

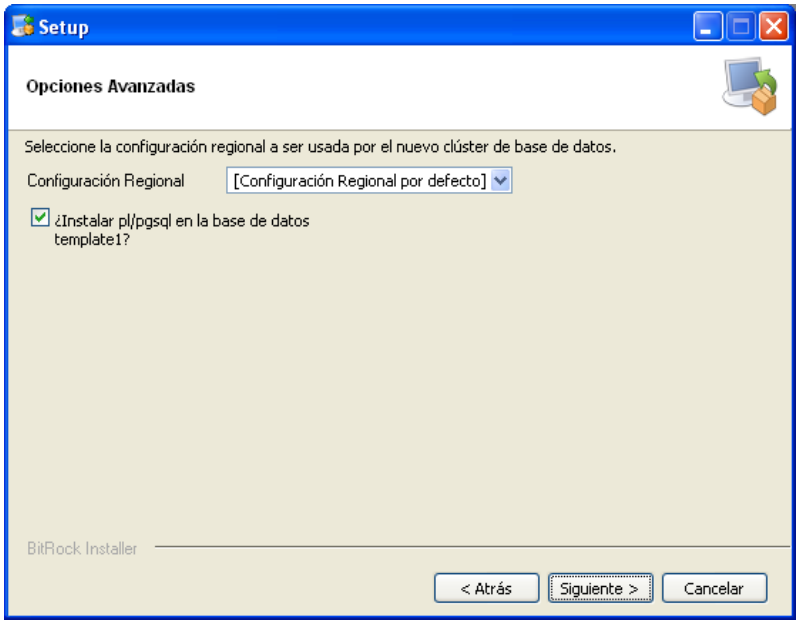


Figura 6.34. Configuración del idioma.

Como último paso ejecutamos la instalación en el servidor

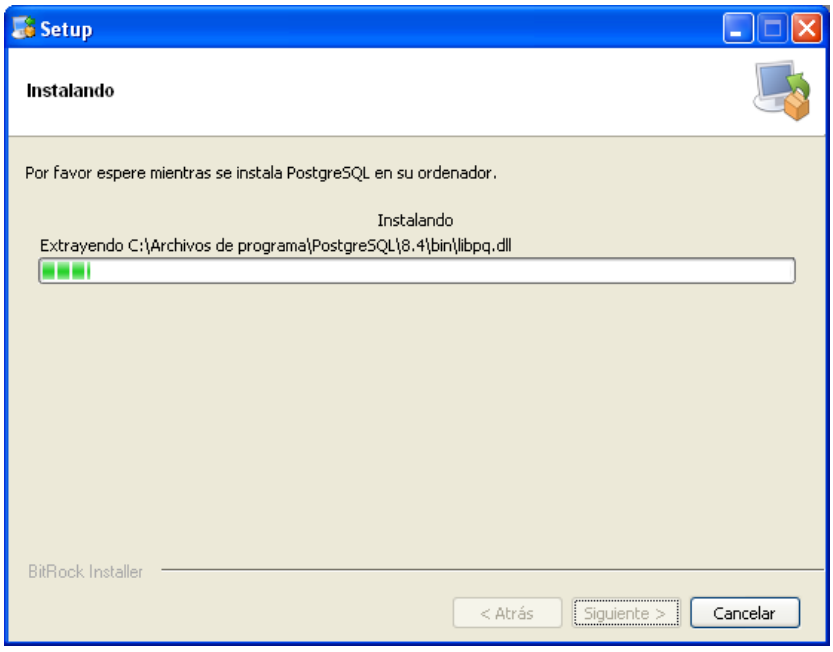


Figura 6.35. Listo para instalar.

Figura 6.36. Instalado

6.9.5.3 Instalación del sistema de gestión de proyectos

Primer paso: Base de datos

Se procedió a crear la base de datos que utilizará el Sistema en PostgreSQL con el nombre "ldphg" junto con sus tablas, funciones y triggers.

La creación de las tablas, funciones y triggers se realizó mediante la ejecución de SQL en la base de datos.

Segundo paso: Instalar los pre-requisitos para proceder a instalar el sistema de registro y sanciones de los jugadores.

Descargamos y ejecutamos el framework dotnetfx35.

Una vez ejecutado el paquete se genero un asistente grafico para seguir la instalación en el cual debemos seleccionar que hemos leído los términos de contrato de licencia.

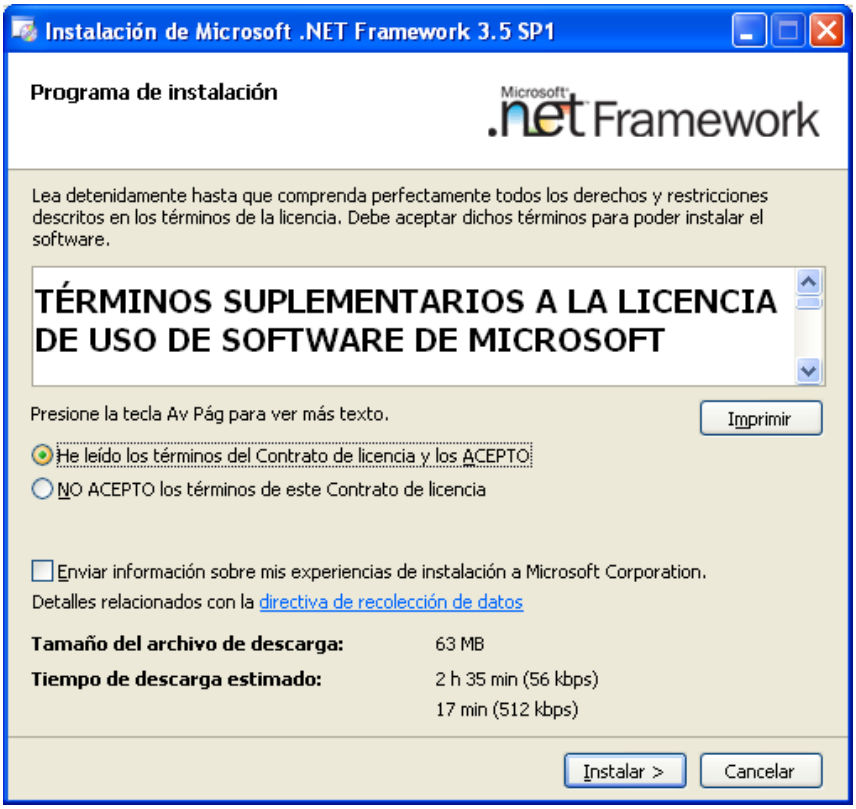


Figura 6.36. Instalar Framework

El programa está listo para instalarse.

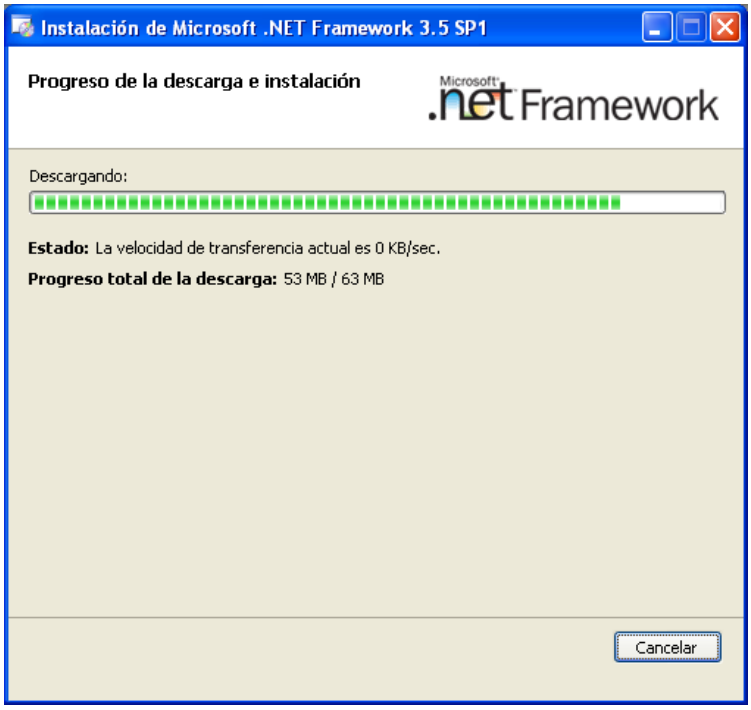


Figura 6.37. Listo para instalar Framework

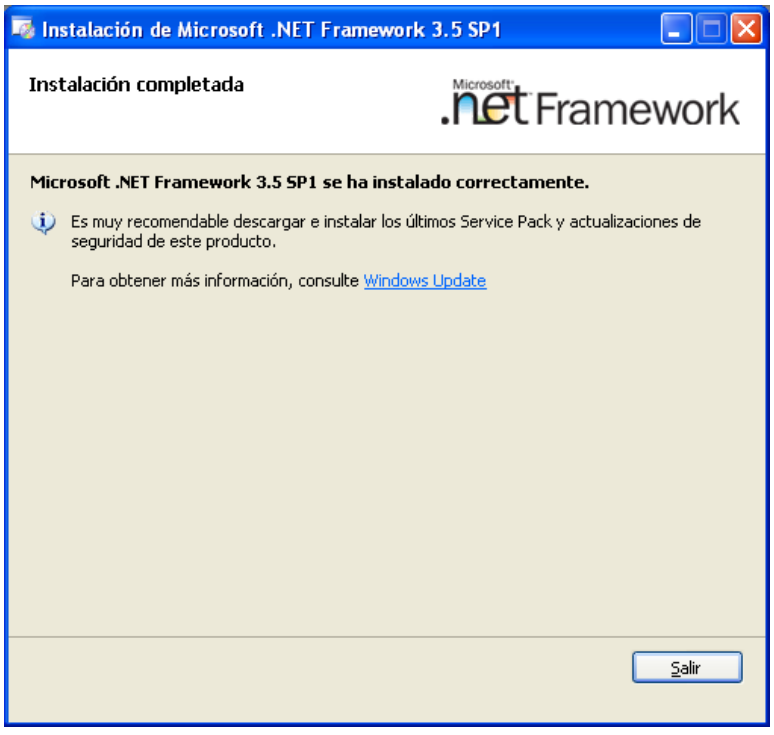


Figura 6.38. Instalado Framework.

Descargamos y ejecutamos Crystal Report Reditribuible CRRedist2008_x86.

Una vez ejecutado el paquete se instala automáticamente.

Tercer paso: Instalar el sistema de registro y sanciones de los jugadores.

Ejecutamos el instalador del sistema de registro y sanciones de los jugadores en el cual solo damos en siguiente y se instala el sistema.

6.10 Conclusiones y Recomendaciones de la propuesta

6.10.1 Conclusiones

- El sistema se ajusta fácilmente a las diversas formas de administrar los registros de los jugadores de cada equipo que es filial a la Liga.
- Mediante la utilización de Diagramas UML, se implementó de mejor manera, la forma que interactúa el usuario con el sistema.
- Los administradores de la Liga cuentan con una potente base que evita datos duplicados y una mejor organización en su información.
- El lenguaje de programación PHP que se utilizó para el desarrollo es totalmente compatible con el Gestor de Base de Datos PostgresSQL, permitiendo así una mejor consulta de la información que se trata en el sistema de registro y sanciones de los jugadores.
- El lenguaje de programación sharpdevelop que se utilizó para el desarrollo es totalmente compatible con el Gestor de Base de Datos PostgresSQL, permitiendo así un mejor control de la información que se trata en el sistema de registro y sanciones de los jugadores.
- La base de datos creada en PostgreSQL cuenta con todas las características para brindar seguridad, y confianza en el almacenamiento de información del sistema.
- El Sistema de registro y sanciones de los jugadores cuenta con pantallas según el tipo de usuario, para que cada uno de ellos puedan manipular y controlar de la mejor manera los recursos de acuerdo a los permisos otorgados.
- Con la aplicación de pruebas en el sistema se pudo encontrar y corregir errores que permitieron asegurar el correcto funcionamiento en su implantación en la Liga Deportiva Parroquial Huachi Grande.

- La interfaz del sistema es de fácil navegación y tiene búsquedas inteligentes, para que el usuario pueda interactuar de la mejor manera con el sistema y recupere datos fácilmente.
- Los manuales para los diferentes usuarios ayudan a comprender y despejar cualquier duda o inquietud que tengan, ya que fueron elaborados minuciosamente y se detalla paso a paso como funciona el sistema.

6.10.2 Recomendaciones

- Es necesario capacitar al personal que vaya a utilizar el sistema, sobre el manejo de aplicaciones informáticas; leer detenidamente el manual de usuario para que puedan tener una mejor idea de las ventajas que pueden tener al momento de manejar el sistema.
- Antes de emitir carnets deportivos en el sistema se deberá definir los parámetros apropiados con que la Liga Deportiva Parroquial Huachi Grande se administrará.
- Realizar respaldos periódicos de la base de datos para salvaguardar la información de la Liga, puesto que si ocurre un daño en la información y no existe un respaldo podría ocasionar graves problemas como pérdidas de de diferente índole respecto a la Liga.
- Los usuarios del sistema deben tener cuidado con el manejo de su contraseña pues el acceso de personas no autorizadas podría provocar daños en la correcta administración del sistema de registro y sanciones de los jugadores.

Bibliografía

Información Bibliográfica de Libros

- MEDINA, Washington (2008) *Guía para el desarrollo de trabajos de graduación.*
- PAZMAY, Galo (2004) *Guía práctica para la elaboración de tesis y trabajos de investigación.* Editorial Freire.
- LAWRENCE, Shari (2002) *Ingeniería de Software Teoría y práctica*, Editorial Pentice Hall

Información Bibliográfica de Internet

- http://sharpdevelop.malavida.com/d3784-descargar-windows
- http://www.abcdatos.com/tutoriales/tutorial/l1637.html
- http://es.wikipedia.org/wiki/SharpDevelop
- http://www.icsharpcode.net/OpenSource/SD/
- http://www.sqlmax.com/func1.asp
- http://www.sqlmax.com/func2.asp
- http://es.wikipedia.org/wiki/Base_de_datos
- http://es.wikipedia.org/wiki/PostgreSQL
- http://www.monografias.com/trabajos7/sisinf/sisinf.shtml
- http://es.wikipedia.org/wiki/Sistema_de_informaci%C3%B3n
- http://es.wikipedia.org/wiki/Software
- http://www.todoliga.com.uy/index.php/component/content/article/2-otrasnot/63-habilitacion-de-los-jugadores.html
- http://www.ofi.org.uy/circulares/2009/circ1966_03-12-09.pdf
- http://www.grupo-maser.com/PAG_Cursos/Auto/auto2/auto2/PAGINA%20PRINCIPAL/Automatizacion/Automatizacion.htm
- http://www.monografias.com/trabajos28/sistema-inscripcion/sistema-inscripcion.shtml
- http://www.derechoecuador.com/index.php?option=com_content&task=view&id=2897&Itemid=426
- http://www.abcdatos.com/programas/programa/l11051.html

- http://www.freedownloadmanager.org/es/downloads/Wincup_7769_p/free.htm

Glosario de Terminos

Automatizacion.- Sistema de producción en el que se usan máquinas en lugar de mano de obra; Proceso para lograr operaciones automáticas

Software.- Se conoce como **software** al equipamiento lógico o soporte lógico de un sistema informático; comprende el conjunto de los componentes **lógicos** necesarios que hacen posible la realización de tareas específicas, en contraposición a los componentes físicos, que son llamados hardware.

SharpDevelop.- Es un entorno de desarrollo integrado libre para los lenguajes de programación C#, Visual Basic .NET y Boo.

XML.- siglas en inglés de e**X**tensible **M**arkup **L**anguage ('lenguaje de marcas extensible'), es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C).

PostgreSQL.- PostgreSQL es un sistema de gestión de base de datos relacional orientada a objetos y libre, publicado bajo la licencia BSD.

Base de Datos.- Es un conjunto de información almacenada en memoria auxiliar que permite acceso directo y un conjunto de programas que manipulan esos datos.

SGBD: Sistema Gestor de Base de Datos o DBMS, es una agrupación de programas que sirven para definir, construir y manipular una base de datos.

HTML HiperText Markup Language o Lenguaje de Marcación de Hipertexto, es un lenguaje se utiliza comúnmente para establecer la estructura y contenido de un sitio web, tanto de texto, objetos e imágenes.

Php.- Hypertext Pre-Processor, es un lenguaje de script incrustado dentro del HTML. La mayor parte de su sintaxis ha sido tomada de C, Java y Perl con algunas características especificas de si mismo. La meta del lenguaje es permitir rápidamente a los desarrolladores la generación dinámica de páginas.

IDE Integrated Development Environment o Entorno integrado de desarrollo, aplicación compuesta por un conjunto de herramientas útiles para un programador.

Open Source.- Código abierto es el término de con el que se conoce al software distribuido y desarrollado libremente.

UML.- Unified Modeling Lenguage o Lenguaje Unificado de Modelado, es un lenguaje gráfico para especificar, visualizar, construir y documentar los sistemas de software, representa un conjunto de las mejores prácticas que han probado ser exitosas en el modelado de sistemas grandes y complejos.

Usuario, Un usuario generalmente se identifica frente al sistema o servicio utilizando un nombre de usuario (nick) y una contraseña. Un usuario registrado accede a un servicio a través de un login luego de su autentificación.

ANEXOS

ANEXO 1: Estructura del Cuestionario

FACULTAD DE INGENIERÍA EN SISTEMAS ELECTRÓNICA E INDUSTRIAL

CARRERA DE INGENIERÍA EN SISTEMAS

OBJETIVO DEL CUESTIONARIO:

El presente cuestionario se ha planificado con el objeto de recabar información referente al Sistema de registro y sanciones de los jugadores, la misma que será manejada en forma responsable y exclusivamente para solucionar un problema de la Liga, la información es anónima y tendrá la reserva del caso, le solicitó comedidamente conteste con la verdad en el siguiente cuestionario.

INSTRUCCIONES:

Marque con una X o escriba la respuesta que UD. considere conveniente.

1. La información del jugador que se recopila por la Liga se encuentra almacenada en:

- ☐ Archivos físicos
- ☐ Archivos digitales
- ☐ De ambas formas
- ☐ Ninguna

2. ¿Sabe que es un sistema automatizado?

- ☐ Si
- ☐ No

3¿Cree usted que con el sistema automatizado se optimice tiempo?

- ☐ Si
- ☐ No

4.- ¿Cree usted que con la automatización de los procesos se realizara un mejor control de los jugadores?

- ☐ Si
- ☐ No

5.- ¿Cree usted que con la implantación del sistema la Liga tendrá mejores ingresos y salida de datos?

☐ Si

☐ No

6.- ¿Cree usted que con el registro de los jugadores podrán realizar un mejor control?

☐ Si

☐ No

7.- ¿Cree usted que con la implantación del sistema podrán emitir las sanciones a los jugadores de manera eficaz?

☐ Si

☐ No

ANEXO 2: Manual de configuración

Antecedentes.

Para que el sistema y el portal funcionen correctamente se necesita:

- Wamp Server 2.0 en adelante.
- Motor de base de datos PostgreSQL.
- Sharp Develop.
- Crystal Report Redistribuible
- Framework (3.5 en adelante)

Instalación del Sistema y Portal

Primer Paso: Base de Datos

Crear la base de datos junto a las tablas que necesita el Sistema de registro y sanciones de los jugadores para su funcionamiento, para ello se debe utilizar una herramienta que nos ayude, esta puede ser en modo gráfico o en la ventana de comandos de PostgreSQL o de la forma que crea pertinente.

Crear una base de datos de preferencia con nombre "**ldphg"** y un usuario con nombre **"postgres"** y contraseña "**sa**", abra una consulta SQL o query en un terminal de la base de datos y pegue el siguiente código sql:

```
CREATE TABLE liga
(
  idliga serial NOT NULL,
  nombre character(100) NOT NULL,
  descripcion character(200),
  anofundacion character(4),
  CONSTRAINT pk_liga PRIMARY KEY (idliga)
)

CREATE TABLE empleados
```

```
(
  idempleados serial NOT NULL,
  cedula character(10) NOT NULL,
  nombre character(50) NOT NULL,
  apellido character(50) NOT NULL,
  direccion character(100) NOT NULL,
  telefono character(12),
  email character(100),
  idcargo serial NOT NULL,
  idliga integer,
  usuario character(50),
  contrasena character(50),
  CONSTRAINT pk_empleados PRIMARY KEY (idempleados),
  CONSTRAINT fk_liga FOREIGN KEY (idliga)
      REFERENCES liga (idliga) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)

CREATE TABLE categoria
(
  idcategoria serial NOT NULL,
  descripcion character(50) NOT NULL,
  idliga integer,
  CONSTRAINT pk_categoria PRIMARY KEY (idcategoria),
  CONSTRAINT fk_liga1 FOREIGN KEY (idliga)
      REFERENCES liga (idliga) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)

CREATE TABLE equipo
(
  idequipo serial NOT NULL,
  nombre character(50) NOT NULL,
  anofundacion character(4),
```

```
  CONSTRAINT pk_equipo PRIMARY KEY (idequipo)
)

CREATE TABLE inscripciones
(
  idinscripcion serial NOT NULL,
  valorgarantia numeric,
  valorinscripcion numeric,
  fechamaximoinscripcion date NOT NULL,
  idcategoria integer,
  idequipo integer,
  periodo character(500),
  CONSTRAINT pk_inscripciones PRIMARY KEY (idinscripcion),
  CONSTRAINT fk_categoria FOREIGN KEY (idcategoria)
     REFERENCES categoria (idcategoria) MATCH SIMPLE
     ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT fk_equipo FOREIGN KEY (idequipo)
     REFERENCES equipo (idequipo) MATCH SIMPLE
     ON UPDATE NO ACTION ON DELETE NO ACTION
)

CREATE TABLE serie
(
  idserie serial NOT NULL,
  descripcion character(100),
  CONSTRAINT pk_serie PRIMARY KEY (idserie)
)

CREATE TABLE equiposerie
(
  idequiposerie serial NOT NULL,
  idequipo integer,
  idserie integer,
  CONSTRAINT pk_equiposerie PRIMARY KEY (idequiposerie),
```

```
  CONSTRAINT fk_equipo12 FOREIGN KEY (idequipo)
      REFERENCES equipo (idequipo) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT fk_serie FOREIGN KEY (idserie)
      REFERENCES serie (idserie) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)

CREATE TABLE calendario
(
  fechaemitecalendario date NOT NULL,
  idcalendario serial NOT NULL,
  fechapartido date NOT NULL,
  hora interval NOT NULL,
  equipo1 integer NOT NULL,
  equipo2 integer NOT NULL,
  idcategoria integer NOT NULL,
  lugar integer NOT NULL,
  CONSTRAINT pk_calendario PRIMARY KEY (idcalendario, fechaemitecalendario)
)

CREATE TABLE estadio
(
  idestadio serial NOT NULL,
  lugar character(500),
  CONSTRAINT pk_estadio PRIMARY KEY (idestadio)
)

CREATE TABLE parametroarbitro
(
  periodo character(100) NOT NULL,
  valor numeric(4,2) NOT NULL,
  CONSTRAINT parametroarbitro_pkey PRIMARY KEY (periodo)
)
```

```
CREATE TABLE cargo
(
  idcargo serial NOT NULL,
  descripcion character(100) NOT NULL,
  CONSTRAINT pk_cargo PRIMARY KEY (idcargo)
)

CREATE TABLE multaequipo
(
  idmultaequipo serial NOT NULL,
  idequipo integer NOT NULL,
  valor numeric(4,2) DEFAULT 0,
  CONSTRAINT multaequipo_pkey PRIMARY KEY (idmultaequipo)
)

CREATE TABLE periodo
(
  idperiodo serial NOT NULL,
  desde date NOT NULL,
  hasta date NOT NULL,
  CONSTRAINT pk_periodo PRIMARY KEY (idperiodo)
)

CREATE TABLE tiemposancion
(
  idtiemposancion serial NOT NULL,
  descripcion character(200) NOT NULL,
  CONSTRAINT pk_tiemposancion PRIMARY KEY (idtiemposancion)
)

CREATE TABLE posiciones
(
  idposiciones serial NOT NULL,
```

```
  fecha date NOT NULL,
  puntos integer,
  partidosjugados integer NOT NULL DEFAULT 0,
  partidosganados integer DEFAULT 0,
  partidosempatados integer DEFAULT 0,
  partidosperdidos integer DEFAULT 0,
  golesfavor integer DEFAULT 0,
  golescontra integer DEFAULT 0,
  goldiferencia integer DEFAULT 0,
  categoria integer,
  idequipo integer NOT NULL,
  CONSTRAINT pk_posiciones PRIMARY KEY (idposiciones, fecha)
)

CREATE TABLE sanciones
(
  idsancion serial NOT NULL,
  descripcion character(200) NOT NULL,
  idfechassancionadas integer,
  idequipo integer,
  CONSTRAINT pk_sanciones PRIMARY KEY (idsancion),
  CONSTRAINT fk_equipos FOREIGN KEY (idequipo)
     REFERENCES equipo (idequipo) MATCH SIMPLE
     ON UPDATE NO ACTION ON DELETE NO ACTION
)

CREATE TABLE jugadores
(
  idjugador serial NOT NULL,
  cedula character(10) NOT NULL,
  nombre character(100) NOT NULL,
  apellido character(100) NOT NULL,
  numerocancha integer NOT NULL,
  idequipo integer,
```

```
  fechanacimiento date NOT NULL,
  lugarnacimiento character(100) NOT NULL,
  foto character(300),
  CONSTRAINT pk_idjugador PRIMARY KEY (idjugador),
  CONSTRAINT fk_equipo1 FOREIGN KEY (idequipo)
      REFERENCES equipo (idequipo) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)

CREATE TABLE jugadoressancion
(
  idjugador integer NOT NULL,
  idsancion integer NOT NULL,
  fechasancionado date NOT NULL,
  CONSTRAINT "pk_jugadores sancion" PRIMARY KEY (idjugador, idsancion,
fechasancionado),
  CONSTRAINT fk_jugador3 FOREIGN KEY (idjugador)
      REFERENCES jugadores (idjugador) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT fk_sancion FOREIGN KEY (idsancion)
      REFERENCES sanciones (idsancion) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)

CREATE TABLE actadejuego
(
  idactadejuego serial NOT NULL,
  fechadeacta date NOT NULL,
  equipolocal character(50) NOT NULL,
  jugadorlocal1 integer,
  jugadorlocal2 integer,
  jugadorlocal3 integer,
  jugadorlocal4 integer,
  jugadorlocal5 integer,
```

```
jugadorlocal6 integer,
jugadorlocal7 integer,
jugadorlocal8 integer,
jugadorlocal9 integer,
jugadorlocal10 integer,
jugadorlocal11 integer,
jugadorlocal12 integer,
jugadorlocal13 integer,
jugadorlocal14 integer,
multaslocal numeric(4,2) DEFAULT 0,
golesequipolocal integer NOT NULL,
equipovisitante character(50) NOT NULL,
jugadorvisitante1 integer,
jugadorvisitante2 integer,
jugadorvisitante3 integer,
jugadorvisitante4 integer,
jugadorvisitante5 integer,
jugadorvisitante6 integer,
jugadorvisitante7 integer,
jugadorvisitante8 integer,
jugadorvisitante9 integer,
jugadorvisitante10 integer,
jugadorvisitante11 integer,
jugadorvisitante12 integer,
jugadorvisitante13 integer,
jugadorvisitante14 integer,
golesequipovisitante integer NOT NULL,
nombrearbitro character(300),
informearbitral character(10000),
nombrevocal character(300),
informevocal character(2000),
multavisitante numeric(4,2) DEFAULT 0,
detallemultasvisitante character(5000),
detallemultaslocal character(5000),
```

```
  CONSTRAINT actadejuego_pkey PRIMARY KEY (idactadejuego)
)

CREATE TABLE pagoarbitrage
(
  idpagoarbitraje integer NOT NULL DEFAULT
        nextval('pagoarbitraje_idpagoarbitraje_seq'::regclass),
  valor numeric(4,2) DEFAULT 0,
  idactadejuego integer,
  CONSTRAINT pagoarbitraje_pkey PRIMARY KEY (idpagoarbitraje),
  CONSTRAINT actadejuego_fkey FOREIGN KEY (idactadejuego)
     REFERENCES actadejuego (idactadejuego) MATCH SIMPLE
     ON UPDATE NO ACTION ON DELETE NO ACTION
)
```

Segundo Paso: Configuraciones

Este paso es fundamental para el funcionamiento del portal, para que este se conecte con la base de datos creada, se necesita configurar en cada página una conexión, para ello puede utilizar cualquier editor de archivos planos.

En la parte de la función *pg_connect* existen 4 parámetros para lo cual se debe colocar en el orden correspondiente, el primero es el servidor web, el segundo es la base datos, el tercero es el usuario que se conecta al Gestor de Base de Datos y el cuarto es la clave del usuario configurado anteriormente, en cuanto a la función.

La siguiente línea no se bebe cambiar.

$dbconn = pg_connect("host=localhost dbname=ldphg user=postgres password=sa port=5432")
Con los pasos realizados ya se puede utilizar el portal, teniendo en cuenta que para acceder solo pueden realizar consultas más no manipulación de datos**.**

Tercer Paso: Código Fuente

Aquí se detalla el código fuente de las secciones más relevantes del sistema:

Guardar Acta de Juego

```
DatabaseHelper ingresoSancion = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);
#region Acta De Juego
 if (txtVocal.Text!="" && txtArbitro.Text!="" && txtJugadorLocal1.Text !="" &&
                                    txtJugadorLocal2.Text!="" &&
txtJugadorLocal3.Text!="" && txtJugadorLocal4.Text!="" &&
                                    txtJugadorLocal5.Text!="" &&
txtJugadorLocal6.Text!="" && txtJugadorLocal7.Text!="" &&
                                    txtJugadorVisitante1.Text!="" &&
txtJugadorVisitante2.Text!="" && txtJugadorVisitante3.Text!="" &&
                                    txtJugadorVisitante4.Text!="" &&
txtJugadorVisitante5.Text!="" &&
                                    txtJugadorVisitante6.Text!="" &&
txtJugadorVisitante7.Text!="" && txtPagoMesaVisitante.Text!=""
                                   && txtPagoMesaLocal.Text!="" )
```

```
{
        //Si el comando es insertar
        if (comando == ComandoNpgsql.Insertar)
        {
                #region Nuevo Registro
                #region Arbitraje
                  decimal pagoArbitrajeLocal =
(Convert.ToDecimal(txtCostoArbitraje.Text)/2)/100;
                if ( Convert.ToDecimal( txtPagoMesaLocal.Text) >=
pagoArbitrajeLocal)
                {
                        pagoArbitrajeLocal =
Convert.ToDecimal(txtPagoMesaLocal.Text) - pagoArbitrajeLocal;
                }
                else
                {
                        MessageBox.Show("El valor de pago de mesa del equipo local
debe ser mayor al costo de arbitraje",
"ERROR",MessageBoxButtons.OK,MessageBoxIcon.Error);
                        txtPagoMesaLocal.Focus();
                }

        decimal pagoArbitrajeVisitante =
(Convert.ToDecimal(txtCostoArbitraje.Text)/2)/100;
        if ( Convert.ToDecimal( txtPagoMesaVisitante.Text) >= pagoArbitrajeLocal)
        {
                pagoArbitrajeVisitante =
Convert.ToDecimal(txtPagoMesaVisitante.Text) - pagoArbitrajeVisitante;
        }
        else
        {
                MessageBox.Show("El valor de pago de mesa del equipo visitante debe
ser mayor al costo de arbitraje","ERROR",
MessageBoxButtons.OK,MessageBoxIcon.Error);
                txtPagoMesaVisitante.Focus();
        }

  #endregion
  //Instanciar una variable para insertar un registro
  DatabaseHelper a = new DatabaseHelper(ConectarBD.CadenaDeConexion,
Providers.PostgresSQL);
  //Añadir los parametros necesarios para el ingreso
  a.AddParameter("@fechaactadejuego", dtpFecha.Value);
  a.AddParameter("@equipolocal", txtEquipoLocal.Text.Trim());
  a.AddParameter("@jugadorlocal1", cbojugadorLocal1.SelectedValue);
  a.AddParameter("@jugadorlocal2", cbojugadorLocal2.SelectedValue);
  a.AddParameter("@jugadorlocal3", cbojugadorLocal3.SelectedValue);
  a.AddParameter("@jugadorlocal4", cbojugadorLocal4.SelectedValue);
  a.AddParameter("@jugadorlocal5", cbojugadorLocal5.SelectedValue);
  a.AddParameter("@jugadorlocal6", cbojugadorLocal6.SelectedValue);
```

```
   a.AddParameter("@jugadorlocal7", cbojugadorLocal7.SelectedValue);
   a.AddParameter("@jugadorlocal8", cbojugadorLocal8.SelectedValue);
   a.AddParameter("@jugadorlocal9", cbojugadorLocal9.SelectedValue);
   a.AddParameter("@jugadorlocal10", cbojugadorLocal10.SelectedValue);
   a.AddParameter("@jugadorlocal11", cbojugadorLocal11.SelectedValue);
   a.AddParameter("@jugadorlocal12", cbojugadorLocal12.SelectedValue);
   a.AddParameter("@jugadorlocal13", cbojugadorLocal13.SelectedValue);
   a.AddParameter("@jugadorlocal14", cbojugadorLocal14.SelectedValue);
   a.AddParameter("@multaslocal", pagoArbitrajeLocal);
   a.AddParameter("@detallemultaslocal", txtDetalleMultaLocal.Text.Trim());
   a.AddParameter("@goleslocal", Convert.ToInt32(lblGolesLocal.Text.Trim()));
   a.AddParameter("@equipovisitante", txtEquipoVisitante.Text.Trim());
   a.AddParameter("@jugadorvisitante1", cboJugadorVisitante1.SelectedValue);
   a.AddParameter("@jugadorvisitante2", cboJugadorVisitante2.SelectedValue);
   a.AddParameter("@jugadorvisitante3", cboJugadorVisitante3.SelectedValue);
   a.AddParameter("@jugadorvisitante4", cboJugadorVisitante4.SelectedValue);
   a.AddParameter("@jugadorvisitante5", cboJugadorVisitante5.SelectedValue);
   a.AddParameter("@jugadorvisitante6", cboJugadorVisitante6.SelectedValue);
   a.AddParameter("@jugadorvisitante7", cboJugadorVisitante7.SelectedValue);
   a.AddParameter("@jugadorvisitante8", cboJugadorVisitante8.SelectedValue);
   a.AddParameter("@jugadorvisitante9", cboJugadorVisitante9.SelectedValue);
   a.AddParameter("@jugadorvisitante10", cboJugadorVisitante10.SelectedValue);
   a.AddParameter("@jugadorvisitante11", cboJugadorVisitante11.SelectedValue);
   a.AddParameter("@jugadorvisitante12", cboJugadorVisitante12.SelectedValue);
   a.AddParameter("@jugadorvisitante13", cboJugadorVisitante13.SelectedValue);
   a.AddParameter("@jugadorvisitante14", cboJugadorVisitante14.SelectedValue);
   a.AddParameter("@pagomultasvisitante", pagoArbitrajeVisitante);
   a.AddParameter("@nombrearbitro", txtDetalleMultaVisitante.Text.Trim());
   a.AddParameter("@golesvisitante",
Convert.ToInt32(lblGolesVisitante.Text.Trim()));
   a.AddParameter("@nombrearbitro", txtArbitro.Text.Trim());
   a.AddParameter("@informearbitral", txtInformeArbitro.Text.Trim());
   a.AddParameter("@nombrevocal", txtVocal.Text.Trim());
   a.AddParameter("@informevocal", txtInformeVocal.Text.Trim());
   //Variable para saber si se inserto el registro
   int registrosAfectados = a.ExecuteNonQuery("insertaractajuego",
CommandType.StoredProcedure);
   #endregion

   #region Amarillas

   #region AmarillaLocal1
   if (chkAmarillaLocal1.Checked)
   {

         ingresoSancion.AddParameter("@idjugador",cbojugadorLocal1.SelectedValue
);
         DataSet dsIdSancion = new DataSet();
         DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);
```

```
        recuperar.AddParameter("@idequipo",Convert.ToInt32(txtIdEquipoLocal.Tex
t));
        dsIdSancion =
recuperar.ExecuteDataSet("seleccionarsancionporidequipoamarilla",CommandType.St
oredProcedure);

        ingresoSancion.AddParameter("@idSancion",dsIdSancion.Tables[0].Rows[0][
0]);
        ingresoSancion.AddParameter("@fechasancionado",dtpFecha.Value);
        //Variable para saber si se inserto el registro
        int registros = ingresoSancion.ExecuteNonQuery("insertarsancion",
CommandType.StoredProcedure);

  }
  #endregion
  #region AmarillaLocal2
  if (chkAmarillaLocal2.Checked)
  {

        ingresoSancion.AddParameter("@idjugador",cbojugadorLocal2.SelectedValue
);
        DataSet dsIdSancion = new DataSet();
        DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);

        recuperar.AddParameter("@idequipo",Convert.ToInt32(txtIdEquipoLocal.Tex
t));
        dsIdSancion =
recuperar.ExecuteDataSet("seleccionarsancionporidequipoamarilla",CommandType.St
oredProcedure);

        ingresoSancion.AddParameter("@idSancion",dsIdSancion.Tables[0].Rows[0][
0]);
        ingresoSancion.AddParameter("@fechasancionado",dtpFecha.Value);
        //Variable para saber si se inserto el registro
        int registros = ingresoSancion.ExecuteNonQuery("insertarsancion",
CommandType.StoredProcedure);
    }
    #endregion
    #region AmarillaLocal3
    if (chkAmarillaLocal3.Checked)
    {

        ingresoSancion.AddParameter("@idjugador",cbojugadorLocal3.SelectedValue
);
        DataSet dsIdSancion = new DataSet();
        DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);
```

```
        recuperar.AddParameter("@idequipo",Convert.ToInt32(txtIdEquipoLocal.Tex
t));
        dsIdSancion =
recuperar.ExecuteDataSet("seleccionarsancionporidequipoamarilla",CommandType.St
oredProcedure);

        ingresoSancion.AddParameter("@idSancion",dsIdSancion.Tables[0].Rows[0][
0]);
        ingresoSancion.AddParameter("@fechasancionado",dtpFecha.Value);
        //Variable para saber si se inserto el registro
        int registros = ingresoSancion.ExecuteNonQuery("insertarsancion",
CommandType.StoredProcedure);
}
#endregion
#region AmarillaLocal4
 if (chkAmarillaLocal4.Checked)
{

        ingresoSancion.AddParameter("@idjugador",cbojugadorLocal4.SelectedValue
);
        DataSet dsIdSancion = new DataSet();
        DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);

        recuperar.AddParameter("@idequipo",Convert.ToInt32(txtIdEquipoLocal.Tex
t));
        dsIdSancion =
recuperar.ExecuteDataSet("seleccionarsancionporidequipoamarilla",CommandType.St
oredProcedure);

        ingresoSancion.AddParameter("@idSancion",dsIdSancion.Tables[0].Rows[0][
0]);
        ingresoSancion.AddParameter("@fechasancionado",dtpFecha.Value);
        //Variable para saber si se inserto el registro
        int registros = ingresoSancion.ExecuteNonQuery("insertarsancion",
CommandType.StoredProcedure);
}
#endregion
#region AmarillaLocal5
if (chkAmarillaLocal5.Checked)
{
        ingresoSancion.AddParameter("@idjugador",cbojugadorLocal5.SelectedValue
);
DataSet dsIdSancion = new DataSet();
DatabaseHelper recuperar = new DatabaseHelper(ConectarBD.CadenaDeConexion,
Providers.PostgresSQL);

        recuperar.AddParameter("@idequipo",Convert.ToInt32(txtIdEquipoLocal.Tex
t));
```

```
        dsIdSancion =
recuperar.ExecuteDataSet("seleccionarsancionporidequipoamarilla",CommandType.St
oredProcedure);

        ingresoSancion.AddParameter("@idSancion",dsIdSancion.Tables[0].Rows[0][
0]);
        ingresoSancion.AddParameter("@fechasancionado",dtpFecha.Value);
        //Variable para saber si se inserto el registro
        int registros = ingresoSancion.ExecuteNonQuery("insertarsancion",
CommandType.StoredProcedure);
}
#endregion
#region AmarillaLocal6
if (chkAmarillaLocal6.Checked)
{
        ingresoSancion.AddParameter("@idjugador",cbojugadorLocal6.SelectedValue
);
        DataSet dsIdSancion = new DataSet();
        DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);

        recuperar.AddParameter("@idequipo",Convert.ToInt32(txtIdEquipoLocal.Tex
t));
        dsIdSancion =
recuperar.ExecuteDataSet("seleccionarsancionporidequipoamarilla",CommandType.St
oredProcedure);

        ingresoSancion.AddParameter("@idSancion",dsIdSancion.Tables[0].Rows[0][
0]);
        ingresoSancion.AddParameter("@fechasancionado",dtpFecha.Value);
        //Variable para saber si se inserto el registro
        int registros = ingresoSancion.ExecuteNonQuery("insertarsancion",
CommandType.StoredProcedure);
 }
 #endregion
 #region AmarillaLocal7
 if (chkAmarillaLocal7.Checked)
 {
        ingresoSancion.AddParameter("@idjugador",cbojugadorLocal7.SelectedValue
);
        DataSet dsIdSancion = new DataSet();
        DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);

        recuperar.AddParameter("@idequipo",Convert.ToInt32(txtIdEquipoLocal.Tex
t));
        dsIdSancion =
recuperar.ExecuteDataSet("seleccionarsancionporidequipoamarilla",CommandType.St
oredProcedure);
```

```
        ingresoSancion.AddParameter("@idSancion",dsIdSancion.Tables[0].Rows[0][
0]);
        ingresoSancion.AddParameter("@fechasancionado",dtpFecha.Value);
        //Variable para saber si se inserto el registro
        int registros = ingresoSancion.ExecuteNonQuery("insertarsancion",
CommandType.StoredProcedure);
}
#endregion
#region AmarillaLocal8
if (chkAmarillaLocal8.Checked)
{
        ingresoSancion.AddParameter("@idjugador",cbojugadorLocal8.SelectedValue
);
        DataSet dsIdSancion = new DataSet();
        DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);

        recuperar.AddParameter("@idequipo",Convert.ToInt32(txtIdEquipoLocal.Tex
t));
        dsIdSancion =
recuperar.ExecuteDataSet("seleccionarsancionporidequipoamarilla",CommandType.St
oredProcedure);

        ingresoSancion.AddParameter("@idSancion",dsIdSancion.Tables[0].Rows[0][
0]);
        ingresoSancion.AddParameter("@fechasancionado",dtpFecha.Value);
        //Variable para saber si se inserto el registro
        int registros = ingresoSancion.ExecuteNonQuery("insertarsancion",
CommandType.StoredProcedure);
 }
 #endregion
 #region AmarillaLocal9
 if (chkAmarillaLocal9.Checked)
 {
        ingresoSancion.AddParameter("@idjugador",cbojugadorLocal9.SelectedValue
);
        DataSet dsIdSancion = new DataSet();
        DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);

        recuperar.AddParameter("@idequipo",Convert.ToInt32(txtIdEquipoLocal.Tex
t));
        dsIdSancion =
recuperar.ExecuteDataSet("seleccionarsancionporidequipoamarilla",CommandType.St
oredProcedure);

        ingresoSancion.AddParameter("@idSancion",dsIdSancion.Tables[0].Rows[0][
0]);
        ingresoSancion.AddParameter("@fechasancionado",dtpFecha.Value);
```

```
        //Variable para saber si se inserto el registro
        int registros = ingresoSancion.ExecuteNonQuery("insertarsancion",
CommandType.StoredProcedure);
}
#endregion
#region AmarillaLocal10
if (chkAmarillaLocal10.Checked)
{
        ingresoSancion.AddParameter("@idjugador",cbojugadorLocal10.SelectedValu
e);
        DataSet dsIdSancion = new DataSet();
        DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);

        recuperar.AddParameter("@idequipo",Convert.ToInt32(txtIdEquipoLocal.Tex
t));
        dsIdSancion =
recuperar.ExecuteDataSet("seleccionarsancionporidequipoamarilla",CommandType.St
oredProcedure);

        ingresoSancion.AddParameter("@idSancion",dsIdSancion.Tables[0].Rows[0][
0]);
        ingresoSancion.AddParameter("@fechasancionado",dtpFecha.Value);
        //Variable para saber si se inserto el registro
        int registros = ingresoSancion.ExecuteNonQuery("insertarsancion",
CommandType.StoredProcedure);
}
#endregion
#region AmarillaLocal11
if (chkAmarillaLocal11.Checked)
{
        ingresoSancion.AddParameter("@idjugador",cbojugadorLocal11.SelectedValu
e);
        DataSet dsIdSancion = new DataSet();
        DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);

        recuperar.AddParameter("@idequipo",Convert.ToInt32(txtIdEquipoLocal.Tex
t));
        dsIdSancion =
recuperar.ExecuteDataSet("seleccionarsancionporidequipoamarilla",CommandType.St
oredProcedure);

        ingresoSancion.AddParameter("@idSancion",dsIdSancion.Tables[0].Rows[0][
0]);
        ingresoSancion.AddParameter("@fechasancionado",dtpFecha.Value);
        //Variable para saber si se inserto el registro
        int registros = ingresoSancion.ExecuteNonQuery("insertarsancion",
CommandType.StoredProcedure);
 }
```

```
 #endregion
 #region AmarillaLocal12
 if (chkAmarillaLocal12.Checked)
 {
        ingresoSancion.AddParameter("@idjugador",cbojugadorLocal12.SelectedValu
e);
        DataSet dsIdSancion = new DataSet();
        DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);

        recuperar.AddParameter("@idequipo",Convert.ToInt32(txtIdEquipoLocal.Tex
t));
        dsIdSancion =
recuperar.ExecuteDataSet("seleccionarsancionporidequipoamarilla",CommandType.St
oredProcedure);

        ingresoSancion.AddParameter("@idSancion",dsIdSancion.Tables[0].Rows[0][
0]);
        ingresoSancion.AddParameter("@fechasancionado",dtpFecha.Value);
        //Variable para saber si se inserto el registro
        int registros = ingresoSancion.ExecuteNonQuery("insertarsancion",
CommandType.StoredProcedure);
}
#endregion
#region AmarillaLocal13
if (chkAmarillaLocal13.Checked)
{
        ingresoSancion.AddParameter("@idjugador",cbojugadorLocal13.SelectedValu
e);
        DataSet dsIdSancion = new DataSet();
        DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);

        recuperar.AddParameter("@idequipo",Convert.ToInt32(txtIdEquipoLocal.Tex
t));
        dsIdSancion =
recuperar.ExecuteDataSet("seleccionarsancionporidequipoamarilla",CommandType.St
oredProcedure);

        ingresoSancion.AddParameter("@idSancion",dsIdSancion.Tables[0].Rows[0][
0]);
        ingresoSancion.AddParameter("@fechasancionado",dtpFecha.Value);
        //Variable para saber si se inserto el registro
        int registros = ingresoSancion.ExecuteNonQuery("insertarsancion",
CommandType.StoredProcedure);
  }
  #endregion
  #region AmarillaLocal14
  if (chkAmarillaLocal14.Checked)
```

```
  {
        ingresoSancion.AddParameter("@idjugador",cbojugadorLocal14.SelectedValu
e);
        DataSet dsIdSancion = new DataSet();
        DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);

        recuperar.AddParameter("@idequipo",Convert.ToInt32(txtIdEquipoLocal.Tex
t));
        dsIdSancion =
recuperar.ExecuteDataSet("seleccionarsancionporidequipoamarilla",CommandType.St
oredProcedure);

        ingresoSancion.AddParameter("@idSancion",dsIdSancion.Tables[0].Rows[0][
0]);
        ingresoSancion.AddParameter("@fechasancionado",dtpFecha.Value);
        //Variable para saber si se inserto el registro
        int registros = ingresoSancion.ExecuteNonQuery("insertarsancion",
CommandType.StoredProcedure);
}
#endregion

#region AmarillaVisitante1
if (chkAmarillavisitante1.Checked)
{
        ingresoSancion.AddParameter("@idjugador",cboJugadorVisitante1.SelectedVa
lue);
        DataSet dsIdSancion = new DataSet();
        DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);

        recuperar.AddParameter("@idequipo",Convert.ToInt32(txtIdEquipoVisitante.
Text));
        dsIdSancion =
recuperar.ExecuteDataSet("seleccionarsancionporidequipoamarilla",CommandType.St
oredProcedure);

        ingresoSancion.AddParameter("@idSancion",dsIdSancion.Tables[0].Rows[0][
0]);
        ingresoSancion.AddParameter("@fechasancionado",dtpFecha.Value);
        //Variable para saber si se inserto el registro
        int registros = ingresoSancion.ExecuteNonQuery("insertarsancion",
CommandType.StoredProcedure);
}
#endregion
 #region AmarillaVisitante2
 if (chkAmarillavisitante2.Checked)
{
        ingresoSancion.AddParameter("@idjugador",cboJugadorVisitante2.SelectedVa
lue);
```

```
        DataSet dsIdSancion = new DataSet();
        DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);

        recuperar.AddParameter("@idequipo",Convert.ToInt32(txtIdEquipoVisitante.
Text));
        dsIdSancion =
recuperar.ExecuteDataSet("seleccionarsancionporidequipoamarilla",CommandType.St
oredProcedure);

        ingresoSancion.AddParameter("@idSancion",dsIdSancion.Tables[0].Rows[0][
0]);
        ingresoSancion.AddParameter("@fechasancionado",dtpFecha.Value);
        //Variable para saber si se inserto el registro
        int registros = ingresoSancion.ExecuteNonQuery("insertarsancion",
CommandType.StoredProcedure);
}
#endregion
#region AmarillaVisitante3
if (chkAmarillavisitante3.Checked)
{
        ingresoSancion.AddParameter("@idjugador",cboJugadorVisitante3.SelectedVa
lue);
        DataSet dsIdSancion = new DataSet();
        DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);

        recuperar.AddParameter("@idequipo",Convert.ToInt32(txtIdEquipoVisitante.
Text));
        dsIdSancion =
recuperar.ExecuteDataSet("seleccionarsancionporidequipoamarilla",CommandType.St
oredProcedure);

        ingresoSancion.AddParameter("@idSancion",dsIdSancion.Tables[0].Rows[0][
0]);
        ingresoSancion.AddParameter("@fechasancionado",dtpFecha.Value);
        //Variable para saber si se inserto el registro
        int registros = ingresoSancion.ExecuteNonQuery("insertarsancion",
CommandType.StoredProcedure);
  }
  #endregion
  #region AmarillaVisitante4
  if (chkAmarillavisitante4.Checked)
  {
        ingresoSancion.AddParameter("@idjugador",cboJugadorVisitante4.SelectedVa
lue);
        DataSet dsIdSancion = new DataSet();
        DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);
```

```
        recuperar.AddParameter("@idequipo",Convert.ToInt32(txtIdEquipoVisitante.
Text));
        dsIdSancion =
recuperar.ExecuteDataSet("seleccionarsancionporidequipoamarilla",CommandType.St
oredProcedure);

        ingresoSancion.AddParameter("@idSancion",dsIdSancion.Tables[0].Rows[0][
0]);
        ingresoSancion.AddParameter("@fechasancionado",dtpFecha.Value);
        //Variable para saber si se inserto el registro
        int registros = ingresoSancion.ExecuteNonQuery("insertarsancion",
CommandType.StoredProcedure);
  }
  #endregion
  #region AmarillaVisitante5
  if (chkAmarillavisitante5.Checked)
  {
        ingresoSancion.AddParameter("@idjugador",cboJugadorVisitante5.SelectedVa
lue);
        DataSet dsIdSancion = new DataSet();
        DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);

        recuperar.AddParameter("@idequipo",Convert.ToInt32(txtIdEquipoVisitante.
Text));
        dsIdSancion =
recuperar.ExecuteDataSet("seleccionarsancionporidequipoamarilla",CommandType.St
oredProcedure);

        ingresoSancion.AddParameter("@idSancion",dsIdSancion.Tables[0].Rows[0][
0]);
        ingresoSancion.AddParameter("@fechasancionado",dtpFecha.Value);
        //Variable para saber si se inserto el registro
        int registros = ingresoSancion.ExecuteNonQuery("insertarsancion",
CommandType.StoredProcedure);
}
#endregion
#region AmarillaVisitante6
if (chkAmarillavisitante6.Checked)
{
        ingresoSancion.AddParameter("@idjugador",cboJugadorVisitante6.SelectedVa
lue);
        DataSet dsIdSancion = new DataSet();
        DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);

        recuperar.AddParameter("@idequipo",Convert.ToInt32(txtIdEquipoVisitante.
Text));
```

```
        dsIdSancion =
recuperar.ExecuteDataSet("seleccionarsancionporidequipoamarilla",CommandType.St
oredProcedure);

        ingresoSancion.AddParameter("@idSancion",dsIdSancion.Tables[0].Rows[0][
0]);
        ingresoSancion.AddParameter("@fechasancionado",dtpFecha.Value);
        //Variable para saber si se inserto el registro
        int registros = ingresoSancion.ExecuteNonQuery("insertarsancion",
CommandType.StoredProcedure);
 }
 #endregion
 #region AmarillaVisitante7
 if (chkAmarillavisitante7.Checked)
 {
        ingresoSancion.AddParameter("@idjugador",cboJugadorVisitante7.SelectedVa
lue);
        DataSet dsIdSancion = new DataSet();
        DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);

        recuperar.AddParameter("@idequipo",Convert.ToInt32(txtIdEquipoVisitante.
Text));
        dsIdSancion =
recuperar.ExecuteDataSet("seleccionarsancionporidequipoamarilla",CommandType.St
oredProcedure);

        ingresoSancion.AddParameter("@idSancion",dsIdSancion.Tables[0].Rows[0][
0]);
        ingresoSancion.AddParameter("@fechasancionado",dtpFecha.Value);
        //Variable para saber si se inserto el registro
        int registros = ingresoSancion.ExecuteNonQuery("insertarsancion",
CommandType.StoredProcedure);
}
#endregion
#region AmarillaVisitante8
if (chkAmarillavisitante8.Checked)
{
        ingresoSancion.AddParameter("@idjugador",cboJugadorVisitante8.SelectedVa
lue);
        DataSet dsIdSancion = new DataSet();
        DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);

        recuperar.AddParameter("@idequipo",Convert.ToInt32(txtIdEquipoVisitante.
Text));
        dsIdSancion =
recuperar.ExecuteDataSet("seleccionarsancionporidequipoamarilla",CommandType.St
oredProcedure);
```

```
        ingresoSancion.AddParameter("@idSancion",dsIdSancion.Tables[0].Rows[0][
0]);
        ingresoSancion.AddParameter("@fechasancionado",dtpFecha.Value);
        //Variable para saber si se inserto el registro
        int registros = ingresoSancion.ExecuteNonQuery("insertarsancion",
CommandType.StoredProcedure);
}
#endregion
#region AmarillaVisitante9
if (chkAmarillavisitante9.Checked)
{
        ingresoSancion.AddParameter("@idjugador",cboJugadorVisitante9.SelectedVa
lue);
        DataSet dsIdSancion = new DataSet();
        DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);

        recuperar.AddParameter("@idequipo",Convert.ToInt32(txtIdEquipoVisitante.
Text));
        dsIdSancion =
recuperar.ExecuteDataSet("seleccionarsancionporidequipoamarilla",CommandType.St
oredProcedure);

        ingresoSancion.AddParameter("@idSancion",dsIdSancion.Tables[0].Rows[0][
0]);
        ingresoSancion.AddParameter("@fechasancionado",dtpFecha.Value);
        //Variable para saber si se inserto el registro
        int registros = ingresoSancion.ExecuteNonQuery("insertarsancion",
CommandType.StoredProcedure);
}
#endregion
#region AmarillaVisitante10
if (chkAmarillavisitante10.Checked)
{
        ingresoSancion.AddParameter("@idjugador",cboJugadorVisitante10.Selected
Value);
        DataSet dsIdSancion = new DataSet();
        DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);

        recuperar.AddParameter("@idequipo",Convert.ToInt32(txtIdEquipoVisitante.
Text));
        dsIdSancion =
recuperar.ExecuteDataSet("seleccionarsancionporidequipoamarilla",CommandType.St
oredProcedure);

        ingresoSancion.AddParameter("@idSancion",dsIdSancion.Tables[0].Rows[0][
0]);
        ingresoSancion.AddParameter("@fechasancionado",dtpFecha.Value);
```

```
        //Variable para saber si se inserto el registro
        int registros = ingresoSancion.ExecuteNonQuery("insertarsancion",
CommandType.StoredProcedure);
}
#endregion
#region AmarillaVisitante11
if (chkAmarillavisitante11.Checked)
{
        ingresoSancion.AddParameter("@idjugador",cboJugadorVisitante11.Selected
Value);
        DataSet dsIdSancion = new DataSet();
        DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);

        recuperar.AddParameter("@idequipo",Convert.ToInt32(txtIdEquipoVisitante.
Text));
        dsIdSancion =
recuperar.ExecuteDataSet("seleccionarsancionporidequipoamarilla",CommandType.St
oredProcedure);

        ingresoSancion.AddParameter("@idSancion",dsIdSancion.Tables[0].Rows[0][
0]);
        ingresoSancion.AddParameter("@fechasancionado",dtpFecha.Value);
        //Variable para saber si se inserto el registro
        int registros = ingresoSancion.ExecuteNonQuery("insertarsancion",
CommandType.StoredProcedure);
}
#endregion
#region AmarillaVisitante12
if (chkAmarillavisitante12.Checked)
{
        ingresoSancion.AddParameter("@idjugador",cboJugadorVisitante12.Selected
Value);
                DataSet dsIdSancion = new DataSet();
                DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);

        recuperar.AddParameter("@idequipo",Convert.ToInt32(txtIdEquipoVisitante.
Text));
                dsIdSancion =
recuperar.ExecuteDataSet("seleccionarsancionporidequipoamarilla",CommandType.St
oredProcedure);

        ingresoSancion.AddParameter("@idSancion",dsIdSancion.Tables[0].Rows[0][
0]);
                ingresoSancion.AddParameter("@fechasancionado",dtpFecha.Value);
                //Variable para saber si se inserto el registro
              int registros = ingresoSancion.ExecuteNonQuery("insertarsancion",
CommandType.StoredProcedure);
            }
```

```
            #endregion
            #region AmarillaVisitante13
            if (chkAmarillavisitante13.Checked)
            {

        ingresoSancion.AddParameter("@idjugador",cboJugadorVisitante13.Selected
Value);
                    DataSet dsIdSancion = new DataSet();
                    DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);

        recuperar.AddParameter("@idequipo",Convert.ToInt32(txtIdEquipoVisitante.
Text));
                    dsIdSancion =
recuperar.ExecuteDataSet("seleccionarsancionporidequipoamarilla",CommandType.St
oredProcedure);

        ingresoSancion.AddParameter("@idSancion",dsIdSancion.Tables[0].Rows[0][
0]);
                    ingresoSancion.AddParameter("@fechasancionado",dtpFecha.Value);
                    //Variable para saber si se inserto el registro
                int registros = ingresoSancion.ExecuteNonQuery("insertarsancion",
CommandType.StoredProcedure);
            }
            #endregion
            #region AmarillaVisitante14
            if (chkAmarillavisitante14.Checked)
            {

        ingresoSancion.AddParameter("@idjugador",cboJugadorVisitante14.Selected
Value);
                    DataSet dsIdSancion = new DataSet();
                    DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);

        recuperar.AddParameter("@idequipo",Convert.ToInt32(txtIdEquipoVisitante.
Text));
                    dsIdSancion =
recuperar.ExecuteDataSet("seleccionarsancionporidequipoamarilla",CommandType.St
oredProcedure);

        ingresoSancion.AddParameter("@idSancion",dsIdSancion.Tables[0].Rows[0][
0]);
                    ingresoSancion.AddParameter("@fechasancionado",dtpFecha.Value);
                    //Variable para saber si se inserto el registro
                int registros = ingresoSancion.ExecuteNonQuery("insertarsancion",
CommandType.StoredProcedure);
            }
            #endregion
            #endregion
```

```
  #region Rojas
  #region RojaLocal1
  if (chkRojalocal1.Checked)
 {

        ingresoSancion.AddParameter("@idjugador",cbojugadorLocal1.SelectedValue
);
        DataSet dsIdSancion = new DataSet();
        DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);

        recuperar.AddParameter("@idequipo",Convert.ToInt32(txtIdEquipoLocal.Tex
t));
        dsIdSancion =
recuperar.ExecuteDataSet("seleccionarsancionporidequiporoja",CommandType.Stored
Procedure);

        ingresoSancion.AddParameter("@idSancion",dsIdSancion.Tables[0].Rows[0][
0]);

        ingresoSancion.AddParameter("@fechasancionado",dtpFecha.Value);
        //Variable para saber si se inserto el registro
         int registros = ingresoSancion.ExecuteNonQuery("insertarsancion",
CommandType.StoredProcedure);
}
#endregion
#region RojaLocal2
 if (chkRojalocal2.Checked)
{
        ingresoSancion.AddParameter("@idjugador",cbojugadorLocal2.SelectedValue
);
                DataSet dsIdSancion = new DataSet();
                DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);

        recuperar.AddParameter("@idequipo",Convert.ToInt32(txtIdEquipoLocal.Tex
t));
                dsIdSancion =
recuperar.ExecuteDataSet("seleccionarsancionporidequiporoja",CommandType.Stored
Procedure);

        ingresoSancion.AddParameter("@idSancion",dsIdSancion.Tables[0].Rows[0][
0]);

        ingresoSancion.AddParameter("@fechasancionado",dtpFecha.Value);
        //Variable para saber si se inserto el registro
        int registros = ingresoSancion.ExecuteNonQuery("insertarsancion",
CommandType.StoredProcedure);
                                        }
```

```
#endregion
#region RojaLocal3
 if (chkRojalocal3.Checked)
{
        ingresoSancion.AddParameter("@idjugador",cbojugadorLocal3.SelectedValue
);
        DataSet dsIdSancion = new DataSet();
        DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);

        recuperar.AddParameter("@idequipo",Convert.ToInt32(txtIdEquipoLocal.Tex
t));
        dsIdSancion =
recuperar.ExecuteDataSet("seleccionarsancionporidequiporoja",CommandType.Stored
Procedure);

        ingresoSancion.AddParameter("@idSancion",dsIdSancion.Tables[0].Rows[0][
0]);

        ingresoSancion.AddParameter("@fechasancionado",dtpFecha.Value);
        //Variable para saber si se inserto el registro
         int registros = ingresoSancion.ExecuteNonQuery("insertarsancion",
CommandType.StoredProcedure);
}
#endregion
#region RojaLocal4
if (chkRojalocal4.Checked)
{
        ingresoSancion.AddParameter("@idjugador",cbojugadorLocal4.SelectedValue
);
        DataSet dsIdSancion = new DataSet();
        DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);

        recuperar.AddParameter("@idequipo",Convert.ToInt32(txtIdEquipoLocal.Tex
t));
        dsIdSancion =
recuperar.ExecuteDataSet("seleccionarsancionporidequiporoja",CommandType.Stored
Procedure);

        ingresoSancion.AddParameter("@idSancion",dsIdSancion.Tables[0].Rows[0][
0]);

        ingresoSancion.AddParameter("@fechasancionado",dtpFecha.Value);
        //Variable para saber si se inserto el registro
         int registros = ingresoSancion.ExecuteNonQuery("insertarsancion",
CommandType.StoredProcedure);
                                                }
#endregion
#region RojaLocal5
```

```
if (chkRojalocal5.Checked)
{
        ingresoSancion.AddParameter("@idjugador",cbojugadorLocal5.SelectedValue
);
        DataSet dsIdSancion = new DataSet();
        DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);

        recuperar.AddParameter("@idequipo",Convert.ToInt32(txtIdEquipoLocal.Tex
t));
        dsIdSancion =
recuperar.ExecuteDataSet("seleccionarsancionporidequiporoja",CommandType.Stored
Procedure);

        ingresoSancion.AddParameter("@idSancion",dsIdSancion.Tables[0].Rows[0][
0]);

        ingresoSancion.AddParameter("@fechasancionado",dtpFecha.Value);
                                                    //Variable para saber si se inserto el
registro
        int registros = ingresoSancion.ExecuteNonQuery("insertarsancion",
CommandType.StoredProcedure);
}
 #endregion
#region RojaLocal6
if (chkRojalocal6.Checked)
{
        ingresoSancion.AddParameter("@idjugador",cbojugadorLocal6.SelectedValue
);
        DataSet dsIdSancion = new DataSet();
        DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);

        recuperar.AddParameter("@idequipo",Convert.ToInt32(txtIdEquipoLocal.Tex
t));
        dsIdSancion =
recuperar.ExecuteDataSet("seleccionarsancionporidequiporoja",CommandType.Stored
Procedure);

        ingresoSancion.AddParameter("@idSancion",dsIdSancion.Tables[0].Rows[0][
0]);

        ingresoSancion.AddParameter("@fechasancionado",dtpFecha.Value);
        //Variable para saber si se inserto el registro
       int registros = ingresoSancion.ExecuteNonQuery("insertarsancion",
CommandType.StoredProcedure);
}
#endregion
#region RojaLocal7
if (chkRojalocal7.Checked)
```

```
{

        ingresoSancion.AddParameter("@idjugador",cbojugadorLocal7.SelectedValue
);
        DataSet dsIdSancion = new DataSet();
        DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);

        recuperar.AddParameter("@idequipo",Convert.ToInt32(txtIdEquipoLocal.Tex
t));
        dsIdSancion =
recuperar.ExecuteDataSet("seleccionarsancionporidequiporoja",CommandType.Stored
Procedure);

        ingresoSancion.AddParameter("@idSancion",dsIdSancion.Tables[0].Rows[0][
0]);

        ingresoSancion.AddParameter("@fechasancionado",dtpFecha.Value);
        //Variable para saber si se inserto el registro
        int registros = ingresoSancion.ExecuteNonQuery("insertarsancion",
CommandType.StoredProcedure);
}
#endregion
#region RojaLocal8
if (chkRojalocal8.Checked)
{
        ingresoSancion.AddParameter("@idjugador",cbojugadorLocal8.SelectedValue
);
        DataSet dsIdSancion = new DataSet();
        DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);

        recuperar.AddParameter("@idequipo",Convert.ToInt32(txtIdEquipoLocal.Tex
t));
        dsIdSancion =
recuperar.ExecuteDataSet("seleccionarsancionporidequiporoja",CommandType.Stored
Procedure);

        ingresoSancion.AddParameter("@idSancion",dsIdSancion.Tables[0].Rows[0][
0]);

        ingresoSancion.AddParameter("@fechasancionado",dtpFecha.Value);
        //Variable para saber si se inserto el registro
       int registros = ingresoSancion.ExecuteNonQuery("insertarsancion",
CommandType.StoredProcedure);
}
#endregion
#region RojaLocal9
if (chkRojalocal9.Checked)
```

```
{
        ingresoSancion.AddParameter("@idjugador",cbojugadorLocal9.SelectedValue
);
        DataSet dsIdSancion = new DataSet();
        DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);

        recuperar.AddParameter("@idequipo",Convert.ToInt32(txtIdEquipoLocal.Tex
t));
        dsIdSancion =
recuperar.ExecuteDataSet("seleccionarsancionporidequiporoja",CommandType.Stored
Procedure);

        ingresoSancion.AddParameter("@idSancion",dsIdSancion.Tables[0].Rows[0][
0]);

        ingresoSancion.AddParameter("@fechasancionado",dtpFecha.Value);
        //Variable para saber si se inserto el registro
        int registros = ingresoSancion.ExecuteNonQuery("insertarsancion",
CommandType.StoredProcedure);
}
#endregion
#region RojaLocal10
if (chkRojalocal10.Checked)
{
        ingresoSancion.AddParameter("@idjugador",cbojugadorLocal10.SelectedValu
e);
        DataSet dsIdSancion = new DataSet();
        DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);

        recuperar.AddParameter("@idequipo",Convert.ToInt32(txtIdEquipoLocal.Tex
t));
        dsIdSancion =
recuperar.ExecuteDataSet("seleccionarsancionporidequiporoja",CommandType.Stored
Procedure);

        ingresoSancion.AddParameter("@idSancion",dsIdSancion.Tables[0].Rows[0][
0]);

        ingresoSancion.AddParameter("@fechasancionado",dtpFecha.Value);
        //Variable para saber si se inserto el registro
        int registros = ingresoSancion.ExecuteNonQuery("insertarsancion",
CommandType.StoredProcedure);
}
#endregion
#region RojaLocal11
if (chkRojalocal11.Checked)
```

```
{
        ingresoSancion.AddParameter("@idjugador",cbojugadorLocal11.SelectedValu
e);
        DataSet dsIdSancion = new DataSet();
        DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);

        recuperar.AddParameter("@idequipo",Convert.ToInt32(txtIdEquipoLocal.Tex
t));
        dsIdSancion =
recuperar.ExecuteDataSet("seleccionarsancionporidequiporoja",CommandType.Stored
Procedure);

        ingresoSancion.AddParameter("@idSancion",dsIdSancion.Tables[0].Rows[0][
0]);

        ingresoSancion.AddParameter("@fechasancionado",dtpFecha.Value);
        //Variable para saber si se inserto el registro
        int registros = ingresoSancion.ExecuteNonQuery("insertarsancion",
CommandType.StoredProcedure);
}
#endregion
#region RojaLocal12
if (chkRojalocal12.Checked)
{
        ingresoSancion.AddParameter("@idjugador",cbojugadorLocal12.SelectedValu
e);
        DataSet dsIdSancion = new DataSet();
        DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);

        recuperar.AddParameter("@idequipo",Convert.ToInt32(txtIdEquipoLocal.Tex
t));
        dsIdSancion =
recuperar.ExecuteDataSet("seleccionarsancionporidequiporoja",CommandType.Stored
Procedure);

        ingresoSancion.AddParameter("@idSancion",dsIdSancion.Tables[0].Rows[0][
0]);

        ingresoSancion.AddParameter("@fechasancionado",dtpFecha.Value);
        //Variable para saber si se inserto el registro
        int registros = ingresoSancion.ExecuteNonQuery("insertarsancion",
CommandType.StoredProcedure);
}
 #endregion
#region RojaLocal13
if (chkRojalocal13.Checked)
```

```
{
        ingresoSancion.AddParameter("@idjugador",cbojugadorLocal13.SelectedValu
e);
        DataSet dsIdSancion = new DataSet();
        DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);

        recuperar.AddParameter("@idequipo",Convert.ToInt32(txtIdEquipoLocal.Tex
t));
        dsIdSancion =
recuperar.ExecuteDataSet("seleccionarsancionporidequiporoja",CommandType.Stored
Procedure);

        ingresoSancion.AddParameter("@idSancion",dsIdSancion.Tables[0].Rows[0][
0]);

        ingresoSancion.AddParameter("@fechasancionado",dtpFecha.Value);
        //Variable para saber si se inserto el registro
        int registros = ingresoSancion.ExecuteNonQuery("insertarsancion",
CommandType.StoredProcedure);
}
#endregion
#region RojaLocal14
if (chkRojalocal14.Checked)
{
        ingresoSancion.AddParameter("@idjugador",cbojugadorLocal14.SelectedValu
e);
        DataSet dsIdSancion = new DataSet();
        DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);

        recuperar.AddParameter("@idequipo",Convert.ToInt32(txtIdEquipoLocal.Tex
t));
        dsIdSancion =
recuperar.ExecuteDataSet("seleccionarsancionporidequiporoja",CommandType.Stored
Procedure);

        ingresoSancion.AddParameter("@idSancion",dsIdSancion.Tables[0].Rows[0][
0]);

        ingresoSancion.AddParameter("@fechasancionado",dtpFecha.Value);
        //Variable para saber si se inserto el registro
        int registros = ingresoSancion.ExecuteNonQuery("insertarsancion",
CommandType.StoredProcedure);
}
#endregion


#region RojaVisitante1
if (chkRojavisitante1.Checked)
```

```
{
        ingresoSancion.AddParameter("@idjugador",cboJugadorVisitante1.SelectedVa
lue);
        DataSet dsIdSancion = new DataSet();
        DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);

        recuperar.AddParameter("@idequipo",Convert.ToInt32(txtIdEquipoVisitante.
Text));
        dsIdSancion =
recuperar.ExecuteDataSet("seleccionarsancionporidequiporoja",CommandType.Stored
Procedure);

        ingresoSancion.AddParameter("@idSancion",dsIdSancion.Tables[0].Rows[0][
0]);

        ingresoSancion.AddParameter("@fechasancionado",dtpFecha.Value);
        //Variable para saber si se inserto el registro
        int registros = ingresoSancion.ExecuteNonQuery("insertarsancion",
CommandType.StoredProcedure);
}
#endregion
#region RojaVisitante2
 if (chkRojavisitante2.Checked)
{
        ingresoSancion.AddParameter("@idjugador",cboJugadorVisitante2.SelectedVa
lue);
        DataSet dsIdSancion = new DataSet();
        DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);

        recuperar.AddParameter("@idequipo",Convert.ToInt32(txtIdEquipoVisitante.
Text));
        dsIdSancion =
recuperar.ExecuteDataSet("seleccionarsancionporidequiporoja",CommandType.Stored
Procedure);

        ingresoSancion.AddParameter("@idSancion",dsIdSancion.Tables[0].Rows[0][
0]);

        ingresoSancion.AddParameter("@fechasancionado",dtpFecha.Value);
        //Variable para saber si se inserto el registro
        int registros = ingresoSancion.ExecuteNonQuery("insertarsancion",
CommandType.StoredProcedure);
}
#endregion
#region RojaVisitante3
if (chkRojavisitante3.Checked)
```

```
{
        ingresoSancion.AddParameter("@idjugador",cboJugadorVisitante3.SelectedVa
lue);
        DataSet dsIdSancion = new DataSet();
        DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);

        recuperar.AddParameter("@idequipo",Convert.ToInt32(txtIdEquipoVisitante.
Text));
        dsIdSancion =
recuperar.ExecuteDataSet("seleccionarsancionporidequiporoja",CommandType.Stored
Procedure);

        ingresoSancion.AddParameter("@idSancion",dsIdSancion.Tables[0].Rows[0][
0]);

        ingresoSancion.AddParameter("@fechasancionado",dtpFecha.Value);
        //Variable para saber si se inserto el registro
        int registros = ingresoSancion.ExecuteNonQuery("insertarsancion",
CommandType.StoredProcedure);
}
#endregion
#region RojaVisitante4
if (chkRojavisitante4.Checked)
{
        ingresoSancion.AddParameter("@idjugador",cboJugadorVisitante4.SelectedVa
lue);
        DataSet dsIdSancion = new DataSet();
        DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);

        recuperar.AddParameter("@idequipo",Convert.ToInt32(txtIdEquipoVisitante.
Text));
        dsIdSancion =
recuperar.ExecuteDataSet("seleccionarsancionporidequiporoja",CommandType.Stored
Procedure);

        ingresoSancion.AddParameter("@idSancion",dsIdSancion.Tables[0].Rows[0][
0]);

        ingresoSancion.AddParameter("@fechasancionado",dtpFecha.Value);
        //Variable para saber si se inserto el registro
        int registros = ingresoSancion.ExecuteNonQuery("insertarsancion",
CommandType.StoredProcedure);
}
#endregion
#region RojaVisitante5
if (chkRojavisitante5.Checked)
```

```
{
        ingresoSancion.AddParameter("@idjugador",cboJugadorVisitante5.SelectedVa
lue);
        DataSet dsIdSancion = new DataSet();
        DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);

        recuperar.AddParameter("@idequipo",Convert.ToInt32(txtIdEquipoVisitante.
Text));
        dsIdSancion =
recuperar.ExecuteDataSet("seleccionarsancionporidequiporoja",CommandType.Stored
Procedure);

        ingresoSancion.AddParameter("@idSancion",dsIdSancion.Tables[0].Rows[0][
0]);

        ingresoSancion.AddParameter("@fechasancionado",dtpFecha.Value);
        //Variable para saber si se inserto el registro
        int registros = ingresoSancion.ExecuteNonQuery("insertarsancion",
CommandType.StoredProcedure);
}
#endregion
#region RojaVisitante6
if (chkRojavisitante6.Checked)
{
        ingresoSancion.AddParameter("@idjugador",cboJugadorVisitante6.SelectedVa
lue);
        DataSet dsIdSancion = new DataSet();
        DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);

        recuperar.AddParameter("@idequipo",Convert.ToInt32(txtIdEquipoVisitante.
Text));
        dsIdSancion =
recuperar.ExecuteDataSet("seleccionarsancionporidequiporoja",CommandType.Stored
Procedure);

        ingresoSancion.AddParameter("@idSancion",dsIdSancion.Tables[0].Rows[0][
0]);

        ingresoSancion.AddParameter("@fechasancionado",dtpFecha.Value);
        //Variable para saber si se inserto el registro
        int registros = ingresoSancion.ExecuteNonQuery("insertarsancion",
CommandType.StoredProcedure);
}
#endregion
#region RojaVisitante7
if (chkRojavisitante7.Checked)
```

```
{
        ingresoSancion.AddParameter("@idjugador",cboJugadorVisitante7.SelectedVa
lue);
        DataSet dsIdSancion = new DataSet();
        DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);

        recuperar.AddParameter("@idequipo",Convert.ToInt32(txtIdEquipoVisitante.
Text));
        dsIdSancion =
recuperar.ExecuteDataSet("seleccionarsancionporidequiporoja",CommandType.Stored
Procedure);

        ingresoSancion.AddParameter("@idSancion",dsIdSancion.Tables[0].Rows[0][
0]);

        ingresoSancion.AddParameter("@fechasancionado",dtpFecha.Value);
        //Variable para saber si se inserto el registro
        int registros = ingresoSancion.ExecuteNonQuery("insertarsancion",
CommandType.StoredProcedure);
}
#endregion
#region RojaVisitante8
if (chkRojavisitante8.Checked)
{
        ingresoSancion.AddParameter("@idjugador",cboJugadorVisitante8.SelectedVa
lue);
        DataSet dsIdSancion = new DataSet();
        DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);

        recuperar.AddParameter("@idequipo",Convert.ToInt32(txtIdEquipoVisitante.
Text));
        dsIdSancion =
recuperar.ExecuteDataSet("seleccionarsancionporidequiporoja",CommandType.Stored
Procedure);

        ingresoSancion.AddParameter("@idSancion",dsIdSancion.Tables[0].Rows[0][
0]);
        ingresoSancion.AddParameter("@fechasancionado",dtpFecha.Value);
        //Variable para saber si se inserto el registro
        int registros = ingresoSancion.ExecuteNonQuery("insertarsancion",
CommandType.StoredProcedure);
}
#endregion
#region RojaVisitante9
if (chkRojavisitante9.Checked)
{
        ingresoSancion.AddParameter("@idjugador",cboJugadorVisitante9.SelectedVa
lue);
```

```
        DataSet dsIdSancion = new DataSet();
        DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);

        recuperar.AddParameter("@idequipo",Convert.ToInt32(txtIdEquipoVisitante.
Text));
        dsIdSancion =
recuperar.ExecuteDataSet("seleccionarsancionporidequiporoja",CommandType.Stored
Procedure);

        ingresoSancion.AddParameter("@idSancion",dsIdSancion.Tables[0].Rows[0][
0]);
        ingresoSancion.AddParameter("@fechasancionado",dtpFecha.Value);
        //Variable para saber si se inserto el registro
        int registros = ingresoSancion.ExecuteNonQuery("insertarsancion",
CommandType.StoredProcedure);
}
#endregion
#region RojaVisitante10
if (chkRojavisitante10.Checked)
{
        ingresoSancion.AddParameter("@idjugador",cboJugadorVisitante10.Selected
Value);
        DataSet dsIdSancion = new DataSet();
        DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);

        recuperar.AddParameter("@idequipo",Convert.ToInt32(txtIdEquipoVisitante.
Text));
        dsIdSancion =
recuperar.ExecuteDataSet("seleccionarsancionporidequiporoja",CommandType.Stored
Procedure);

        ingresoSancion.AddParameter("@idSancion",dsIdSancion.Tables[0].Rows[0][
0]);
        ingresoSancion.AddParameter("@fechasancionado",dtpFecha.Value);
        //Variable para saber si se inserto el registro
        int registros = ingresoSancion.ExecuteNonQuery("insertarsancion",
CommandType.StoredProcedure);
}
#endregion
#region RojaVisitante11
if (chkRojavisitante11.Checked)
{
        ingresoSancion.AddParameter("@idjugador",cboJugadorVisitante11.Selected
Value);
        DataSet dsIdSancion = new DataSet();
        DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);
```

```
        recuperar.AddParameter("@idequipo",Convert.ToInt32(txtIdEquipoVisitante.
Text));
        dsIdSancion =
recuperar.ExecuteDataSet("seleccionarsancionporidequiporoja",CommandType.Stored
Procedure);

        ingresoSancion.AddParameter("@idSancion",dsIdSancion.Tables[0].Rows[0][
0]);

        ingresoSancion.AddParameter("@fechasancionado",dtpFecha.Value);
        //Variable para saber si se inserto el registro
         int registros = ingresoSancion.ExecuteNonQuery("insertarsancion",
CommandType.StoredProcedure);
}
#endregion
#region RojaVisitante12
if (chkRojavisitante12.Checked)
{
        ingresoSancion.AddParameter("@idjugador",cboJugadorVisitante12.Selected
Value);
        DataSet dsIdSancion = new DataSet();
        DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);

        recuperar.AddParameter("@idequipo",Convert.ToInt32(txtIdEquipoVisitante.
Text));
        dsIdSancion =
recuperar.ExecuteDataSet("seleccionarsancionporidequiporoja",CommandType.Stored
Procedure);

        ingresoSancion.AddParameter("@idSancion",dsIdSancion.Tables[0].Rows[0][
0]);

        ingresoSancion.AddParameter("@fechasancionado",dtpFecha.Value);
        //Variable para saber si se inserto el registro
         int registros = ingresoSancion.ExecuteNonQuery("insertarsancion",
CommandType.StoredProcedure);
}
#endregion
#region RojaVisitante13
if (chkRojavisitante13.Checked)
{
        ingresoSancion.AddParameter("@idjugador",cboJugadorVisitante13.Selected
Value);
        DataSet dsIdSancion = new DataSet();
        DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);
```

```
        recuperar.AddParameter("@idequipo",Convert.ToInt32(txtIdEquipoVisitante.
Text));
        dsIdSancion =
recuperar.ExecuteDataSet("seleccionarsancionporidequiporoja",CommandType.Stored
Procedure);

        ingresoSancion.AddParameter("@idSancion",dsIdSancion.Tables[0].Rows[0][
0]);

        ingresoSancion.AddParameter("@fechasancionado",dtpFecha.Value);
        //Variable para saber si se inserto el registro
        int registros = ingresoSancion.ExecuteNonQuery("insertarsancion",
CommandType.StoredProcedure);
}
#endregion
#region RojaVisitante14
if (chkRojavisitante14.Checked)
{
        ingresoSancion.AddParameter("@idjugador",cboJugadorVisitante14.Selected
Value);
        DataSet dsIdSancion = new DataSet();
        DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);

        recuperar.AddParameter("@idequipo",Convert.ToInt32(txtIdEquipoVisitante.
Text));
        dsIdSancion =
recuperar.ExecuteDataSet("seleccionarsancionporidequiporoja",CommandType.Stored
Procedure);

        ingresoSancion.AddParameter("@idSancion",dsIdSancion.Tables[0].Rows[0][
0]);

        ingresoSancion.AddParameter("@fechasancionado",dtpFecha.Value);
        //Variable para saber si se inserto el registro
        int registros = ingresoSancion.ExecuteNonQuery("insertarsancion",
CommandType.StoredProcedure);
}
#endregion
#endregion
#region TablaPosicion
#region Gana Local
if(Convert.ToInt32(lblGolesLocal.Text) > Convert.ToInt32(lblGolesVisitante.Text))
{
        DataSet dsPosiciones = new DataSet();
        DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion,Providers.PostgresSQL);
        dsPosiciones =
recuperar.ExecuteDataSet("seleccionarposicion",CommandType.StoredProcedure);
```

```
        if (dsPosiciones.Tables[0].Rows.Count>0)
        {
                for (int i=0;i<dsPosiciones.Tables[0].Rows.Count;i++)
                {

                        #region Ganador
                if (dsPosiciones.Tables[0].Rows[i]["idequipo"].ToString().Trim() ==
txtIdEquipoLocal.Text.Trim())
                   {
        int puntos = Convert.ToInt32(dsPosiciones.Tables[0].Rows[i]["puntos"]) + 3;
                        int partidosganados =
                Convert.ToInt32(dsPosiciones.Tables[0].Rows[i]["partidosganados"]) +
                1;
                        int partidosjugados =
Convert.ToInt32(dsPosiciones.Tables[0].Rows[i]["partidosjugados"]) + 1;
                        int golesfavor =
Convert.ToInt32(dsPosiciones.Tables[0].Rows[i]["golesfavor"]) +
Convert.ToInt32(lblGolesLocal.Text);
                        int golescontra =
Convert.ToInt32(dsPosiciones.Tables[0].Rows[i]["golescontra"]) +
Convert.ToInt32(lblGolesVisitante.Text);
                        int goldiferencia = golesfavor - golescontra;
                        DatabaseHelper upDate = new
DatabaseHelper(ConectarBD.CadenaDeConexion,Providers.PostgresSQL);

        upDate.AddParameter("@fecha",dtpFecha.Value);

        upDate.AddParameter("@puntos",puntos);

        upDate.AddParameter("@partidosjugados",partidosjugados);

        upDate.AddParameter("@partidosganados",partidosganados);

        upDate.AddParameter("@partidosempatados",Convert.ToInt32(dsPosiciones.
Tables[0].Rows[i]["partidosempatados"]));

        upDate.AddParameter("@partidosperdidos",Convert.ToInt32(dsPosiciones.Ta
bles[0].Rows[i]["partidosperdidos"]));

        upDate.AddParameter("@golesfavor",golesfavor);

        upDate.AddParameter("@golescontra",golescontra);

        upDate.AddParameter("@goldiferencia",goldiferencia);

        upDate.AddParameter("@categoria",Convert.ToInt32( txtIdCategoria.Text));

        upDate.AddParameter("@equipo",Convert.ToInt32(txtIdEquipoLocal.Text));
                                                int actualizar =
upDate.ExecuteNonQuery("updateposicion",CommandType.StoredProcedure);
```

```
}
#endregion
#region Perdedor
if (dsPosiciones.Tables[0].Rows[i]["idequipo"].ToString().Trim() ==
txtIdEquipoVisitante.Text.Trim())
{
        int puntosperdedor =
Convert.ToInt32(dsPosiciones.Tables[0].Rows[i]["puntos"]);
        int partidosperdidos =
Convert.ToInt32(dsPosiciones.Tables[0].Rows[i]["partidosperdidos"]) + 1;
        int partidosjugados =
Convert.ToInt32(dsPosiciones.Tables[0].Rows[i]["partidosjugados"]) + 1;
        int golesfavor =
Convert.ToInt32(dsPosiciones.Tables[0].Rows[i]["golesfavor"]) +
Convert.ToInt32(lblGolesVisitante.Text);
        int golescontra =
Convert.ToInt32(dsPosiciones.Tables[0].Rows[i]["golescontra"]) +
Convert.ToInt32(lblGolesLocal.Text);
        int goldiferencia = golesfavor - golescontra;
        DatabaseHelper upDateperdedor = new
DatabaseHelper(ConectarBD.CadenaDeConexion,Providers.PostgresSQL);

        upDateperdedor.AddParameter("@fecha",dtpFecha.Value);

        upDateperdedor.AddParameter("@puntos",puntosperdedor);

        upDateperdedor.AddParameter("@partidosjugados",partidosjugados);

        upDateperdedor.AddParameter("@partidosganados",Convert.ToInt32(dsPosici
ones.Tables[0].Rows[i]["partidosganados"]));

        upDateperdedor.AddParameter("@partidosempatados",Convert.ToInt32(dsPos
iciones.Tables[0].Rows[i]["partidosempatados"]));

        upDateperdedor.AddParameter("@partidosperdidos",partidosperdidos);

        upDateperdedor.AddParameter("@golesfavor",golesfavor);

        upDateperdedor.AddParameter("@golescontra",golescontra);

        upDateperdedor.AddParameter("@goldiferencia",goldiferencia);

        upDateperdedor.AddParameter("@categoria",Convert.ToInt32(
txtIdCategoria.Text));

        upDateperdedor.AddParameter("@equipo",Convert.ToInt32(txtIdEquipoVisit
ante.Text));
                                                int actualizar =
upDateperdedor.ExecuteNonQuery("updateposicion",CommandType.StoredProcedure
);
```

```
                    }
                    #endregion
                }
        }

}
#endregion

#region Gana Visitante
if(Convert.ToInt32(lblGolesLocal.Text) < Convert.ToInt32(lblGolesVisitante.Text))
{
        DataSet dsPosiciones = new DataSet();
        DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion,Providers.PostgresSQL);
        dsPosiciones =
recuperar.ExecuteDataSet("seleccionarposicion",CommandType.StoredProcedure);
        if (dsPosiciones.Tables[0].Rows.Count>0)
        {
                for (int i=0;i<dsPosiciones.Tables[0].Rows.Count;i++)
                {
                        #region Ganador
                        if
(dsPosiciones.Tables[0].Rows[i]["idequipo"].ToString().Trim() ==
txtIdEquipoVisitante.Text.Trim())
                        {

                                int puntos =
Convert.ToInt32(dsPosiciones.Tables[0].Rows[i]["puntos"]) + 3;
                                int partidosganados =
Convert.ToInt32(dsPosiciones.Tables[0].Rows[i]["partidosganados"]) + 1;
                                int partidosjugados =
Convert.ToInt32(dsPosiciones.Tables[0].Rows[i]["partidosjugados"]) + 1;
                                int golesfavor =
Convert.ToInt32(dsPosiciones.Tables[0].Rows[i]["golesfavor"]) +
Convert.ToInt32(lblGolesVisitante.Text);
                                int golescontra =
Convert.ToInt32(dsPosiciones.Tables[0].Rows[i]["golescontra"]) +
Convert.ToInt32(lblGolesLocal.Text);
                                int goldiferencia = golesfavor - golescontra;
                                DatabaseHelper upDate = new
DatabaseHelper(ConectarBD.CadenaDeConexion,Providers.PostgresSQL);

        upDate.AddParameter("@fecha",dtpFecha.Value);

        upDate.AddParameter("@puntos",puntos);

        upDate.AddParameter("@partidosjugados",partidosjugados);

        upDate.AddParameter("@partidosganados",partidosganados);
```

```
        upDate.AddParameter("@partidosempatados",Convert.ToInt32(dsPosiciones.
Tables[0].Rows[i]["partidosempatados"]));

        upDate.AddParameter("@partidosperdidos",Convert.ToInt32(dsPosiciones.Ta
bles[0].Rows[i]["partidosperdidos"]));

        upDate.AddParameter("@golesfavor",golesfavor);

        upDate.AddParameter("@golescontra",golescontra);

        upDate.AddParameter("@goldiferencia",goldiferencia);

        upDate.AddParameter("@categoria",Convert.ToInt32( txtIdCategoria.Text));

        upDate.AddParameter("@equipo",Convert.ToInt32(txtIdEquipoVisitante.Text
));
        int actualizar =
upDate.ExecuteNonQuery("updateposicion",CommandType.StoredProcedure);
        }
        #endregion
        #region Perdedor
        if (dsPosiciones.Tables[0].Rows[i]["idequipo"].ToString().Trim() ==
txtIdEquipoLocal.Text.Trim())
        {
                int puntosperdedor =
Convert.ToInt32(dsPosiciones.Tables[0].Rows[i]["puntos"]);
                int partidosperdidos =
Convert.ToInt32(dsPosiciones.Tables[0].Rows[i]["partidosperdidos"]) + 1;
                int partidosjugados =
Convert.ToInt32(dsPosiciones.Tables[0].Rows[i]["partidosjugados"]) + 1;
                int golesfavor =
Convert.ToInt32(dsPosiciones.Tables[0].Rows[i]["golesfavor"]) +
Convert.ToInt32(lblGolesLocal.Text);
                int golescontra =
Convert.ToInt32(dsPosiciones.Tables[0].Rows[i]["golescontra"]) +
Convert.ToInt32(lblGolesVisitante.Text);
                int goldiferencia = golesfavor - golescontra;
                DatabaseHelper upDateperdedor = new
DatabaseHelper(ConectarBD.CadenaDeConexion,Providers.PostgresSQL);

        upDateperdedor.AddParameter("@fecha",dtpFecha.Value);

        upDateperdedor.AddParameter("@puntos",puntosperdedor);

        upDateperdedor.AddParameter("@partidosjugados",partidosjugados);

        upDateperdedor.AddParameter("@partidosganados",Convert.ToInt32(dsPosici
ones.Tables[0].Rows[i]["partidosganados"]));
```

```
        upDateperdedor.AddParameter("@partidosempatados",Convert.ToInt32(dsPos
iciones.Tables[0].Rows[i]["partidosempatados"]));

        upDateperdedor.AddParameter("@partidosperdidos",partidosperdidos);

        upDateperdedor.AddParameter("@golesfavor",golesfavor);

        upDateperdedor.AddParameter("@golescontra",golescontra);

        upDateperdedor.AddParameter("@goldiferencia",goldiferencia);

        upDateperdedor.AddParameter("@categoria",Convert.ToInt32(
txtIdCategoria.Text));

        upDateperdedor.AddParameter("@equipo",Convert.ToInt32(txtIdEquipoLocal
.Text));
        int actualizar =
upDateperdedor.ExecuteNonQuery("updateposicion",CommandType.StoredProcedure
);
                                                    }
        #endregion
          }
        }
}
#endregion

#region Empate
if(Convert.ToInt32(lblGolesLocal.Text) == Convert.ToInt32(lblGolesVisitante.Text))
{
        DataSet dsPosiciones = new DataSet();
        DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion,Providers.PostgresSQL);
        dsPosiciones =
recuperar.ExecuteDataSet("seleccionarposicion",CommandType.StoredProcedure);
        if (dsPosiciones.Tables[0].Rows.Count>0)
        {
                for (int i=0;i<dsPosiciones.Tables[0].Rows.Count;i++)
                  {
                        #region Visitante
                        if
(dsPosiciones.Tables[0].Rows[i]["idequipo"].ToString().Trim() ==
txtIdEquipoVisitante.Text.Trim())
                        {
                                int puntos =
Convert.ToInt32(dsPosiciones.Tables[0].Rows[i]["puntos"]) + 1;
                                int partidosganados =
Convert.ToInt32(dsPosiciones.Tables[0].Rows[i]["partidosganados"]);
                                int partidosjugados =
Convert.ToInt32(dsPosiciones.Tables[0].Rows[i]["partidosjugados"]) + 1;
```

```
                    int golesfavor =
Convert.ToInt32(dsPosiciones.Tables[0].Rows[i]["golesfavor"]) +
Convert.ToInt32(lblGolesVisitante.Text);
                    int golescontra =
Convert.ToInt32(dsPosiciones.Tables[0].Rows[i]["golescontra"]) +
Convert.ToInt32(lblGolesLocal.Text);
                    int goldiferencia = golesfavor - golescontra;
                    DatabaseHelper upDate = new
DatabaseHelper(ConectarBD.CadenaDeConexion,Providers.PostgresSQL);

        upDate.AddParameter("@fecha",dtpFecha.Value);

        upDate.AddParameter("@puntos",puntos);

        upDate.AddParameter("@partidosjugados",partidosjugados);

        upDate.AddParameter("@partidosganados",partidosganados);

        upDate.AddParameter("@partidosempatados",(Convert.ToInt32(dsPosiciones.
Tables[0].Rows[i]["partidosempatados"]) + 1));

        upDate.AddParameter("@partidosperdidos",Convert.ToInt32(dsPosiciones.Ta
bles[0].Rows[i]["partidosperdidos"]));

        upDate.AddParameter("@golesfavor",golesfavor);

        upDate.AddParameter("@golescontra",golescontra);

        upDate.AddParameter("@goldiferencia",goldiferencia);

        upDate.AddParameter("@categoria",Convert.ToInt32( txtIdCategoria.Text));

        upDate.AddParameter("@equipo",Convert.ToInt32(txtIdEquipoVisitante.Text
));
        int actualizar =
upDate.ExecuteNonQuery("updateposicion",CommandType.StoredProcedure);
}
#endregion
 #region Local
if (dsPosiciones.Tables[0].Rows[i]["idequipo"].ToString().Trim() ==
txtIdEquipoLocal.Text.Trim())
{
        int puntosperdedor =
Convert.ToInt32(dsPosiciones.Tables[0].Rows[i]["puntos"]) + 1;
        int partidosperdidos =
Convert.ToInt32(dsPosiciones.Tables[0].Rows[i]["partidosperdidos"]);
        int partidosjugados =
Convert.ToInt32(dsPosiciones.Tables[0].Rows[i]["partidosjugados"]) + 1;
```

```
        int golesfavor =
Convert.ToInt32(dsPosiciones.Tables[0].Rows[i]["golesfavor"]) +
Convert.ToInt32(lblGolesLocal.Text);
        int golescontra =
Convert.ToInt32(dsPosiciones.Tables[0].Rows[i]["golescontra"]) +
Convert.ToInt32(lblGolesVisitante.Text);
        int goldiferencia = golesfavor - golescontra;
        DatabaseHelper upDateperdedor = new
DatabaseHelper(ConectarBD.CadenaDeConexion,Providers.PostgresSQL);

        upDateperdedor.AddParameter("@fecha",dtpFecha.Value);

        upDateperdedor.AddParameter("@puntos",puntosperdedor);

        upDateperdedor.AddParameter("@partidosjugados",partidosjugados);

        upDateperdedor.AddParameter("@partidosganados",Convert.ToInt32(dsPosici
ones.Tables[0].Rows[i]["partidosganados"]));

        upDateperdedor.AddParameter("@partidosempatados",(Convert.ToInt32(dsPo
siciones.Tables[0].Rows[i]["partidosempatados"])+1));

        upDateperdedor.AddParameter("@partidosperdidos",partidosperdidos);

        upDateperdedor.AddParameter("@golesfavor",golesfavor);

        upDateperdedor.AddParameter("@golescontra",golescontra);

        upDateperdedor.AddParameter("@goldiferencia",goldiferencia);

        upDateperdedor.AddParameter("@categoria",Convert.ToInt32(
txtIdCategoria.Text));

        upDateperdedor.AddParameter("@equipo",Convert.ToInt32(txtIdEquipoLocal
.Text));
                                                int actualizar =
upDateperdedor.ExecuteNonQuery("updateposicion",CommandType.StoredProcedure
);
   }
  #endregion
   }
}
}
#endregion
#endregion
//Si la variable es diferente de 0  se inserto caso contrario
//No se puede insertar el registro
if (registrosAfectados != 0)
{
```

```
        MessageBox.Show("Registro
Insertado","INFORMACION",MessageBoxButtons.OK,MessageBoxIcon.Information);
}
else
{
        MessageBox.Show("No se pudo
insertar","ERROR",MessageBoxButtons.OK,MessageBoxIcon.Error);
}
}
 //Asignar ninguno al comando luego de insertar o actualizar
 comando = ComandoNpgsql.Ninguno;
 }
 else
 {
         MessageBox.Show("Faltan algunos campos por llenar", "ERROR",
MessageBoxButtons.OK, MessageBoxIcon.Error);
 }
 #endregion
```

Insertar inscripciones

```
if (dtpFechaMaximo.Text != "" && cboCategoria.Text!= "" && cboEquipo.Text !="")
{
        string periodo = "Desde: " + dtpDesde.Value.ToString() + " Hasta: " +
dtpHasta.Value.ToString();
        //Si el comando es insertar
        if (comando == ComandoNpgsql.Insertar)
        {
                #region Ingreso a Tabla de Posiciones

                int puntosperdedor = 0;
                int partidosganados = 0;
                int partidosjugados = 0;
                int partidosempatados = 0;
                int partidoperdidos = 0;
                int golesfavor = 0;
                int golescontra = 0;
                int goldiferencia = 0;
                DatabaseHelper upDate = new
DatabaseHelper(ConectarBD.CadenaDeConexion,Providers.PostgresSQL);

        upDate.AddParameter("@fecha",dtpFechaMaximo.Value);

        upDate.AddParameter("@puntos",puntosperdedor);

        upDate.AddParameter("@partidosjugados",partidosjugados);
```

```
        upDate.AddParameter("@partidosganados",partidosganados);

upDate.AddParameter("@partidosempatados",partidosempatados);

upDate.AddParameter("@partidosperdidos",partidoperdidos);

upDate.AddParameter("@golesfavor",golesfavor);

        upDate.AddParameter("@golescontra",golescontra);

        upDate.AddParameter("@goldiferencia",goldiferencia);

        upDate.AddParameter("@categoria",cboCategoria.SelectedValue);

        upDate.AddParameter("@equipo",cboEquipo.SelectedValue);
        int actualizar =
upDate.ExecuteNonQuery("insertarposicion",CommandType.StoredProcedure);
        #endregion

        //Instanciar una variable para insertar un registro
        DatabaseHelper a = new DatabaseHelper(ConectarBD.CadenaDeConexion,
Providers.PostgresSQL);
        //Añadir los parametros necesarios para el ingreso
            a.AddParameter("@garantia",txtGarantia.Text.Trim());
                a.AddParameter("@inscripcion",txtInscripcion.Text.Trim());
                a.AddParameter("@periodo",periodo);
                a.AddParameter("@fechamaximoinscripcion",Convert.ToDateTime(
dtpFechaMaximo.Text));
                a.AddParameter("@idcategoria",cboCategoria.SelectedValue);
                a.AddParameter("@idequipo",cboEquipo.SelectedValue);
                //Variable para saber si se inserto el registro
                int registrosAfectados = a.ExecuteNonQuery("insertarinscripcion",
CommandType.StoredProcedure);
                //Si la variable es diferente de 0  se inserto caso contrario
                //No se puede insertar el registro
                if (registrosAfectados != 0)
                {
                   MessageBox.Show("Registro
Insertado","INFORMACION",MessageBoxButtons.OK,MessageBoxIcon.Information);
                }
                else
                {
                   MessageBox.Show("No se pudo
insertar","ERROR",MessageBoxButtons.OK,MessageBoxIcon.Error);
                }
                //Ejecutar el procedimiento para recuperar los datos despues de ser
guardados
                dsDatos = a.ExecuteDataSet("seleccionarinscripcion",
CommandType.StoredProcedure);
```

```
                //Llamar al metodo que muestra los datos en controles
                MostrarDatosEnControles();
                MoverComboBox();
                //Habilitar o desabilitar los controles
                EstadoControles(true);

            }
            //Si el comando es actualizar o modificar
            if (comando == ComandoNpgsql.Actualizar)
            {
              //Instanciar una variable para actualizar un registro
                DatabaseHelper a = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);
                //Añadir los parametros necesarios para la actualizacion
                a.AddParameter("@garantia",txtGarantia.Text.Trim());
                a.AddParameter("@inscripcion",txtInscripcion.Text.Trim());
                a.AddParameter("@periodo",periodo);
                a.AddParameter("@fechamaximoinscripcion",Convert.ToDateTime(
dtpFechaMaximo.Text));
                a.AddParameter("@idcategoria",cboCategoria.SelectedValue);
                a.AddParameter("@idequipo",cboEquipo.SelectedValue);
                a.AddParameter("@idinscripcion",txtIDInscripcion.Text.Trim());
                //Variable para saber si se actualizo el registro
                int registrosAfectados = a.ExecuteNonQuery("Updateinscripcion",
CommandType.StoredProcedure);
                //Si la variable es diferente de 0  se actualizo caso contrario
                //No se puede actualizar el registro
                if (registrosAfectados != 0)
                {
                   MessageBox.Show("Datos Actualizados", "INFORMACION",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                }
                else
                {
                   MessageBox.Show("No se pudo
actualizar","ERROR",MessageBoxButtons.OK,MessageBoxIcon.Error);
                }
                //Ejecutar el procedimiento para recuperar los datos despues de ser
guardados
                dsDatos = a.ExecuteDataSet("seleccionarinscripcion",
CommandType.StoredProcedure);
                //Llamar al metodo que muestra los datos en controles
                MostrarDatosEnControles();
                //Habilitar o desabilitar los controles
                EstadoControles(true);
            }
            //Asignar ninguno al comando luego de insertar o actualizar
            comando = ComandoNpgsql.Ninguno;
        }
        else
```

```
            {
                MessageBox.Show("Los campos son requeridos", "ERROR",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                txtGarantia.Focus();
            }
```

Insertar Equipo

```
if (txtNombre.Text.Trim() != "")
            {
                //Si el comando es insertar
                if (comando == ComandoNpgsql.Insertar)
                {
                    //Instanciar una variable para insertar un registro
                    DatabaseHelper a = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);
                    //Añadir los parametros necesarios para el ingreso
                    a.AddParameter("@nombre",txtNombre.Text.Trim());
                    a.AddParameter("@anofundacion",txtAnoFundacion.Text.Trim());
                    //Variable para saber si se inserto el registro
                    int registrosAfectados = a.ExecuteNonQuery("insertarequipo",
CommandType.StoredProcedure);
                    //Si la variable es diferente de 0  se inserto caso contrario
                    //No se puede insertar el registro
                    if (registrosAfectados != 0)
                    {
                            string amarilla = "AMARILLA";
                            string roja = "ROJA";
                            DatabaseHelper ingresoAmarilla = new
DatabaseHelper(ConectarBD.CadenaDeConexion,Providers.PostgresSQL);
                            ingresoAmarilla.AddParameter("@amarilla",amarilla);
                            #region Recuperar ID Fechas Sancionadas
                            DatabaseHelper recuperarIdTiempoSancion = new
DatabaseHelper(ConectarBD.CadenaDeConexion,Providers.PostgresSQL);

        recuperarIdTiempoSancion.AddParameter("@idequipo",txtIdEquipos.Text.Tri
m());
                            DataSet recuperarid = new DataSet();
                            recuperarid =
recuperarIdTiempoSancion.ExecuteDataSet("seleccionarfechassancionadasporidequipo
",CommandType.StoredProcedure);
                            #endregion

        ingresoAmarilla.AddParameter("@idfechassancionadas",Convert.ToInt32(rec
uperarid.Tables[0].Rows[0][0]));
                            #region Recuperar ultimo equipo
                            DatabaseHelper recuperarUltimoDatoEquipo = new
DatabaseHelper(ConectarBD.CadenaDeConexion,Providers.PostgresSQL);
```

```
                    DataSet dsUltimoEqupo = new DataSet();
                    dsUltimoEqupo =
recuperarUltimoDatoEquipo.ExecuteDataSet("seleccionarultimoequipo",CommandTy
pe.StoredProcedure);
                    #endregion

        ingresoAmarilla.AddParameter("@idequipo",Convert.ToInt32(dsUltimoEqup
o.Tables[0].Rows[0][0]));
                    int seInsertoAmarilla =
ingresoAmarilla.ExecuteNonQuery("insertarsancionesfrmequipo",CommandType.Stor
edProcedure);

                    DatabaseHelper ingresoRoja = new
DatabaseHelper(ConectarBD.CadenaDeConexion,Providers.PostgresSQL);
                    ingresoRoja.AddParameter("@roja",roja);
                    #region Recuperar ID Fechas Sancionadas
                    DatabaseHelper recuperarIdTiempoSancionroja = new
DatabaseHelper(ConectarBD.CadenaDeConexion,Providers.PostgresSQL);

        recuperarIdTiempoSancionroja.AddParameter("@idequipo",txtIdEquipos.Text.
Trim());
                    DataSet recuperaridroja = new DataSet();
                    recuperaridroja =
recuperarIdTiempoSancionroja.ExecuteDataSet("seleccionarfechassancionadasporideq
uiporoja",CommandType.StoredProcedure);
                    #endregion

        ingresoRoja.AddParameter("@idfechassancionadas",Convert.ToInt32(recuper
aridroja.Tables[0].Rows[0][0]));
                    #region Recuperar ultimo equipo
                    DatabaseHelper recuperarUltimoDatoEquipoRoja = new
DatabaseHelper(ConectarBD.CadenaDeConexion,Providers.PostgresSQL);
                    DataSet dsUltimoEqupoRoja = new DataSet();
                    dsUltimoEqupoRoja =
recuperarUltimoDatoEquipoRoja.ExecuteDataSet("seleccionarultimoequipo",Comman
dType.StoredProcedure);
                    #endregion

        ingresoRoja.AddParameter("@idequipo",Convert.ToInt32(dsUltimoEqupoRoj
a.Tables[0].Rows[0][0]));
                    int seInsertoRoja =
ingresoRoja.ExecuteNonQuery("insertarsancionesfrmequipo",CommandType.StoredPr
ocedure);
                        MessageBox.Show("Registro
Insertado","INFORMACION",MessageBoxButtons.OK,MessageBoxIcon.Information);
                    }
                    else
                    {
                        MessageBox.Show("No se pudo
insertar","ERROR",MessageBoxButtons.OK,MessageBoxIcon.Error);
```

```
                }
                //Ejecutar el procedimiento para recuperar los datos despues de ser
guardados
                dsDatos = a.ExecuteDataSet("seleccionarequipo",
CommandType.StoredProcedure);
                //Llamar al metodo que muestra los datos en controles
                MostrarDatosEnControles();
                //Habilitar o desabilitar los controles
                EstadoControles(true);

            }
            //Si el comando es actualizar o modificar
            if (comando == ComandoNpgsql.Actualizar)
            {
              //Instanciar una variable para actualizar un registro
                DatabaseHelper a = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);
                //Añadir los parametros necesarios para la actualizacion
                a.AddParameter("@nombre",txtNombre.Text.Trim());
                a.AddParameter("@anofundacion",txtAnoFundacion.Text.Trim());
                a.AddParameter("@idequipo",txtIdEquipos.Text);
                //Variable para saber si se actualizo el registro
                int registrosAfectados = a.ExecuteNonQuery("Updateequipo",
CommandType.StoredProcedure);
                //Si la variable es diferente de 0  se actualizo caso contrario
                //No se puede actualizar el registro
                if (registrosAfectados != 0)
                {
                   MessageBox.Show("Datos Actualizados", "INFORMACION",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                }
                else
                {
                   MessageBox.Show("No se pudo
actualizar","ERROR",MessageBoxButtons.OK,MessageBoxIcon.Error);
                }
                //Ejecutar el procedimiento para recuperar los datos despues de ser
guardados
                dsDatos = a.ExecuteDataSet("seleccionarequipo",
CommandType.StoredProcedure);
                //Llamar al metodo que muestra los datos en controles
                MostrarDatosEnControles();
                //Habilitar o desabilitar los controles
                EstadoControles(true);
            }
            //Asignar ninguno al comando luego de insertar o actualizar
            comando = ComandoNpgsql.Ninguno;
        }
        else
        {
```

```
            MessageBox.Show("El campo descripcion es requerido", "ERROR",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            txtNombre.Focus();
        }
```

Funcion Validar Cedula

```
private bool ValidarCedula()
    {
      //Codigo C# para validar la cedula
      //int par;
      //int impar;
      bool bandera = true;
      string cadena;
      int suma = 0;
      int digitoVerficador;
      int valor = 0;
      //cadena = Convert.ToString(txtCedula.Text);
      cadena = txtCedula.Text.Trim();
      if (cadena.Length == 10)
      {
        for (int i = 0; i <= 8; i++)
        {
          if (i % 2 == 0)
          {
            //string cadena = Convert.ToString(txtCedula[i].text);
            valor = Convert.ToInt32(cadena.Substring(i, 1)) * 2;
            if (valor > 9)
              valor -= 9;
            suma += valor;
          }
          else
          {
            //suma += Convert.ToInt32(txtCedula[i].text);
            suma += Convert.ToInt32(cadena.Substring(i, 1));
          }
        }
        digitoVerficador = 10 - (suma % 10);
        //if (digitoVerficador == Convert.ToInt32(txtCedula[10].text))
        if (digitoVerficador != Convert.ToInt32(cadena.Substring(9, 1)))
        {
          bandera = false;
        }

        //txtCedula
      }
      else
      {
        bandera = false;
      }
```

```
        return bandera;

    }
```

Consultas Jugadores Busqueda rápida

```
void TxtCedulaTextChanged(object sender, EventArgs e)
                {
                        try
        {
          //Chequear que el TextBox no este vacío.
          if (txtCedula.Text != "")
          {
            //Establecer la cadena con el criterio de busqueda.
            string clausulaWhere = "cedula = " + txtCedula.Text.Trim() + "";
            General general = new General();
            dvDatosjugador = general.Filtrar(dvDatosjugador,clausulaWhere);
            dgvJugador.DataSource = dvDatosjugador;

            lblMensaje.Text = "Recuperados  " + dvDatosjugador.Count.ToString() +
"   registros";

          }
        }
        catch (NpgsqlException error)
        {
          ControlExcepciones.TratarErroresWindows(error);
        }
        catch (Exception error)
        {
          ControlExcepciones.TratarErroresWindows(error);

        }
                }


void TxtNombreTextChanged(object sender, EventArgs e)
                {
                        try
        {
          //Chequear que el TextBox no este vacío.
          if (txtNombre.Text != "")
          {
            //Establecer la cadena con el criterio de busqueda.
            string clausulaWhere = "Nombre LIKE '" + txtNombre.Text.Trim() + "%'";
            General general = new General();
            dvDatosjugador = general.Filtrar(dvDatosjugador, clausulaWhere);
```

```
            dgvJugador.DataSource = dvDatosjugador;

            lblMensaje.Text = "Recuperados  " + dvDatosjugador.Count.ToString() +
"   registros";

          }
        }
        catch (NpgsqlException error)
        {
          ControlExcepciones.TratarErroresWindows(error);
        }
        catch (Exception error)
        {
          ControlExcepciones.TratarErroresWindows(error);

        }
              }


void TxtApellidoTextChanged(object sender, EventArgs e)
              {
                    try
        {
          //Chequear que el TextBox no este vacío.
          if (txtApellido.Text != "")
          {
            //Establecer la cadena con el criterio de busqueda.
            string clausulaWhere = "Apellido LIKE '" + txtApellido.Text.Trim() +
"%'";
            General general = new General();
            dvDatosjugador = general.Filtrar(dvDatosjugador, clausulaWhere);
            dgvJugador.DataSource = dvDatosjugador;

            lblMensaje.Text = "Recuperados  " + dvDatosjugador.Count.ToString() +
"   registros";

          }
        }
        catch (NpgsqlException error)
        {
          ControlExcepciones.TratarErroresWindows(error);
        }
        catch (Exception error)
        {
          ControlExcepciones.TratarErroresWindows(error);

        }
              }
```

Insertar Estadio

```
if (txtLugar.Text.Trim() != "" )
            {
                //Si el comando es insertar
               if (comando == ComandoNpgsql.Insertar)
               {

                  //Instanciar una variable para insertar un registro
                  DatabaseHelper a = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);
                  //Añadir los parametros necesarios para el ingreso
                  a.AddParameter("@lugar",txtLugar.Text.Trim());
                  //Variable para saber si se inserto el registro
                  int registrosAfectados = a.ExecuteNonQuery("insertarestadio",
CommandType.StoredProcedure);
                  //Si la variable es diferente de 0  se inserto caso contrario
                  //No se puede insertar el registro
                  if (registrosAfectados != 0)
                  {
                     MessageBox.Show("Registro
Insertado","INFORMACION",MessageBoxButtons.OK,MessageBoxIcon.Information);
                  }
                  else
                  {
                     MessageBox.Show("No se pudo
insertar","ERROR",MessageBoxButtons.OK,MessageBoxIcon.Error);
                  }
                  //Ejecutar el procedimiento para recuperar los datos despues de ser
guardados
                  dsDatos = a.ExecuteDataSet("seleccionarestadio",
CommandType.StoredProcedure);
                  //Llamar al metodo que muestra los datos en controles
                  MostrarDatosEnControles();
                  //Habilitar o desabilitar los controles
                  EstadoControles(true);

               }
               //Si el comando es actualizar o modificar
               if (comando == ComandoNpgsql.Actualizar)
               {
                  //Instanciar una variable para insertar un registro
                  DatabaseHelper a = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);
                  //Añadir los parametros necesarios para el ingreso
                  a.AddParameter("@lugar",txtLugar.Text.Trim());

a.AddParameter("@idestadio",Convert.ToInt32(txtIdEstadio.Text.Trim()));
                  //Variable para saber si se actualizo el registro
```

```
                int registrosAfectados = a.ExecuteNonQuery("Updateestadio",
CommandType.StoredProcedure);
                //Si la variable es diferente de 0  se actualizo caso contrario
                //No se puede actualizar el registro
                if (registrosAfectados != 0)
                {
                   MessageBox.Show("Datos Actualizados", "INFORMACION",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                }
                else
                {
                   MessageBox.Show("No se pudo
actualizar","ERROR",MessageBoxButtons.OK,MessageBoxIcon.Error);
                }
                //Ejecutar el procedimiento para recuperar los datos despues de ser
guardados
                dsDatos = a.ExecuteDataSet("seleccionarestadio",
CommandType.StoredProcedure);
                //Llamar al metodo que muestra los datos en controles
                MostrarDatosEnControles();
                //Habilitar o desabilitar los controles
                EstadoControles(true);
             }
             //Asignar ninguno al comando luego de insertar o actualizar
             comando = ComandoNpgsql.Ninguno;
          }
          else
          {
             MessageBox.Show("Los campos son requeridos", "ERROR",
MessageBoxButtons.OK, MessageBoxIcon.Error);
             txtLugar.Focus();
          }
```

Eliminar estadio

```
//Si hay registro se puede eliminar caso contrario no
          if (totalRegistros > 0)
          {
              //Realizar una pregunta antes de ser eliminado el registro
             if (DialogResult.Yes == MessageBox.Show("Esta seguro de eliminar el
registro ?",
                                 "ELIMINAR",
                                 MessageBoxButtons.YesNo,
                MessageBoxIcon.Information))
             {
                //Obtener el registro a eliminar.
```

```
                DatabaseHelper recuperar = new
DatabaseHelper(ConectarBD.CadenaDeConexion, Providers.PostgresSQL);
                recuperar.AddParameter("@idestadio", Convert.ToInt32(
txtIdEstadio.Text.Trim()));
                //Variable para saber si se elimino o no el registro
                int seElimino = recuperar.ExecuteNonQuery("eliminarestadio",
CommandType.StoredProcedure);
                //Condicion si es resultado de la variable seElimino es diferente de cero
                //se puede eliminar caso contrario un error
                if (seElimino != 0)

                    MessageBox.Show("Registro
Eliminado","INFORMACION",MessageBoxButtons.OK,MessageBoxIcon.Information
);
                else
                    MessageBox.Show("No se puede
Eliminar","ERROR",MessageBoxButtons.OK,MessageBoxIcon.Error);
                //Ejecutar el procedimiento para poder ver los cambios que se ha
realizado
                //Despues de ser eliminados
                dsDatos = recuperar.ExecuteDataSet("seleccionarestadio",
CommandType.StoredProcedure);
                //Metodo para ver los datos en los controles aqui llamamos al metodo
                MostrarDatosEnControles();
            }
        }
        else
        {
            MessageBox.Show("No hay registros a eliminar", "Aviso",
MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
```

ANEXO 3: Manual de usuario

Para poder ingresar al sistema se debe ejecutar el programa LDPHG:

Todas las páginas para los usuarios visitantes contendrán la siguiente información:

Pantalla de inicio de sesión.

Cuando se accede al sistema la primera pantalla a la que es redirigido es a la de ingreso.

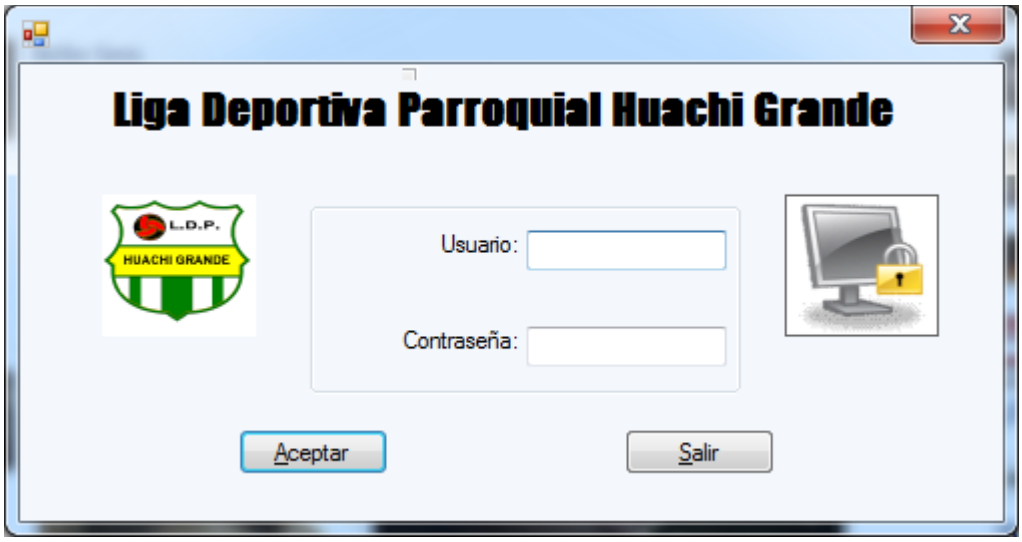


Figura A3.1 Pantalla de inicio de sesión

En esta página el usuario que desea entrar al sistema deberá ingresar el nombre de usuario y su contraseña datos que serán validados y permitirán su acceso; se asignará los permisos de manipulación de datos concedidos al usuario. Cuando el usuario ha sido validado la caja de login cambiará por los datos del usuario en el cual podrá modificar sus datos o cerrar sesión.

Menú administrador

Cuando el usuario se encuentre habilitado aparecerá un menú con las siguientes opciones: Archivo-Ingresos:

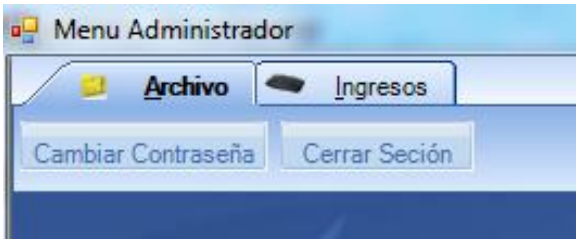


Figura A3.2 Menú administrador

Pantalla Cambio de contraseña

Seleccionando esta opción aparecerán los textos para ingresar la contraseña anterior y la nueva contraseña.

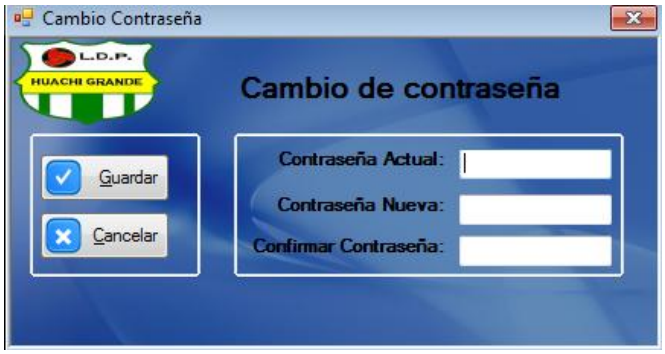


Figura A3.3 Cambio de contraseña

Los botones que se muestra son el de guardar y cancelar. Guardamos los datos una vez ingresadas las claves y que todos los datos estén correctos.

Pantalla Liga

Esta pantalla llamada como frmLiga.cs, contiene la información básica de la Liga, existen botones como nuevo, eliminar, modificar, cancelar y guardar.

Figura A3.4 Datos de la Liga

En esta pantalla podemos agregar un registro con la opción nuevo, a la vez podemos eliminar un registro, podemos modificar un registro con respecto la Liga.

Pantalla cargo

Esta pantalla llamada como frmCargo.cs, contiene la información básica de los cargos que existen en la Liga, existen botones como nuevo, eliminar, modificar, cancelar y guardar.

Figura A3.5 Cargos en la Liga

En esta pantalla podemos modificar cualquier registro de los cargos como son insertar, eliminar y modificar cargos existentes en la Liga.

Pantalla empleado

Esta pantalla llamada como frmEmpleados.cs, contiene la información básica de los empleados que existen en la Liga, existen botones como nuevo, eliminar, modificar, cancelar y guardar.

Figura A3.6 Forma empleado

En esta pantalla podemos manipular los datos de los empleados existentes en la Liga ya sea agregando, eliminando o modificando un empleado.

Menú Secretaria

Cuando el usuario se encuentre habilitado aparecerá un menú con las siguientes opciones: Archivo-Ingresos-Consultas-Reportes:

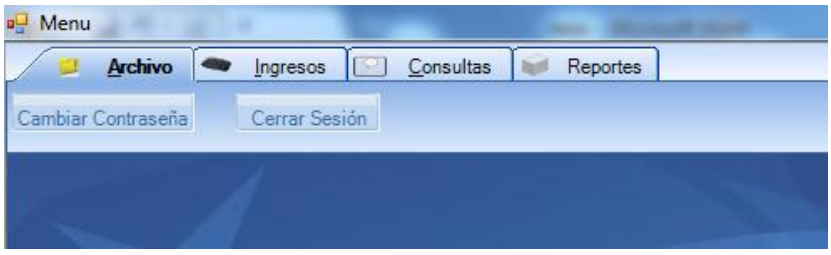


Figura A3.7 Menú Secretaria

Opción Archivo

Dentro de esta opción tenemos los siguientes submenús que son: Cambiar contraseña y cerrar sesión

Opción Ingresos

Dentro de esta opción tenemos los siguientes submenus que son: empleados, categorías, equipos, periodo, inscripciones, jugadores, tiempo sancion, sanciones, estadio, costo arbitraje, calendario, acta de juego.

Figura A3.8 Opción ingresos

Pantalla empleado

Esta pantalla llamada como frmEmpleados.cs, contiene la información básica de los empleados que existen en la Liga, existen botones como nuevo, eliminar, modificar, cancelar y guardar.

Figura A3.9 Forma empleado

En esta pantalla podemos manipular los datos de los empleados existentes en la Liga ya sea agregando, eliminando o modificando un empleado.

Pantalla categoria

Esta pantalla llamada como frmCategoria.cs, contiene la información básica de las categorías que existen en la Liga, existen botones como nuevo, eliminar, modificar, cancelar y guardar.

Figura A4.10 Forma categorías

En esta pantalla podemos manipular los datos de las categorías existentes en la Liga ya sea agregando, eliminando o modificando una categoria.

Pantalla equipos

Esta pantalla llamada como frmEquipo.cs, contiene la información básica de los equipos que existen en la Liga, existen botones como nuevo, eliminar, modificar, cancelar y guardar.

Figura A4.11 Forma equipos

En esta pantalla podemos manipular los datos del equipo existente en la Liga ya sea agregando, eliminando o modificando un equipo.

Pantalla periodo

Esta pantalla llamada como frmPeriodo.cs, contiene la información básica de los periodos que existen en la Liga, existen botones como nuevo, eliminar, modificar, cancelar y guardar.

Figura A4.12 Forma periodo

En esta pantalla podemos manipular los datos de los periodos existentes en la Liga ya sea agregando, eliminando o modificando un periodo.

Pantalla inscripciones

Esta pantalla llamada como frmInscripcion.cs, contiene la información básica de los equipos inscritos que existen en la Liga, existen botones como nuevo, eliminar, modificar, cancelar y guardar.

Figura A4.13 Forma inscripciones

En esta pantalla podemos manipular los datos de las inscripciones existentes en la Liga ya sea agregando, eliminando o modificando los equipos inscritos.

Pantalla jugador

Esta pantalla llamada como frmJugador.cs, contiene la información básica del jugador que existe en la Liga, existen botones como nuevo, eliminar, modificar, cancelar y guardar.

Figura A4.14 Forma jugador

En esta pantalla podemos manipular los datos del jugador existente en la Liga ya sea agregando, eliminando o modificando los equipos inscritos.

Pantalla tiempo sancion

Esta pantalla llamada como frmTiempoSancion.cs, contiene la información básica de la duración de las sanciones que existe en la Liga, existen botones como nuevo, eliminar, modificar, cancelar y guardar.

Figura A4.15 Forma Tiempo sanción

En esta pantalla podemos manipular los datos del tiempo de sanción existente en la Liga ya sea agregando, eliminando o modificando los tiempos de sancion.

Pantalla sanciones

Esta pantalla llamada como frmSanciones.cs, contiene la información básica de las sanciones impuestas al jugador que existe en la Liga, existen botones como nuevo, eliminar, modificar, cancelar y guardar.

Figura A4.16 Forma sanciones

En esta pantalla podemos manipular los datos de las sanciones impuestas a los jugadores existentes en la Liga ya sea agregando, eliminando o modificando las sanciones.

Pantalla estadio

Esta pantalla llamada como frmEstadio.cs, contiene la información básica de los estadios que posee la Liga, existen botones como nuevo, eliminar, modificar, cancelar y guardar.

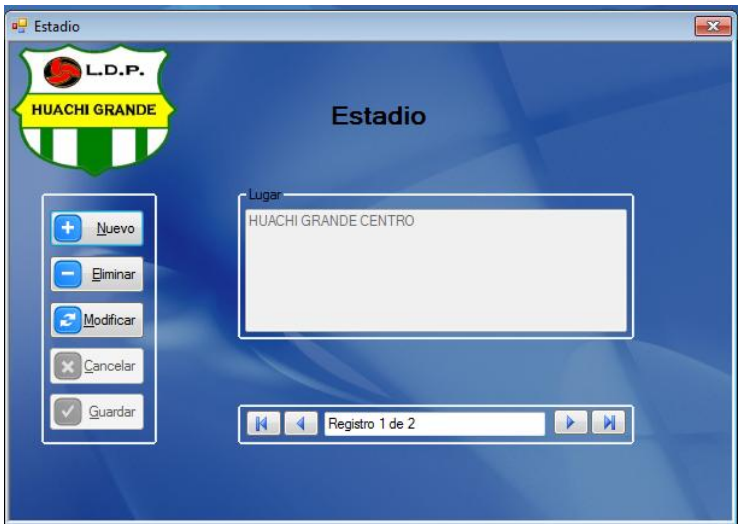


Figura A4.17 Forma estadio

En esta pantalla podemos manipular los datos de los estadios existentes en la Liga ya sea agregando, eliminando o modificando los estadios de la Liga.

Pantalla Costo Arbitraje

Esta pantalla llamada como frmCostoArbitraje.cs, contiene la información básica del costo de arbitraje para el periodo actual, existen botones como nuevo, modificar, cancelar y guardar.

Figura A4.18 Forma costo arbitraje

En esta pantalla podemos manipular los datos de los costos de arbitraje de el ultimo periodo vigente en la Liga ya sea agregando o modificando los costos existentes.

Pantalla calendario

Esta pantalla llamada como frmCalendario.cs, contiene la información básica del los equipos existentes divididos por categorías que existe en la Liga, existen botones como nuevo, eliminar, modificar, cancelar y guardar.

Figura A4.19 Forma calendario

En esta pantalla podemos manipular los datos del ultimo calendario a realizarse en la Liga ya sea agregando, eliminando o modificando los últimos calendarios emitidos.

Pantalla jugador

Esta pantalla llamada como frmActaDeJuego.cs, contiene la información básica las actas de juego que se realizan una ves ingresado las hojas de control en la Liga, existen botones como cancelar y guardar.

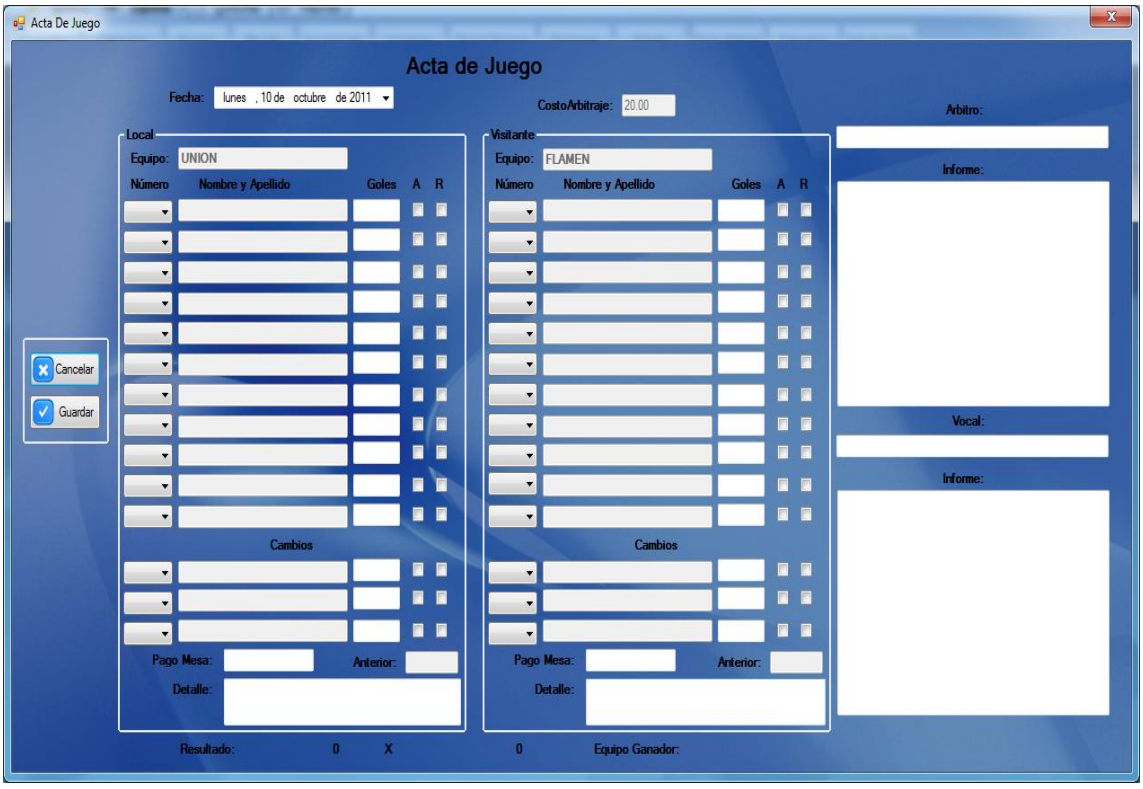


Figura A4.20 Forma acta de juego

En esta pantalla podemos manipular los datos de la acta de juego que se realiza a travez del los resultados de los partidos jugados a travez del ultimo calendario emitido.

Opcion consultas

Dentro de esta opción tenemos los siguientes submenus que son: sanciones, calendario, posiciones y jugadores.

Figura A4.21 Forma opción consultas

Pantalla consulta sanciones

Esta pantalla llamada como frmConsultaSanciones.cs, contiene la información básica de las sanciones emitidas a los jugadores que existen en la Liga.

Figura A4.22 Forma consulta sanciones

En esta pantalla podemos consultar los datos de las sanciones que se han impuesta a los jugadores de acuerdo a las fechas que deseamos.

Pantalla consulta calendario

Esta pantalla llamada como frmConsultaCalendario.cs, contiene la información básica del último calendario de juego emitido en la Liga.

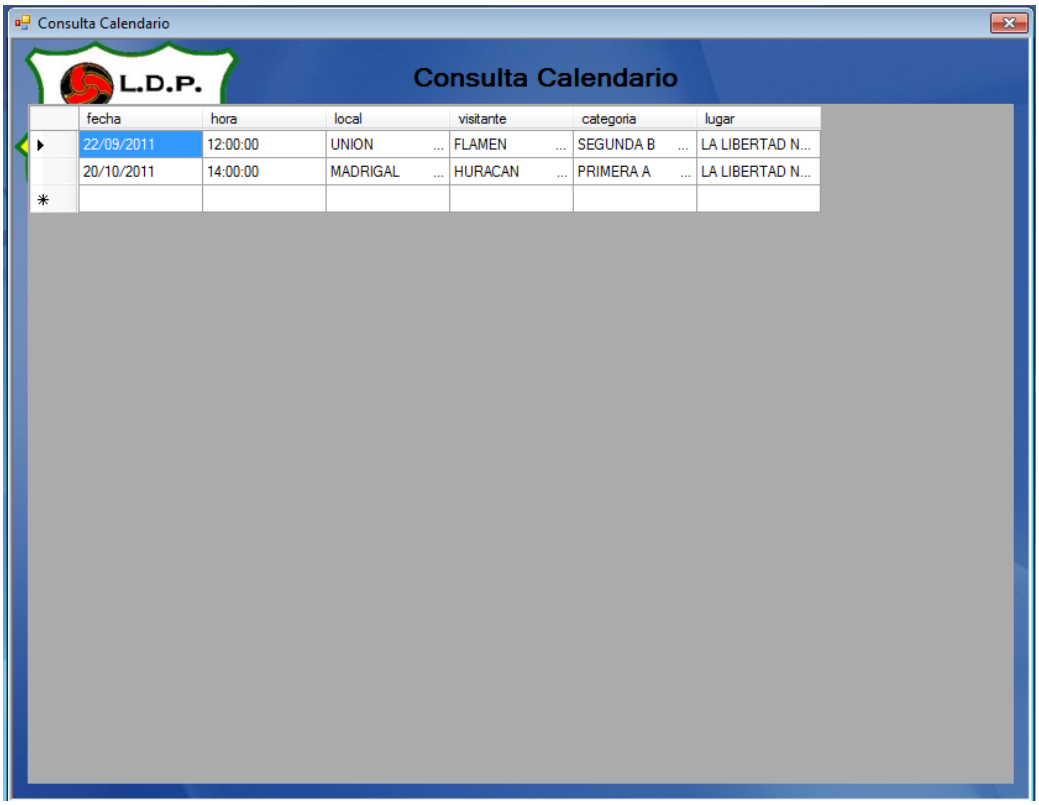


Figura A4.23 Forma consulta calendario

En esta pantalla podemos consultar los datos del último calendario de juego que se ha emitido por medio de la Liga.

Pantalla consulta tabla de posiciones

Esta pantalla llamada como frmConsultaTablaPosiciones.cs, contiene la información básica de la tabla de posiciones de acuerdo a la categoría.

Figura A4.24 Forma consulta tabla de posiciones

En esta pantalla podemos consultar los datos de las tablas de posiciones de acuerdo a su categoría siendo esta actualizada.

Pantalla consulta jugador

Esta pantalla llamada como frmConsultaJugador.cs, contiene la información básica de los jugadores que existen en la Liga.

Figura A4.25 Forma consulta jugador

En esta pantalla podemos consultar los datos de los jugadores existentes en la Liga buscando de acuerdo a su numero de cedula, nombre, apellido.

Opcion consultas

Dentro de esta opción tenemos los siguientes submenus que son: calendario, posiciones, jugadores y jugadores por equipo.

Figura A4.26 opcion reportes

Pantalla reporte calendario

Esta pantalla llamada como frmReporteCalendario.cs, contiene la información básica del último calendario de juego emitido en la Liga, en este reporte lo puede imprimir.

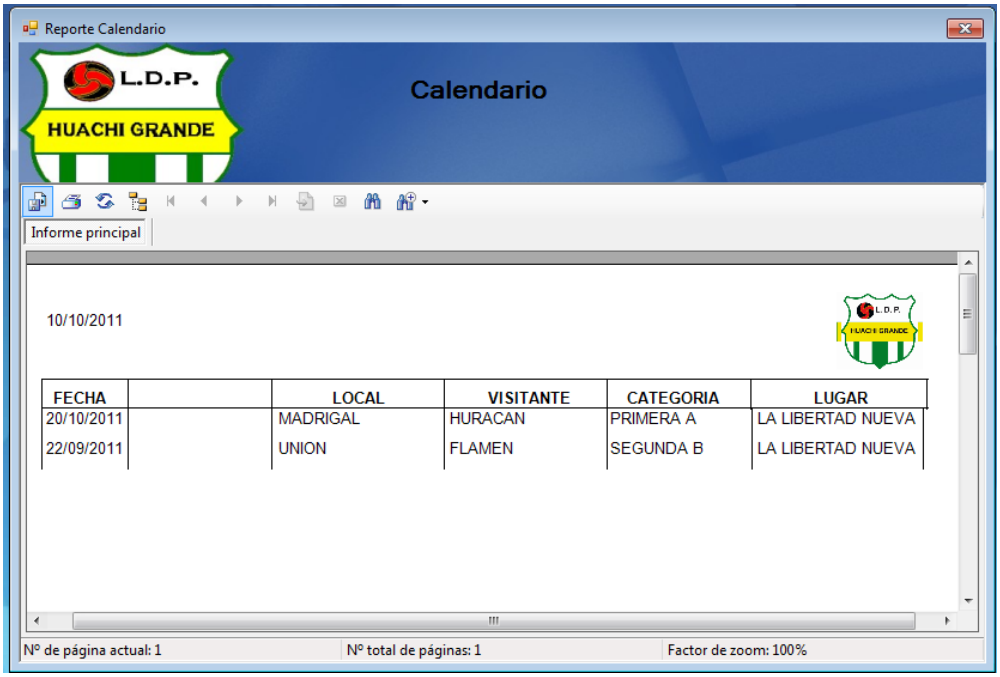


Figura A4.27 Reporte calendario

En esta pantalla podemos imprimir los datos del último calendario de juego que se ha emitido por medio de la Liga.

Pantalla reporte tabla de posiciones

Esta pantalla llamada como frmReportePosiciones.cs, contiene la información básica de la tabla de posiciones de acuerdo a la categoría.

Figura A4.28 Reporte tabla de posiciones

En esta pantalla podemos imprimir los datos de las tablas de posiciones de acuerdo a su categoría siendo esta actualizada.

Pantalla reporte jugador

Esta pantalla llamada como frmReporteJugador.cs, contiene la información básica de los jugadores que existen en la Liga.

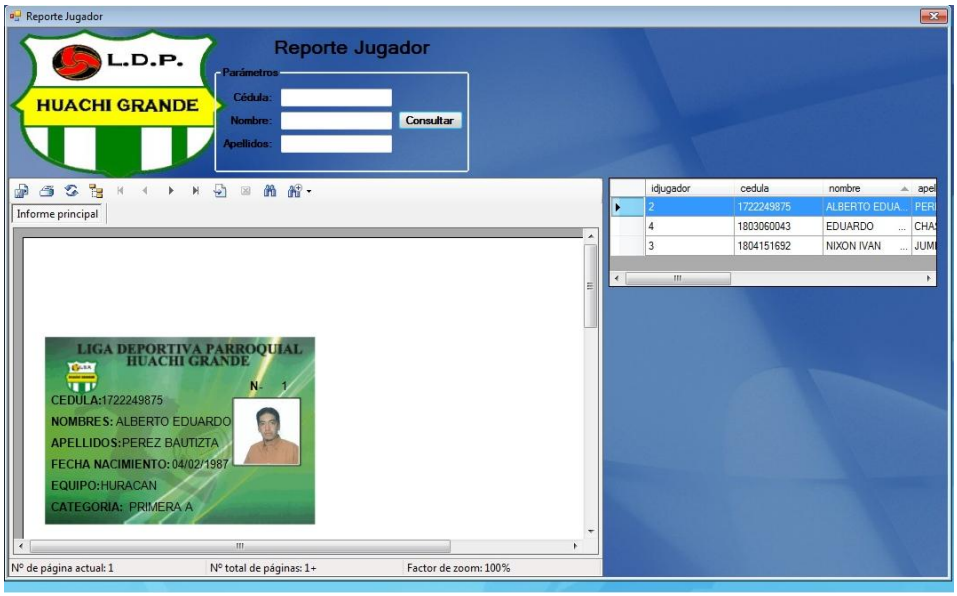


Figura A4.29 Reporte jugador

En esta pantalla podemos imprimir los datos de los jugadores existentes en la Liga buscando de acuerdo a su numero de cedula, nombre, apellido de acuerdo al periodo que esta vigente.

Pantalla reporte jugador por equipo

Esta pantalla llamada como frmReportePorEquipo.cs, contiene la información básica de los jugadores distribuidos pos sus equipos que existen en la Liga.

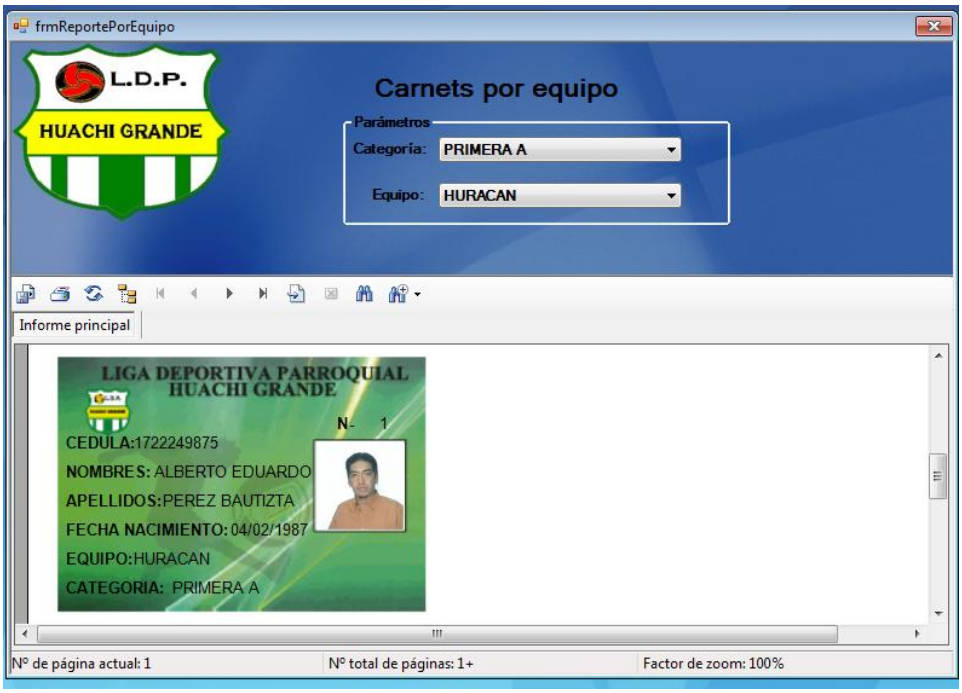


Figura A4.30 Reporte por equipos

En esta pantalla podemos imprimir los datos de los jugadores existentes en la Liga de acuerdo a los parámetros establecidos como la categoría y que equipo en este se imprimen todos los carnets que se encuentren habilitados en ese equipo.

Pantalla inicio del portal

Esta pantalla llamada como index.php, contiene la información básica de la liga como es la bienvenida y algunas reglas internas.

Figura A4.31 Inicio del portal de consultas

Pantalla calendario

Esta pantalla llamada como calendario.php, contiene la información básica del último calendario de juego emitido en la Liga.

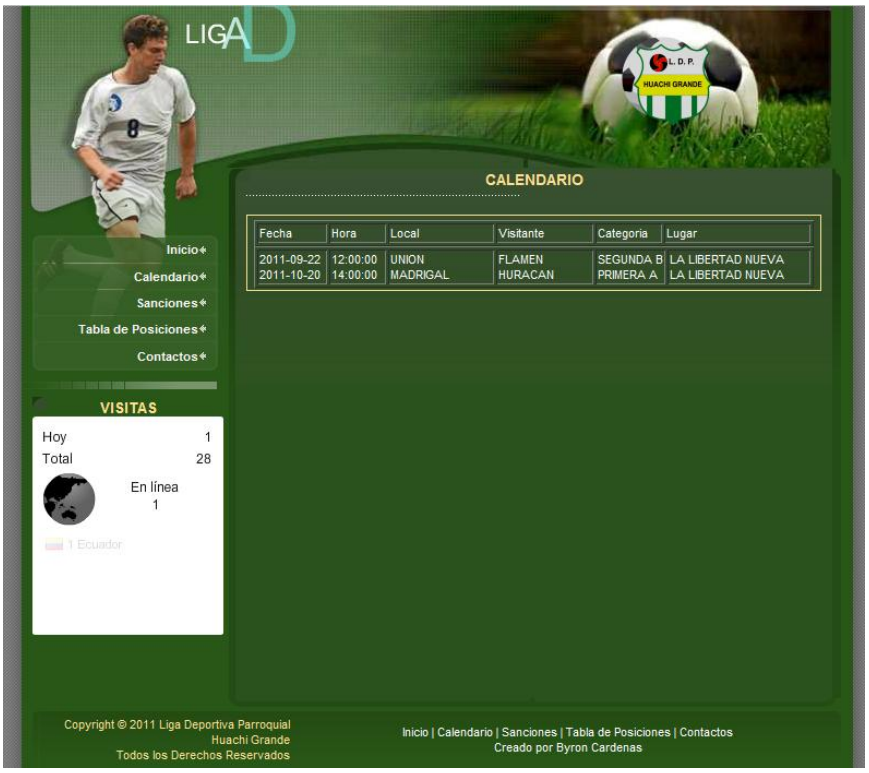


Figura A4.32 Consulta calendario

En esta pantalla podemos consultar los datos del último calendario de juego que se ha emitido por medio de la Liga.

Pantalla sanciones

Esta pantalla llamada como sanciones.php, contiene la información básica del último calendario de juego emitido en la Liga.

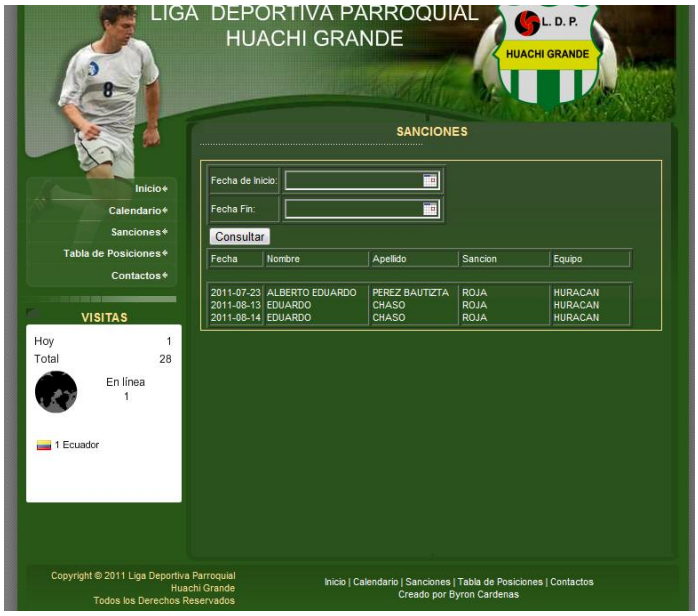


Figura A4.33 Consulta sanciones

En esta pantalla podemos consultar los datos de las sanciones que se han impuesta a los jugadores de acuerdo a las fechas que deseamos.

Pantalla tabla de posiciones

Esta pantalla llamada como tabla.php, contiene la información básica de la tabla de posiciones de acuerdo a la categoría.

Figura A4.34 Consulta tabla de posiciones

En esta pantalla podemos consultar los datos de las tablas de posiciones de acuerdo a su categoría siendo esta actualizada.

Pantalla contactos

Esta pantalla llamada como contactos.php, contiene la información básica de donde esta ubicada la Liga y a la se puede enviar mensajes a un correo electrónico de la Liga.

Figura A4.35 Direccion y mensajes.