BENEMÉRITA UNIVERSIDAD AUTONOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA COMPUTACION

"TUTORIAL EN LINEA DE TEORIA DE JUEGOS"

TESIS PROFESIONAL

QUE PARA OBTENER EL TITULO DE:

LICENCIADO EN CIENCIAS DE LA COMPUTACION

PRESENTA

FERNANDO RODRÍGUEZ BERNAL

ASESOR

M.C.GARCÉS BÁEZ JOSÉ ALFONSO

PUEBLA PUE 2008

Resumen

Como resumen de este trabajo de investigación de tesis llamado Tutorial en Línea de Teoría de Juegos encontraras puntos básicos de lo que es la teoría de juegos en donde se puede ocupar.

Encontraras temas relacionado con UML desde su estandarización, d diferentes tipos de diagramas y ejemplos de diagramas UML, se muestran temas relacionados con las herramientas de programación que se ocuparon para realizar el sistema como SQL, sus orígenes y características principales, JSP y MYSQL con algunas de las consultas que se pueden realizar a una base de datos.

Encontraras puntos relacionados con lo que es un Sistema en Línea y los requerimientos para que este sistema pueda funcionar correctamente, Casos de Uso del usuario y administrador que serán las funciones que desempeñaran dentro del Sistema, Su diseño del Sistema así como su arquitectura de la base de datos y los diagramas que llevaron a realizar la Base de Datos.

La implementación de las herramientas que se ocuparon para realizar la interfaz del sistema y su funcionamiento.

Se mostraran los resultados del sistema con las pantallas que el usuario y administrador tendrán que interactuar así como su código de estas paliaciones

Índice

Introducción

El presente trabajo de tesis el cual es titulado Tutorial en Línea de Teoría de Juegos, pretende mostrar lo que es la teoría de juegos.

El objetivo de este que el lector tenga los conocimientos necesarios de lo que es la teoría de juegos y todos los términos que esta implica y la importancia de que se difundan juegos de esta teoría en la WEB.

Este tutorial permitirá que las personas que estén interesadas en la teoría de juegos puedan interactuar con juegos en línea, de esta área ya que solo encuentran ejercicio sobre matemáticas.

En este Sistema de Tutorial en Línea de Teoría de Juegos los usuarios tendrán las opciones no solo de ver información de teoría de juegos si no también podrán interactuar con los ejercicios que se le proporcionen y tener asesoría en línea, así como poder enviar sus comentarios al administrador para proporcionar nuevos juegos y tener una asesoría en línea.

Se tiene contemplado que el Sistema de Tutorial en Línea de Teoría de Juegos contenga un administrador que es el que va a administrar las cuentas de usuario así como es el que va a proporcionar los nuevos juegos en línea a los que los usuarios tendrán acceso.

El sistema que se menciona es una aplicación en WEB para la que se utilizo una arquitectura de software libre que incluye código PHP, Código Java Script, como lenguaje de programación, apache como servidor WEB, apache como servidor Web y MySQL como servidor de Base de Datos, y Flash como diseño de animación.

Este documentó de Tesis esta organizado de la siguiente forma:

CAPITULO I. Marco Teórico. Se definen los conceptos básicos de la Teoría de Juegos, UML, como Herramientas de Programación, SQL, Lenguaje de Manipulación de datos, JSP, MySQL.

CAPITULO II. Análisis. Se analizara las especificaciones de requerimientos, también se mostraran los casos de uso.

CAPITULO III. Diseño. Se Presenta la arquitectura del sistema, las tablas de la base de dato, diccionario de datos.

CAPITULO IV. Implementación Se muestra las herramientas que se usaron para desarrollar este sistema, las herramientas para la interfaz grafica.

CAPITULO V. Pruebas. Muestra los resultados que se obtuvieron mediante las pruebas, mostrando las ventanas de acceso a usuarios con sus restricciones, pantalla de juegos, comentarios como modo usuario y la aplicación del administrador, baja, modifica, alta de usuarios y alta de juegos.

CAPITULO I Marco Teórico

1.1 Teoría de Juegos

Dentro del marco histórico de la teoría de juegos no hay que olvidar a esos "anónimos" personajes que contribuyeron de manera notable, ya no sólo al uso, sino también a la axiomatización de esta rama que no hace más de un siglo se tenía por desconocida.

No obstante, cabe reseñar que estos grandes científicos, ya no sólo matemáticos, sino también economistas, han desarrollado todo su trabajo en el siglo XX consiguiendo grandes aplicaciones de esta teoría al mundo actual y no sólo en el tema económico sino en otros muchos aspectos mencionados con anterioridad.

La *teoría de juegos* puede decir que nace hacia 1913 con los trabajos de Ernst Zermelo, un matemático y filósofo alemán, en diversos tipos de juegos, como el ajedrez, demostrando que son resolubles. Hacia los años 20, los matemáticos Emile Borel y Von Neumann estudian los equilibrios de tipo *minimax* (del que comentaremos algo posteriormente) en los juegos de suma cero, es decir, aquellos juegos en los que lo que gana uno lo pierde el otro.

Sin embargo, el primer gran avance en la teoría de juegos ocurre allá por los años 40 con la publicación de un libro denominado *The Theory of Games and Economic Behavior* escrito por Von Neumann y Oscar Morgensten en el año 1944. En este libro se divulgaba una formalización general de los juegos en su forma *extendida y normal*, se introdujo el concepto de estrategias en los juegos extensivos y propuso aplicaciones. Sin lugar a dudas era un libro en el cual se trató de manera rigurosa de axiomatizar la teoría de juegos. Su importancia fue tal que se llegó a decir que diez libros como el escrito por Neumann y Morgensten podrían garantizar el futuro de la economía.

En los años 50 hubo un desarrollo importantísimo de estas ideas en la Universidad de Princeton. Sin duda alguna, de los más importantes destacamos a Duncan Luce que junto a Howard Raiffa en 1957 difundieron resultados de estas ideas en su libro introductorio denominado *Games and Decisions*. Harold Kuhn en 1953 trabajó en definir el concepto de la información en los juegos y Lloyd Shapley, también ese mismo año, permitió establecer una forma de atacar a los juegos cooperativos, esos en los que los jugadores pueden establecer contratos.

Pero sin lugar a dudas el máximo representante de todos ellos es John Nash quien en 1950 definió el concepto de *equilibrio nash* consiguiendo extender la teoría a los juegos no-cooperativos mucho más generales que los de suma cero en los que había trabajado Von Neumann. La demostración de

que todo juego no-cooperativo tenía al menos un punto de equilibrio fue la tesis de John Nash de aproximadamente 27 hojas. [1]

Señalemos un hecho que llama mucho la atención para darse cuenta de la importancia de la teoría de juegos, y es que durante esa época, el Departamento de Defensa de los EE.UU. fue el que financió las investigaciones en el tema debido a que la mayor parte de las aplicaciones de los juegos de tipo *suma cero* se concentraban en estrategias militares.

En los años 60, más concretamente en 1967, destacamos al húngaro John Harsanyi quien extendió la teoría de juegos a juegos de información incompleta, es decir, aquellos en que los jugadores no conocen todas las características del juego como, por ejemplo, lo que obtienen el resto de jugadores de recompensa.

En los años 70 nos encontramos con un problema: la multiplicidad de equilibrios de Nash, muchos de los cuales no eran soluciones razonables a juegos. En este campo, el alemán Reinhard Selten definió en 1975 el concepto de equilibrio perfecto en el subjuego para juegos de información completa. Además dio una generalización para el caso de juegos de información imperfecta.

Avanzando unos 30 años nos hallamos en la época actual y en este tiempo también existe gente interesada en la teoría de juegos. Así nos encontramos con dos economistas estadounidenses llamados Thomas Schelling y Robert Aumann quienes estudiando ejemplos tan fáciles como el dilema del prisionero, el modelo halcón-paloma o la guerra de sexos, han llegado a la conclusión que estos ejemplos pueden ayudar a tomar una decisión política o económica adecuada, prevenir guerras comerciales, de precios o hasta incluso conflictos bélicos, así como averiguar por qué va a ser más o menos eficiente un determinado trabajo en equipo.

Un juego de enfrentamiento es un proceso en que dos o más personas toman decisiones y acciones, la estructura de las cuales está inscrita en un conjunto de reglas (que pueden ser formales o informales), a fines de obtener beneficio y derrotar al contri cario. Cada combinación de decisiones y acciones determina una **situación** particular, y, dado que las decisiones y acciones de los agentes involucrados (llamados los **jugadores**) pueden ser combinadas de numerosas formas, las situaciones generadas también serán numerosas y su magnitud igual a las de las combinaciones de decisiones y acciones de los agentes.

El conjunto total de situaciones posibles es el **cuadro situacional del juego**. Siguiendo con este razonamiento, encontramos que cada situación (es decir, cada punto del cuadro situacional) genera una combinación de premios determinada. El premio que le da a un jugador una situación particular puede ser comparado con los premios que le ofrecen las otras situaciones.

1

Un concepto importante es el de **pago**. Como se dijo, cada situación particular ofrece una combinación de premios, de la manera siguiente: si se trata de dos jugadores, la situación ofrece un premio para el primero y otro para el segundo. Si se trata de tres jugadores, la situación genera un premio para cada jugador. Ésta es la lógica de los premios y las situaciones. A cada premio se le llama pago.

La Teoría de Juegos nos ayuda a analizar juegos en los que dos o más personas compiten por un único premio o pago (juegos de **suma cero** de los pagos) y juegos en los que se compite por premios que pueden ser obtenidos simultáneamente (juegos de **suma no-cero**. La Teoría de Juegos enseña que la interacción de los jugadores generará una situación más probable, o un conjunto de situaciones igualmente probables. A esta situación o conjunto de situaciones se les llamará la **solución del juego**. La solución del juego se sustenta en que la conducta de cada jugador llega a *engancharse* con la de los otros, derivando todo esto en situaciones más fuertes que otras. Las situaciones más fuertes son las que serán producidas con la mayor probabilidad, y debido a esto es que se considera que la solución o desenlace del problema del juego corresponde a la situación o situaciones más fuertes, más probables.

Viene aquí lo que yo considero el corazón de los beneficios ofrecidos por la Teoría de Juegos. El análisis de un juego lleva muchas veces a que se determine cuál va a ser el punto final de solución de dicho juego A este resultado se le denominará resultado **inminente o fatal del juego**. Debo decir, no obstante, que en la realidad existen muchos juegos cuyo final es *imposible de determinar*, incluso con la ayuda de la Teoría de Juegos: estos juegos no tienen resultados inminentes, o, si es que los tienen, éstos no son previsibles y la Teoría de Juegos no puede predecirlos.

Tal es el caso de un juego de ajedrez, el cual es un juego de suma cero: todo lo que la Teoría de Juegos nos puede decir acerca de este juego es que uno de los dos jugadores ganará y el otro perderá el juego. Al margen de esta grave circunstancia, la Teoría de Juegos sí puede ayudarnos a determinar los resultados de otros muchos importantes juegos y situaciones de negociaciones e intereses en conflicto.

La Teoría de Juegos es importante porque permite hallar los resultados inminentes o fatales de numerosos juegos diversos que debemos enfrentar cotidianamente en el mundo real. La Teoría de Juegos no deja de ser importante sólo porque no puede analizar la totalidad de los juegos.

Para usar la Teoría de Juegos como una aplicación para una situación real, se requiere construir modelos simplificados de la realidad. En estos modelos, se tendrá que representar a cada jugador con sus respectivas formas de conducta. Cuando se trata de un juego en que usted enfrenta a un único rival, normalmente puede usted decir que conoce perfectamente cuál es su propia forma de actuar, pero ignora o conoce sólo en parte la de su rival u

oponente. Por esto se hace más fácil representar simplificadamente su propia conducta que representar la conducta del rival.

En cualquier caso, se requiere representar adecuadamente las conductas de los dos (o más) jugadores que intervienen. Nuestra conducta será conocida con certidumbre, mientras que la del rival sólo en forma probable (en lenguaje científico, *estocástica*.

A veces se necesitará plantear dos o más representaciones de la conducta probable del rival. Cada representación recibe el nombre de escenario. Cada escenario es un juego simple. El conjunto de dos o más escenarios es un juego compuesto.

En la vida cotidiana se presentan muy a menudo fenómenos o situaciones que involucran a dos o más partes con intereses diferentes y con la posibilidad de llevar a cabo diversas acciones para lograr su objetivo. Este tipo de situaciones se llama *situaciones conflictivas* o, para abreviar más, *conflictos*.

Un conflicto típico se caracteriza por tres componentes básicos:

- Partes interesadas
- Decisiones posibles
- Intereses de las partes

Las situaciones conflictivas extraídas de la vida cotidiana suelen ser bastante difíciles de analizar por su complejidad. A esto hay que añadir diversos factores, aunque haya alguna parte de ellos que ni influya en el desarrollo del conflicto ni en el resultado. Por eso, para analizar estos conflictos es necesario olvidarse de estos factores secundarios, de manera que si las condiciones son óptimas se pueda construir un modelo normal y simplificado. Dicho modelo se suele denominar *juego*. Este concepto se diferencia del conflicto en que se desarrolla de acuerdo a unas reglas definidas de manera total.

La necesidad de estudiar conflictos susceptibles de ser representados con modelos matemáticos simples (juegos) ha dado lugar a un aparato matemático llamado ***teoría de juegos***.

Las partes interesadas se llaman *jugadores*. Cada acción posible dentro del marco de las reglas establecidas se denomina *estrategia*. Bajo condiciones de conflicto, cada jugador escoge su estrategia (la que más le puede convenir por ejemplo) y como resultado se genera un conjunto de estrategias entre todos los jugadores que se denomina *situación*. Para medir el interés de cada jugador en una situación concreta se asigna un número que expresa, por decirlo de manera pedante, la satisfacción de sus intereses en dicha situación. A este número se le llama *pago* del jugador en la situación dada.

Bajo estas circunstancias, el desarrollo del conflicto no es otra cosa que la elección por parte de los jugadores de determinadas estrategias y la

obtención de un pago proveniente de otros jugadores o de alguna otra fuente. Esto da lugar a lo que se conoce como *teoría de juegos con utilidad*.

Por ejemplo la evaluación de una situación por parte de un jugador basándose en su utilidad no siempre es posible y, en algunas situaciones, puede no-tener sentido. La solución entonces no es abordar la situación de manera cuantitativa sino indagar en las preferencias comparativas que una determinada situación proporciona a cada jugador. La teoría de juegos bajo este enfoque empieza a llamarse *teoría de juegos con preferencias*.

Una vez que hayamos elaborado nuestro principio de optimalizad, nuestro segundo paso es poner en práctica estos principios, es decir, establecer la existencia de *estrategias óptimas*. Para terminar, buscaremos esas estrategias.

Todo lo expuesto anteriormente se basa en un concepto: *equilibrio*. Se denomina *situación de equilibrio* a una situación que todos los jugadores están interesados en preservar ya que cada jugador obtiene su pago máximo.

Como se mencionó, la teoría de juegos debe recurrir al soporte de la teoría y técnicas de optimización. A diferencia de la aplicación que recibe en otros campos de estudio, la optimización es utilizada en la teoría de juegos para producir criterios de decisión-acción que aprovechen la escasa información para producir la mejor adivinanza educada que fuese posible. En un extremo están los problemas en los que se carece absolutamente de información. En tales casos, la teoría debe presentar los mecanismos de decisión-acción que permitan operar en tinieblas.

Naturalmente, aunque actuar en oscuridad con los criterios óptimos de la teoría de juegos es menos riesgoso que si se prescinde de ellos, el riesgo siempre es alto cuando se carece de información. En el otro extremo, cuando tenemos todos los datos necesarios, surge la situación ya mencionada en que sólo se requiere optimización pura, y en que la teoría de juegos ya no es requerida. Partiendo del punto de información cero, a medida que se dispone de información en mayor cantidad y calidad, las técnicas de optimización aplicadas a la teoría de juegos producen mecanismos de decisión-acción de cada vez menor riesgo.

Por todo lo dicho, la teoría de juegos debe ser vista como un artefacto destinado a reducir el riesgo implícito por situaciones en que se carece de información respecto a los intereses y modos de acción del competidor.

1.1.2 Estadística y teoría de juegos

El riesgo forma parte de todo proceso de competencia de naturaleza trascendente.

Como vimos, la teoría de juegos reduce el riesgo de incurrir en fracaso como resultado de nuestras propias decisiones, pero no lo elimina. El riesgo es la contrapartida de la probabilidad de los eventos. Si un evento es p% probable de suceder, entonces el riesgo de que no suceda ese evento es (1-p) %. Veamos un ejemplo muy simplificado de una situación real: si decidimos comprar una patente cara porque pensamos que el mercado va a demandar los servicios derivados de ella, entonces el riesgo implícito es el complemento de la probabilidad de que en verdad haya una respuesta favorable.

Al tratar con probabilidades, tratamos con riesgos. Si compramos la patente, lo hacemos porque es probable que haya considerable interés por parte de nuestros clientes potenciales. Si sucediera que (recordemos que se trata de un ejemplo muy simplificado) la demanda no es atraída por el producto o servicio derivado de la patente, entonces el evento probable no se produjo, y el riesgo cristalizó. Todo evento riesgoso siempre puede terminar de manera desfavorable.

La teoría de juegos puede crear los mecanismos para definir la naturaleza del riesgo que enfrentamos (en muchos casos podrá incluso "inventar" criterios), pero no termina con le problema del riesgo. Por lo tanto, si un individuo se decide a adquirir un terreno porque la teoría de juegos le muestra que es probable que una serie de agentes se interesen en arrendar diferentes secciones de tal espacio, ha de tomar en conciencia del nivel de riesgo asumido. Pero si este proceso se llega a repetir con exactamente las mismas características en todos los casos en un número grande de veces, entonces el conjunto total de acciones ofrecerá ventaja para nuestro agente.

Esta es una consideración interesante, pero dado que es teórica no es aconsejable tomarla como una referencia concluyente.

1.1.3 Clasificación de los juegos

Las situaciones de conflicto reales conducen a una gran diversidad de juegos. En la actualidad no existe ninguna clasificación universal de los juegos, aunque éstos se diferencian por diversos criterios como: número de participantes, número de estrategias, relación entre los jugadores, tipo de pago, número de movimientos, cantidad de información que posee cada jugador.
Vamos a analizar sin entrar en muchos detalles estas diferencias.

- **Número de jugadores.** Dependiendo del número de jugadores se definen tres tipos: *juegos de un jugador* (sin consideración en teoría de juegos),

juegos de dos jugadores (la más estudiada) y *juegos n-personales* con un proceso de simulación y resolución muy dificultoso.

- **Número de estrategias.** Los juegos se dividen en *juegos finitos* en los que cada jugador tiene un número finito de estrategias, y *juegos infinitos* en los que al menos un jugador posee infinitas estrategias.

- **Relación entre los jugadores.** Se clasifican en *juegos sin coaliciones* en los que los jugadores no pueden firmar ni acuerdos ni coaliciones, *juegos con coaliciones* y *juegos cooperativos* en los cuales los acuerdos se firman con anterioridad y deben ser respetados obligatoriamente.

- **Tipo de pago.** Se distinguen los *juegos de suma cero* en el que el beneficio de un jugador implica la pérdida en misma cantidad de otro y *juegos de suma no nula*.

- **Número de movimientos.** Los juegos se dividen en *juegos de un paso* que terminan cuando cada jugador realiza un movimiento, y *juegos multiplazos* los cuales también se dividen en *juegos de posición* (cada jugador puede realizar más de un movimiento en el tiempo, *juegos estocásticos* (al elegir una nueva posición existe probabilidad de volver a la anterior), *juegos de tipo duelo* (se caracterizan por el instante en el cual se hace el movimiento y por la probabilidad de obtener un pago dependiendo del tiempo transcurrido.

- **Información disponible.** Los juegos se clasifican en *juegos de información completa* en los que cada jugador conoce los movimientos hechos por los demás, y *juegos de información incompleta* en los que no se conocen todas las jugadas anteriores.

Obviamente existen otros tipos de juegos. Dependiendo de su clase, se trata de elaborar su método de solución. No obstante, cabe destacar que aquí no se enmarca los juegos de manera rigurosa en determinadas clases, es decir, que un mismo juego puede pertenecer a diferentes clases.

1.1.4 Uso de la Teoría de Juegos

Sin duda alguna, una de nuestras mayores preguntas es saber para qué sirve toda esta maraña de ideas y de conceptos, Bueno la respuesta está ahí fuera. La teoría de juegos ha tenido una aplicación enorme en muchos campos del mundo actual, desde la economía hasta la biología.

Y a continuación se muestra algunos ejemplos, espero que estos sean de bastante utilidad para entender la magnitud de estas ideas:

- **El análisis de las negociaciones.** Las negociaciones entre sindicato y empresa, por ejemplo, se pueden analizar como juegos en que las partes tratan de dividir el excedente de la empresa antes de pagar los salarios.

- **Análisis de licitaciones.** Las empresas y el Estado utilizan procesos de licitación para comprar o vender bienes y servicios. Es importante saber cuáles son los mecanismos de licitación adecuados ante cada tipo de licitación y sus debilidades.

- **Análisis empresarial.** La teoría de juegos permite evaluar la eficiencia que puede tener un determinado equipo de trabajo en una empresa.

- **El comportamiento de las firmas ante la entrada de competencia.** Las firmas pueden ser agresivas frente a la nueva competencia, reduciendo precios y aumentando el gasto publicitario o pueden acomodar la entrada, tratando de llegar a un entendimiento con la firma entrante.

- **Los juegos de atrición.** En este tipo de juegos lo que se evalúa es la capacidad para resistir y, por lo tanto, permiten evaluar la situación de defensa de un país.

- **Estrategias en comercio internacional.** En el comercio internacional, los gobiernos protegen la producción nacional a costa de la extranjera, evaluando el costo que podría tener una posible reacción de los gobiernos extranjeros.

- **Análisis político.** Las reglas electorales alteran las plataformas electorales de los candidatos y se pueden estudiar las consecuencias de distintos tipos de reglas. Incluso se puede evaluar la influencia en la población de un gobierno conservador o uno progresista.

- **Evolución en las especies biológicas.** Las especies que conocemos son el producto de un largo proceso de interacciones con otras especies. Los genes y la influencia de éstos sobre su comportamiento y características físicas hacen que individuos de una especie tengan distinta capacidad reproductora, con lo que los genes más exitosos en el "juego de la reproducción" son los que sobreviven.

1.1.5 Más sobre la Teoría de Juegos

Sin duda alguna, dos de las personalidades más a destacar en la Teoría de Juegos son John Von Neumann y John Nash que revolucionaron el mundo de los juegos con dos teoremas que, posiblemente, son los más importantes con respecto a los juegos: *El teorema del mínimax* y el *teorema del punto de equilibrio* también llamado *Teorema de Nash*. Sin entrar en ninguna matemática, vamos a enunciar estos teoremas con una breve reseña biográfica de cada uno y sus aportaciones a la teoría de juegos.

John Von Neumann (1903 – 1957)

Matemático húngaro nacionalizado estadounidense. Desde niño dio muestras de unas extraordinarias dotes en matemáticas. Colaboró con Hilbert en la Universidad de Gotinga contribuyendo de manera fundamental a la teoría de la demostración. Fue el autor de la primera teoría axiomática de los llamados espacios de Hilbert y sus operadores.

En 1943, el ejército estadounidense le reclamó para la fabricación de la bomba atómica. Sus convicciones anticomunistas propiciaron que interviniera también en la fabricación de la bomba de hidrógeno y en el desarrollo de misiles balísticos.
[2]

Interesado también por la robótica, propuso dos modelos de máquinas autoras reproductoras (una de ellas de manera similar a la reproducción de cristales.

En 1955 se le detectó un cáncer en estado muy avanzado apartándolo de toda actividad hasta su muerte. Destaquemos que hasta última hora estaba convencido que el cerebro humano se podía modernizar.

Su aportación a la teoría de juegos ha sido inmensa. Se le considera el "padre" de dicha teoría. Jugaba mucho al póquer y pronto se sintió atraído por el engaño, el faroleo y las segundas intenciones pensando que existía en ello algo que era "no trivial". Desde mediados de los años 20 y hasta los 40 se entretuvo en investigar la estructura matemática del póquer y otros juegos. No tardó en darse cuenta que sus teoremas podrían aplicarse a ámbitos económicos y políticos. Junto a Oskar Morgensten escribió la que pudiera ser su obra maestra *Theory of Games and Economic Behavior* en 1944. En torno al año 1928 demostró una de las ideas de Émile Borel: *el teorema del minimax*.***[2]***

[2] http://es.wikipedia.org/wiki/John_von_Neumann

Teorema Minimax: *Dado cualquier juego de suma cero, existe un número α , tal que tiene asociadas dos estrategias, a saber: Una estrategia mixta (estrategia maximin) para A que le garantiza recibir como mínimo α , y una estrategia mixta para B que le garantiza ceder a lo sumo α (estrategia minimax. [2]*

John Forbes Nash (1928 -)

Matemático estadounidense. En el colegio no destacó por su brillantez intelectual sino más bien por su torpeza en las relaciones sociales. En un principio iba a estudiar Ingeniería Química, pero su gran capacidad para las matemáticas hicieron que se decantara por esta área. [3]

A los 20 años pidió ser admitido en la Universidad de Princeton donde por aquel entonces estudiaban personalidades como Einstein o el propio Von Neumann.[3] Allí elaboró grandes postulados matemáticos no sólo en la teoría de juegos sino también en topología y geometría algebraica.

En 1955 se le diagnostica una enfermedad llamada esquizofrenia paranoica consistente en delirios, alucinaciones y conductas inhabituales. Sus delirios se centrarían en mensajes cifrados y conspiraciones. En 1959 viajó a Europa para pedir asilo político ya que pensaba que era perseguido por los comunistas. Durante los años siguientes permaneció ingresado en hospitales por periodos de cinco a ocho meses. Jamás pudo curarse de su enfermedad, lo que le costó, entre otras cosas, su matrimonio.

Al igual que Von Neumann, Nash destacó desde un principio en la teoría de juegos. A los 21 años escribió una disertación de 27 páginas en la que expuso por primera vez la solución para juegos estratégicos no-cooperativos utilizando dos teoremas emergentes en aquella época: *el teorema del punto fijo de Kakutani* y *el teorema del punto fijo de Brouwer* ya que, en realidad, el punto de equilibrio que encontró Nash no es otra cosa que un "punto fijo". Esta disertación pasaría a ser su tesis doctoral que en un principio la iba a dirigir el mismo John Von Neumann, pero éste se negó en rotundo al considerarlo una trivialidad.

Sin embargo, el término de *equilibrio Nash* no llega hasta 1950. Ese mismo año publica la *Solución de Negociaciones de Nash* y en 1951 publicó el *Programa de Nash* en el que reducía los juegos cooperativos a un marco no-cooperativo.

Gracias a sus avances en la teoría de juegos fue miembro de la RAND, una institución dedicada a la investigación estratégica que en ese momento reclutaban talentos para la aplicación de la teoría de juegos a las circunstancias mundiales.

[3] http://es.wikipedia.org/wiki/John_Forbes_Nash

Toda esta labor en la teoría de juegos le valió el reconocimiento en 1994 con la concesión del Premio Nobel de Economía por algo que el mismo Nash definió como *"mi trabajo más trivial"*.

Teorema del equilibrio de Nash: *Todo juego no-cooperativo tiene como mínimo un punto de equilibrio en estrategias combinadas denominado equilibrio Nash [3]*

Resumiendo, podemos decir que el edificio de la teoría de juegos reposa sobre estos dos grandes teoremas. Podemos incluso suponer que el *teorema de Nash* es una generalización del de Von Neumann, pero también constituye un desvío radical del mismo.

A pesar que Von Neumann era la piedra angular en los juegos de suma cero y de dos jugadores, éstos no tenían relevancia en el mundo real. Nash introdujo los términos de *juegos cooperativos y no-cooperativos*. Al ampliar la teoría para que incluyera juegos que implican cooperación y competencia, Nash consiguió abrir la puerta a las aplicaciones de la teoría de juegos a la economía, política, sociología y, en última instancia, a la biología evolutiva.

1.2 UML Lenguaje Unificado de Modelado

(**UML**, por sus siglas en inglés, *Unified Modeling Language*) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; Aún cuando todavía no es un estándar oficial, está apoyado en gran manera por el OMG (Object Management Group.[4]

Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables.

El punto importante para notar aquí es que UML es un "lenguaje" para especificar y no un método o un proceso. UML se usa para definir un sistema de software; para detallar los artefactos en el sistema; para documentar y construir -es el lenguaje en el que está descrito el modelo. UML se puede usar en una gran variedad de formas para soportar una metodología de desarrollo de software (tal como el Proceso Unificado de Rational) -pero no especifica en sí mismo qué metodología o proceso usar.

UML cuenta con varios tipos de diagramas, los cuales muestran diferentes aspectos de las entidades representadas.

1.2.1 Estandarización de UML

Además de haberse convertido en un estándar *de facto*, UML es un estándar industrial promovido por el grupo OMG al mismo nivel que el estándar CORBA para intercambio de objetos distribuidos. Para la revisión de UML se formaron dos "corrientes" que promovían la aparición de la nueva versión desde distintos puntos de vista. Finalmente se impuso la visión más industrial frente a la académica. Recientemente se ha publicado la versión 2.0 en la que aparecen muchas novedades y cambios que, fundamentalmente, se centran en resolver carencias prácticas. Además, esta versión recibe diversas mejoras que provienen del lenguaje.

Historia

Cuando la Rational Software Corporation contrató en 1994 a James Rumbaugh quien trabajaba para General Electric. Así consiguió tener a los creadores de las dos técnicas más populares para modelado de sistemas orientados a objetos:: la Técnica de Modelado a Objetos de Rumbaugh, que era mejor en análisis, y el Método Booch de Grady Booch, que era mejor en diseño. Juntos, Rumbaugh y Booch comenzaron a compatibilizar sus métodos sentando las bases del Lenguaje Unificado.

1.2.2 DIAGRAMAS UML

En UML 2.0 hay 13 tipos de diagramas. Para comprenderlos, a veces es útil categorizar jerárquicamente, como se muestra en la siguiente figura. **[5]**

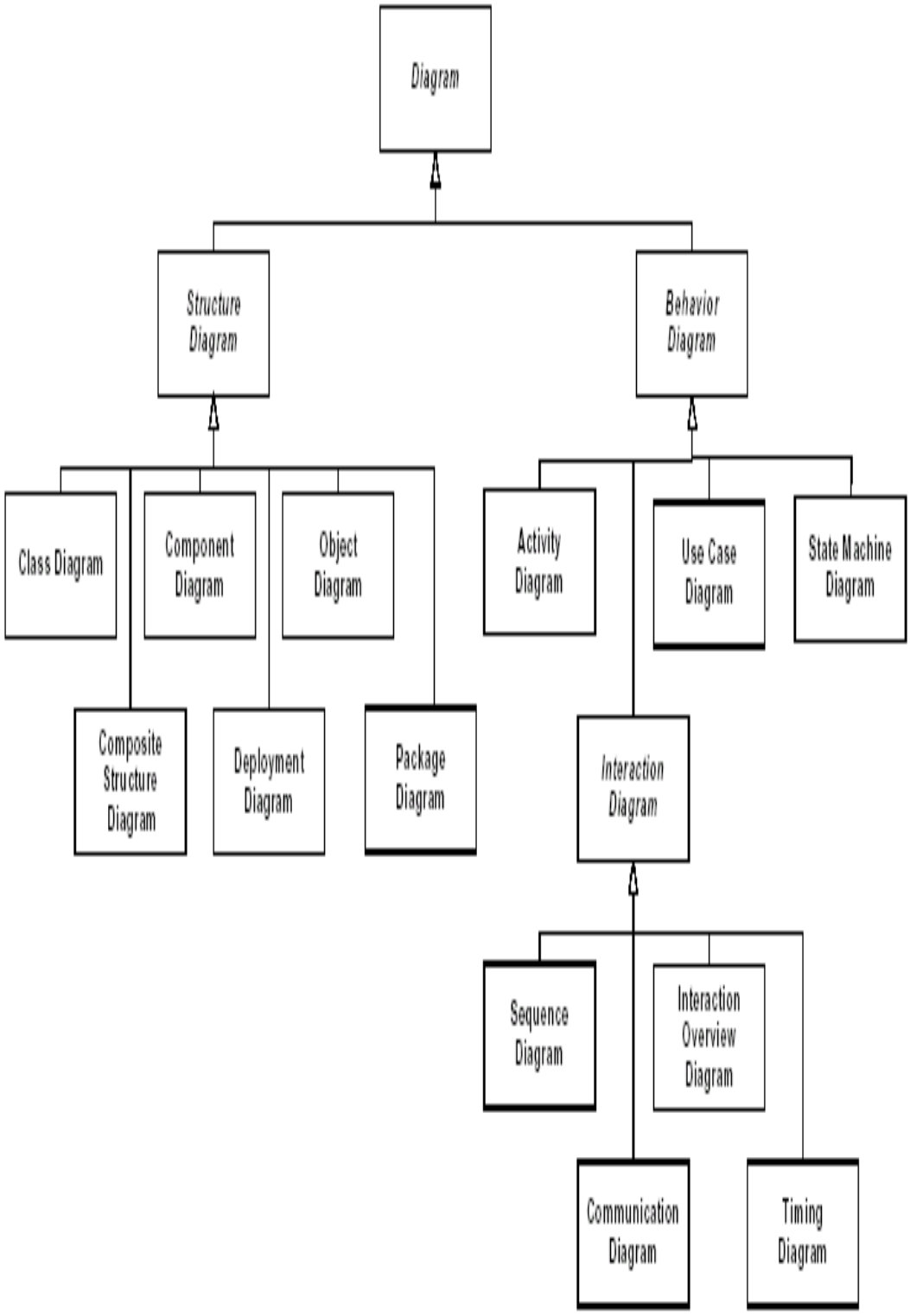


Diagram
Structure Diagram
Behavior Diagram
Class Diagram
Component Diagram
Object Diagram
Activity Diagram
Use Case Diagram
State Machine Diagram
Composite Structure Diagram
Deployment Diagram
Package Diagram
Interaction Diagram
Sequence Diagram
Interaction Overview Diagram
Communication Diagram
Timing Diagram

FIGURA 1.2

1.2.3 Jerarquías de Diagramas

- ***Diagramas de estructura*** enfatizan en los elementos que deben existir en el sistema modelado:

1.2.3.1 Diagramas de Clases

El Diagrama de Clases es el diagrama principal para el análisis y diseño. Un diagrama de clases presenta las clases del sistema con sus relaciones estructurales y de herencia. La definición de clase incluye definiciones para atributos y operaciones. El modelo de casos de uso aporta información para establecer las clases, objetos, atributos y operaciones. A continuación se presenta un ejemplo de Diagrama de Clases.

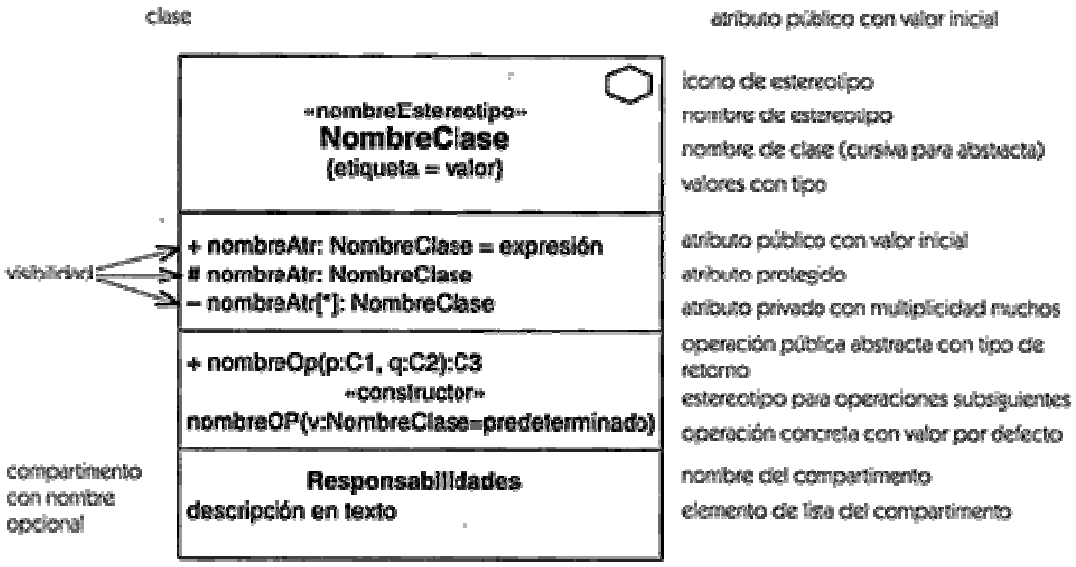


FIGURA 1.2.3.1

1.2.3.2 Diagramas de Componentes

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones. Muestran las opciones de realización incluyendo código fuente, binario y ejecutable. Los componentes representan todos los tipos de elementos software que entran en la fabricación de aplicaciones

informáticas. Pueden ser simples archivos, paquetes de Ada, bibliotecas cargadas dinámicamente, etc. Las relaciones de dependencia se utilizan en los diagramas de componentes para indicar que un componente utiliza los servicios ofrecidos por otro componente.

Un diagrama de componentes representa las dependencias entre componentes software, incluyendo componentes de código fuente, componentes del código binario, y componentes ejecutables. Un módulo de software se puede representar como componente. Algunos componentes existen en tiempo de compilación, algunos en tiempo de enlace y algunos en tiempo de ejecución, otros en varias de éstas.

Un componente de sólo compilación es aquel que es significativo únicamente en tiempo de compilación. Un componente ejecutable es un programa ejecutable.

Un diagrama de componentes tiene sólo una versión con descriptores, no tiene versión con instancias. Para mostrar las instancias de los componentes se debe usar un diagrama de despliegue.

Un diagrama de componentes muestra clasificadores de componentes, las clases definidas en ellos, y las relaciones entre ellas. Los clasificadores de componentes también se pueden anidar dentro de otros clasificadores de componentes para mostrar relaciones de definición.

Un diagrama que contiene clasificadores de componentes y de nodo se puede utilizar para mostrar las dependencias del compilador, que se representa como flechas con líneas discontinuas (dependencias) de un componente cliente a un componente proveedor del que depende. Los tipos de dependencias son específicos del lenguaje y se pueden representar como estereotipos de las dependencias.

El diagrama también puede usarse para mostrar interfaces y las dependencias de llamada entre componentes, usando flechas con líneas discontinuas desde los componentes a las interfaces de otros componentes.

El diagrama de componente hace parte de la vista física de un sistema, la cual modela la estructura de implementación de la aplicación por sí misma, su organización en componentes y su despliegue en nodos de ejecución. Esta vista proporciona la oportunidad de establecer correspondencias entre las clases y los componentes de implementación y nodos. La vista de implementación se representa con los diagramas de componentes.

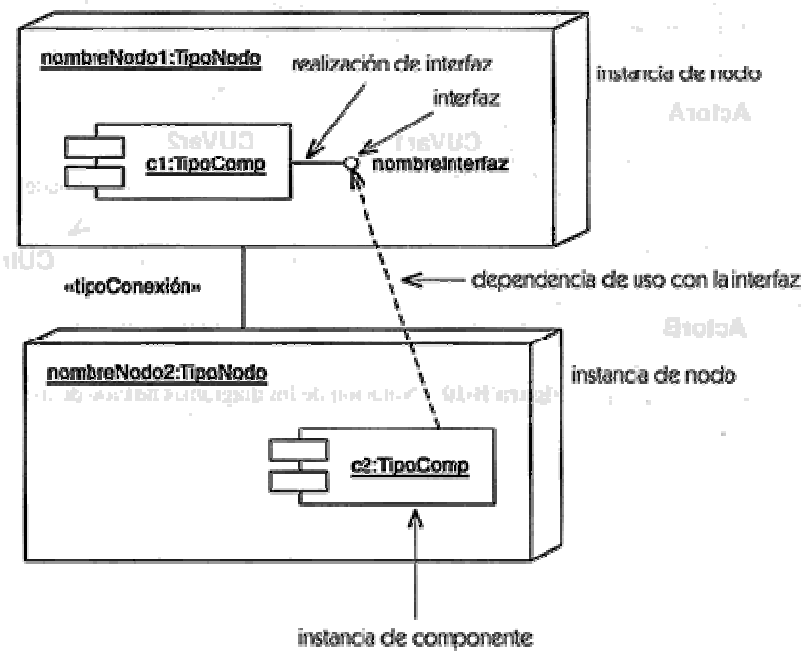


Figura 1.2.3.2

1.2.3.3 Diagramas de Objetos

Objeto es una entidad discreta con límites bien definidos y con identidad, es una unidad atómica que encapsula estado y comportamiento. La encapsulación en un objeto permite una alta cohesión y un bajo acoplamiento. El Objeto es reconocido también como una instancia de la clase a la cual pertenece.

La encapsulación presenta tres ventajas básicas:

1. Se protegen los datos de accesos indebidos
2. El acoplamiento entre las clases se disminuye
3. Favorece la modularidad y el mantenimiento

Un objeto se puede ver desde dos perspectivas relacionadas: como una entidad de un determinado instante de tiempo que posee un valor específico (Un objeto puede caracterizar una entidad física -coche-) y como un poseedor de identidad que tiene distintos valores a lo largo del tiempo (abstracta -ecuación matemática-).

Cada objeto posee su propia identidad exclusiva y se puede hacer referencia a él mediante una denominación exclusiva que permite accederle. El Modelado de Objetos permite representar el ciclo de vida de los objetos a través de sus interacciones. En UML, un objeto se representa por un rectángulo con un nombre subrayado.

- Objeto = Identidad + Estado + Comportamiento
- El estado está representado por los valores de los atributos.
- Un atributo toma un valor en un dominio concreto.

La regla general para la notación de instancias consiste en utilizar el mismo símbolo geométrico que el descriptor. En la instancia se muestran los posibles valores pero las propiedades compartidas sólo se ponen de manifiesto en el descriptor. La notación canónica es un rectángulo con tres compartimientos. En el primero va el nombre del objeto, en el segundo sus atributos y en el tercero sus operaciones. Este último puede ser omitido si así se prefiere.

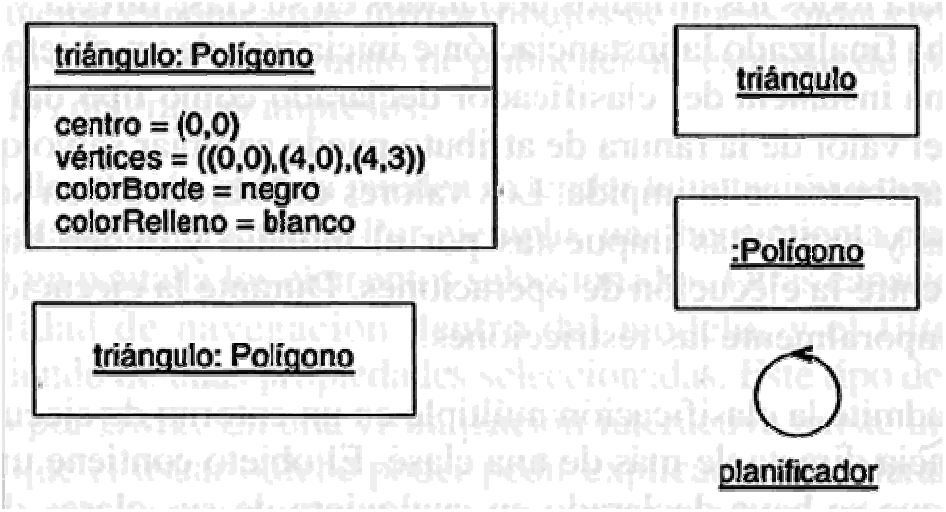
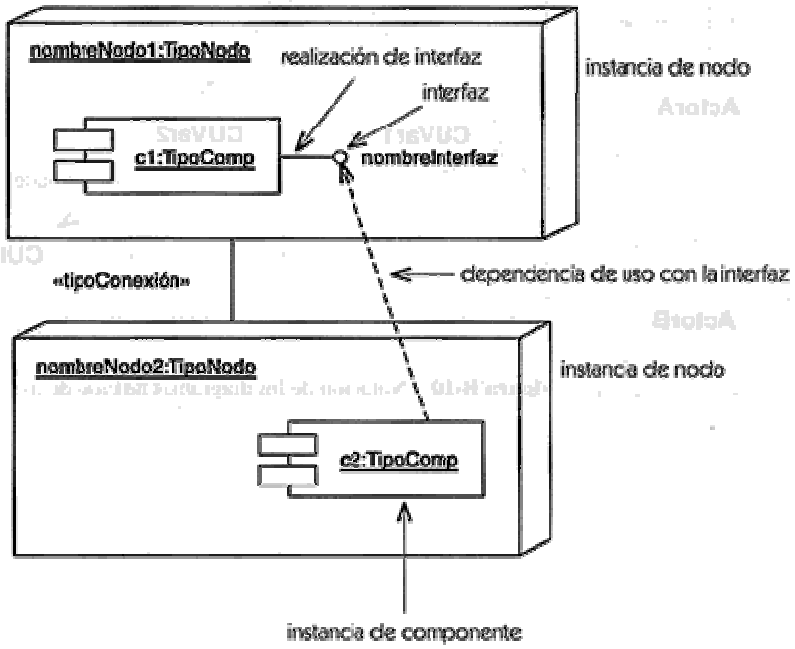


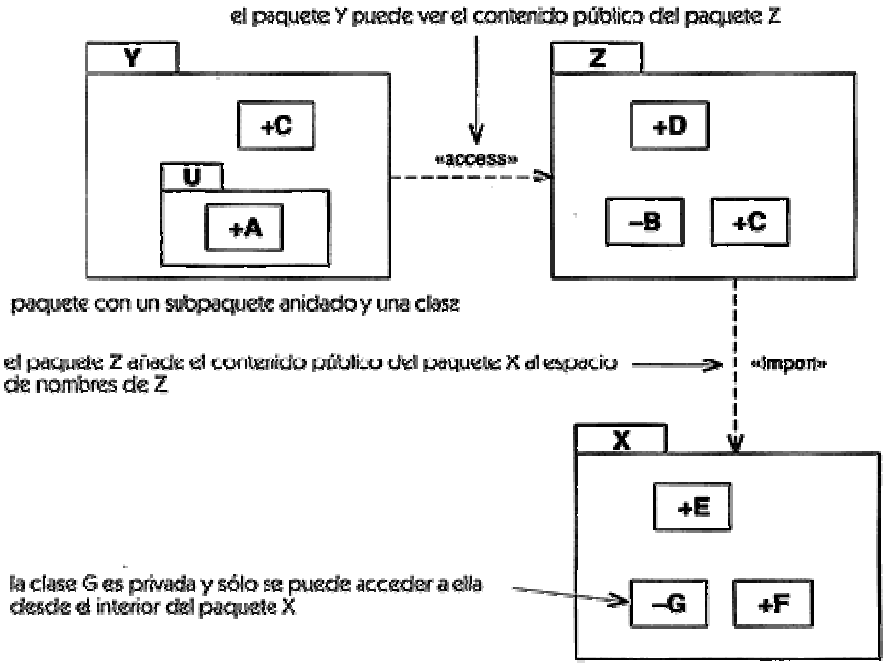
Figura 1.2.3.3

Más Diagramas de estructura

- Diagrama de estructura compuesta (UML 2.0)
- Diagrama de despliegue



- Diagrama de paquetes



1.2.3.4 Diagrama de Actividades

El Diagrama de Actividad es una especialización del Diagrama de Estado, organizado respecto de las acciones y usado para especificar:

- Un método
- Un caso de uso
- Un proceso de negocio (Workflow)

Un estado de actividad representa una actividad: un paso en el flujo de trabajo o la ejecución de una operación. Un grafo de actividades describe grupos secuenciales y concurrentes de actividades. Los grafos de actividades se muestran en diagramas de actividades. Las actividades se enlazan por transiciones automáticas. Cuando una actividad termina se desencadena el paso a la siguiente actividad.

- Un diagrama de actividades es provechoso para entender el comportamiento de alto nivel de la ejecución de un sistema, sin profundizar en los detalles internos de los mensajes. Los parámetros de entrada y salida de una acción se pueden mostrar usando las relaciones de flujo que conectan la acción y un estado de flujo de objeto.

Un grafo de actividades contiene estados de actividad que representa la ejecución de una secuencia en un procedimiento, o el funcionamiento de una actividad en un flujo de trabajo. En vez de esperar un evento, como en un estado de espera normal, un estado de actividad espera la terminación de su

cómputo. Cuando la actividad termina, entonces la ejecución procede al siguiente estado de actividad dentro del diagrama. Una transición de terminación es activada en un diagrama de actividades cuando se completa la actividad precedente. Los estados de actividad no tienen transiciones con eventos explícitos, peor pueden ser abortados por transiciones en estados que los

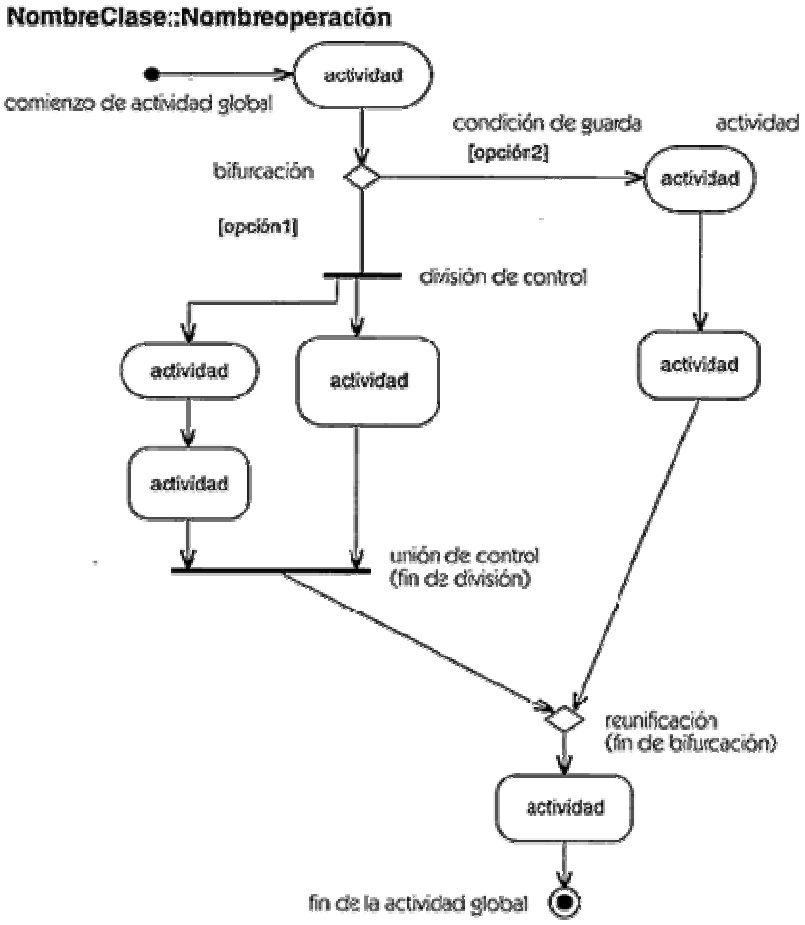


Figura 1.2.3.4

1.2.3.5 Diagramas de Caso de Uso

Casos de Uso es una técnica para capturar información de cómo un sistema o negocio trabaja, o de cómo se desea que trabaje. No pertenece estrictamente al enfoque orientado a objeto, es una técnica para captura de requisitos.

- Los Casos de Uso (Ivar Jacobson) describen bajo la forma de acciones y reacciones el comportamiento de un sistema desde el p.d.v. del usuario.
-
- Permiten definir los límites del sistema y las relaciones entre el sistema y el entorno.
-
- Los Casos de Uso son descripciones de la funcionalidad del sistema independiente de la implementación.
-
- Comparación con respecto a los Diagramas de Flujo de Datos del Enfoque Estructurado.
-
- Los Casos de Uso cubren la carencia existente en métodos previos (OMT, Booch) en cuanto a la determinación de requisitos.
-
- Los Casos de Uso peticionan el conjunto de necesidades atendiendo a la categoría de usuarios que participan en el mismo.
-
- Están basados en el lenguaje natural, es decir, es accesible por los usuarios.
-

Actores

- Principales: personas que usan el sistema.
-
- Secundarios: personas que mantienen o administran el sistema.
-
- Material externo: dispositivos materiales imprescindibles que forman parte del ámbito de la aplicación y deben ser utilizados.
-
- Otros sistemas: sistemas con los que el sistema interactúa.

La misma persona física puede interpretar varios papeles como actores distintos, el nombre del actor describe el papel desempeñado.
Los Casos de Uso se determinan observando y precisando, actor por actor, las secuencias de interacción, los escenarios, desde el punto de vista del usuario. Los casos de uso intervienen durante todo el ciclo de vida. El proceso de desarrollo estará dirigido por los casos de uso. Un escenario es una instancia de un caso de uso.

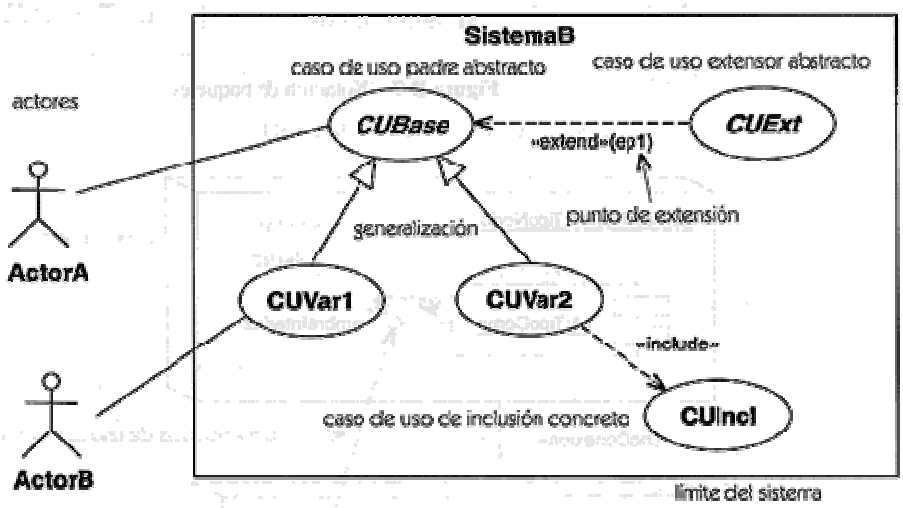


Figura 1.2.3.5

1.2.3.6 UML define cuatro tipos de relación en los Diagramas de Casos de Uso

Comunicación

- *Inclusión:* una instancia del Caso de Uso origen incluye también el comportamiento descrito por el Caso de Uso destino. «include» reemplazó al denominado «uses»
-
- *Extensión*: el Caso de Uso origen extiende el comportamiento del Caso de Uso destino. «extend»
-
- *Herencia:* el Caso de Uso origen hereda la especificación del Caso de Uso destino y posiblemente la modifica o amplía.

Parámetros para la construcción de un caso de uso:

Un caso de uso debe ser simple, inteligible, claro y conciso. Generalmente hay pocos actores asociados a cada Caso de Uso. Preguntas clave:

1. ¿Cuáles son las tareas del actor?
2. ¿Qué información crea, guarda, modifica, destruye o lee el actor?
3. Debe el actor notificar al sistema los cambios externos?
4. Debe el sistema informar al actor de los cambios internos?

La descripción del Caso de Uso comprende:

1. El inicio: ¿Cuándo y qué actor lo produce?
2. El fin: ¿Cuándo se produce y qué valor devuelve?
3. La interacción actor-caso de uso: ¿Qué mensajes intercambian ambos?
4. Objetivo del caso de uso: ¿Qué lleva a cabo o intenta?
5. cronología y origen de las interacciones
6. Repeticiones de comportamiento: ¿Qué operaciones son iteradas?

7. Situaciones opcionales: ¿Qué ejecuciones alternativas se presentan en el caso de uso?

1.2.3.7 Diagramas de Estado

Muestra el conjunto de estados por los cuales pasa un objeto durante su vida en una aplicación, junto con los cambios que permiten pasar de un estado a otro.

Los Diagramas de Estado representan autómatas de estados finitos, desde el p.d.v. de los estados y las transiciones. Son útiles sólo para los objetos con un comportamiento significativo. Cada objeto está en un estado en cierto instante. El estado está caracterizado parcialmente por los valores algunos de los atributos del objeto. El estado en el que se encuentra un objeto determina su comportamiento.

Cada objeto sigue el comportamiento descrito en el Diagrama de Estados asociado a su clase. Los Diagramas de Estados y escenarios son complementarios, los Diagramas de Estados son autómatas jerárquicos que permiten expresar concurrencia, sincronización y jerarquías de objetos, son grafos dirigidos y deterministas. La transición entre estados es instantánea y se debe a la ocurrencia de un evento.

Estado

Identifica un periodo de tiempo del objeto (no instantáneo) en el cual el objeto está esperando alguna operación, tiene cierto estado característico o puede recibir cierto tipo de estímulos. Se representa mediante un rectángulo con los bordes redondeados, que puede tener tres compartimientos: uno para el nombre, otro para el valor característico de los atributos del objeto en ese estado y otro para las acciones que se realizan al entrar, salir o estar en un estado (entry, exit o do, respectivamente.

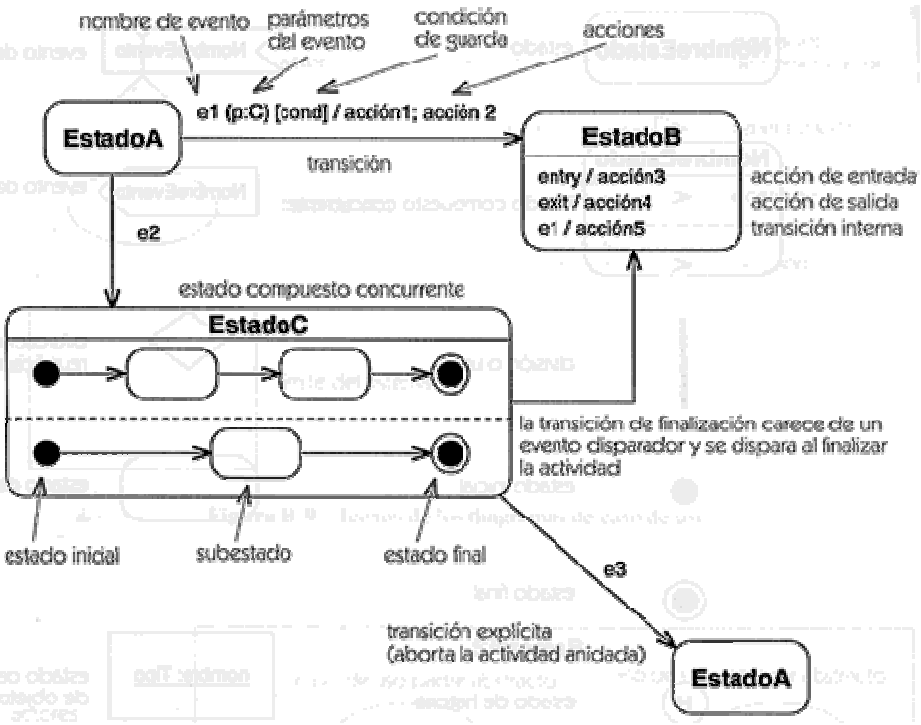
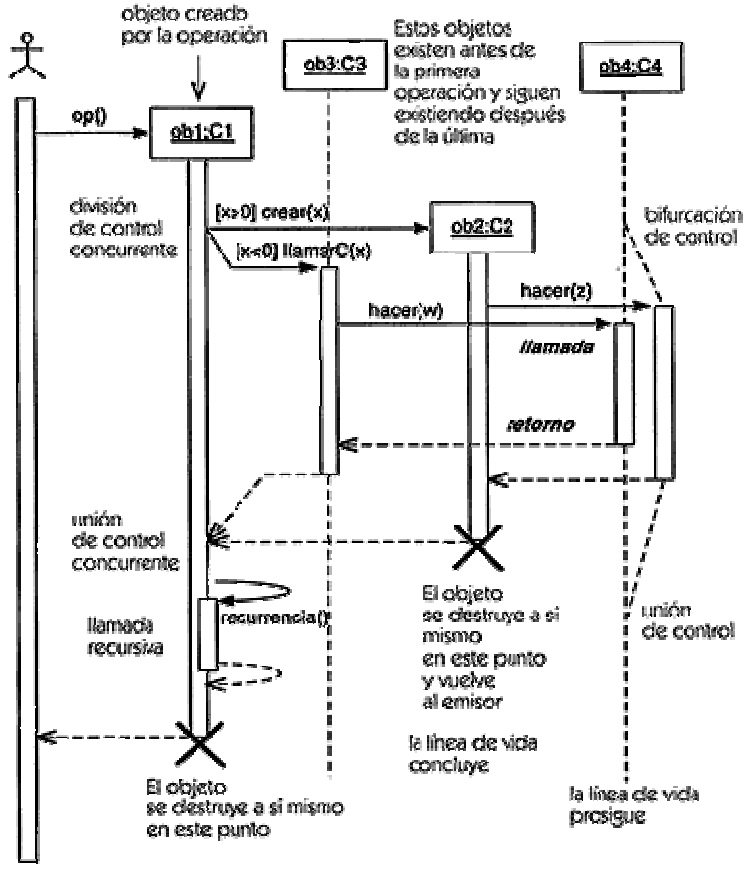


Figura 1.2.3.7

- ***Diagramas de Interacción***, un subtipo de diagramas de comportamiento, que enfatiza sobre el flujo de control y de datos entre los elementos del sistema modelado:

- Diagrama de secuencia



- Diagrama de comunicación
- Diagrama de tiempos (UML 2.0)
- Diagrama de vista de interacción (UML 2.0)[4]

[4] http://www.creangel.com/uml/

1.3 HERRAMIESTAS DE PROGRAMACION

1.3.1 Tipos de Software libre (Open Source) para modelado en UML

- ArgoUml, Herramienta de modelado UML escrito en java Puede ser usado para modelar varios tipos de diagramas UML
- Umbrello Herramienta para modelado UML
- MonoUML Herramienta CASE para la plataforma
- UMLet Herramienta para modelado rápido de UML también escrita en Java
- gModeler Herramienta para modelado de UML basada en Flash (utilizable desde el navegador), que permite generar código Action Script 2.0 Compatible
- StarUML Herramienta de modelado para Windows desarrollada en Delphi. Bastante estable y usable

Software propietario gratuito para modelado en UML

- Visual Paradigm for UML, Herramienta de modelado UML y herramienta CASE que, como en el caso anterior, cuenta con una versión gratuita denominada Community Edition
- Omondo plugin para Eclipse. Herramienta de modelado UML para Java

TCM herramienta para crear diversoso tipos de diagramas incluidos UML

JUDE Community Herramienta de modelado UML[6]

1.3.2 SQL

El Lenguaje de Consulta Estructurado (Structured Query Language) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones sobre las mismas. Aúna características del álgebra y el cálculo relacional permitiendo lanzar consultas con el fin de recuperar información de interés de una base de datos, de una forma sencilla.

1.3.3 Orígenes y evolución

Los orígenes del SQL están ligados a los de las bases de datos relaciónales. En 1970 Codd propone el modelo relacional y asociado a este un sublenguaje de acceso a los datos basado en el cálculo de predicados.

Basándose en estas ideas los laboratorios de IBM definen el lenguaje SEQUEL (Structured English QUEry Language) que más tarde sería ampliamente implementado por el SGBD experimental System R, desarrollado en 1977 también por IBM. Sin embargo, fue Oracle quien lo introdujo por primera vez en 1979 en un programa comercial.

El SEQUEL terminaría siendo el predecesor de SQL, siendo este una versión evolucionada del primero. El SQL pasa a ser el lenguaje por excelencia de los diversos SGBD relacionales surgidos en los años siguientes y es por fin estandarizado en 1986 por el ANSI, dando lugar a la primera versión estándar de este lenguaje, el SQL-86 o SQL1. Al año siguiente este estándar es también adoptado por la ISO.

Sin embargo este primer estándar no cubre todas las necesidades de los desarrolladores e incluye funcionalidades de definición de almacenamiento que se consideraron suprimir. Así que en 1992 se lanza un nuevo estándar ampliado y revisado del SQL llamado SQL-92 o SQL2.

En la actualidad el SQL es el estándar *de facto* de la inmensa mayoría de los SGBD comerciales. Y, aunque la diversidad de añadidos particulares que incluyen las distintas implementaciones comerciales del lenguaje es amplia, el soporte al estándar SQL-92 es general y muy amplio.

El ANSI SQL sufrió varias revisiones y agregados a lo largo del tiempo:

Año	Nombre	Alias	Comentarios
1986	SQL-86	SQL-87	Primera publicación hecha por ANSI. Confirmada por ISO en 1987.
1989	SQL-89		Revisión menor.
1992	SQL-92	SQL2	Revisión mayor.
1999	SQL:1999	SQL3	Se agregaron expresiones regulares, consultas recursivas (para relaciones jerárquicas), triggers y algunas características orientadas a objetos.
2003	SQL:2003		Introducen algunas características de XML, cambios en las funciones, estandarización del objeto y de las columnas auto numérico. (Ver. Eisenberg et al.: *SQL: 2003 Has Been Published*.)

1.3.4 Características generales

El SQL es un lenguaje de acceso a bases de datos que explota la flexibilidad y potencia de los sistemas relacionales permitiendo gran variedad de operaciones sobre los mismos. Es un lenguaje declarativo de alto nivel o de no-procedimiento, que gracias a su fuerte base teórica y su orientación al manejo de conjuntos de registros, y no a registros individuales, permite una alta productividad en codificación. De esta forma una sola sentencia puede equivaler a uno o más programas que utilizasen un lenguaje de bajo nivel orientado a registro.

1.3.5 Funcionalidad

El SQL proporciona una rica funcionalidad más allá de la simple consulta (o recuperación) de datos. Asume el papel de lenguaje de definición de datos (LDD), lenguaje de definición de vistas (LDV) y lenguaje de manipulación de datos (LMD. Además permite la concesión y denegación de permisos, la implementación de restricciones de integridad y controles de transacción, y la alteración de esquemas. Las primeras versiones del SQL incluían funciones propias de lenguaje de definición de almacenamiento (LDA) pero fueron suprimidas en los estándares más recientes con el fin de mantener el lenguaje sólo a nivel conceptual y externo.

1.3.6Modos de uso

El SQL permite fundamentalmente dos modos de uso:

- Un uso interactivo, destinado principalmente a los usuarios finales avanzados u ocasionales, en el que las diversas sentencias SQL se escriben y ejecutan en línea de comandos, o un entorno semejante.
- Un uso integrado, destinado al uso por parte de los programadores dentro de programas escritos en cualquier *lenguaje de programación anfitrión*. En este caso el SQL asume el papel de *su lenguaje de datos*.

En el caso de hacer un uso embebido del lenguaje podemos utilizar dos técnicas alternativas de programación. En una de ellas, en la que el lenguaje se denomina SQL estático, las sentencias utilizadas no cambian durante la ejecución del programa. En la otra, donde el lenguaje recibe el nombre de SQL dinámico, se produce una modificación total o parcial de las sentencias en el transcurso de la ejecución del programa. La utilización de SQL dinámico permite mayor flexibilidad y mayor complejidad en las sentencias, pero como contra punto obtenemos una eficiencia menor y el uso de técnicas de programación más complejas en el manejo de memoria y variables.

1.3.7 Optimización

Como ya se dijo arriba, y como suele ser común en los lenguajes de acceso a bases de datos de alto nivel, el SQL es un lenguaje declarativo. O

sea, que especifica qué es lo que se quiere y no cómo conseguirlo, por lo que una sentencia no establece explícitamente un orden de ejecución... El orden de ejecución interno de una sentencia puede afectar gravemente a la eficiencia del SGBD, por lo que se hace necesario que éste lleve a cabo una optimización antes de la ejecución de la misma. Muchas veces, el uso de índices acelera una instrucción de consulta, pero ralentiza la actualización de los datos, dependiendo del uso de la aplicación, sé priorizará el acceso indexado o una rápida actualización de la información. La optimización difiere sensiblemente en cada motor de base de datos y depende de muchos factores. Existe una ampliación de SQL conocida como FSQL (Fuzzy SQL, SQL difuso) que permite el acceso a bases de datos difusas, usando la lógica difusa. Este lenguaje ha sido implementado a nivel experimental y está evolucionando rápidamente.

1.3.8 Lenguaje de Manipulación de datos (DML)

El lenguaje de Manipulación de datos, en inglés Data Manipulation Language (DML), es el que se encarga de la modificación de los datos dentro de la base de datos. Mediante este grupo de comandos, es posible consultar y modificar todos los datos de la base de datos. Es el principal componente del SQL. Existen cuatro operaciones básicas: INSERT, UPDATE, DELETE y SELECT.

INSERT

Este comando SQL inserta registros en una tabla específica. Se pueden insertar valores específicos o valores provenientes de otra tabla o vista.

Ejemplo 1:
INSERT INTO *NOMBRE_TABLA* (ID, NOMBRE, FECHA, DESCRIPCION) VALUES (1,'Juan Jose','2006/01/01',10)

Ejemplo 2:
INSERT INTO *NOMBRE_TABLA* (ID, NOMBRE, FECHA, DESCRIPCION)
SELECT ID, NOMBRE, FECHA, DESCRIPCION
FROM *NOMBRE_TABLA_2*

Este comando SQL modifica los valores de los campos de registros ya existentes en una tabla especifica.

Ejemplo 1:
UPDATE *TABLA_NOMBRE* SET NOMBRE='Jesus Jaime', FECHA='2006/01/01' WHERE ID=2
DELETE

Este comando SQL elimina registros de una tabla especifica.

```
Ejemplo 1:
DELETE FROM TABLA_NOMBRE WHERE ID = 2
SELECT
```

Este comando SQL permite devolver información de una o más tablas de la base de datos. Es, de lejos, el comando más versátil del lenguaje SQL. Existen muchas cláusulas asociadas a la sentencia SELECT (GROUP BY, ORDER, HAVING, UNION) y muchos motores tienen incorporadas otras cláusulas no estándar (CUBE, ROLLAP, GROUPING).

```
Ejemplo 1:
SELECT * FROM TABLA_NOMBRE ORDER BY ID, FECHA, NOMBRE

Ejemplo 2:
SELECT NOMBRE, DESCRIPCION FROM TABLA_NOMBRE WHERE
FECHA >= '2006/1/01' ORDER BY ID, FECHA, NOMBRE

Ejemplo 3:
SELECT NOMBRE, COUNT(*) AS CANTIDAD FROM
```

1.3.9 Lenguaje de Definición de datos DDL

El lenguaje de Definición de datos, en inglés Data Definition Language (DDL), es el que se encarga de la modificación de la estructura de los objetos de la base de datos. Existen cuatro operaciones básicas: CREATE, ALTER, DROP y TRUNCATE.

CREATE

Este comando crea un objeto dentro de la base de datos. Puede ser una tabla, vista, índice, trigger, función, procedimiento o cualquier otro objeto que el motor de la base de datos soporte.

```
Ejemplo 1 (creación de una tabla):

CREATE TABLE TABLA_NOMBRE (
 my_field1   INT         UNSIGNED,
 my_field2   VARCHAR (50),
 my_field3   DATE        NOT NULL,
 PRIMARY KEY (my_field1, my_field2)
)
```

ALTER

Este comando permite modificar la estructura de un objeto. Se pueden agregar/quitar campos a una tabla, modificar el tipo de un campo, agregar/quitar índices a una tabla, modificar un trigger, etc.

Ejemplo 1 (agregar columna a una tabla):

```
ALTER TABLE TABLA_NOMBRE (
 ADD NUEVO_CAMPO INT UNSIGNED
)
DROP
```

Este comando elimina un objeto de la base de datos puede ser una tabla, vista, índice, trigger, función, procedimiento o cualquier otro objeto que el motor de la base de datos soporte. Se puede combinar con la sentencia ALTER.

Ejemplo 1:

```
DROP TABLE TABLA_NOMBRE
```

Ejemplo 2:

```
ALTER TABLE TABLA_NOMBRE
(
   DROP COLUMN CAMPO_NOMBRE1
)
TRUNCATE
```

Este comando trunca todo el contenido de una tabla. La ventaja sobre el comando DELETE, es que si se quiere borrar todo el contenido de la tabla, es mucho más rápido, especialmente si la tabla es muy grande, la desventaja es que TRUNCATE solo sirve cuando se quiere eliminar absolutamente todos los registros, ya que no se permite la cláusula WHERE. Si bien, en un principio, esta sentencia parecería ser DML (Lenguaje de Manipulación de Datos), es en realidad una DDL, ya que internamente, el comando trúncate borra la tabla y la vuelve a crear y no ejecuta ninguna transacción.

Ejemplo 1:

```
TRUNCATE TABLE TABLA_NOMBRE
```

1.3.10 Sistemas de gestión de base de datos

Los sistemas de gestión de base de datos con soporte SQL más utilizados son:

- DB2
- Oracle
- SQL Server
- Sybase ASE
- MySQL
- PostgreSQL
- Firebird

1.3.11 JSP

JavaServer Pages (JSP), en el campo de la Informática, es la tecnología para generar páginas web de forma dinámica en el servidor, desarrollado por Sun Microsystems, basado en scripts que utilizan una variante del lenguaje java.

La tecnología JSP, o de JavaServer Pagés, es una tecnología Java que permite a los programadores generar dinámicamente HTML, XML o algún otro tipo de página web. Esta tecnología permite al código Java y a algunas acciones predefinidas ser embebidas en el contenido estático. En las JSP, se escribe el texto que va a ser devuelto en la salida (normalmente código HTML) incluyendo código java dentro de él para poder modificar o generar contenido dinámicamente. El código java se incluye dentro de las marcas de etiqueta <% y %>.

En una posterior especificación, se incluyeron taglib; esto es, la posibilidad de definir etiquetas nuevas que ejecuten código de clases java. La asociación de las etiquetas con las clases java se declara en archivos de configuración en XML.

La principal ventaja de JSP frente a otros lenguajes es que permite integrarse con clases Java (class) lo que permite separar en niveles las aplicaciones web, almacenando en clases java las partes que consumen más recursos así como las que requieren más seguridad, y dejando la parte encargada de formatear el documento HTML en el archivo JSP.

Además Java se caracteriza por ser un lenguaje que puede ejecutarse en cualquier sistema, lo que sumado a JSP le da mucha versatilidad.

Sin embargo JSP no se puede considerar un script al 100% ya que antes de ejecutarse el servidor web compila el script y genera un servlet, por lo tanto se puede decir que aunque este proceso sea transparente para el programador

no deja de ser una aplicación compilada. La ventaja de esto es algo más de rapidez y disponer del API de Java en su totalidad.

Debido a esto la tecnología JSP, así como Java está teniendo mucho peso en el desarrollo web profesional (sobre todo en intranets.

Microsoft, la más directa competencia de Sun, ha visto en esta estrategia de Sun una amenaza lo que le ha llevado a que su plataforma. incluya su lenguaje de scripts ASP.NET que permite ser integrado con clases. (Ya estén hechas en C++, Visual Basic o C#) del mismo modo que JSP se integra con clases Java.

1.3.12 MySQL

MySQL es un sistema de gestión de bases de datos relacional, fue creada por la empresa sueca MySQL AB, la cual tiene el copyright del código fuente del servidor SQL, así como también de la marca.

MySQL es un software de código abierto, licenciado bajo la GPL del GNU, aunque MySQL AB distribuye una versión comercial, en lo único que se diferencia de la versión libre, es en el soporte técnico que se ofrece, y la posibilidad de integrar este gestor en un software propietario, ya que de otra manera, se vulneraría la licencia GPL.

El lenguaje de programación que utiliza MySQL es Structured Query Language (SQL) que fue desarrollado por IBM en 1981 y desde entonces es utilizado de forma generalizada en las bases de datos relacionales. [7]

1.3.12.1Historia de MySQL

MySQL surgió alrededor de la década del 90, Michael Windenis comenzó a usar mSQL para conectar tablas usando sus propias rutinas de bajo nivel (ISAM. Tras unas primeras pruebas, llegó a la conclusión de que mSQL no era lo bastante flexible ni rápido para lo que necesitaba, por lo que tuvo que desarrollar nuevas funciones. Esto resulto en una interfaz SQL a su base de datos, totalmente compatible a MySQL.

El origen del nombre MySQL no se sabe con certeza de donde proviene, por una lado se dice que en sus librerías han llevado el prefijo “my” durante los diez últimos años, por otra parte, la hija de uno de los desarrolladores se llama My. Así que no está claramente definido cuál de estas dos causas han dado lugar al nombre de este conocido gestor de bases de datos.

1.3.13 Características principales

Inicialmente, MySQL carecía de algunos elementos esenciales en las bases de datos relacionales, tales como integridad referencial y transacciones. A pesar de esto, atrajo a los desarrolladores de páginas web con contenido dinámico, debido a su simplicidad, de tal manera que los elementos faltantes fueron complementados por la vía de las aplicaciones que la utilizan. Poco a

poco estos elementos faltantes, están siendo incorporados tanto por desarrolladores internos, como por desarrolladores de software libre.

1.3.14 GUIA DE MYSQL

Introducción

El objetivo de este Tutorial en Línea es mostrar el uso del programa cliente MySQL para crear y usar una sencilla base de datos. mysql (algunas veces referido como "monitor mysql") es un programa interactivo que permite conectarnos a un servidor MySQL, ejecutar algunas consultas, y ver los resultados. mysql puede ser usado también en modo batch: Es decir, se pueden colocar toda una serie de consultas en un archivo, y posteriormente decirle a mysql que ejecute dichas consultas. **[8]**

Este tutorial asume que mysql está instalado en alguna máquina y que disponemos de un servidor MySQL al cual podemos conectarnos. Si este no es el caso, tenemos que contactar con nuestro administrador MySQL. (Si nosotros somos los administradores, es necesario consultar la documentación de MySQL que se refieren a la instalación y configuración del servidor MySQL.

Para ver la lista de opciones proporcionadas por mysql, lo invocamos con la opción --help:

shell> mysql --help A continuación se describe el proceso completo de creación y uso de una base de datos en MySQL. Si se está interesado sólo en el acceso y uso de una base de datos existente, se pueden omitir las secciones que describen como crear la base de datos y las tablas correspondientes.

Puesto que es imposible que se describan a detalle muchos de los tópicos cubiertos aquí, se recomienda que se consulte el manual de MySQL para obtener más información al respecto.

1.3.14.1 Conectándose y desconectándose al servidor MySQL

Para conectarse al servidor, usualmente necesitamos de un nombre de usuario (login) y de una contraseña (password), y si el servidor al que nos deseamos conectar está en una máquina diferente de la nuestra, también necesitamos indicar el nombre o la dirección IP de dicho servidor. Una vez que conocemos estos tres valores, podemos conectarnos de la siguiente manera:

shell> mysql -h NombreDelServidor

-u Nombre De Usuario

-p Cuando ejecutamos este comando, se nos pedirá que proporcionemos también la contraseña para el nombre de usuario que estamos usando.

Si la conexión al servidor MySQL se pudo establecer de manera satisfactoria, recibiremos el mensaje de bienvenida y estaremos en el prompt de mysql:

shell>mysql -h casita -u root -p

Enter password: ******

Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 5563 to server version: 3.23.41

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>Este prompt nos indica que mysql está listo para recibir comandos.

Algunas instalaciones permiten que los usuarios se conecten de manera anónima al servidor corriendo en la máquina local. Si es el caso de nuestra máquina, debemos de ser capaces de conectarnos al servidor invocando a mysql sin ninguna opción:

shell> mysql Después de que nos hemos conectado de manera satisfactoria, podemos desconectarnos en cualquier momento al escribir "quit", "exit", o presionar CONTROL+D.

1.3.14.2 Ejecutando algunas consultas

En este momento debimos de haber podido conectarnos ya al servidor MySQL, aún cuando no hemos seleccionado alguna base de datos para trabajar. Lo que haremos a continuación es escribir algunos comandos para irnos familiarizando con el funcionamiento de mysql

```
mysql> SELECT VERSION(), CURRENT_DATE;
+-----------+--------------------------+
| VERSION() | CURRENT_DATE |
+-----------+--------------------------+
| 3.23.41   | 2002-10-01   |
+-----------+-------------+---------------+
1 row in set (0.03 sec)
```

mysql> Esta comando ilustra distintas cosas acerca de mysql:

Un comando normalmente consiste de una sentencia SQL seguida por un punto una coma.

Cuando emitimos un comando, mysql lo manda al servidor para que lo ejecute, nos muestra los resultados y regresa el prompt indicando que está listo para recibir más consultas.

mysql muestra los resultados de la consulta como una tabla (filas y columnas. La primera fila contiene etiquetas para las columnas. Las filas siguientes muestran los resultados de la consulta. Normalmente las etiquetas de las columnas son los nombres de los campos de las tablas que estamos usando en alguna consulta. Si lo que estamos recuperando es el valor de una expresión (como en el ejemplo anterior) las etiquetas en las columnas son la expresión en sí.

MySQL muestra cuántas filas fueron regresadas y cuanto tiempo tardó en ejecutarse la consulta, lo cual puede darnos una idea de la eficiencia del servidor, aunque estos valores pueden ser un tanto imprecisos ya que no se muestra la hora del CPU, y porque pueden verse afectados por otros factores, tales como la carga del servidor y la velocidad de comunicación en una red. Las palabras clave pueden ser escritas usando mayúsculas y minúsculas. Las siguientes consultas son equivalentes:

```
mysql> SELECT

 VERSION(),CURRENT_DATE;

mysql> select versión(),

 current_date;

mysql> SeLeCt versión(),

 current_DATE;
```

Aquí está otra consulta que demuestra cómo se pueden escribir algunas expresiones matemáticas y trigonométricas:

```
mysql> SELECT SIN(PI()/4), (4+1)*5;
+------------+------------+
| SIN(PI()/4)  | (4+1)*5 |
+------------+------------+
| 0.707107    | 25         |
+------------+------------+
```

Aunque hasta este momento se han escrito sentencias sencillas de una sola línea, es posible escribir más de una sentencia por línea, siempre y cuando estén separadas por punto y coma:

```
mysql> SELECT VERSION(); SELECT NOW();
+-----------+
| VERSION() |
+-----------+
| 3.23.41   |
+-----------+
1 row in set (0.01 sec)
```

```
+---------------------+
| NOW()               |
+---------------------+
| 2002-10-28 14:26:04 |
+---------------------+
```
1 row in set (0.01 sec)Un comando no necesita ser escrito en una sola línea, así que los comandos que requieran de varias líneas no son un problema. mysql determinará en donde finaliza la sentencia cuando encuentre el punto y coma, no cuando encuentre el fin de línea.

Aquí está un ejemplo que muestra una consulta simple escrita en varias líneas:

```
mysql> SELECT
-> USER(),
-> CURRENT_DATE;
+---------------+--------------------------+
| USER()        | CURRENT_DATE |
+---------------+--------------------------+
| root@localhost | 2002-09-14       |
+---------------+--------------------------+
```
1 row in set (0.00 sec)

mysql>En este ejemplo debe notarse como cambia el prompt (de mysql> a ->) cuando se escribe una consulta en varias líneas. Esta es la manera en cómo mysql indica que está esperando a que finalice la consulta. Sin embargo si deseamos no terminar de escribir la consulta, podemos hacerlo al escribir \c como se muestra en el siguiente ejemplo:

```
mysql>SELECT
->USER(),
->\c
```
mysql> De nuevo, se nos regresa el comando el prompt mysql> que nos indica que mysql está listo para una nueva consulta.

En la siguiente tabla se muestran cada uno de los prompts que podemos obtener y una breve descripción de su significado para mysql:

Prompt Significado
mysql> Listo para una nueva consulta.
-> Esperando la línea siguiente de una consulta multi-línea.
'> Esperando la siguiente línea para completar una cadena que comienza con una comilla sencilla (').
"> Esperando la siguiente línea para completar una cadena que comienza con una comilla doble (").

Los comandos multi-línea comúnmente ocurren por accidente cuando tecleamos ENTER, pero olvidamos escribir el punto y coma. En este caso mysql se queda esperando para que finalicemos la consulta:

mysql>SELECTUSER()
->Si esto llega a suceder, muy probablemente mysql estará esperando por un punto y coma, de manera que si escribimos el punto y coma podrá completar la consulta y mysql podrá ejecutarla:

```
mysql> SELECT USER()
-> ;
+----------------+
| USER()         |
+----------------+
| root@localhost |
+----------------+
1 row in set (0.00 sec)
```

mysql>Los prompts '> y "> ocurren durante la escritura de cadenas. En mysql podemos escribir cadenas utilizando comillas sencillas o comillas dobles (por ejemplo, 'hola' y "hola"), y mysql nos permite escribir cadenas que ocupen multiples líneas. De manera que cuando veamos el prompt '> o ">, mysql nos indica que hemos empezado a escribir una cadena, pero no la hemos finalizado con la comilla correspondiente.

Aunque esto puede suceder si estamos escribiendo una cadena muy grande, es más frecuente que obtengamos alguno de estos prompts si inadvertidamente escribimos alguna de estas comillas.

Por ejemplo:

mysql> SELECT * FROM mi _ tabla WHERE nombre = "Lupita AND edad < 30;
">Si escribimos esta consulta SELECT y entonces presionamos ENTER para ver el resultado, no sucederá nada. En lugar de preocuparnos porque la consulta ha tomado mucho tiempo, debemos notar la pista que nos da mysql cambiando el prompt. Esto nos indica que mysql está esperando que finalicemos la cadena iniciada ("Lupita.

En este caso, ¿qué es lo que debemos hacer? La cosa más simple es cancelar la consulta. Sin embargo, no basta con escribir \c, ya que mysql interpreta esto como parte de la cadena que estamos escribiendo. En lugar de esto, debemos escribir antes la comilla correspondiente y después \c:

mysql> SELECT * FROM mi _ tabla WHERE nombre = "Lupita AND edad < 30;
"> " \c
mysql>El prompt cambiará de nuevo al ya conocido mysql>, indicándonos que mysql está listo para una nueva consulta.

Es sumamente importante conocer lo que significan los prompts '> y ">, ya que si en algún momento nos aparece alguno de ellos, todas las líneas que escribamos a continuación serán consideradas como parte de la cadena, inclusive cuando escribimos QUIT. Esto puede ser confuso, especialmente si no sabemos que es necesario escribir la comilla correspondiente para finalizar

la cadena, para que podamos escribir después algún otro comando, o terminar la consulta que deseamos ejecutar.

1.3.14.3 Creando y usando una base de datos

Ahora que conocemos como escribir y ejecutar sentencias, es tiempo de acceder a una base de datos.

Supongamos que tenemos diversas mascotas en casa (nuestro pequeño zoológico) y deseamos tener registros de los datos acerca de ellas. Podemos hacer esto al crear tablas que guarden esta información, para que posteriormente la consulta de estos datos sea bastante fácil y de manera muy práctica. Esta sección muestra cómo crear una base de datos, crear una tabla, incorporar datos en una tabla, y recuperar datos de las tablas de diversas maneras

La base de datos "zoológicos" será muy simple (deliberadamente), pero no es difícil pensar de situaciones del mundo real en la cual una base de datos similar puede ser usada.

Primeramente usaremos la sentencia SHOW para ver cuáles son las bases de datos existentes en el servidor al que estamos conectados:

```
mysql> SHOW DATABASES;
+----------+
| Database |
+----------+
| mysql    |
| test     |
+----------+
2 rows in set (0.00 sec)
```

mysql>Es probable que la lista de bases de datos que veamos sea diferente en nuestro caso, pero seguramente las bases de datos "mysql" y "test" estarán entre ellas. En particular, la base de datos "mysql" es requerida, ya que ésta tiene la información de los privilegios de los usuarios de MySQL. La base de datos "test" es creada durante la instalación de MySQL con el propósito de servir como área de trabajo para los usuarios que inician en el aprendizaje de MySQL.

Se debe anotar también que es posible que no veamos todas las bases de datos si no tiene el privilegio SHOW DATABASES. Se recomienda revisar la sección del manual de MySQL dedicada a los comandos GRANT y REVOKE.

Si la base de datos "test" existe, hay que intentar acceder a ella:

```
mysql> USE test


Data base changed
```

mysql>Observar que USE, al igual que QUIT, no requieren el uso del punto y coma, aunque si se usa éste, no hay ningún problema. El comando USE es especial también de otra manera: éste debe ser usado en una sola línea.

Podría usar la base de datos "test" (si tenemos acceso a ella) para los ejemplos que vienen a continuación, pero cualquier cosa que hagamos puede ser eliminada por cualquier otro usuario que tenga acceso a esta base de datos. Por esta razón, es recomendable que preguntemos al administrador MySQL acerca de la base de datos que podemos usar. Supongamos que deseamos tener una base de datos llamada "zoológico" (nótese que no se está acentuando la palabra) a la cual sólo nosotros tengamos acceso, para ello el administrador necesita ejecutar un comando como el siguiente:

mysql> GRANT ALL on zoologico.* TO MiNombreUsuario@MiComputadora
-> IDENTIFIED BY 'Mí contraseña'; en donde MiNombreUsuario es el nombre de usuario asignado dentro del contexto de MySQL, Mí computadora es el nombre o la dirección IP de la computadora desde la que nos conectamos al servidor MySQL, y Mi Contraseña es la contraseña que se nos ha asignado, igualmente, dentro del ambiente de MySQL exclusivamente. Ambos, nombre de usuario y contraseña no tienen nada que ver con el nombre de usuario y contraseña manejados por el sistema operativo (sí es el caso.

Si el administrador creó la base de datos al momento de asignar los permisos, podemos hacer uso de ella. De otro modo, nosotros debemos crearla:

mysql> USE zoológico
ERROR 1049: Unknown data base 'zoológico'
mysql>El mensaje anterior indica que la base de datos no ha sido creada, por lo tanto necesitamos crearla.

mysql> CREATE DATABASE zoologico;

Query OK, 1 row affected (0.00 sec)

mysql> USE zoológico

Data base changed

mysql>

Bajo el sistema operativo Unix, los nombres de las bases de datos son sensibles al uso de mayúsculas y minúsculas (no como las palabras clave de SQL), por lo tanto debemos de tener cuidado de escribir correctamente el nombre de la base de datos. Esto es cierto también para los nombres de las tablas.

Al crear una base de datos no se selecciona ésta de manera automática; debemos hacerlo de manera explícita, por ello usamos el comando USE en el ejemplo anterior.

La base de datos se crea sólo una vez, pero nosotros debemos seleccionarla cada vez que iniciamos una sesión con mysql. Por ello es recomendable que se indique la base de datos sobre la que vamos a trabajar al momento de invocar al monitor de MySQL. Por ejemplo:

shell>mysql -h casita -u blueman -p zoologico

Enter password: ******

Welcome to the MySQL monitor. Commands end with ; or \g.

Your MySQL connection id is 17 to server version: 3.23.38-nt

Type 'help;' or '\h' for help. Type '\c' to clear the buffer

mysql>Observar que "zoológico" no es la contraseña que se está proporcionando desde la línea de comandos, sino el nombre de la base de datos a la que deseamos acceder. Si deseamos proporcionar la contraseña en la línea de comandos después de la opción "-p", debemos de hacerlo sin dejar espacios (por ejemplo, -phola123, no como -p hola123. Sin embargo, escribir nuestra contraseña desde la línea de comandos no es recomendado, ya que es bastante inseguro.

1.3.14.4 Creando una tabla

Crear la base de datos es la parte más fácil, pero en este momento la base de datos está vacía, como lo indica el comando SHOW TABLES:

mysql> SHOW TABLES;

Empty set (0.00 sec)La parte un tanto complicada es decidir la estructura que debe tener nuestra base de datos: qué tablas se necesitan y qué columnas estarán en cada tabla.

En principio, necesitamos una tabla que contenga un registro para cada una de nuestras mascotas.

Ésta puede ser una tabla llamada mascotas, y debe contener por lo menos el nombre de cada uno de nuestros animalitos. Ya que el nombre en sí no es muy interesante, la tabla debe contener alguna otra información. Por ejemplo, si más de una persona en nuestra familia tiene una mascota, es probable que tengamos que guardar la información acerca de quién es el dueño de cada mascota. Así mismo, también sería interesante contar con alguna información más descriptiva tal como la especie, y el sexo de cada mascota.

¿Y qué sucede con la edad?. Esto puede ser también de interés, pero no es una buena idea almacenar este dato en la base de datos. La edad cambia conforme pasa el tiempo, lo cual significa que debemos de actualizar los registros frecuentemente. En vez de esto, es una mejor idea guardar un valor fijo, tal como la fecha de nacimiento. Entonces, cuando necesitemos la edad, la podemos calcular como la diferencia entre la fecha actual y la fecha de nacimiento. MySQL proporciona funciones para hacer operaciones entre fechas, así que no hay ningún problema.

Al almacenar la fecha de nacimiento en lugar de la edad tenemos algunas otras ventajas:

Podemos usar la base de datos para tareas tales como generar recordatorios para cada cumpleaños próximo de nuestras mascotas. Podemos calcular la edad con relación a otras fechas que la fecha actual. Por ejemplo, si almacenamos la fecha en que murió nuestra mascota en la base de datos, es fácil calcular que edad tenía nuestro animalito cuando falleció. Es probable que estemos pensando en otro tipo de información que sería igualmente útil en la tabla "mascotas", pero para nosotros será suficiente por ahora contar con información de nombre, propietario, especie, nacimiento y fallecimiento.

Usaremos la sentencia CREATE TABLE para indicar como estarán conformados los registros de nuestras mascotas.

mysql> CREATE TABLE mascotas(

-> nombre VARCHAR(20), propietario VARCHAR(20),

-> especie VARCHAR(20), sexo CHAR(1), nacimiento DATE,
-> fallecimento DATE);

Query OK, 0 rows affected (0.02 sec)

mysql>VARCHAR es una buena elección para los campos nombre, propietario, y especie, ya que los valores que almacenarán son de longitud variable. No es necesario que la longitud de estas columnas sea la misma, ni tampoco que sea de 20. Se puede especificar cualquier longitud entre 1 y 255, lo que se considere más adecuado. Si resulta que la elección de la longitud de los campos que hemos hecho no resultó adecuada, MySQL proporciona una sentencia ALTER TABLE que nos puede ayudar a solventar este problema.

El campo sexo puede ser representado en una variedad de formas, por ejemplo, "m" y "f", o tal vez "masculino" y "femenino", aunque resulta más simple la primera opción.

El uso del tipo de dato DATE para los campos nacimiento y fallecimiento debe de resultar obvio.

Ahora que hemos creado la tabla, la sentencia SHOW TABLES debe producir algo como:

```
mysql> SHOW TABLES;
+--------------------+
| Tables_in_zoologico |
+--------------------+
| mascotas           |
+--------------------+
1 row in set (0.00 sec)
```

mysql>Para verificar que la tabla fue creada como nosotros esperábamos, usaremos la sentencia DESCRIBE:

```
mysql> DESCRIBE mascotas;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| nombre      |varchar(20)  | YES  |     | NULL    |       |
| propietario |varchar(20)  | YES  |     | NULL    |       |
| especie     |varchar(20)  | YES  |     | NULL    |       |
| sexo        |char(1)      | YES  |     | NULL    |       |
| nacimiento  |date         | YES  |     | NULL    |       |
| fallecimento|date         | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
6 rows in set (0.01 sec)
```

mysql>Podemos hacer uso de la sentencia DESCRIBE en cualquier momento, por ejempo, si olvidamos los nombres ó el tipo de las columnas en la tabla.

1.3.14.5 Cargando datos en una tabla

Después de haber creado la tabla, ahora podemos incorporar algunos datos en ella, para lo cual haremos uso de las sentencias INSERT y LOAD DATA.

Supongamos que los registros de nuestras mascotas pueden ser descritos por los datos mostrados en la siguiente tabla.

Nombre	Propietario	Especie	Sexo	Nacimiento	Fallecimiento
Fluffy	Arnoldo	Gato	F	1999-02-04	
Mau	Juan	Gato	M	1998-03-17	
Buffy	Arnoldo	Perro	F	1999-05-13	
FanFan	Benito	Perro	M	2000-08-27	
Kaiser	Diana	Perro	M	1998-08-31	1997-07-29
Chispa	Omar	Ave	F	1998-09-11	
Wicho	Tomás	Ave		2000-02-	

				09	
Skim	Benito	Serpiente	M	2001-04-29	

Debemos observar que MySQL espera recibir fechas en el formato YYYY-MM-DD, que puede ser diferente a lo que nosotros estamos acostumbrados.

Ya que estamos iniciando con una tabla vacía, la manera más fácil de poblarla es crear un archivo de texto que contenga un registro por línea para cada uno de nuestros animalitos para que posteriormente carguemos el contenido del archivo en la tabla únicamente con una sentencia.

Por tanto, debemos de crear un archivo de texto "mascotas.txt" que contenga un registro por línea con valores separados por tabuladores, cuidando que el orden de las columnas sea el mismo que utilizamos en la sentencia CREATE TABLE. Para valores que no conozcamos podemos usar valores nulos (NULL). Para representar estos valores en nuestro archivo debemos usar \N.

Para cargar el contenido del archivo en la tabla mascotas, usaremos el siguiente comando:

mysql> LOAD DATA LOCAL INFILE "mascotas.txt" INTO TABLE mascotas; La sentencia LOAD DATA nos permite especificar cuál es el separador de columnas, y el separador de registros, por default el tabulador es el separador de columnas (campos), y el salto de línea es el separador de registros, que en este caso son suficientes para que la sentencia LOAD DATA lea correctamente el archivo "mascotas.txt".

Si lo que deseamos es añadir un registro a la vez, entonces debemos hacer uso de la sentencia INSERT. En la manera más simple, debemos proporcionar un valor para cada columna en el orden en el cual fueron listados en la sentencia CREATE TABLE. Supongamos que nuestra hermana Diana compra un nuevo hámster nombrado Pelusa. Podemos usar la sentencia INSERT para agregar su registro en nuestra base de datos.

mysql> INSERT INTO mascotas

-> VALUES('Pelusa','Diana','Hamster','f','2000-03-30',NULL);

Notar que los valores de cadenas y fechas deben estar encerrados entre comillas. También, con la sentencia INSERT podemos insertar el valor NULL directamente para representar un valor nulo, un valor que no conocemos. En este caso no se usa \N como en el caso de la sentencia LOAD DATA.

De este ejemplo, debemos ser capaces de ver que es un poco más la tarea que se tiene que realizar si inicialmente cargamos los registros con varias sentencias INSERT en lugar de una única sentencia LOAD DATA.

1.3.14.6 Recuperando información de una tabla

La sentencia SELECT es usada para obtener la información guardada en una tabla. La forma general de esta sentencia es:

SELECT La Información que deseamos FROM De Que Tabla WHERE Condición A Satisfacer Aquí, La Información que deseamos es la información que queremos ver. Esta puede ser una lista de columnas, o un * para indicar "todas las columnas". De Que Tabla indica el nombre de la tabla de la cual vamos a obtener los datos. La cláusula WHERE es opcional. Si está presente, la Condición Satisfacer especifica las condiciones que los registros deben satisfacer para que puedan ser mostrados.

1.3.14.7 Seleccionando todos los datos

La manera más simple de la sentencia SELECT es cuando se recuperan todos los datos de una tabla:

mysql> SELECT * FROM mascotas;

```
+--------+------------+-----------+------+------------+--------------+
| nombre | propietario | especie   | sexo | nacimiento | fallecimiento |
+--------+------------+-----------+------+------------+--------------+
| Fluffy | Arnoldo     | Gato      | f    | 1999-02-04 | NULL          |
| Mau    | Juan        | Gato      | m    | 1998-03-17 | NULL          |
| Buffy  | Arnoldo     | Perro     | f    | 1999-05-13 | NULL          |
| FanFan | Benito      | Perro     | m    | 2000-08-27 | NULL          |
| Kaiser | Diana       | Perro     | m    | 1998-08-31 | 1997-07-29    |
| Chispa | Omar        | Ave       | f    | 1998-09-11 | NULL          |
| Wicho  | Tomás       | Ave       | NULL | 2000-02-09 | NULL          |
| Skim   | Benito      | Serpiente | m    | 2001-04-29 | NULL          |
| Pelusa | Diana       | Hamster   | f    | 2000-03-30 | NULL          |
+--------+------------+-----------+------+------------+--------------+
```

9 rows in set (0.00 sec)Esta forma del SELECT es útil si deseamos ver los datos completos de la tabla, por ejemplo, para asegurarnos que están todos los registros después de la carga de un archivo.

Por ejemplo, en este caso que estamos tratando, al consultar los registros de la tabla, nos damos cuenta de que hay un error en el archivo de datos (mascotas.txt): parece que Kaiser ha nacido después de que ha fallecido!. Al revisar un poco el pedigree de Kaiser encontramos que la fecha correcta de nacimiento es el año 1989, no 1998.

Hay por lo menos un par de maneras de solucionar este problema:

Editar el archivo "mascotas.txt" para corregir el error, eliminar los datos de la tabla mascotas con la sentencia DELETE, y cargar los datos nuevamente con el comando LOAD DATA:

mysql> DELETE FROM mascotas;

mysql> LOAD DATA LOCAL INFILE "mascotas.txt" INTO TABLE mascotas;

Sin embargo, si hacemos esto, debemos ingresar los datos de Pelusa, la mascota de nuestra hermana Diana.

La segunda opción consiste en corregir sólo el registro erróneo con una sentencia UPDATE:

mysql> UPDATE mascotas SET nacimiento="1989-08-31" WHERE

nombre="Kaiser";

Como se mostró anteriormente, es muy fácil recuperar los datos de una tabla completa. Pero típicamente no deseamos hacer esto, particularmente cuando las tablas son demasiado grandes. En vez de ello, estaremos más interesados en responder preguntas particulares, en cuyo caso debemos especificar algunas restricciones para la información que deseamos ver.

1.3.14.8 Seleccionando registros particulares

Podemos seleccionar sólo registros particulares de una tabla. Por ejemplo, si deseamos verificar el cambio que hicimos a la fecha de nacimiento de Kaiser, seleccionamos sólo el registro de Kaiser de la siguiente manera:

mysql> SELECT * FROM mascotas WHERE nombre="Kaiser";

```
+--------+------------+---------+------+------------+--------------+
| nombre | propietario | especie | sexo | nacimiento | fallecimiento |
+--------+------------+---------+------+------------+--------------+
| Kaiser | Diana      | Perro   | m    | 1989-08-31 | 1997-07-29   |
+--------+------------+---------+------+------------+--------------+
```

1 row in set (0.00 sec)La salida mostrada confirma que el año ha sido corregido de 1998 a 1989.

La comparación de cadenas es normalmente no sensitiva, así que podemos especificar el nombre como "kaiser", "KAISER", etc. El resultado de la consulta será el mismo.

Podemos además especificar condiciones sobre cualquier columna, no sólo el "nombre". Por ejemplo, si deseamos conocer qué mascotas nacieron después del 2000, tendría que usar la columna "nacimiento":

mysql> SELECT * FROM mascotas WHERE nacimiento >= "2000-1-1";

```
+--------+------------+----------+------+------------+--------------+
| nombre | propietario | especie  | sexo | nacimiento | fallecimiento |
+--------+------------+----------+------+------------+--------------+
| FanFan | Benito     | Perro    | m    | 2000-08-27 | NULL         |
```

```
| Wicho  | Tomás       | Ave       | NULL | 2000-02-09 | NULL          |
| Skim   | Benito      | Serpiente | m    | 2001-04-29 | NULL          |
| Pelusa | Diana       | Hamster   | f    | 2000-03-30 | NULL          |
+--------+-------------+-----------+------+------------+---------------+
```

4 rows in set (0.00 sec)Podemos también combinar condiciones, por ejemplo, para localizar a los perros hembras:

```
mysql> SELECT * FROM mascotas WHERE especie="Perro" AND sexo="f";
```

```
+-------+-------------+---------+------+------------+---------------+
| nombre | propietario | especie | sexo | nacimiento | fallecimiento |
+--------+-------------+---------+------+------------+---------------+
| Buffy  | Arnoldo     | Perro   | f    | 1999-05-13 | NULL          |
+--------+-------------+---------+------+------------+---------------+
```

1 row in set (0.00 sec)La consulta anterior usa el operador lógico AND. Hay también un operador lógico OR:

```
mysql> SELECT * FROM mascotas WHERE especie = "Ave" OR especie =
"Gato";
```

```
+--------+-------------+---------+------+------------+---------------+
| nombre | propietario | especie | sexo | nacimiento | fallecimiento |
+--------+-------------+---------+------+------------+---------------+
| Fluffy | Arnoldo     | Gato    | f    | 1999-02-04 | NULL          |
| Mau    | Juan        | Gato    | m    | 1998-03-17 | NULL          |
| Chispa | Omar        | Ave     | f    | 1998-09-11 | NULL          |
| Wicho  | Tomás       | Ave     | NULL | 2000-02-09 | NULL          |
+--------+-------------+---------+------+------------+---------------+
```

4 rows in set (0.00 sec)El operador AND y el operador OR pueden ser intercambiados. Si hacemos esto, es buena idea usar paréntesis para indicar como deben ser agrupadas las condiciones:

```
mysql> SELECT * FROM mascotas WHERE (especie = "Gato" AND sexo =
"m")
-> OR (especie = "Perro" AND sexo = "f");
```

```
+--------+-------------+---------+------+------------+---------------+
| nombre | propietario | especie | sexo | nacimiento | fallecimiento |
+--------+-------------+---------+------+------------+---------------+
| Mau    | Juan        | Gato    | m    | 1998-03-17 | NULL          |
| Buffy  | Arnoldo     | Perro   | f    | 1999-05-13 | NULL          |
+--------+-------------+---------+------+------------+---------------+
```

2 rows in set (0.00 sec)Seleccionando columnas particulares

Si no deseamos ver los registros completos de una tabla, entonces tenemos que usar los nombres de las columnas en las que estamos interesados separándolas por coma. Por ejemplo, si deseamos conocer la fecha de nacimiento de nuestras mascotas, debe seleccionar la columna "nombre" y "nacimiento":

mysql> SELECT nombre, nacimiento FROM mascotas;

```
+--------+------------+
| nombre | nacimiento |
+--------+------------+
| Fluffy | 1999-02-04 |
| Mau    | 1998-03-17 |
| Buffy  | 1999-05-13 |
| FanFan | 2000-08-27 |
| Kaiser | 1989-08-31 |
| Chispa | 1998-09-11 |
| Wicho  | 2000-02-09 |
| Skim   | 2001-04-29 |
| Pelusa | 2000-03-30 |
+--------+------------+
```

9 rows in set (0.00 sec)Para conocer quién tiene alguna mascota, usaremos la siguiente consulta:

mysql> SELECT propietario FROM mascotas;

```
+-------------+
| Propietario |
+-------------+
| Arnoldo     |
| Juan        |
| Arnoldo     |
| Benito      |
| Diana       |
| Omar        |
| Tomás       |
| Benito      |
| Diana       |
+-------------+
```

9 rows in set (0.00 sec)Sin embargo, debemos notar que la consulta recupera el nombre del propietario de cada mascota, y algunos de ellos aparecen más de una vez. Para minimizar la salida, agregaremos la palabra clave DISTINCT:

mysql> SELECT DISTINCT propietario FROM mascotas;

```
+-------------+
| propietario |
+-------------+
| Arnoldo     |
| Juan        |
| Benito      |
| Diana       |
| Omar        |
| Tomás       |
+-------------+
```

6 rows in set (0.03 sec)Se puede usar también una claúsula WHERE para combinar selección de filas con selección de columnas. Por ejemplo, para obtener la fecha de nacimiento de los perritos y los gatitos, usaremos la siguiente consulta:

```
mysql> SELECT nombre, especie, nacimiento FROM mascotas
-> WHERE especie = "perro" OR especie = "gato";

+--------+---------+------------+
| nombre | especie | nacimiento |
+--------+---------+------------+
| Fluffy | Gato    | 1999-02-04 |
| Mau    | Gato    | 1998-03-17 |
| Buffy  | Perro   | 1999-05-13 |
| FanFan | Perro   | 2000-08-27 |
| Kaiser | Perro   | 1989-08-31 |
+--------+---------+------------+
5 rows in set (0.00 sec)
```

1.3.14.9 Ordenando registros

Se debe notar en los ejemplos anteriores que las filas regresadas son mostradas sin ningún orden en particular. Sin embargo, frecuentemente es más fácil examinar la salida de una consulta cuando las filas son ordenadas en alguna forma útil. Para ordenar los resultados, tenemos que usar una cláusula ORDER BY.

Aquí aparecen algunos datos ordenados por fecha de nacimiento:

```
mysql> SELECT nombre, nacimiento FROM mascotas ORDER BY nacimiento;

+--------+------------+
| nombre | nacimiento |
+--------+------------+
| Kaiser | 1989-08-31 |
| Mau    | 1998-03-17 |
| Chispa | 1998-09-11 |
| Fluffy | 1999-02-04 |
| Buffy  | 1999-05-13 |
| Wicho  | 2000-02-09 |
| Pelusa | 2000-03-30 |
| FanFan | 2000-08-27 |
| Skim   | 2001-04-29 |
+--------+------------+
9 rows in set (0.00 sec)
```

En las columnas de tipo caracter, el ordenamiento es ejecutado normalmente de forma no sensitiva, es decir, no hay diferencia entre mayúsculas y minúsculas. Sin embargo, se puede forzar un ordenamiento sensitivo al usar el operador BINARY.

Para ordenar en orden inverso, debemos agregar la palabra clave DESC al nombre de la columna que estamos usando en el ordenamiento:

```
mysql> SELECT nombre, nacimiento FROM mascotas ORDER BY
-> nacimiento DESC;

+--------+------------+
| nombre | nacimiento |
+--------+------------+
| Skim   | 2001-04-29 |
| FanFan | 2000-08-27 |
| Pelusa | 2000-03-30 |
| Wicho  | 2000-02-09 |
| Buffy  | 1999-05-13 |
| Fluffy | 1999-02-04 |
| Chispa | 1998-09-11 |
| Mau    | 1998-03-17 |
| Kaiser | 1989-08-31 |
+--------+------------+
```

9 rows in set (0.00 sec)Podemos ordenar múltiples columnas. Por ejemplo, para ordenar por tipo de animal, y poner al inicio los animalitos más pequeños de edad, usaremos la siguiente consulta:

```
mysql> SELECT nombre, especie, nacimiento FROM mascotas
-> ORDER BY especie, nacimiento DESC;

+--------+-----------+------------+
| nombre | especie   | nacimiento |
+--------+-----------+------------+
| Wicho  | Ave       | 2000-02-09 |
| Chispa | Ave       | 1998-09-11 |
| Fluffy | Gato      | 1999-02-04 |
| Mau    | Gato      | 1998-03-17 |
| Pelusa | Hamster   | 2000-03-30 |
| FanFan | Perro     | 2000-08-27 |
| Buffy  | Perro     | 1999-05-13 |
| Kaiser | Perro     | 1989-08-31 |
| Skim   | Serpiente | 2001-04-29 |
+--------+-----------+------------+
```

9 rows in set (0.00 sec)Notar que la palabra clave DESC aplica sólo a la columna nombrada que le precede.

1.3.14.10 Cálculos con fechas

MySQL proporciona diversas funciones que se pueden usar para efectuar cálculos sobre fechas, por ejemplo, para calcular edades o extraer partes de una fecha (día, mes, año, etc.).

Para determinar la edad de cada una de nuestras mascotas, tenemos que calcular la diferencia de años de la fecha actual y la fecha de nacimiento, y entonces substraer uno si la fecha actual ocurre antes en el calendario que la

fecha de nacimiento. Las siguientes consultas muestran la fecha actual, la fecha de nacimiento y la edad para cada mascota.

```
mysql> SELECT nombre, nacimiento, CURRENT_DATE,
-> (YEAR(CURRENT_DATE) - YEAR(nacimiento))
-> - (RIGHT(CURRENT_DATE,5) < RIGHT(nacimiento,5)) AS edad FROM mascotas;
+--------+------------+--------------+------+
| nombre | nacimiento | CURRENT_DATE | edad |
+--------+------------+--------------+------+
| Fluffy | 1999-02-04 | 2002-12-23 | 3 |
| Mau | 1998-03-17 | 2002-12-23 | 4 |
| Buffy | 1999-05-13 | 2002-12-23 | 3 |
| FanFan | 2000-08-27 | 2002-12-23 | 2 |
| Kaiser | 1989-08-31 | 2002-12-23 | 13 |
| Chispa | 1998-09-11 | 2002-12-23 | 4 |
| Wicho | 2000-02-09 | 2002-12-23 | 2 |
| Skim | 2001-04-29 | 2002-12-23 | 1 |
| Pelusa | 2000-03-30 | 2002-12-23 | 2 |
+--------+------------+--------------+------+
9 rows in set (0.01 sec)Aquí, YEAR( ) obtiene únicamente el año y RIGHT( ) obtiene los cinco caracteres más a la derecha De cada una de las fechas, que representan el mes y el día (MM-DD). La parte de la expresión que compara los valores MM-DD se evalúa a 1 o 0, y permite ajustar el valor de la edad en el caso de que el valor MM-DD de la fecha actual ocurra antes del valor MM-DD de la fecha de nacimiento.
```

Dado que la expresión en sí es bastante fea, se ha usado un alias (edad) que es el que aparece como etiqueta en la columna que muestra el resultado de la consulta.

Esta consulta debe trabajar bien, pero el resultado puede ser de alguna manera más útil si las filas son presentadas en algún orden. Para ello haremos uso de la cláusula ORDER BY.

Por ejemplo, para ordenar por nombre, usaremos la siguiente consulta:

```
mysql> SELECT nombre, nacimiento, CURRENT_DATE,
-> (YEAR(CURRENT_DATE) - YEAR(nacimiento))
-> - (RIGHT(CURRENT_DATE,5) < RIGHT(nacimiento,5))
-> AS edad FROM mascotas ORDER BY nombre;
+--------+------------+--------------+------+
| nombre | nacimiento | CURRENT_DATE | edad |
+--------+------------+--------------+------+
| Buffy | 1999-05-13 | 2002-12-23 | 3 |
| Chispa | 1998-09-11 | 2002-12-23 | 4 |
| FanFan | 2000-08-27 | 2002-12-23 | 2 |
| Fluffy | 1999-02-04 | 2002-12-23 | 3 |
| Kaiser | 1989-08-31 | 2002-12-23 | 13 |
| Mau | 1998-03-17 | 2002-12-23 | 4 |
```

Pelusa	2000-03-30	2002-12-23	2
Skim	2001-04-29	2002-12-23	1
Wicho	2000-02-09	2002-12-23	2
+--------+------------+--------------+------+
9 rows in set (0.00 sec)Para ordenar por edad en lugar de nombre, únicamente tenemos que usar una cláusula ORDER BY diferente:

```
mysql> SELECT nombre, nacimiento, CURRENT_DATE,
-> (YEAR(CURRENT_DATE) - YEAR(nacimiento))
-> - (RIGHT(CURRENT_DATE,5) < RIGHT(nacimiento,5))
-> AS edad FROM mascotas ORDER BY edad;
+--------+------------+--------------+------+
| nombre | nacimiento | CURRENT_DATE | edad |
+--------+------------+--------------+------+
| Skim | 2001-04-29 | 2002-12-23 | 1 |
| FanFan | 2000-08-27 | 2002-12-23 | 2 |
| Wicho | 2000-02-09 | 2002-12-23 | 2 |
| Pelusa | 2000-03-30 | 2002-12-23 | 2 |
| Fluffy | 1999-02-04 | 2002-12-23 | 3 |
| Buffy | 1999-05-13 | 2002-12-23 | 3 |
| Mau | 1998-03-17 | 2002-12-23 | 4 |
| Chispa | 1998-09-11 | 2002-12-23 | 4 |
| Kaiser | 1989-08-31 | 2002-12-23 | 13 |
+--------+------------+--------------+------+
9 rows in set (0.01 sec)
```

Una consulta similar puede ser usada para determinar la edad que tenía una mascota cuando falleció. Para determinar que animalitos ya fallecieron, la condición es que el valor en el campo fallecimiento no sea nulo (NULL. Entonces, para los registros con valor no-nulo, calculamos la diferencia entre los valores fallecimiento y nacimiento.

```
mysql> SELECT nombre, nacimiento, fallecimiento,
-> (YEAR(fallecimiento) - YEAR(nacimiento))
-> - (RIGHT(fallecimiento,5) < RIGHT(nacimiento,5))
-> AS edad FROM mascotas WHERE fallecimiento IS NOT NULL;
+--------+------------+--------------+------+
| nombre | nacimiento | fallecimiento | edad |
+--------+------------+--------------+------+
| Kaiser | 1989-08-31 | 1997-07-29 | 7 |
+--------+------------+--------------+------+
1 row in set (0.01 sec)
```

La consulta usa fallecimiento IS NOT NULL, en vez de falleciimiento < > NULL porque NULL es una valor especial. Esto será explicando más a detalle posteriormente.

¿Qué sucede si deseamos conocer cuáles de nuestras mascotas cumplen años el próximo mes? Para este tipo de cálculos, el año y el día son irrelevantes; simplemente tenemos que extraer el valor del mes en la columna nacimiento. Como se mencionó anteriormente, MySQL proporciona diversas

funciones para trabajar y manipular fechas, en este caso haremos uso de la función MONTH( ). Para ver cómo trabaja, vamos a ejecutar una consulta muy simple que muestra tanto el valor de una fecha como el valor que regresa la función MONTH( ).

```
mysql> SELECT nombre, nacimiento, MONTH(nacimiento) FROM mascotas;
+--------+------------+------------------+
| nombre | nacimiento | MONTH(nacimiento) |
+--------+------------+------------------+
| Fluffy | 1999-02-04 | 2 |
| Mau | 1998-03-17 | 3 |
| Buffy | 1999-05-13 | 5 |
| FanFan | 2000-08-27 | 8 |
| Kaiser | 1989-08-31 | 8 |
| Chispa | 1998-09-11 | 9 |
| Wicho | 2000-02-09 | 2 |
| Skim | 2001-04-29 | 4 |
| Pelusa | 2000-03-30 | 3 |
+--------+------------+------------------+
9 rows in set (0.00 sec)
```

Encontrar los animalitos cuyo cumpleaños es el próximo mes es muy sencillo. Suponiendo que el mes actual es Abril (valor 4), entonces tenemos que buscar los registros cuyo valor de mes sea 5 (Mayo).

```
mysql> SELECT nombre, nacimiento FROM mascotas WHERE
MONTH(nacimiento) = 5;
+--------+------------+
| nombre | nacimiento |
+--------+------------+
| Buffy | 1999-05-13 |
+--------+------------+
1 row in set (0.00 sec)
```

Aquí habrá por supuesto una complicación si el mes actual es Diciembre. No podemos simplemente agregar uno al número del mes (12) y buscar los registros cuyo mes de nacimiento sea 13 porque dicho mes no existe. En vez de esto, tenemos que buscar los animalitos que nacieron en Enero (mes 1).

Sin embargo, lo mejor es que podemos escribir una consulta que funcione no importando cuál sea el mes actual. La función DATE_ADD( ) nos permite agregar un intervalo de tiempo a una fecha dada. Si agregamos un mes al valor regresado por la función NOW( ), y entonces extraemos el valor del mes con la función MONTH( ), el resultado es que siempre obtendremos el mes siguiente.

La consulta que resuelve nuestro problema queda así:

```
mysql> SELECT nombre, nacimiento FROM mascotas
-> WHERE MONTH(nacimiento) = MONTH(DATE_ADD(NOW(), INTERVAL 1
MONTH));
```

Trabajando con valores nulos

El valor NULL puede sorprendermos mientras no hayamos trabajado con él. Conceptualmente, NULL significa un valor que hace falta, o un valor desconocido, y es tratado de una manera diferente a otros valores. Para verificar si un valor es NULL no podemos usar los operadores de comparación tales como =, > o <.

Para probar esto ejecutemos la siguiente consulta:

```
mysql> SELECT 1 = NULL, 1 <> NULL, 1 < NULL, 1 > NULL;
+----------+-----------+----------+----------+
| 1 = NULL | 1 <> NULL | 1 < NULL | 1 > NULL |
+----------+-----------+----------+----------+
| NULL | NULL | NULL | NULL |
+----------+-----------+----------+----------+
1 row in set (0.00 sec)
```

Claramente observamos que no obtenemos resultados con algún significado con estos operadores. Es por ello que tenemos que usar los operadores IS NULL e IS NOT NULL:

```
mysql> SELECT 1 IS NULL, 1 IS NOT NULL;
+-----------+---------------+
| 1 IS NULL | 1 IS NOT NULL |
+-----------+---------------+
| 0 | 1 |
+-----------+---------------+
1 row in set (0.00 sec)
```

En MySQL, 0 o NULL significan falso y cualquier otro valor significa verdadero. El valor que se considera verdadero por default es 1.

Cuando se usa un ORDER BY, los valores NULL son siempre ordenados primero, aún cuando se use la cláusula DESC.

CAPITULO II ANALISIS

2.1 Descripción de Problema

2.1.1 Sistema en línea

Un sistema en línea se refiere a una aplicación que corre en un sistema operativo multitareas, multiprocesos, multiusuario conectados a la red (WEB).

Un sistema colector de datos, es un sistema en línea que acepta y almacena datos desde varias terminales, pero no actualiza ningún archivo maestro.

Un sistema de procesamiento de transacciones es un sistema en línea que actualiza los archivos necesarios en la medida que ingresa el trabajo, tal como en un sistema de procesamiento de órdenes de compra.

2.2 Objetivos Específicos

La Teoría de Juegos es importante porque no se ha desarrollado en el estado de Puebla ni en la Facultad de Computación un sistema sobre esta materia en el área de matemáticas, estadísticas, probabilidad y computación donde los usuarios o estudiantes pueden llevar acabo aprender sobre esta materia.

2.3 Requisitos del sistema

Es un requisito necesario para el lector conocer el funcionamiento básico de la arquitectura SQL y MYSQL para la correcta comprensión de los siguientes apartados de la base de datos y comandos de SQL.

SOFTWARE
Sistema Operativo
Windows 98, ME, 2000, XP, Vista
Servidor WEB
Internet Explorer
Mozila
Servidor Local (Reactor Server)

Gestor de Base de Datos
Reactor server
SQL yong
PphMyAdmin
HARDWARE
Memoria RAM 128 MB
Procesador INTEL Pentium II
CPU 533MHZ
Disco Duro 20GB

2.4 Requerimientos Funcionales

- Sistema interactivo en WEB para que lo usuarios entren en cualquier momento.
- Que incluya conceptos Básicos sobre la Teoría de Juegos
- Modulo de registro de Usuarios

  El cual el usuario al momento de entrar al sistema podrá darse de alta

- Modulo de Juegos Interactivos.

  El cual contiene las funciones de juegos de Teoría de Juegos, Matemáticas y Cálculo.

- Modulo de Auto-Evaluación

  El cual contiene un ejercicio en el cual el usuario podrá seleccionar la respuesta que desee.

- Modulo de Asesoría en Línea.

  El cual podrá acceder en cualquier momento solo si esta registrado

- Modulo de comentarios al Sistema

  El cual será enviado directamente al correo (Email) del administrador

- Modulo Administrador el cual tendrá las opciones de

  - Dar de Baja a un Usuario
  - Modificar a un Usuario
  - Dar de alta nuevos Juegos

2.5 Requerimientos no funcionales

- Fácil de usar
- Cuenta con arquitectura cliente-servidor
- Rendimiento Bueno

2.6 Diagrama Caso de Uso

Como Objetivo general, se propone diseñar un modelo para la simulación interactivo y eficiente de ejercicios de teoría de juegos que comprenda información integrada sobre el mismo, permitiendo la ejecución de operaciones de un ambiente computacional.

2.6.1CASO DE USO USUARIO

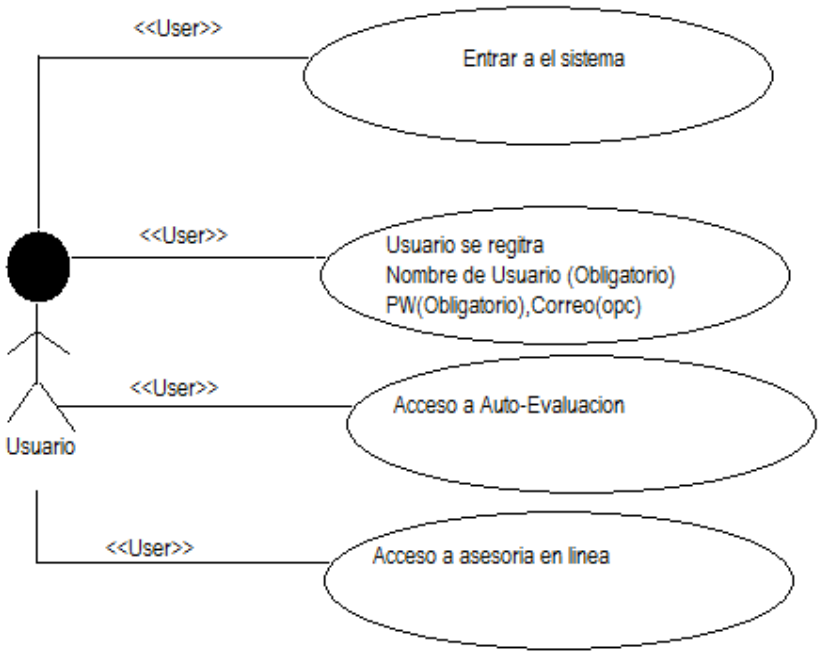


Figura 2.6.1

2.6.1.1Caso de uso: Usuario entrar al Sistema

Caso de uso	Entrar al sistema
Actores	Usuario
Tipo	Primario
Descripción	El usuario tendrá que acceder a la pagina desde el intro de la pagina y pasando por un cargador para entrar a la página principal

2.6.1.2 Caso de uso: Usuario se registra

Caso de Uso	Generar Accesos sistema al Usuario
Actores	Usuario
Tipo	Primario y esencial
Descripción	El usuario al momento de entrar al sistema tendrá la opción de registrarse con un nombre de usuario y contraseña que serán obligatorios y su correo que será opcional

2.6.1.3 Caso de uso: Usuario auto-evaluación

Caso de uso	Auto-Evaluación de Teoría de Juegos

Actores	Usuario – Administrador
Tipo	Esencial
Descripción	El Usuario tendrá la facilidad de entrar a él modulo en donde se encuentren ejercicios de Teoría de juegos e interactuar en línea con él

2.6.1.4 Caso de Uso: Asesoría en Línea

Caso de Uso	Asesoría en Línea
Actores	Usuario-Administrador
Tipo	Esencial
Descripción	El usuario tendrá la facilidad de platicar con otras personas o con un especialista del tema de Teoría de Juegos en línea en tiempo real

2.7 DIAGRAMAS ADMINISTRADOR

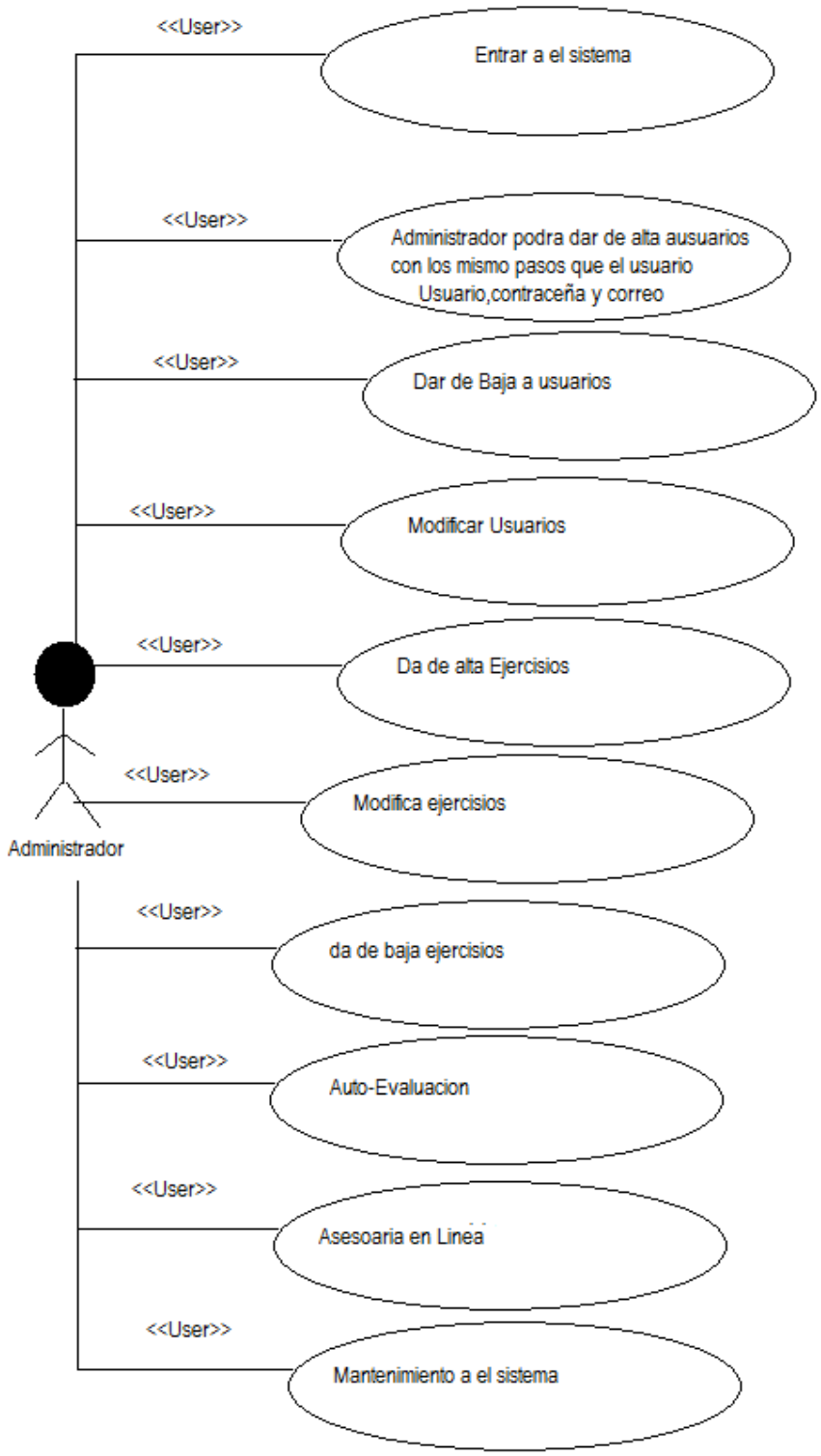


Figura 2.7.1

2.7.1 Caso de Uso Administrador Entrar al Sistema

Caso de uso	Entrar al sistema
Actores	Administrador
Tipo	Primario
Descripción	El Administrador tendrá que acceder a la pagina desde el intro de la pagina y pasando por un cargador para entrar a la página principal y después entrar a link oculto para entrar a las funciones del administrador

2.7.2 Caso de Uso Administrador Generar Usuarios

Caso de uso	Generar Usuarios
Actores	Administrador
Tipo	Primario
Descripción	El Administrador tendrá la función de generar cuantas de usuarios que no pudieron acceder.

2.7.3 Caso de Uso Administrador Dar de Baja Usuarios

Caso de uso	Dar de Baja a Usuarios
Actores	Administrador
Tipo	Primario
Descripción	El Administrador tendrá la cualidad de dar de baja un usuario solo con su clave que se le asigno al momento de registrarse en los sistemas no es necesario tener su nombre de usuario o su Pasword.

2.7.4 Caso de Uso Administrador Modificar Un usuario

Caso de uso	Modificar un Usuario
Actores	Administrador
Tipo	Primario
Descripción	El Administrador tendrá la cualidad de modificar a un usuario sus datos que son Nombre de usuario y Pasword desde el sistema o dentro de la Base Datos.

2.7.5 Caso de Uso Administrador Dar de alta Juegos

Caso de uso	Dar de alta juegos
Actores	Administrador
Tipo	Primario y Esencial
Descripción	El Administrador tendrá la cualidad de dar de alta los juegos con sus respectivas respuestas y su grado de dificultad.

2.7.6 Caso de Uso Administrador Modificar un Juego

Caso de uso	Modificar un Juego
Actores	Administrador
Tipo	Primario y Esencial
Descripción	El Administrador tendrá la cualidad de modificar un juego en cualquier momento con sus respuestas y grado de dificultar que el requiera.

2.7.7 Caso de Uso Administrador Dar de baja un Juego

Caso de uso	Dar de Baja un juego
Actores	Administrador
Tipo	Primario y Esencial
Descripción	El Administrador tendrá la cualidad de dar de baja un juego cuando considere que ya no es útil para el sistema.

2.7.8 Caso de uso: Administrador Auto-Evaluación

Caso de uso	Auto-Evaluación de Teoría de Juegos
Actores	Usuario – Administrador
Tipo	Esencial
Descripción	El Administrador tendrá la facilidad de entrar a él modulo en donde se encuentren ejercicios de Teoría de juegos e interactuar en línea con él en cualquier momento

2.7.9 Caso de Uso: Asesoría en Línea

Caso de Uso	Asesoría en Línea
Actores	Usuario-Administrador
Tipo	Esencial
Descripción	El usuario tendrá la facilidad de platicar con otras personas o con un especialista del tema de Teoría de Juegos en línea en tiempo real

CAPITULO III DISEÑO

3.1 Descripción del Sistema

Uno de los intereses de este sistema es introducir a los usuarios en el Tutorial en Línea, es una buena base para que las personas que no conocen sobre este tema al momento de entrar a la WEB y necesiten información sobre la Teoría de Juegos no-solo encuentren información en la rama de Economía sino también en el área de Computación y puedan aprender más de la Teoría de Juegos.

Se plantea que este Tutorial en Línea de Teoría de Juegos tenga un administrador que gestione todas las acciones y las solicitudes de los usuarios como implementar nuevos juegos, modificar la imagen o el acceso al sistema.

El sistema se encuentra en una aplicación WEB en la cual se requiere código que soporte la WEB que en este caso fue lenguaje de programación PHP, servidor WEB Apache, Gestor de Base de Datos MySQL, código Java Script y Flash.

3.2 Arquitectura del Sistema

La arquitectura que se utilizará en el Tutorial en Línea de Teoría de Juegos es una aplicación WEB (Internet) la cual el usuario tendrá que ejecutar en su navegador Web él nombré de la pagina de inicio del sistema el cual consta de una Interfaz Gráfica de Usuario que esta será la que se encargue de facilitar al usuario y mostrarle todas las pantallas con las que el usuario se relacionara mientras este en contacto con el Sistema

3.3Diagrama Modelo de Entidad Relación

El Diagrama de la figura 3.3 consta de entidades, atributos, como podemos ver en los atributos son las claves primaras y las claves foráneas.

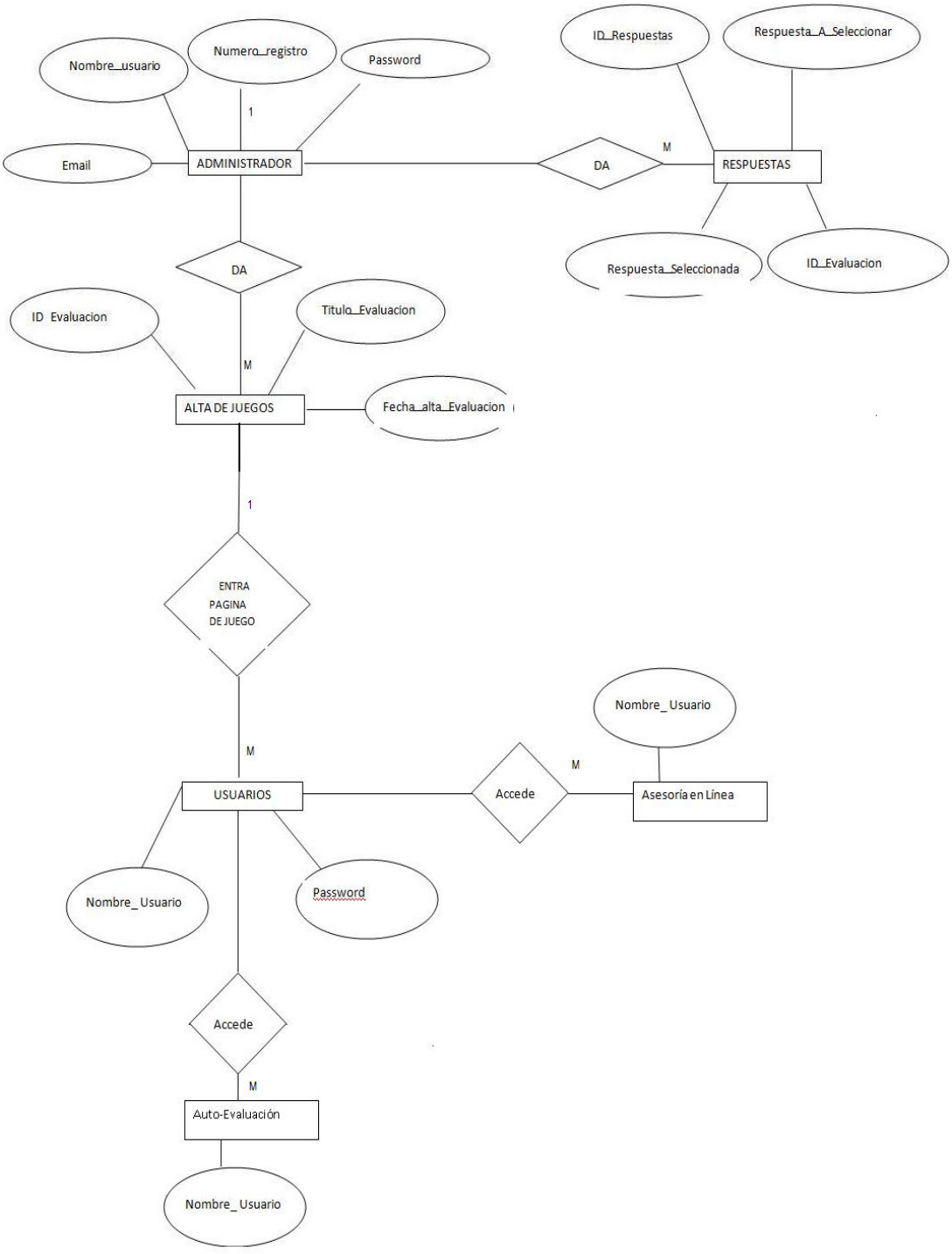


Figura 3.3

3.4 Esquema Relacional

La Figura 3.4 de nuestro esquema relacional esta formado por PK que son llaves primarias y FK que son llaves foráneas.

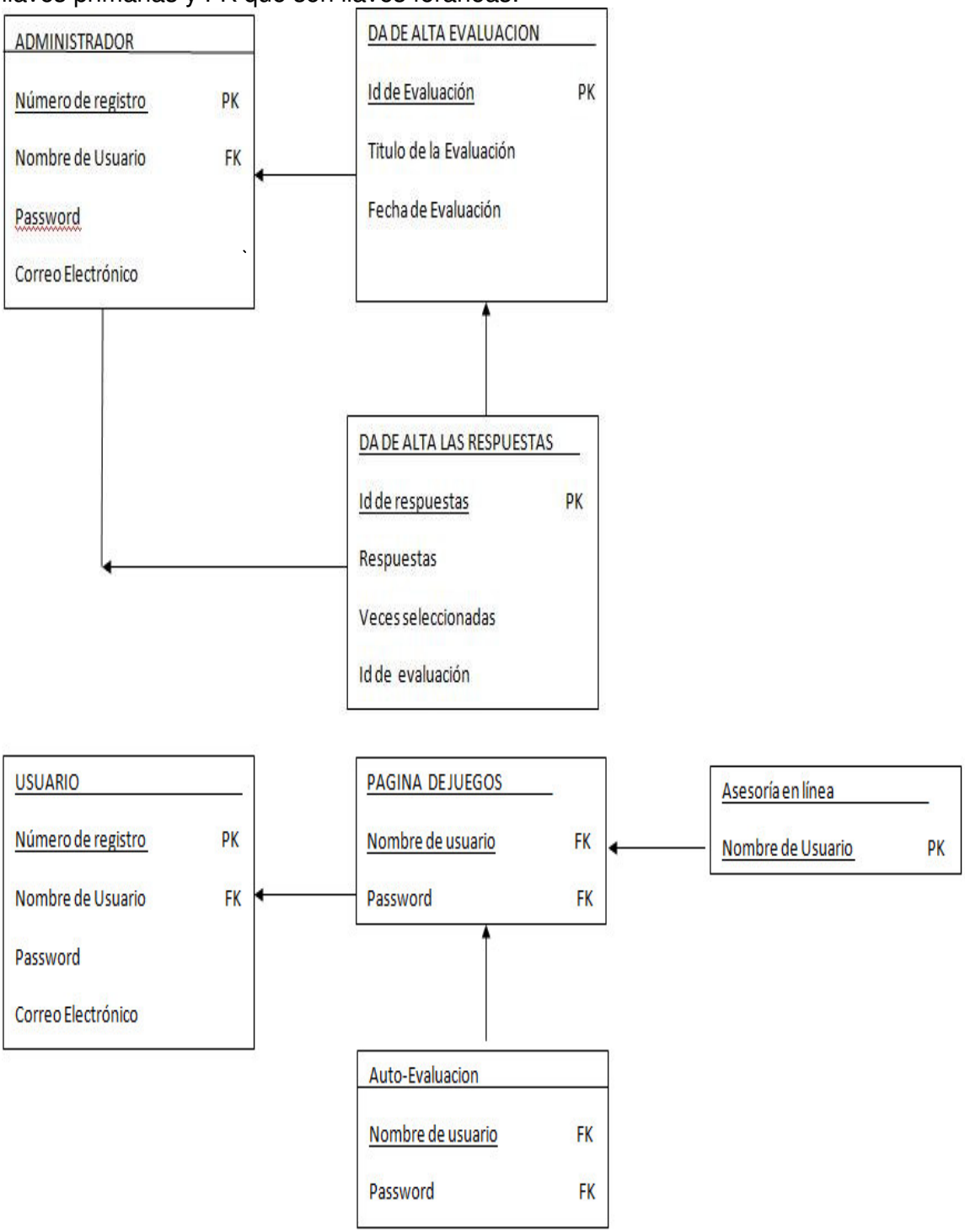


Figura 3.4

3.5 Diccionario de Datos

En este punto se muestran los diccionarios de datos que se ocuparon en la base de datos, en donde se puede observar su campo, tipo

Tabla de usuarios

Proporcionara:

Nombre de Usuario: Servirá para entrar al sistema

Password: Servirá para entrar al sistema

Correo Electrónico

Numero de Ficha será su ficha de registro en el sistema

Campo	Tipo	Nulo	Predeterminado	Comentarios
nic	varchar(16)	Sí	*NULL*	
pw	varchar(12)	Sí	*NULL*	
dirección	varchar(80)	Sí	*NULL*	
numficha	int(4)	Sí	*NULL*	

Tabla 3.5.1

Tabla de Auto-Evaluación

Tendrá los campos de:
ID: Sera el valor de auto incremento de las preguntas que se publicaran en el sistema para tener un control de cuantas preguntas se han realizado

Campo	Tipo	Nulo	Predeterminado	Comentarios
id	int(11)	Sí	*NULL*	
titulo	varchar(50)	Sí		
fecha	int(11)	Sí		

Tabla3.5.2

Tabla de respuestas

En la cual generara una grafica de los avances a las preguntas en el auto evaluación
Contara con un ID
Titulo de la pregunta
La respuesta que eligió

Campo	Tipo	Nulo	Predeterminado	Comentarios
id	int(11)	Sí	*NULL*	
texto	varchar(50)	Sí		
respuestas	int(5)	Sí		
idenc	int(11)	Sí		

Tabla 3.5.3

CAPITULO IV. IMPLEMENTACION

Para la realización de este sistema se presenta a continuación las herramientas que se utilizaron.

- Servidor WEB
- Servido de Base de datos (Mysql)
- Lenguaje de programación (PHP)
- Lenguaje de programación (Java Script)
- Navegador WEB (Internet Explorer)
- Navegador WEB (Mozilla)
- Lenguaje UML
- Flash
- DreamWeaver

Los pasos que se realizaron para crear la base de datos de los usuarios, de las respuestas y las preguntas que se ven en el sistema fueron.

Instalar el Gestor de la Base de Datos de MySQL, se realizo la conexión con PHP donde el servidor apache se encuentra presente

4.1Servidor WEB

El servidor web de Reactor server se utilizó para ejecutar el sistema de Tutorial en Línea de Teoría de Juegos este se manejo en un servidor local llamado Reactor Server, se utilizó ya que es fácil de instalar, portable y fácil de entender.

Figura 4.1.1

4.2 Servidor de Base de Datos

Se utilizo ya que es portable, fácil de utilizar, su instalación es rápida y al momento de generar las tablas y campos ayuda a los usuarios con la guía necesaria, soporta tanto tamaños grandes como pequeños de información y la conectividad con el código PHP es fácil y se pueden emigrar los reportes tanto a Excel, Word y emigrar de un gestor de Base de datos a otro mediante las instrucciones de SQL que el archivó se reduce a instrucciones.

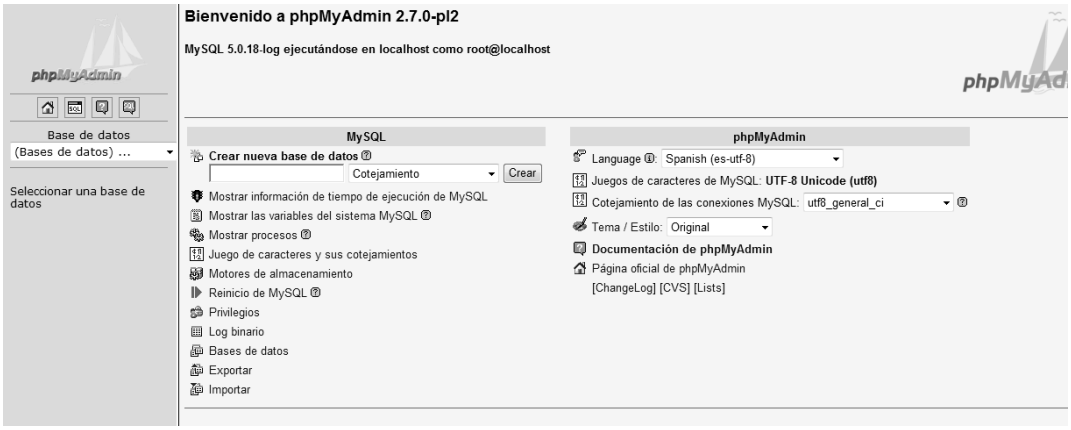


Figura 4.2.1

4.3PHP

Se utilizo este lenguaje de Programación ya que es fácil de entender si distribución de código es libre, permite el acceso basándose en datos y otras funciones que se ocupan en la WEB, es capaz de acceder a aplicaciones que se encuentran en la red. [ ]

```
<?php
function Conectar()
{
   if
(!($link=mysql_connect("localhost","root","password"))
)
   {
      echo "Error conectando a la base de datos.";
      exit();
   }
   if (!mysql_select_db("usuarios",$link))
   {
      echo "Error seleccionando la base de datos.";
      exit();
   }
```

Figura 4.3.1

4.4 Java Script

Se utilizo Java Script para la implantación de Ventanas emergentes de saludo y de datos confirmados así como de errores que el usuario crea, es muy fácil de entender y de ejecutar estas instrucciones.

Este código de Java Script se inserta en el código HTML un código de Java Script es solamente un secuencia de instrucciones que se escriben dentro del HEAD del código HTML y su instrucción para que se lea como código de Java Script es <script lenguaje="javascript">

Ejemplo Código de Java Script para validar los campos de contraseña al momento que el usuario da sus campos y nos son con los mismos de la Base de Datos, esta ventana saldrá al momento de comparar con código PHP la alerta se activara.

```
<html>
<head>
<script language="javascript">
 alert("           El usuario y/o Contraseña no son
validos\r\no no se encuentran en la base de datos");
window.open("principal.html", target="_parent")
</script>
</head>
```

Figura 4.4.1

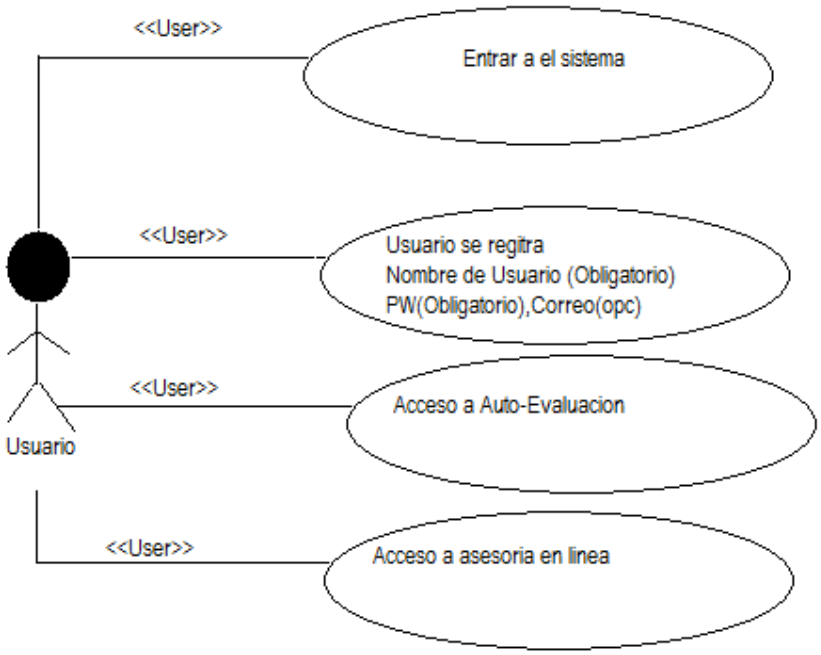
Ejemplo en el cual una ventana aparecerá un enunciado y este a su vez se acercara cada vez mas y se agrandara la letra

```
<script>
i=0;
tamano=0;
hauteur = (navigator.appName != "Microsoft Internet Explorer")?
window.innerHeight : document.body.offsetHeight;
marge = hauteur/3;
document.getElementsByTagName('div')[0].style.marginTop =
marge;
textanim = new Array("TEORIA DE JUEGOS EN LINEA");
function animation() {
document.getElementsByTagName('div')[0].style.fontSize =
""+tamano+"px";
document.getElementById('bienvenidos').innerHTML = textanim[i];
if (tamano < 50) {
tamano++;
}
else {
```

Figura 4.2.2

4.5 UML

Se utilizo para el diseño de la diagramación siguiendo sus bases en este caso fue en los diagramas de Caso de uso del sistema



4.6Macromedia Flash MX

Esta aplicación se ocupó para las animación, mejor presentación del sistema como los botones, banner, la asesoría en línea.

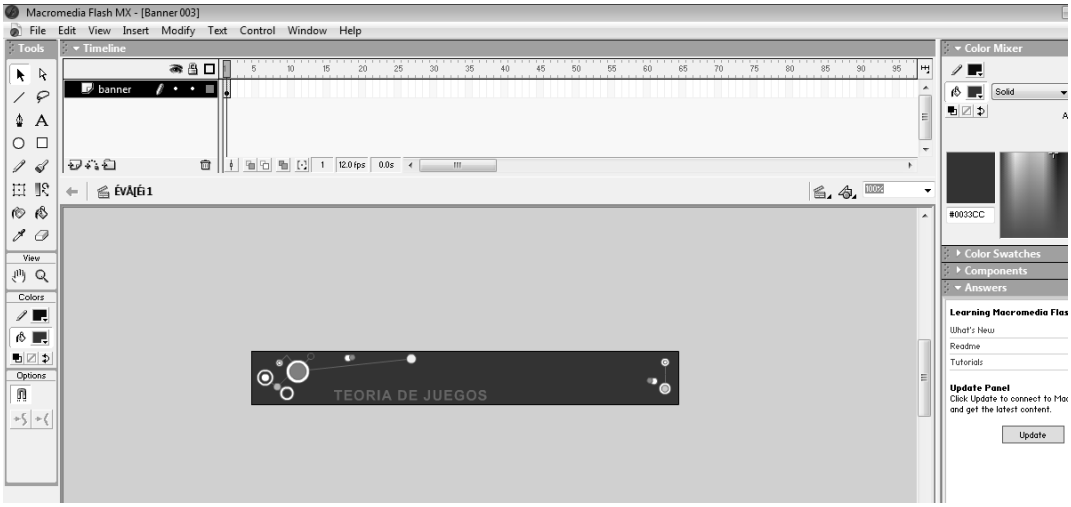


Figura 4.4.1

Figura 4.4.1

4.7 Macromedia Dreamweaver MX

Este Software fue de gran utilidad en el Sistema de Tutorial en Línea de Teoría de Juegos ya que en él se diseño toda la interfaz grafica, se programo el código PHP, Java Script y se incrusto las aplicaciones de Flash.

Se ocupo ya que es portable y tiene las facilidades de ver el código, el diseño y al mismo tiempo se puede observar las dos formas y ver lo que se está diseñando.

Es fácil de instalar y de manejar ya que puede estar tanto en el lenguaje Ingles como en español.

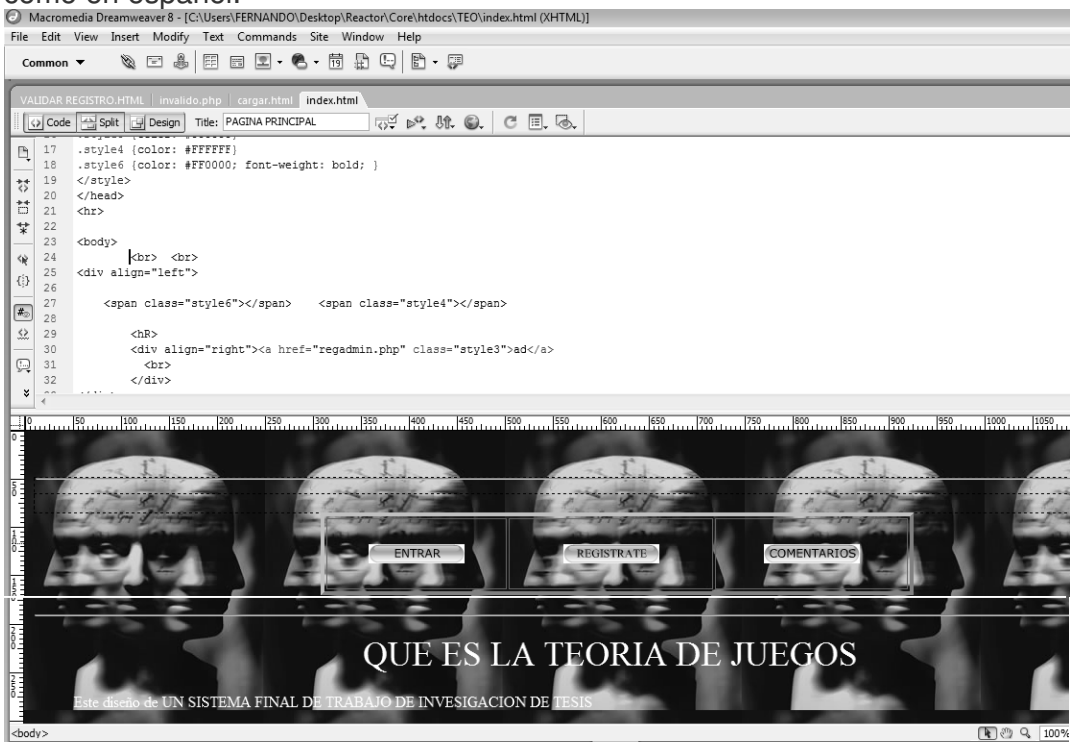
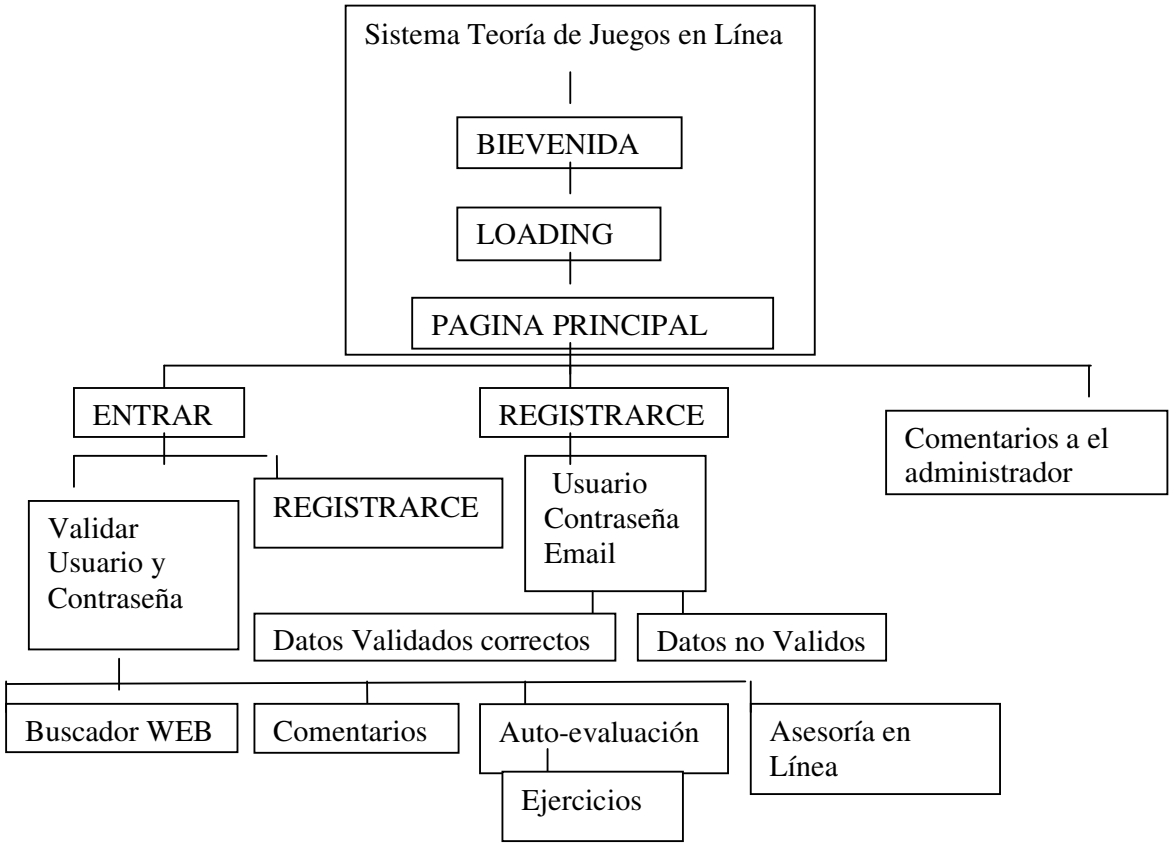
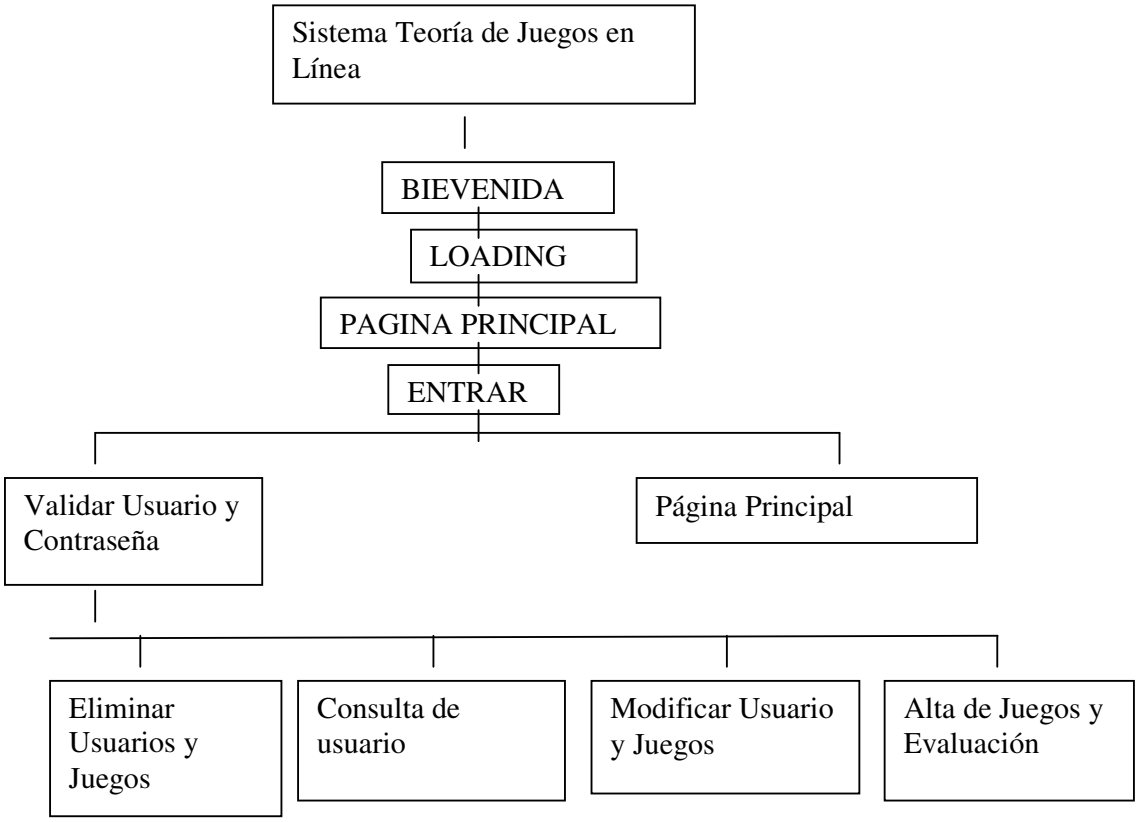


Figura 4.5.1

4.8 Diagrama de Navegación modo Usuario



4.9 Diagrama de Navegación modo Administrador



CAPITULO V. Pruebas

Para concluir con la realización del Sistema de Tutorial en Línea de Teoría de Juegos, se realizaron las siguientes pruebas y se visualizara con la interfaz grafica que se le presentará al usuario basándose en lo realizado en el análisis y diseño del sistema que se mencionaron en los capítulos anteriores

5.1 Pantalla de Entrada a él Tutorial

Primera pantalla la cual contiene un código de JAVA SCRIP el cual se refleja en la las línea de Teoría de Juegos en Línea que al momento de cargar la pagina realizara un Zoom y se estará acercando las letras.

Contiene una liga a la página de introducción

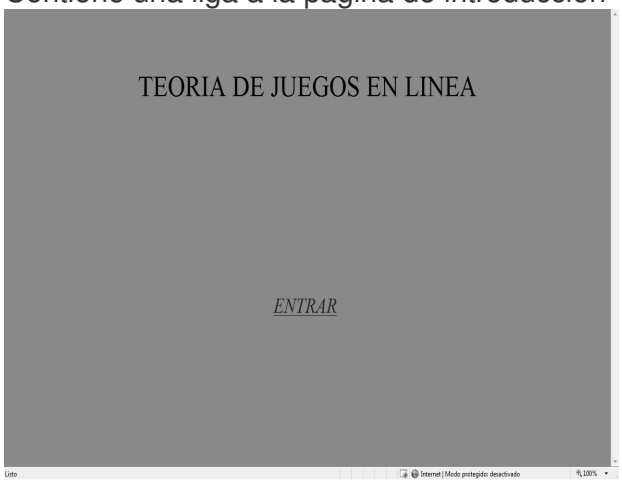


FIGURA 5.1.1

5.2Pantalla de inicio

- Pantalla de inicio del sistema realizada en código HTML(DREAMWEAVER) Y FLASH
- Botón de entrar acceso al sistema realizado con FLASH y vinculado a la página principal del sistema

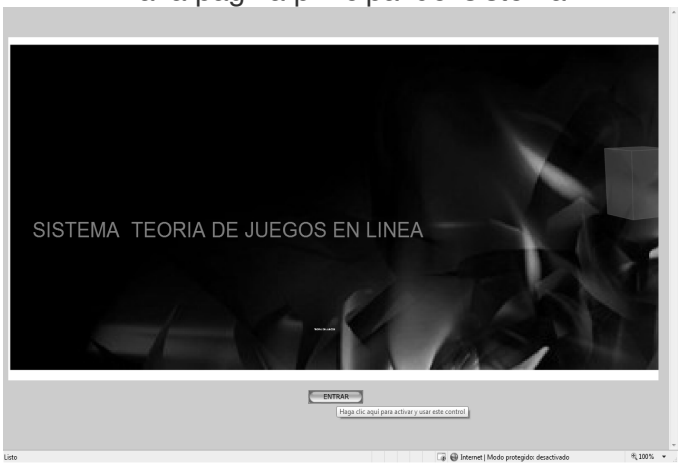


Figura 5.2.1

5.3 PAGINA PRINCIPAL

Pagina inicial donde se encontraran 3 botones para registrarse en el sistema, entrar al sistema y enviar comentario el encargado del sistema.

Realizada con código HTML(DREAMWEABER) versión MX y FLASH versión MX y dividida por conjunto de marcos

Su estructura es la siguiente:

- Encabezado en FLASH que muestra la hora y fecha del equipo de computo de donde esta accediendo

- Botones en flash de
  Entrar al sistema
  Registrarse
  Comentarios

Donde se encuentra él circulo será la opción para entrar como modo ADMINISTRADOR

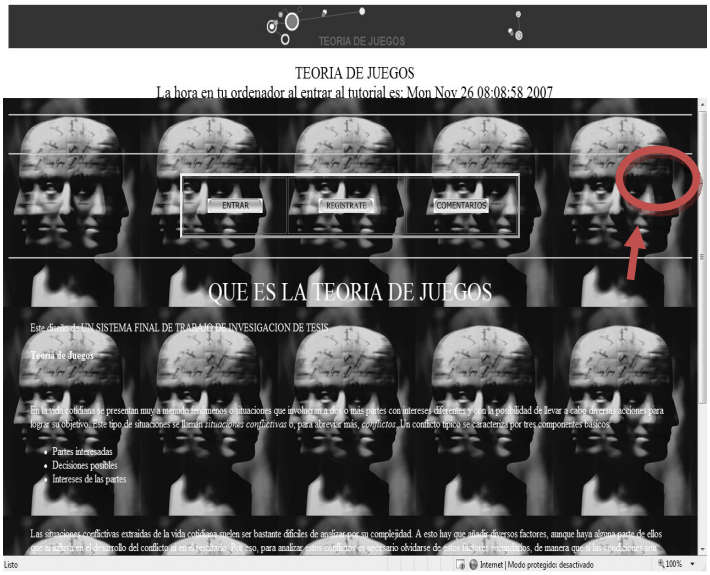


Figura 5.3.1

5.4MODO USUARIO PANTALLA DE REGISTRO DE USUARIOS

Pagina donde el usuario tendrá que proporcionar datos como son

- NOMBRE DE USUARIO
- CONTRACEÑA
- DIRECCION DE CORREO

El nombre de usuario y contraseña son campos obligatorios para entrar al sistema estos datos se almacenaran en la Base de Datos

Estructura:

- Código HTML de interfaz.

- Programación con lenguaje PHP y acceso a la Base de Datos.

- Contiene 3 campos
    1. Nickname
    2. Contraseña aparecerá con asteriscos para mayor seguridad.
    3. Dirección de correo.
- Contiene 3 botones que son:
    1. Enviar: Envía información a la Base de datos
    2. Limpiar: para no regresar a la página principal borra la información que metió
    3. Atrás por si no quiere registrarse regresa a la página de inicio

Figura 5.4.1

5.5 Datos enviados a la Base de Datos Modo Usuario

En esta ventana los datos serán enviados a nuestra base de datos para su almacenamiento con la liga de regresara será enviado a la página principal para que pueda entrar al sistema.

Contiene una liga a la página de comentarios por si desea enviar un comentario o cambiar sus datos

Estructura:

- Código HTML de interfaz.

- Programación con lenguaje PHP y vinculación con la Base de Datos.

Figura 5.5.1

5.6 MODO USUARIO Validando los campos

Esta ventana muestra si los datos que dio el usuario ya están dentro de nuestra Base de Datos le enviara una ventana donde diga que esos datos ya están dentro de nuestra Base de Datos y que intente con otro se le dan la liga a la ventana de registro a él sistemas

Estructura:
Código HTML (Dream weaver)
Codigo java-script

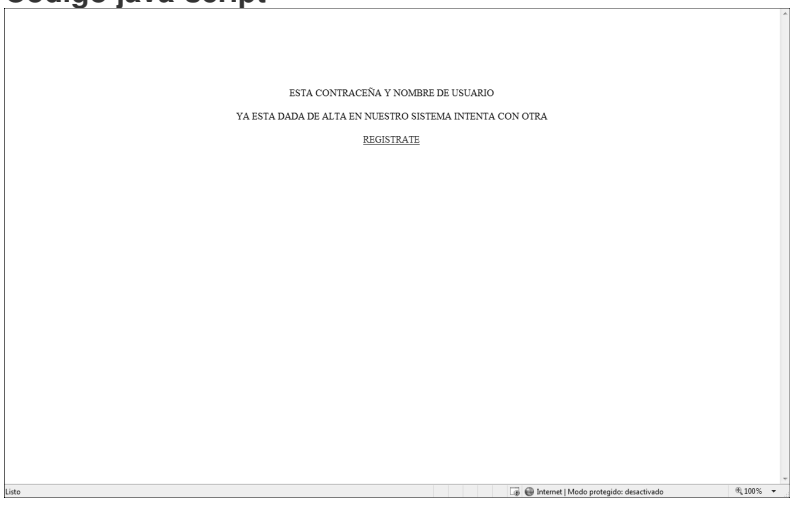


Figura 5.6.1

5.7MODO USUARIO Entrar al sistema Con datos dentro de la base de dato (ya registrados)

Estructura:

- Código HTML de interfaz.
- Código FLASH encabezado
- Programación con lenguaje PHP y acceso a la Base de Datos.
- Contiene 2 campos
    1. Nickname o Apodo no importa los caracteres.
    2. Contraseña aparecerá con asteriscos para mayor seguridad.
        - Los cuales serán validados para entrar a nuestro sistema si se encuentran dentro de la Base de Datos accederá si ningún problema a la pagina donde se encuentra la información de Teoría de juegos en línea.

Figura 5.7.1

5.8 MODO USUARIO Entrar al sistema (no registrados)

Estructura:

- Código HTML de interfaz.
- Código FLASH encabezado
- Programación con lenguaje PHP y acceso a la Base de Datos.

Enviara una alerta que no se encuentra registrado en nuestra base de datos y le dar la opción al momento de aceptar de regresar a la página principal para entrar a la opción de registro

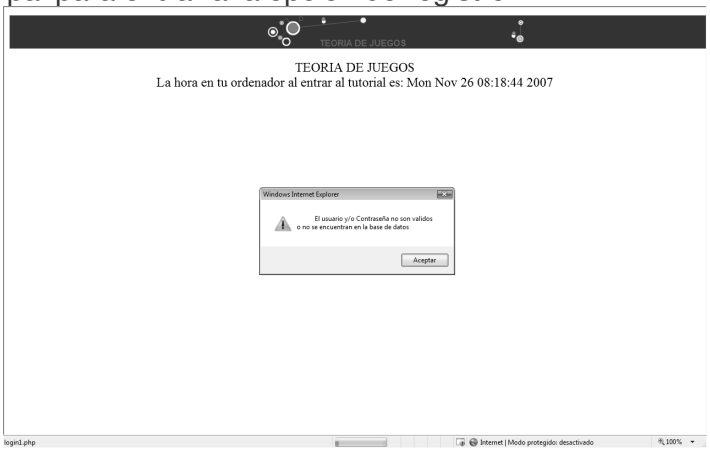


Figura 5.8.1

5.9MODO USUARIO Entrar al sistema (validación correcta) Pagina de juegos

Pagina de juegos en la cual contiene ejemplos de teoría de juegos, probabilidad, estadística, matemáticas.

En la cual el usuario tiene una función de un buscador, enviar comentarios, registrar a otro usuario.

Una de las funciones de este sistema es la AUTO-EVALUACION y LA ASESORIA EN LINEA.

Estructura:
Navegador de WEB realizado con SCRIPT el cual mostrara la información que el usuario busca en cuatro diferentes buscadores.

Botón de comentarios
Registrar un nuevo usuario

Auto-evaluación: La cual servirá para que el usuario pueda practicar la teoría de juegos en línea con preguntas ya establecidas.

Asesoría en Línea: Función donde el usuario puede entrar y pedir ayuda o ayudar a personas que tengan preguntas sobre la Teoría de Juegos todo esto en Línea

Botón de regreso a la página Principal

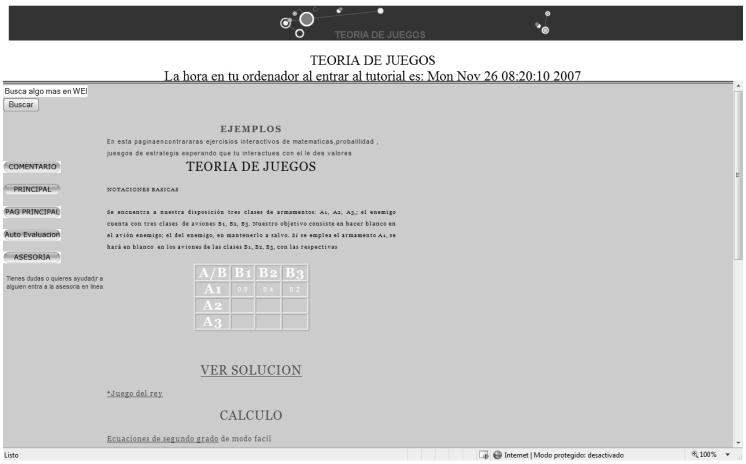


Figura 5.9.1

5.10 MODO USUARIO AUTO-EVALUACION

Pagina de problemas.
Pagina donde el usuario al momento de ingresar a la Auto-Evaluación encontrara las instrucciones que se necesitan para interactuar con el Sistema y las preguntas que se reflejaran con una grafica

Utiliza código PHP con el acceso a la Base de Datos

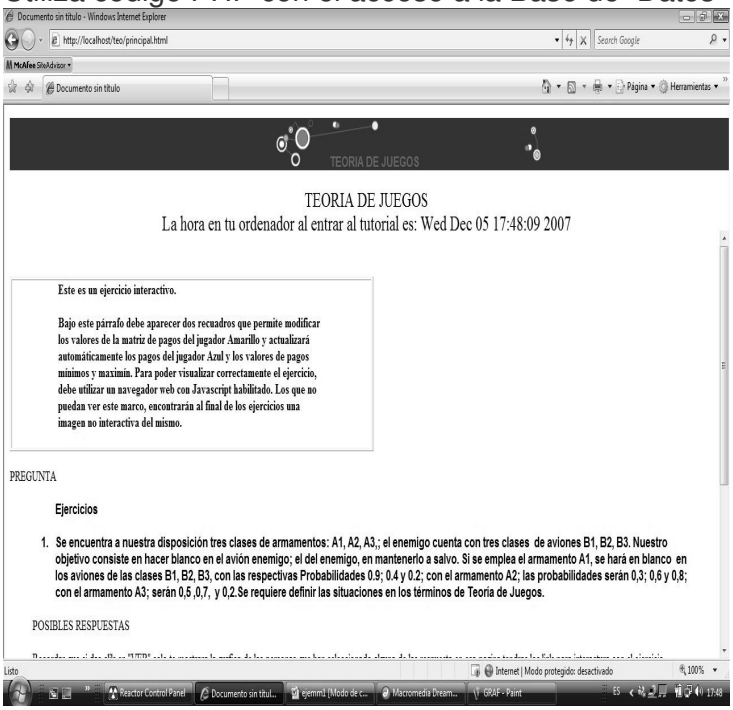
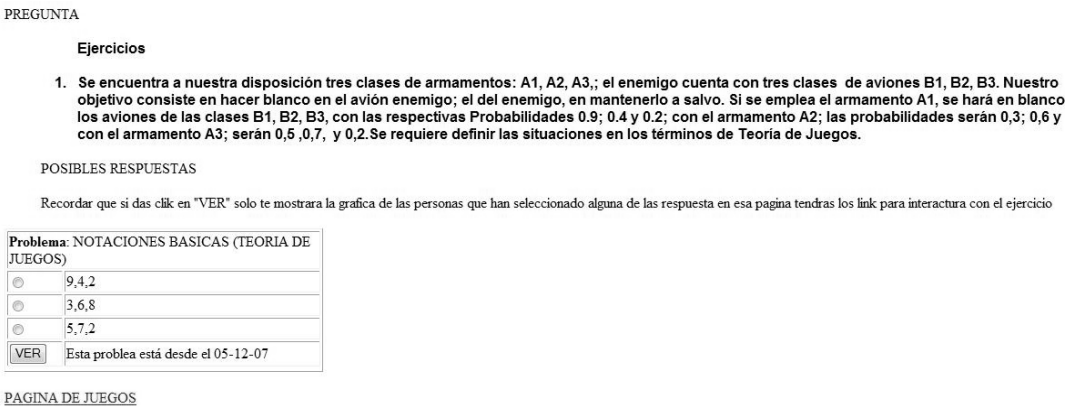


Figura 5.10.1

5.11 Pagina de Preguntas

Pagina donde se encuentran las repuestas posibles código PHP con acceso a la Base de Datos



PREGUNTA

Ejercicios

1. **Se encuentra a nuestra disposición tres clases de armamentos: A1, A2, A3,; el enemigo cuenta con tres clases de aviones B1, B2, B3. Nuestro objetivo consiste en hacer blanco en el avión enemigo; el del enemigo, en mantenerlo a salvo. Si se emplea el armamento A1, se hará en blanco los aviones de las clases B1, B2, B3, con las respectivas Probabilidades 0.9; 0.4 y 0.2; con el armamento A2; las probabilidades serán 0,3; 0,6 y con el armamento A3; serán 0,5 ,0,7, y 0,2.Se requiere definir las situaciones en los términos de Teoría de Juegos.**

POSIBLES RESPUESTAS

Recordar que si das clik en "VER" solo te mostrara la grafica de las personas que han seleccionado alguna de las respuesta en esa pagina tendras los link para interactura con el ejercicio

Problema: NOTACIONES BASICAS (TEORIA DE JUEGOS)	
○	9,4,2
○	3,6,8
○	5,7,2
VER	Esta problea está desde el 05-12-07

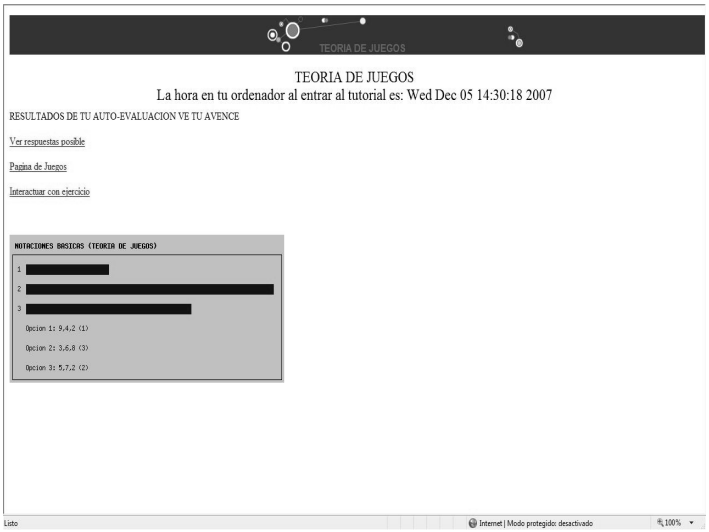
PAGINA DE JUEGOS

Figura 5.11.1

5.12 MODO USUARIO AUTO-EVALUACION

Pagina de consulta de preguntas en modo grafico.

Página en la cual el usuario encontrara todas las personas que han interactuado con el sistema y ver cuáles son las personas que han seleccionado una pregunta en concreto y contiene las ligas para que el usuario interactué con los ejercicios en línea



Figuera 5.12.1

5.13 MODO USUARIO AUTOO EVALUACION JUEGO EN LINEA

- Pagina donde el usuario interactuar con ejercicios reales de teoría de juegos realizada con código HTML, JavaScrip y flash

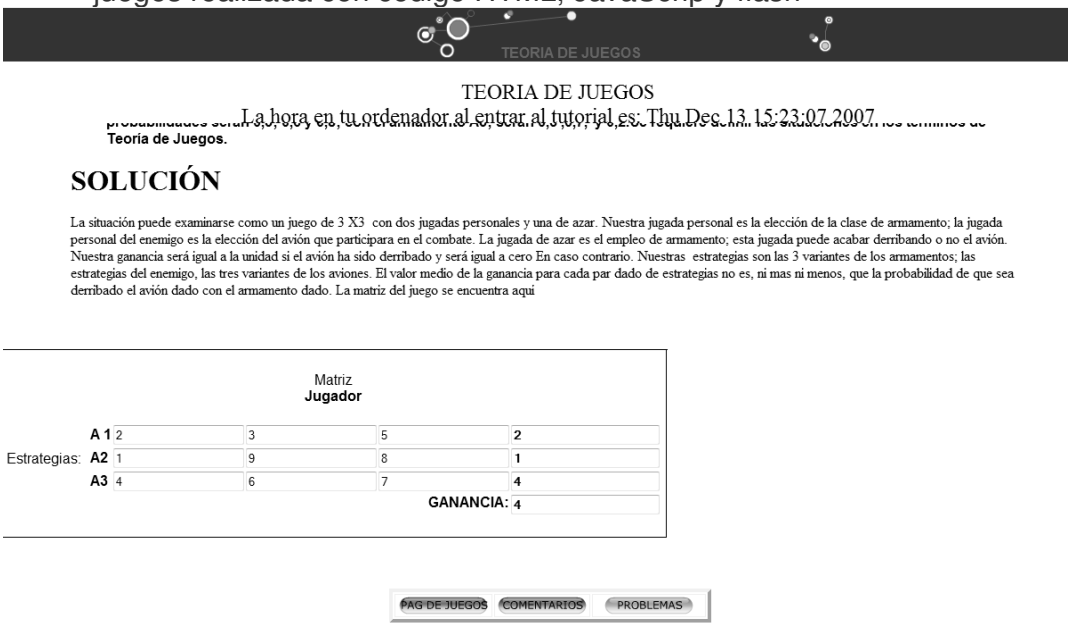


Figura 5.13.1

5.14 MODO USUARIO ASESORIA EN LINEA

- Pagina de asesoría en línea realizada con Código PHP y flash en la cual se le pedirá un nombre de usuario.

Figura 5.14.1

Mensaje de Bienvenida donde le da toda la información de la asesoría en línea

Figura 5.14.2

Pantalla donde podrá dar Asesoría en línea o Pedir asesoría a cualquier persona que esté en línea.

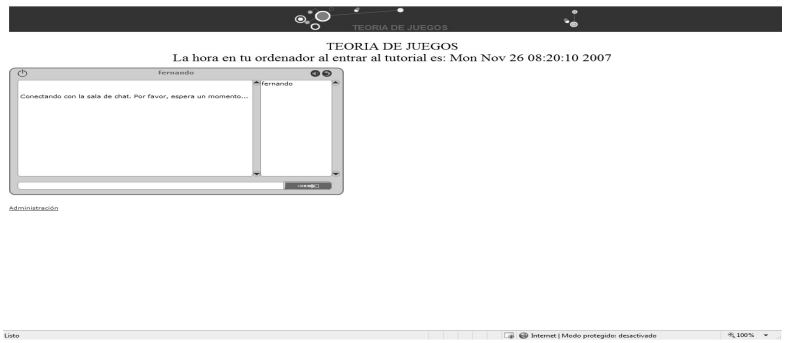


Figura 5.14.3

5.15 MODO ADMINISTRADOR Acceso modo administrador

Esta ventana es de uso del administrador del sistema en él podrá realizar las funciones del administrador modificar un registro, eliminar, insertar un nuevo registro, insertar una nueva evaluación ver el avance de los usuarios.

Estructura:

- Código HTML de interfaz.
- Programación con lenguaje PHP y acceso a la Base de Datos.

Enviara una los datos a la base de dato y verificara si están dentro de ella para entrar a las funciones del administrador.

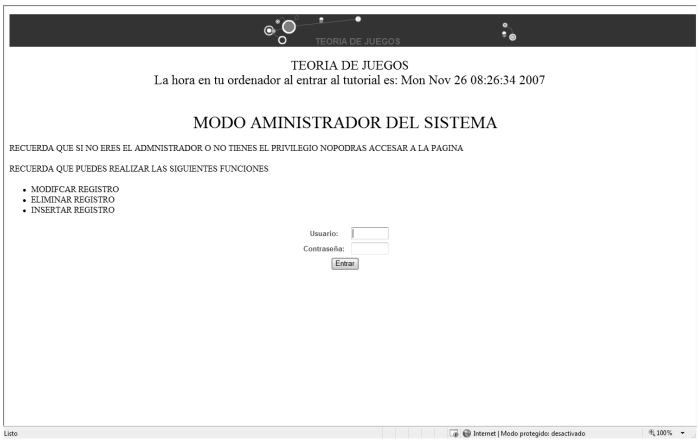


Figura 5.15.1

5.16 Modo administrador validar DATOS

Esta ventana mostrara si al momento de entrar al sistema no son los datos correctos enviará un error y al momento de dar aceptar enviará a la página principal

ESTRUCUTURA:

- Código JAVA de alerta.
- Encabezado en FLASH
- Botón aceptar en JAVA
-

  - Esta opción es solo válida para el administrador
    1. Y enviara a la página donde podrá consultar información
    2. Dar de baja a un usuario
    3. Modificar datos de un usuario

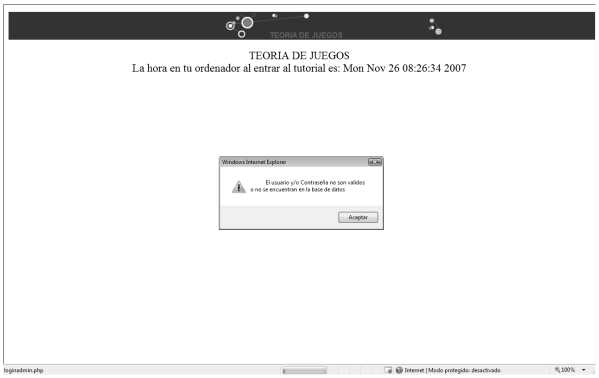


Figura 5.16.1

5.17 Modo administrador Menú de Opciones

ESTRUCTURA

- Esta opción es solo válida para el administrador
  1. Dar de baja a un usuario

2. Modificar datos de un usuario
3. Dar de alta una Auto-Evaluación

Figura 5.17.1

5.18Modo administrador Opciones Modificar

ESTRUCTURA

En esta opción el administrador modificara el nic, contraseña y la dirección de correo:

Él numera de ficha es el número de identificación que se le asigna al usuario y puede actualizar las veces que él quiera

Figura 5.18.1

5.19 Modo administrador Opciones Eliminar

En esta opción el administrador eliminara a un usuario solo con él numeró de registro (IDENTIFICACION)

Tendrá las opciones de
Modificar
Salir
Dar de alta una Evaluación

Figura 5.19.1

5.20 MODO ADMINISTRADOR Consulta de usuarios

En esta ventana el usuario tendrá la facilidad de ver a todos los usuarios que se han registrado en el Sistema.

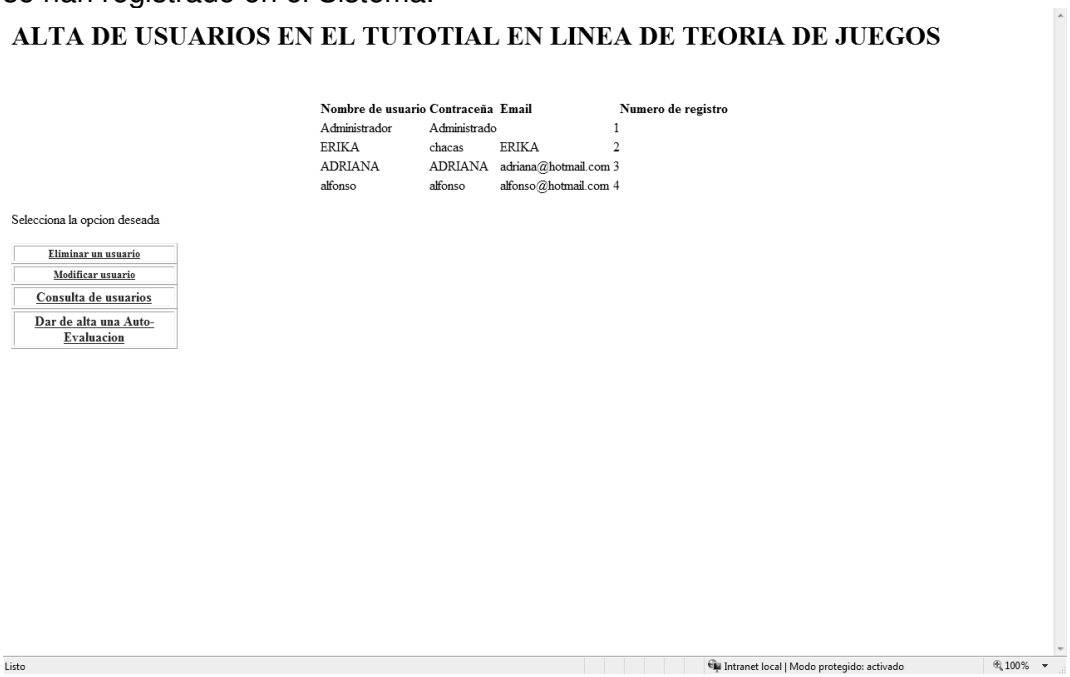


Figura 5.20

5.21 MODO ADMINISTRADOR Dar de alta una nueva Evaluación

Esta ventana es solo para el uso del administrador en la cual el administrador podrá poner en línea nuevo ejercicios en línea, podrá dar él titulo de la Pregunta, así como respuestas las que él considere que son necesarias para la evaluación de los usuarios.

Figura 5.21

5.22 Modo Administrador Alta de Respuestas

Esta ventana muestra las opciones donde el administrador puede poner las preguntas que él necesite para la evaluación.

Figura 5.22.1

5.23 MODO ADMININISTRADOR Fin de la Captura de una nueva Auto-Evaluación.

Ventana en la cual una vez insertado la auto-evaluación le dirá que fue insertada en la base de datos y actualizada en la pagina para que los usuarios puedan interactuar con ella.


TEORIA DE JUEGOS
TEORIA DE JUEGOS
La hora en tu ordenador al entrar al tutorial es: Mon Nov 26 08:40:54 2007
Felicidades!! Si todo ha ido bien, tu AUTO-EVALUACION ha sido insertada!!
Dentro de la Base de Datos y el Tutorial en Linea
VEO COMO AUMENTAN LAS POSIBLES AUTOEVALUACIONES
AUTO-EVALUACION
DAR DE ALTA OTRA EVALUACION
PAGINA PRINCIPAL MODO ADMINISTRADOR
Internet | Modo protegido: desactivado
100%

Figura 5.23.1

Conclusiones

La Teoría de Juegos aporta conocimientos a otras áreas importantes tales como la Ingeniería Artificial, Investigación de Operaciones y simulación entre otros.

Como resultado de este trabajo tenemos un Sistema básico disponible la red para que cualquier persona pueda familiarizarse con algunos conceptos de la Teoría de Juegos

El Tutorial en Línea que se presenta en su primera versión, tiene algunas limitaciones por lo cual se sugiere algunas recomendaciones para mejorarlo tales como:

Interactuar con un sistema donde puedan aprender y ejecutarse, obteniendo así un avance de sus habilidades de teoría de juegos.

Trabajo a Futuro

- Como recomendación del sistema se tiene que mejorar la parte grafica del sistema ya que se le puede dar un mejor diseño para que el usuario le agrade mas la vista del sistema.

- La forma de insertar juegos ya que es necesario hacerlo de una forma más fácil y rápida.

- Forma de interactuar con el sistema ya que es recomendable hacerlo con ejemplos donde se puedan interactuar de una forma más sencilla y donde de los resultados se presente de una forma más rápida.

ANEXO A Código del Sistema de Teoría de Juegos en Línea

Este anexo proporciona los códigos que se utilizaron en el prototipo del sistema de Teoría de Juegos en Lina.

Como los códigos para acceder a la Base de Datos, enviar comentarios, registrarse a la Base de Datos, verificación de datos, auto-evaluación y muchos más códigos para su uso y mejoría.

A.1CODIGO PROTOTIPO DEL SISTEMA

A.1.1Código en el cual da la bienvenida al sistema.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>CARGAR PAGINA</title>
</head>
<body>
<body>
</div>
<div align="center">
  <script>
i=0;
tamano=0;
hauteur = (navigator.appName != "Microsoft Internet Explorer")?
window.innerHeight : document.body.offsetHeight;
marge = hauteur/3;
document.getElementsByTagName('div')[0].style.marginTop = marge;
textanim = new Array("TEORIA DE JUEGOS EN LINEA");
function animation() {
document.getElementsByTagName('div')[0].style.fontSize = ""+tamano+"px";
document.getElementById('bienvenidos').innerHTML = textanim[i];
if (tamano < 50) {
tamano++;
}
else {
i++;
tamano=0;
}
if (i < textanim.length) {
setTimeout('animation()',50)
}
}
setTimeout('animation()',50);
</script><br>
<br>
<br>
<br>
  <a href="entrar.html">ENTRAR</a>
```

A.1.2Código en el cual da la introducción con una animación en FLASH

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>INTRO TEORIA DE JUEGOS EN LINEA</title>
<style type="text/css">
<!--
body {
      background-color: #999999;
}
</style></head>

<body>
<p> </p>
<p>
  <object                classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab
#version=7,0,19,0" width="732" height="363" align="absmiddle" title="INTRO">
    <param name="movie" value="940-2.swf" />
    <param name="quality" value="high" />
    <embed   src="940-2.swf"   width="732"   height="363"   align="absmiddle"
quality="high"      pluginspage="http://www.macromedia.com/go/getflashplayer"
type="application/x-shockwave-flash"></embed>
  </object>
</p>
<p align="center">
  <object                classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab
#version=5,0,0,0" width="100" height="20">
    <param name="movie" value="button9.swf" />
    <param name="quality" value="high" />
    <param name="bgcolor" value="#666666" />
    <embed                  src="button9.swf"                  quality="high"
pluginspage="http://www.macromedia.com/shockwave/download/index.cgi?P1_Prod_V
ersion=ShockwaveFlash"    type="application/x-shockwave-flash"    width="100"
height="20" bgcolor="#666666"></embed>
  </object>
</p>
</body>
</html>
```

A.1.3Código de la Pagina Principal del Sistema en el cual contiene las funciones de.

- Entrar a el sistema
- Registrarse
- Comentarios

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-
1" />
<title>PAGINA PRINCIPAL</title>
<style type="text/css">
<!--
.Estilo1 {   color: #FFFFFF;
      font-size: 36px;
}
.Estilo2 {color: #FF0000}
body {
      background-image: url(02_estrategia_b.jpg);
}
.Estilo3 {color: #999999}
.style2 {
      font-size: 18px;
      color: #FFFFFF;
}
.style3 {color: #000000}
.style4 {color: #FFFFFF}
.style6 {color: #FF0000; font-weight: bold; }
</style>
</head>
<hr>
<body>
          <br>  <br>
<div align="left">

    <span class="style6"></span>    <span class="style4"></span>

            <hR>
        <div            align="right"><a            href="regadmin.php"
class="style3">ad</a>
          <br>
        </div>
</div>
  <table width="618" border="4" align="center">
  <tr bordercolor="#666666">
    <th width="184" height="74" scope="col"><span class="Estilo3">
      <object      classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swfl
ash.cab#version=5,0,0,0" width="100" height="20" title="ENTRAR">
        <param name="BGCOLOR" value="" />
        <param name="movie" value="button2.swf" />
        <param name="quality" value="high" />
        <embed             src="button2.swf"             quality="high"
pluginspage="http://www.macromedia.com/shockwave/download/index.cgi?P1
_Prod_Version=ShockwaveFlash"      type="application/x-shockwave-flash"
width="100" height="20" ></embed>
      </object>
    </span></th>
    <th  width="209"  scope="col"><object  classid="clsid:D27CDB6E-AE6D-
11cf-96B8-444553540000"
codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swfl
ash.cab#version=5,0,0,0" width="100" height="20" title="REGISTRATE">
      <param name="BGCOLOR" value="" />
      <param name="movie" value="button1.swf" />
      <param name="quality" value="high" />
      <embed              src="button1.swf"              quality="high"
pluginspage="http://www.macromedia.com/shockwave/download/index.cgi?P1
```

```
_Prod_Version=ShockwaveFlash"      type="application/x-shockwave-flash"
width="100" height="20" ></embed>
    </object></th>
    <th width="197" scope="col"><object classid="clsid:D27CDB6E-AE6D-
11cf-96B8-444553540000"
codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swfl
ash.cab#version=5,0,0,0" width="100" height="20">
      <param name="movie" value="button4.swf" />
      <param name="quality" value="high" />
      <embed src="button4.swf" width="100" height="20" quality="high"
pluginspage="http://www.macromedia.com/shockwave/download/index.cgi?P1
_Prod_Version=ShockwaveFlash"      type="application/x-shockwave-flash"
></embed>
    </object></th>
    </tr>
</table>
<BR>
<hr>
<blockquote>
  <p align="center" class="Estilo1">QUE ES LA TEORIA DE JUEGOS </p>
  <p class="Estilo2">Este dise&ntilde;o de UN SISTEMA FINAL DE TRABAJO
DE INVESIGACION DE TESIS </p>
  <p class="Estilo2"><strong>Teor&iacute;a de Juegos</strong></p>
  <p class="Estilo2"> </p>
  <p class="Estilo2">En la vida cotidiana se presentan  muy a menudo
fen&oacute;menos o situaciones que involucran a dos o m&aacute;s
partes con  intereses diferentes y con la posibilidad de llevar a cabo
diversas acciones  para lograr su objetivo. Este tipo de situaciones
se llaman <em>situaciones    conflictivas </em>o, para abreviar
m&aacute;s, <em>conflictos</em>. Un conflicto t&iacute;pico    se
caracteriza por tres componentes b&aacute;sicos:</p>
  <ul>
    <li>Partes interesadas</li>
    <li>Decisiones posibles</li>
    <li>Intereses de las partes</li>
  </ul>
  <p class="Estilo2"> </p>
  <p class="Estilo2">Las  situaciones conflictivas extra&iacute;das de
la vida cotidiana suelen ser bastante  dif&iacute;ciles de analizar
por su complejidad. A esto hay que a&ntilde;adir diversos  factores,
aunque haya alguna parte de ellos que ni influya en el desarrollo del
conflicto ni en el resultado. Por eso, para analizar estos conflictos
es  necesario olvidarse de estos factores secundarios, de manera que
si las  condiciones son &oacute;ptimas se pueda construir un modelo
normal y simplificado.  Dicho modelo se suele denominar <em>juego</em>
<p class="Estilo2">La necesidad  de estudiar conflictos susceptibles
de ser representados con modelos  matem&aacute;ticos simples (juegos)
ha    dado    lugar    a    un    aparato    matem&aacute;tico
llamado <strong><em>teor&iacute;a  de juegos</em></strong>.</p>
  <p    class="Estilo2"> Las    partes    interesadas    se
llaman <em>jugadores</em>.  Cada acci&oacute;n posible dentro del marco de
las  reglas  establecidas  se  denomina  <em>estrategia</em>.     Bajo
condiciones de conflicto, cada jugador escoge su estrategia (la que
m&aacute;s le  puede convenir por ejemplo) y como resultado se genera
un conjunto de  estrategias entre todos los jugadores que se denomina
<em>situaci&oacute;n</em>. Para    medir el inter&eacute;s de cada
jugador en una situaci&oacute;n concreta se asigna un n&uacute;mero
que expresa, por decirlo de manera pedante, la satisfacci&oacute;n de
```

```
sus intereses en  dicha situaci&oacute;n. A este n&uacute;mero se le
llama <em>pago</em> del jugador en la  situaci&oacute;n dada.<br />
    p; Bajo  estas circunstancias, el desarrollo del conflicto no es
otra cosa que la  elecci&oacute;n por parte de los jugadores de
determinadas estrategias y la obtenci&oacute;n  de un pago proveniente
de otros jugadores o de alguna otra fuente. Esto da  lugar a lo que se
conoce como <em>teor&iacute;a de juegos con utilidad</em>.</p>
  <p  class="Estilo2">Divi&eacute;rtase  y  en  este  estupendo  sitio
Web!</p>
  <p>  </p>
</blockquote>
</body>
</body>
</html>
```

A.1.4 CODIGO MODO USUARIO Código el cual servirá para registrar un usuario.

```
<!DOCTYPE  html  PUBLIC  "-//W3C//DTD  XHTML  1.0  Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>REGISTRO DE USUARIOS</title>
</head>

<body bgcolor="#FFFFFF">
<p align="left"><font face="Times New Roman, Times, serif"><strong>BIENVENIDO
AL EL TUTORIA DE TEORIA DE JUEGOS EN LINEA SI DECEAS DARTE DE ALTA A EL
SISTEMA SOLO SIGUE LAS SIGUIENTES INSTRUCCIONES </strong></font></p>
<p align="left"><font face="Times New Roman, Times, serif"><strong>1-. SOLO
INSERTA TU NOMBRE DE USUARIO </strong></font></p>
<p  align="left"><font  face="Times  New  Roman,  Times,  serif"><strong>2-.  TU
CONTRACE&Ntilde;A     QUE     SERA     ENVIADA     A     NUESTRA     BASE     DE
DATOS</strong></font></p>
<p  align="left"><font  face="Times  New  Roman,  Times,  serif"><strong>3-.TU
CORREO ELECTRONICO</strong></font></p>
<p align="left">  4.-LOS DATOS MARCADOS CON * SON NECESARIOS PARA REGISTRARTE
EN EL SISTEMA <br />
</p>

  <form action="savealt.php" method="post" target="_parent">
    <div align="center">
      <p>*NOMBRE DE USUARIO<BR>

        <input name="nic1" size="28" />
      </p>
      <p>        <BR>
        *CONTRACE&Ntilde;A <BR>
        <input name="pw1" type="password" size="28" />
      </p>
      <p>        <BR>
        DIRECCION EMAIL <BR>
        <input name="direccion1" size="28" />
      </p>
      <p>        <BR>
        <input type="submit" value="Enviar" name="Enviar" />
         <input type="reset" value="Limpiar" name="Limpiar" />
      </p>
```

```
    </div>
  </form>
</body>
</html>
```

A.1.5 CODIGO USUARIO Código en el cual inserta datos en la Base de Datos

```
<title>Datos</title>
<style type="text/css">
<!--
body,td,th {
       color: #333333;
}
.style1 {color: #000000}
-->
</style><body bgcolor="#FFFFFF">

<p>
  <?php
include("conexion.php");
$link=conectar();
$sql="INSERT                                INTO                                altas
(nic,pw,direccion)VALUES('$nic1','$pw1','$direccion1');";
$resul=mysql_query($sql,$link);
if (!empty($resul))
{
   echo "Usuario registrado";
   echo "<br>";
                                  echo  "En  unos  Instantes  se  activara  tu
Cuenta";
                                                      echo "<br>";
                                     echo                                      "<a
href='principal.html'>Regresar</a>";
}
else
{
$dir = "validar registro.html";
header("Location: ". $dir );
}
//cierrro la conexión
mysql_close($link);
?>
</p>
<p> </p>
<p>SI NECESITAS MAS INFORMACION SOBRE TU CUENTA ENVIA UN CORREO ELECTRONICO
PIDIENDO QUE SE TE MODIFIQUEN LOS DATOS </p>
<p>ATTE</p>
<span class="style1"></span>
<p>ADMINISTRADOR DE SISTEMAS</p>
<p>Modifica tu cuenta  </p>
```

A.1.6CODIGO MODO USUARIO valida los datos antes de entrar a el sistema.

```
      <html>
      <head>
      <title>index</title>
      </head>
      <body bgcolor="#FFFFFF">
        <p> </p>
```

```
        <p> </p>
        <table width="171" height="71" border="0" align="center">
        <form action="login1.php" name="form1" method="post">
          <tr>
            <th width="91" height="2" scope="col"><div align="center"><font
color="#0063C6"       size="2"       face="Arial,       Helvetica,       sans-
serif">Usuario:</font></div></th>
            <td width="79" scope="col"><div align="center">
              <input name="nic" type="text" size="7" maxlength="15">
            </div></td>
          </tr>
          <tr>
            <th     height="23"     scope="row"><div     align="center"><font
color="#0063C6"       size="2"       face="Arial,       Helvetica,       sans-
serif">Contrase&ntilde;a:</font></div></th>
            <td><div align="center">
              <input    name="pw"    type="password"    id="pw"    size="7"
maxlength="15">
            </div></td>
          </tr>
          <tr>
            <th height="38" colspan="2" scope="row"> <p align="center">
              <input name="enviar" type="submit" value="Entrar">
              <br>
            <a   href="registro.php"   class="style2"   target="_parent"><font
color="#0063C6"       size="2"       face="Arial,       Helvetica,       sans-
serif">Registrarse</font></a> </tr>
      </form>
      </table>
        <table width="673" border="0">
          <tr>

          </tr>
        </table>
        <br>
        <table    width="851"    height="124"    border="0"    cellpadding="2"
cellspacing="2">
        </table>
      </body>
      </html>
```

A.1.8 Código PHP con el cual valida los datos en la Base de Datos si son verdaderos o no.

```
<?php
include("conexion.php");
$link=conectar();
mysql_select_db("usuarios",$link);
$query_chequeo = "SELECT nic, pw FROM altas";
$chequeo = mysql_query($query_chequeo, $link) or die(mysql_error());
$row_chequeo = mysql_fetch_assoc($chequeo);
$totalRows_chequeo = mysql_num_rows($chequeo);
?><?php
// *** Validate request to login to this site.
if (!isset($_SESSION)) {
  session_start();
}
```

```
$loginFormAction = $_SERVER['PHP_SELF'];
if (isset($_GET['accesscheck'])) {
  $_SESSION['PrevUrl'] = $_GET['accesscheck'];
}

if (isset($_POST['nic'])) {
  $loginUsername=$_POST['nic'];
  $password=$_POST['pw'];
  $MM_fldUserAuthorization = "";
  $MM_redirectLoginSuccess = "juegos.php";
  $MM_redirectLoginFailed = "invalido.php";
  $MM_redirecttoReferrer = false;
  mysql_select_db("usuarios",$link);

  $LoginRS__query=sprintf("SELECT nic, pw FROM altas WHERE nic='%s' AND
pw='%s'",
    get_magic_quotes_gpc() ? $loginUsername : addslashes($loginUsername),
get_magic_quotes_gpc() ? $password : addslashes($password));

  $LoginRS = mysql_query($LoginRS__query, $link) or die(mysql_error());
  $loginFoundUser = mysql_num_rows($LoginRS);
  if ($loginFoundUser) {
     $loginStrGroup = "";

    //declare two session variables and assign them
    $_SESSION['MM_Username'] = $loginUsername;
    $_SESSION['MM_UserGroup'] = $loginStrGroup;

    if (isset($_SESSION['PrevUrl']) && false) {
      $MM_redirectLoginSuccess = $_SESSION['PrevUrl'];
    }
    header("Location: " . $MM_redirectLoginSuccess );
  }
  else {
    header("Location: ". $MM_redirectLoginFailed );
  }
}
?>
```

A.1.9 Modo Usuario Código con el cual si el usuario al momento de ingresar sus datos son falsos enviara la alerta a el usuario.

```
<html>
<head>
<script language="javascript">
 alert("               El usuario y/o Contraseña no son validos\r\no no se
encuentran en la base de datos");
window.open("principal.html", target="_parent")
</script>
</head>
<body>
</body>
</html>
```

A.1.10 MODO USUARIO Código de Auto-Evaluación
CODIGO A.1.10.1

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Untitled Document</title>
</head>

<body>
Esta Pantalla es solo para Administrador y da de alta una Auto-Evaluacion con
sus posibles respuestas
<form name="form1" method="post" action="enc2.php">
  <p>Titulo AUTO EVALUACION:
    <input type="text" name="titulo">
  </p>
  Pon las respuestas que creeas conveniente para que voten por ellas
  <p>N&uacute;mero de respuestas:
    <input type="text" name="respuestas">
  </p>
  <p>
    <input type="submit" name="Submit" value="Enviar">
  </p>
  <p><a href="funadmin.php">Salir </a></p>
  <p> </p>
  <p> </p>
</form>

</body>
</html>
```

CODIGO A.1.10.2

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Untitled Document</title>
</head>

<body>

<form action="enc3.php" method="post">
  <table border="0">
<?php
  for($i=1;$i<=$respuestas;$i++){
?>
  <tr>
    <td>Respuestas Posibles <?php echo $i; ?></td>
    <td><input name="p<?php echo $i;?>" type="text" size="50"
maxlength="50"></td>
  </tr>
<?php } ?>
  </table>
    <input type="submit" name="Submit" value="Enviar"></p>
      <input name="titulo" type="hidden" value="<?php echo $titulo;?>">
      <input type="hidden" name="respuestas" value="<?php echo
$respuestas;?>">
</form>
```

```
</body>
</html>
```

CODIGO A.1.10.3

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>AUTO EVALUACION</title>
</head>

<body>
<?php

//Conectamos con la base de datos
require('configuracion.inc.php');
$enlace = mysql_connect($host, $usuario, $password);
mysql_select_db($db,$enlace);

//Obtenemos la fecha del sistema
$fecha = time();

//Insertamos la nueva encuesta
$sql = "INSERT INTO encuestas (titulo, fecha) VALUES ('$titulo', '$fecha') ";
$sql = mysql_query($sql);

//Ahora obtenemos el ID de la encuesta que acabamos de insertar
$sql = "SELECT id FROM encuestas ORDER BY fecha DESC LIMIT 0,1";
$sql = mysql_query($sql);
while($row = mysql_fetch_array($sql)){
      $id=$row["id"];
}

//Recorremos todas las preguntas
for($i=1; $i<=$respuestas; $i++){

//Obtenemos el texto de la pregunta
      $preg = p.$i;
      $texto = $$preg;

//Y lo insertamos
      $sql = "INSERT INTO respuestas(texto, votos, idenc) VALUES(\"$texto\",
0, $id)";
      $sql = mysql_query($sql);
      }
 ?>
<div align="center">
  <p><strong>Felicidades!! Si todo ha ido bien, tu AUTO-EVALUACION ha
    sido insertada!! </strong> </p>
  <p>Dentro de la Base de Datos y el Tutorial en Linea </p>
</div>

<p>DA TU PROPIO VOTO Y VEO COMO AUMENTAN LAS POSIBLES AUTOEVALUACIONES </p>
<p><a href="encuenta.php">VOTAR</a></p>
<p><a href="enc1.php">DAR DE ALTA OTRA EVALUACION </a></p>
<p>  </p>
<p><a href="principal.html">PAGINA PRINCIPAL</a> </p>
```

```
</body>
</html>
```

A.1.11MODO USUARIO Almacenar al usuario en la Base de Datos

```
Código PHP
<?php
include("conexion.php");
$link=conectar();
mysql_select_db("usuarios",$link);
$query_chequeo = "SELECT nic, pw FROM altas";
$chequeo = mysql_query($query_chequeo, $link) or die(mysql_error());
$row_chequeo = mysql_fetch_assoc($chequeo);
$totalRows_chequeo = mysql_num_rows($chequeo);
?><?php
// *** Validate request to login to this site.
if (!isset($_SESSION)) {
  session_start();
}

$loginFormAction = $_SERVER['PHP_SELF'];
if (isset($_GET['accesscheck'])) {
  $_SESSION['PrevUrl'] = $_GET['accesscheck'];
}

if (isset($_POST['nic'])) {
  $loginUsername=$_POST['nic'];
  $password=$_POST['pw'];
  $MM_fldUserAuthorization = "";
  $MM_redirectLoginSuccess = "juegos.php";
  $MM_redirectLoginFailed = "invalido.php";
  $MM_redirecttoReferrer = false;
  mysql_select_db("usuarios",$link);

  $LoginRS__query=sprintf("SELECT nic, pw FROM altas WHERE nic='%s' AND
pw='%s'",
    get_magic_quotes_gpc() ? $loginUsername : addslashes($loginUsername),
get_magic_quotes_gpc() ? $password : addslashes($password));

  $LoginRS = mysql_query($LoginRS__query, $link) or die(mysql_error());
  $loginFoundUser = mysql_num_rows($LoginRS);
  if ($loginFoundUser) {
     $loginStrGroup = "";

    //declare two session variables and assign them
    $_SESSION['MM_Username'] = $loginUsername;
    $_SESSION['MM_UserGroup'] = $loginStrGroup;

    if (isset($_SESSION['PrevUrl']) && false) {
      $MM_redirectLoginSuccess = $_SESSION['PrevUrl'];
    }
    header("Location: " . $MM_redirectLoginSuccess );
  }
  else {
    header("Location: ". $MM_redirectLoginFailed );
  }
}
?>
```

A.1.13 MODO USUARIO Registrar usuario

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>REGISTRO DE USUARIOS</title>
</head>
<body bgcolor="#94ECF3">
<p align="left"><strong><font face="Verdana, Arial, Helvetica, sans-
serif">BIENVENIDO AL EL TUTORIA DE TEORIA DE JUEGOS EN LINEA SI DECEAS DARTE
DE ALTA A EL SISTEMA SOLO SIGUE LAS SIGUIENTES INSTRUCCIONES
</font></strong></p>
<p align="left"><strong>1-. SOLO INSERTA TU NOMBRE DE USUARIO </strong></p>
<p align="left"><strong>2-. TU CONTRACE&Ntilde;A QUE SERA ENVIADA A NUESTRA
BASE DE DATOS</strong></p>
<p align="left"><strong>3-.TU CORREO ELECTRONICO</strong></p>
<p align="left">  <br />
</p>
  <form action="savealt.php" method="post" target="_parent">
1.-NOMBRE DE USUARIO
<BR>
      <input name="nic1" size="28" />
        <BR>
Contrase&ntilde;a
<BR>
        <input name="pw1" type="password" size="28" />
            <BR>
Direcci&oacute;n
<BR>
     <input name="direccion1" size="28" />
       <BR>
         <input type="submit" value="Enviar" name="Enviar" />
         <input type="reset" value="Limpiar" name="Limpiar" />


  </form>

</body>
</html>
```

A.1.14 MODO USUARIO

```
Ejemplos

<HTML>
<HEAD>

<SCRIPT LANGUAGE ="JavaScript">
<!-- se oculta la información de los navegadores antiguos

  function calculaEcuacion() {
    var a, b, c, solucion;

    a=parseFloat(document.miFormulario.a.value);
```

```
    b=parseFloat(document.miFormulario.b.value);
    c=parseFloat(document.miFormulario.c.value);

    solucion= (-b + Math.sqrt(Math.pow(b,2)-(4*a*c)))/(2*a);
    document.miFormulario.x1.value=solucion;
    solucion= (-b - Math.sqrt(Math.pow(b,2)-(4*a*c)))/(2*a);
    document.miFormulario.x2.value=solucion;
  }

// final del comentario -->
</SCRIPT>
</HEAD>
<BODY>
<CENTER>
  <H1>ECUACIONES </H1>
<FORM NAME="miFormulario">
<BR><B>Introduce los coeficientes de la</B>
<BR><B>siguiente ecuación de 2º grado: </B><BR><BR>
<INPUT TYPE=TEXT NAME="a" SIZE=3 MAXLENGTH=3> x² +
<INPUT TYPE=TEXT NAME="b" SIZE=3 MAXLENGTH=3> x +
<INPUT TYPE=TEXT NAME="c" SIZE=3 MAXLENGTH=3> = 0<BR>
<BR><B>La soluciones son:</B><BR><BR>
 x = <INPUT TYPE=TEXT NAME="x1" SIZE=5 MAXLENGTH=5> ó
 x = <INPUT TYPE=TEXT NAME="x2" SIZE=5 MAXLENGTH=5>
<BR><BR>
<INPUT          TYPE=BUTTON          VALUE="Solucionar"          NAME="miBoton"
onClick="calculaEcuacion()">
</FORM>
</CENTER>
</BODY>
</HTML>
```

A.1.15 MODO USUARIO

EJEMPLOS

```
<HTML>
<HEAD>
<TITLE>Funciones</TITLE>
<SCRIPT LANGUAGE ="JavaScript">
<!-- se oculta la información de los navegadores antiguos

  function ObtieneMayor(dato1, dato2, dato3) {

    var el_mayor=0;

    if (dato1>dato2){
              el_mayor=dato1;
        }
    else  {
              el_mayor=dato2;
           }
    if (dato3>el_mayor) {
                 el_mayor=dato3;
       }
    return el_mayor;
  }

// final del comentario -->
</SCRIPT>
```

```
</HEAD>
<BODY>
<CENTER>
<SCRIPT LANGUAGE ="JavaScript">
<!-- se oculta la información de los navegadores antiguos

  var primer_dato = 0, segundo_dato = 0, tercer_dato = 0;

 do {
       primer_dato = prompt("Dame el primer valor a comparar...","");
       segundo_dato = prompt("Dame el segundo valor a comparar...","");
       tercer_dato = prompt("Dame el tercer valor a comparar...","");

       primer_dato = parseInt(primer_dato);
       segundo_dato = parseInt(segundo_dato);
       tercer_dato = parseInt(tercer_dato);

      document.write("<H3> El mayor valor de ");
      document.write(primer_dato+", " +
                     segundo_dato+" y " +
                     tercer_dato+" es:</H3>");
      document.write("<H1>"                                                +
ObtieneMayor(primer_dato,segundo_dato,tercer_dato)
                     + "</H1>");
   } while (confirm("¿Deseas continuar?"));

    document.write("<H3>...Hecho</H3>");
// final del comentario -->
</SCRIPT>
</CENTER>
</BODY>
</HTML>
```

A.1.16 MODO ADMINISTRADOR

Código en el cual el administrador entra al sistema

```
<html>
<head>
<title>index</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-
1"></head>
<body bgcolor="#FFFFFF">
  <p> </p>
  <p align="center"><font size="+3">MODO AMINISTRADOR DEL SISTEMA</font></p>
  <p> RECUERDA QUE SI NO ERES EL ADMNISTRADOR O NO TIENES EL PRIVILEGIO
NOPODRAS ACCESAR A LA PAGINA</p>
  <p>RECUERDA QUE PUEDES REALIZAR LAS SIGUIENTES FUNCIONES</p>
  <ul>
    <li> MODIFCAR REGISTRO </li>
    <li>ELIMINAR  REGISTRO </li>
    <li>INSERTAR REGISTRO </li>
  </ul>
  <table width="171" height="71" border="0" align="center">
  <form action="loginadmin.php" name="form1" method="post">
    <tr>
      <th width="91" height="2" scope="col"><div align="center"><font
color="#0063C6" size="2" face="Arial, Helvetica, sans-
serif">Usuario:</font></div></th>
```

```
      <td width="79" scope="col"><div align="center">
        <input name="nic" type="text" size="7" maxlength="15">
      </div></td>
    </tr>
    <tr>
      <th height="23" scope="row"><div align="center"><font color="#0063C6"
size="2" face="Arial, Helvetica, sans-
serif">Contrase&ntilde;a:</font></div></th>
      <td><div align="center">
        <input name="pw" type="password" id="pw" size="7" maxlength="15">
      </div></td>
    </tr>
    <tr>
      <th height="38" colspan="2" scope="row"> <p align="center">
        <input name="enviar" type="submit" value="Entrar">
        <br>

    </tr>
</form>
</table>
  <table width="673" border="0">
    <tr>

    </tr>
  </table>
  <br>
  <table width="851" height="124" border="0" cellpadding="2" cellspacing="2">
  </table>

</body>
</html>
```

A.1.17 MODO ADMINISTRADOR Código en el cual si no proporciona los datos el administrador el mensaje que le generara es.

```
<html>
<head>
<script language="javascript">
 alert("               El usuario y/o Contraseña no son validos\r\no no se
encuentran en la base de datos");
window.open("principal.html", target="_parent")
</script>
</head>
<body>
</body>
</html>
```

A.1.18MODO ADMINISTRADOR Menú Principal del modo administrador.

```
<!DOCTYPE    html    PUBLIC    "-//W3C//DTD    XHTML    1.0    Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>MODO ADMINISTRADOR</title>
<style type="text/css">
<!--
body,td,th {
```

```
      color: #333333;
}
-->
</style></head>

<body>
<?php
if (!isset($_SESSION)) {
  session_start();
}
$MM_authorizedUsers = "";
$MM_donotCheckaccess = "true";


function isAuthorized($strUsers, $strGroups, $UserName, $UserGroup) {

  $isValid = False;

  if (!empty($UserName)) {
    $arrUsers = Explode(",", $strUsers);
    $arrGroups = Explode(",", $strGroups);
    if (in_array($UserName, $arrUsers)) {
      $isValid = true;
    }

    if (in_array($UserGroup, $arrGroups)) {
      $isValid = true;
    }
    if (($strUsers == "") && true) {
      $isValid = true;
    }
  }
  return $isValid;
}

$MM_restrictGoTo = "invalido.php";
if                   (!((isset($_SESSION['MM_Username']))                   &&
(isAuthorized("",$MM_authorizedUsers,               $_SESSION['MM_Username'],
$_SESSION['MM_UserGroup'])))) {
  $MM_qsChar = "?";
  $MM_referrer = $_SERVER['PHP_SELF'];
  if (strpos($MM_restrictGoTo, "?")) $MM_qsChar = "&";
  if (isset($QUERY_STRING) && strlen($QUERY_STRING) > 0)
  $MM_referrer .= "?" . $QUERY_STRING;
  $MM_restrictGoTo   =   $MM_restrictGoTo.   $MM_qsChar   .   "accesscheck="   .
urlencode($MM_referrer);
  header("Location: ". $MM_restrictGoTo);
  exit;
}
?>
<?php
if ($salir)
{
mysql_close($link);
$dir = "principal.html";
header ("location: ".$dir);
}
?>
```

```
<meta    http-equiv="Content-Type"    content="text/html;    charset=iso-8859-
1"><style type="text/css">
<!--
body,td,th {
      color: #FFFFFF;
}
-->
</style></head>
<body bgcolor="FFFFFF">
<form method="POST" target="_parent" name="form1">
  <table width="180" height="42" border="0" align="left">

  <tr bgcolor="#FFFFFF">
    <th height="38" background="modif.php" scope="row">

        <div align="center">
          <p>
            <?php
      echo  "<font  color=´#333333´  size='2'  face='Arial,  Helvetica,  sans-
serif'> Bienvenido <br>{$_SESSION['MM_Username']} </font>";
      ?>
            <br>
            <input name="salir" type="submit" value="Salir">
            <br>
            <font  color="#333333"  size="2"  face="Arial,  Helvetica,  sans-
serif"><a                          href="eliminarme.php">Eliminar</a></font><a
href="eliminarme.php"></a>
            </a><br>
            <font  color="#ffffff"  size="2"  face="Arial,  Helvetica,  sans-
serif"><a href="modif.php">Modificar Datos</a></font><a href="modif.php"></a>
            </p>
          </a>        </p>
          <p><a href="enc1.php">Dar de alta una Auto-Evaluacion</a> </p>
        </div>
    </table>

</form>
<?php
$tiempo_logout = 600;
$arr = file("usuarios.txt");
$contenido = $REMOTE_ADDR.":".time()."
";
for ( $i = 0 ; $i < sizeof($arr) ; $i++ )
{
$tmp = explode(":",$arr[$i]);
if (( $tmp[0] != $REMOTE_ADDR ) && (( time() - $tmp[1] ) < $tiempo_logout ))
{
$contenido .= $REMOTE_ADDR.":".time()."
";
}
}
$fp = fopen("usuarios.txt","w");
fputs($fp,$contenido);
fclose($fp);
$array = file("usuarios.txt");
$usuarios_activos = count($array);
echo"<font    color=´#333333´    size='2'    face='Arial,    Helvetica,    sans-
serif'><b>Usuarios Administrador: $usuarios_activos </b></font>";
?>
```

```
</body>
</html>


</html>
```

A.1.19MODO ADMINISTRADOR Código en el cual el Administrador podrá dar de baja a un usuario.

```
<?php
if (!isset($_SESSION)) {
  session_start();
}
$MM_authorizedUsers = "";
$MM_donotCheckaccess = "true";


function isAuthorized($strUsers, $strGroups, $UserName, $UserGroup) {

  $isValid = False;

  if (!empty($UserName)) {
    $arrUsers = Explode(",", $strUsers);
    $arrGroups = Explode(",", $strGroups);
    if (in_array($UserName, $arrUsers)) {
      $isValid = true;
    }

    if (in_array($UserGroup, $arrGroups)) {
      $isValid = true;
    }
    if (($strUsers == "") && true) {
      $isValid = true;
    }
  }
  return $isValid;
}

$MM_restrictGoTo = "invalido.php";
if                  (!((isset($_SESSION['MM_Username']))                  &&
(isAuthorized("",$MM_authorizedUsers,              $_SESSION['MM_Username'],
$_SESSION['MM_UserGroup'])))) {
  $MM_qsChar = "?";
  $MM_referrer = $_SERVER['PHP_SELF'];
  if (strpos($MM_restrictGoTo, "?")) $MM_qsChar = "&";
  if (isset($QUERY_STRING) && strlen($QUERY_STRING) > 0)
  $MM_referrer .= "?" . $QUERY_STRING;
  $MM_restrictGoTo  =  $MM_restrictGoTo.  $MM_qsChar  .  "accesscheck="  .
urlencode($MM_referrer);
  header("Location: ". $MM_restrictGoTo);
  exit;
}
?>
<?php
if ($salir)
{
mysql_close($link);
$dir = "eliminarme.php";
header ("location: ".$dir);
```

```
}
?>

<?php
if($borrar){
include("conexion.php");
$link=Conectar();
  $sql = "DELETE FROM altas WHERE numficha='$numficha'";
  mysql_select_db("usuarios", $link);
  mysql_query($sql, $link) or die(mysql_error());
  $dir = "eliminarme.php";
  if (isset($_SERVER['QUERY_STRING']))
  {
    $dir .= (strpos($dir, '?')) ? "&" : "?";
    $dir .= $_SERVER['QUERY_STRING'];
  }
  header(sprintf("Location: %s", $dir));
  $selec = 'SELECT numficha FROM altas';
  $grab=mysql_query($selec, $link) or die(mysql_error());
  mysql_fetch_assoc($grab);
  mysql_num_rows($grab);
  mysql_close($link);
}
?>
```

A.1.20 DAR DE BAJA USUARIO

```
<html>
<head>
<title>BORRAR USUARIO</title>
</head>
<body bgcolor="#FFFFFF">
<table width="180" height="700" border="0" align="left">
<form action="funadmin.php" method="get">

  </form>
</table>

<table width="673" border="0">
  <tr>
    <td width="10" valign="top"></td>

  </tr>
</table>
<table width="353" border="0" cellspacing="2" cellpadding="2">
  <form action="" method="get" name="form2" target="_parent">
    <tr>
      <th width="138" scope="col"><font color="#0063C6" size="2" face="Arial,
Helvetica, sans-serif">Numero de Ficha:</font></th>
      <th width="201" scope="col"><input name="numficha" type="text"></th>
    </tr>
    <tr>
      <th   colspan="2"   scope="row">   <input   name="borrar"   type="submit"
value="Borrar">
      </th>
    </tr>
  </form>
</table>
<p> </p>
<p>
  <?php
```

```
      echo   "<font   color=&acute;#ffffff&acute;   size='2'   face='Arial,
Helvetica, sans-serif'> Bienvenido <br>{$_SESSION['MM_Username']} </font>";
      ?>
        <br>
  <br>
  <br>
  <a href="funadmin.php">Salir</a>  <br>
  <a  href="modif.php"  class="style2"  target="_parent"><font  color="#0063C6"
size="2" face="Arial, Helvetica, sans-serif">Modificar Datos</font></a></p>
<p><a href="enc1.php">Dar de alta-evaluacion</a> </p>
</p>
</body>
</html>
```

A.1.21MODO ADMINISTRADOR MODIFICAR UN USUARIO

```
<?php
if (!isset($_SESSION)) {
  session_start();
}
$MM_authorizedUsers = "";
$MM_donotCheckaccess = "true";


function isAuthorized($strUsers, $strGroups, $UserName, $UserGroup) {

  $isValid = False;

  if (!empty($UserName)) {
    $arrUsers = Explode(",", $strUsers);
    $arrGroups = Explode(",", $strGroups);
    if (in_array($UserName, $arrUsers)) {
      $isValid = true;
    }

    if (in_array($UserGroup, $arrGroups)) {
      $isValid = true;
    }
    if (($strUsers == "") && true) {
      $isValid = true;
    }
  }
  return $isValid;
}

$MM_restrictGoTo = "invalido.php";
if                  (!((isset($_SESSION['MM_Username']))                  &&
(isAuthorized("",$MM_authorizedUsers,               $_SESSION['MM_Username'],
$_SESSION['MM_UserGroup'])))) {
  $MM_qsChar = "?";
  $MM_referrer = $_SERVER['PHP_SELF'];
  if (strpos($MM_restrictGoTo, "?")) $MM_qsChar = "&";
  if (isset($QUERY_STRING) && strlen($QUERY_STRING) > 0)
  $MM_referrer .= "?" . $QUERY_STRING;
  $MM_restrictGoTo   =   $MM_restrictGoTo.   $MM_qsChar   .   "accesscheck="   .
urlencode($MM_referrer);
  header("Location: ". $MM_restrictGoTo);
  exit;
}
```

```
?>
<?php
include ("conexion.php");
$link = Conectar();
if ($salir)
{
mysql_close($link);
$dir = "principal.html";
header ("location: ".$dir);
}
if ($modificar)
{
$sql = "UPDATE altas SET nic='$nic', pw='$pw', direccion='$direccion' WHERE
numficha='$numficha';";
mysql_select_db('usuarios', $link);
mysql_query($sql, $link) or die(mysql_error());
}
mysql_select_db('usuarios', $link);
$modificar = mysql_query("SELECT *FROM altas WHERE nic='$MM_Username' ",
$link);
$myrow = mysql_fetch_assoc($modificar);
mysql_num_rows($modificar);
?>
<html>
<head>
<title>index</title>
</head>
<body bgcolor="FFFFFF">
<form method="POST" target="_parent" name="form1">
<table width="798" height="124" border="0" cellpadding="2" cellspacing="2">
</table>
<table width="180" height="42" border="0" align="left">
  <tr>
    <th height="38" scope="row">
        <p>
          <?php
      echo "<font color=´#ffffff´ size='2' face='Arial, Helvetica, sans-
serif'> Bienvenido <br>{$_SESSION['MM_Username']} </font>";
      ?>
          <br>
          <input name="salir" type="submit" value="Salir">
          <br><a           href="administrador.php"           class="style2"
target="_parent"></a><br>
  </tr>
</table>
  <tr>
    <td width="10" valign="top"></td>
    <td height="851" colspan="6" valign="top">
<form method="post" name="form1" action="">
<table align="center" borde="1">

  <tr valign="baseline">
    <td nowrap align="right">Nic o apodo:</td>
    <td><input type="text" name="nic" value="<?php echo $myrow['nic']; ?>"
size="32"></td>
  </tr>
  <tr valign="baseline">
    <td nowrap align="right">Contraseña:</td>
```

```
    <td><input  type="text"  name="pw"  value="<?php  echo  $myrow['pw'];  ?>"
size="32"></td>
  </tr>
  <tr valign="baseline">
    <td nowrap align="right">Direccion:</td>
    <td><input     type="text"     name="direccion"     value="<?php     echo
$myrow['direccion']; ?>" size="32"></td>
  </tr>
  <tr valign="baseline">
    <td nowrap align="right">Numero de la ficha:</td>
    <td><?php echo $myrow['numficha']; ?></td>
  </tr>
  <tr valign="baseline">
    <td     colspan="2"     align="right"     nowrap><div     align="center"><a
href="funadmin.php">OTRO</a>
        <input name="actualizar" type="submit" value="Actualizar">
         <input     type="hidden"     name="numficha"     value="<?php     echo
$myrow['numficha']; ?>">
    </div></td>
  </tr>
</table>
</form>
</body>
</html>
```

A.1.22 CONEXIÓN A LA BASE DE DATOS DE USUARIOS Código el cual realiza la conexión con la Base de Datos de los Usuarios.

```
<?php
function Conectar()
{
   if (!($link=mysql_connect("localhost","root","password")))
   {
      echo "Error conectando a la base de datos.";
      exit();
   }
   if (!mysql_select_db("usuarios",$link))
   {
      echo "Error seleccionando la base de datos.";
      exit();
   }
   return $link;
}
?>
```

A.1.23 Conexión a la Base de Datos para dar de alta problemas, y ver las respuestas de los usuarios.

```
<!DOCTYPE    html    PUBLIC    "-//W3C//DTD    XHTML    1.0    Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Untitled Document</title>
</head>
<body><?
$host = "localhost";
$usuario = "root";
$password = "password";
```

```
$db = "usuarios";
?>
</body>
</html>
```

Bibliografía y Referencias

1. Booch Grady;"El lenguaje Unificado Modificado", Addison Wesley,1999
2. Henry F.Korth,Abraham Silberchatz:"Fundamentos de Base de Datos",Addison Wesley,Madrid 2000
3. Dra Maria J. somodevilla Garcia,"Base de Datos cliente-servidor";:Notas diplomado 2006
4. Rasmus Leordorf"Tutorial PHP"
5. Oros Cabello Juan Carlos"Diseño de Paginas WEB interactivas con JavaScript",AlfaOmega & RA-MA,Madrid,España 1999
6. http://es.wikipedia.org/wiki/John_Forbes_Nash
7. http://es.wikipedia.org/wiki/John_von_Neumann
8. http://www.creangel.com/uml/
9. http://monografias.com/teoriajuegos
10. http://Procesama_Net(Taller PHP Net)
11. http://Desarrollo web/manuals/12/registrophp
12. http://wikipedia.org/wiki/sql
13. http://wikipedia.org/wiki/php
14. www.mysql-hispano.org/
15. http://practicaphpprog.html
16. http://desarrolloweb/taller/php