INSTITUTO TECNOLÓGICO DE MÉRIDA

TESIS

"HECODER, HERRAMIENTA COLABORATIVA PARA LA GENERACIÓN DE DIAGRAMAS ENTIDAD-RELACIÓN"

PARA OPTAR AL GRADO DE:

MAESTRO EN INGENIERÍA

PRESENTA:

I.S.C. DIDIER RENÉ MORENO VÁZQUEZ

ASESOR:

M.C. MARIO RENÁN MORENO SABIDO

MÉRIDA, YUCATÁN, MÉXICO.

04 DE JUNIO DE 2013

"2013. Año de la Lealtad Institucional y Centenario del Ejército Mexicano"

DEPENDENCIA: DIV. DE EST. DE POSG. E INV.
OFICIO N° X/165/2013

MÉRIDA, YUCATÁN A 07 DE MAYO DE 2013

ASUNTO: SE AUTORIZA IMPRESIÓN

C. DIDIER RENÉ MORENO VÁZQUEZ
PASANTE DE MAESTRIA EN INGENIERÍA
P R E S E N T E.

De acuerdo al fallo emitido por su asesor **EL M.C. MARIO RENÁN MORENO SABIDO**, y la comisión revisora integrada por la M.C. Grelty del Socorro Canul Novelo, el M.S.C. Nora Leticia Cuevas Cuevas, y el Dr. José Ramón Atoche Enseñat, considerando que cubre los requisitos establecidos en el Reglamento de Titulación de los Institutos Tecnológicos le autorizamos la impresión de su trabajo profesional con la **TESIS**:

"HECODER, HERRAMIENTA COLABORATIVA PARA LA GENERACIÓN DE DIAGRAMAS ENTIDAD-RELACIÓN"

ATENTAMENTE
IN HOC SIGNO VINCES

M.C RAMIRO ALPIZAR CARRILLO.
JEFE DE LA DIVISION DE ESTUDIOS
DE POSGRADO E INVESTIGACIÓN

S. E. P.
INSTITUTO TECNOLOGICO
DE MERIDA
DIVISION DE ESTUDIOS DE
POSGRADO E INVESTIGACION

RAC/fjaa

DEDICATORIAS

Dedico este trabajo a Dios, a mis padres Dolores Vázquez y Raúl Moreno, a mis hermanos Gladys y Raúl, a mi familia, pero sobre todo al amor de mi vida Soledad por que estuvo conmigo en aquellos momentos en el que el estudio y el trabajo ocuparon mi tiempo y esfuerzo. Gracias por tu amor, ayuda y compresión.

Di a la sabiduría: "Tú eres mi hermana", y a la inteligencia: "Eres de mi sangre." –

Proverbios 7:4

AGRADECIMIENTOS

- A Dios por haberme permitido llegar hasta este punto y haberme dado salud, sabiduría y fortaleza para lograr mis objetivos.
- Agradezco todo el apoyo y cariño brindado por parte de mis padres, hermanos y esposa que siempre me alentaron a continuar con mis estudios y a superarme.
- Al M.C. Mario Renán Moreno Sabido por sus contribuciones a esta tesis, así como por su calidad humana, apoyo y colaboración para la realización de la misma.
- A la M.I.E. Danice Cano Barrón que junto con el Departamento de Ingeniería en Sistemas del Instituto Tecnológico Superior de Motul proporcionaron las bases para el desarrollo del proyecto.
- A mis revisores: M.C. Grelty del Socorro Canul Novelo, M.C. Nora Leticia Cuevas Cuevas, Dr. José Ramón Atoche Enseñat y al M.C. Mario Renán Moreno Sabido, por sus contribuciones en la revisión del presente trabajo.
- Al personal docente perteneciente a la Maestría en Ingeniera del Instituto Tecnológico de Mérida, en especial al Dr. Jesús Sandoval Gío y al Dr. José Ramón Atoche Enseñat que compartieron sus conocimientos y experiencia.
- A mis amigos y compañeros del ITM por los momentos tan gratos que guardo en mi memoria, en especial a Danice, Julián y Ángel por convertirse en mi familia más cercana.
- Y a todas las personas que de una forma u otra colaboraron en la realización de ésta tesis.

RESUMEN

HECODER, es un software educativo que trabaja de forma colaborativa permitiendo a los estudiantes generar diagramas de tipo Entidad – Relación (E-R) de bases de datos en tiempo real, propicia la colaboración, retroalimentación y participación de todos los estudiantes por los que este conformado un equipo de trabajo para lograr la solución de un ejercicio de E-R.

Permite al Maestro administrar tareas, ejercicios, equipos de trabajo y monitorear en tiempo real los avance sobre tareas o ejercicios que estén resolviendo los equipos de trabajo, así como poder interactuar con ellos en diversas formas y espacios de tiempo por medio de chat o la base de datos de conocimientos.

HECODER elimina la barrera de un lugar físico como un aula escolar así como un horario establecido para poder trabajar en equipo y aportar los conocimientos de cada integrante para retroalimentar a todo el equipo.

ABSTRACT

HECODER, is an educational software that works collaboratively allowing students to generate diagrams of type Entity - Relationship (ER) database in real time, allows for collaboration, feedback and participation of all students by which this formed a team to achieve resolution of an exercise of ER.

Allows the teacher manage tasks, exercises, work teams and monitor in real time the progress on tasks or exercises that are solving teams and interact with them in various forms and spaces of time through chat or base knowledge data.

HECODER removes a physical barrier such as a classroom and a schedule to bring teamwork and knowledge of each member to provide feedback to the team.

INDICE DE CONTENIDO

INDICE DE TABLAS

INDICE DE FIGURAS

CAPÍTULO 1: INTRODUCCIÓN

En este capítulo se mencionan los antecedentes relacionados con las Plataformas de Software Colaborativos. De igual forma se plantea la problemática encontrada, así como la propuesta de solución, la cual consiste en el desarrollo de una plataforma colaborativa que permita la generación en tiempo real de diagramas de tipo Entidad – Relación de bases de datos.

Para finalizar este capítulo se establecen los objetivos, así como la justificación, el alcance y las limitaciones del proyecto de software.

CAPÍTULO 1: INTRODUCCIÓN

1.1 Antecedentes

En la actualidad es cada vez menos probable el realizar las tareas diarias sin el uso directo o indirecto de una computadora o aparato tecnológico. Es un hecho que la tecnología constantemente evoluciona, se reinventa y busca siempre la forma más simple y rápida de hacer el trabajo de forma eficaz y eficiente para lograr aprovechar al máximo el tiempo.

Debido a lo anterior, los términos de: trabajo en equipo, colaboración, internet, redes sociales, chat, celulares, tabletas, computadoras, ya son parte del lenguaje coloquial de las personas, por lo que el uso y aprovechamiento de los grandes avances en la tecnología no son exclusivos del sector público o privado, ya que estos prácticamente han permeado a la gran parte de la sociedad en diferentes campos de trabajo o esparcimiento.

Estudios (Carreras, 2005) sustentan que "en los últimos años está cobrando relativa importancia las aplicaciones diseñadas para la colaboración entre personas o grupos de personas... Asi pues emerge el Trabajo Colaborativo Soportado por Computadora (o CSCW, Computer Supported Cooperative Work) con el objetivo de estudiar tanto el desarrollo de las aplicaciones de trabajo como el impacto que provocan en la sociedad".

De forma similar se originan los sistemas de Aprendizaje Colaborativo Soportado por Computadora (o CSCL, Computer Supported Collaborative Learning). "Estos sistemas ofrecen versiones electrónicas de muchas actividades y recursos presentes en las aulas de enseñanza tradicional (presencial). Así, es posible disponer de espacios para trabajo compartido, lecturas y presentaciones online, resultados de evaluaciones y calificaciones, listados de bibliografía, repositorios de ejercicios y materiales, etc. Contando también con herramientas de comunicación síncrona y/o asíncrona como chat, foro y email, todo lo cual da soporte tanto a la comunicación como a la colaboración entre los estudiantes." (Costaguta, 2009)

De igual forma el autor (Costaguta, 2009) señala que el CSCL rápidamente fue adoptado en el ámbito de la educación a distancia por las facilidades expuestas en el párrafo anterior, y porque a través del soporte computacional logra independizar a los estudiantes de las variables tiempo y espacio (estudiantes ubicados en puntos geográficos distantes, e incluso, contribuyendo en momentos diferentes en el tiempo, pueden trabajar colaborativamente).

Enfocando el tema de las Tecnologías de la Información y de la Comunicación (TIC) en el ámbito Académico se puede afirmar que el uso de las TIC ha propiciado un cambio en los procesos y metodologías en que los docentes imparten sus cátedras y también la forma de cómo los educandos adquieren nuevos conocimientos. En este sentido estudios (Madrid, 2007) señalan que "el uso de las TIC en las universidades del mundo ha sido uno de los principales factores de inducción al cambio y adaptación a las nuevas formas de hacer y de pensar iniciadas a partir de los ochenta en los distintos sectores de la sociedad".

De igual forma (Madrid, 2007) afirma que "el uso de las TIC han facilitado a un gran número de estudiantes el acceso a la información, y han modificado significativamente el proceso de enseñanza- aprendizaje." Por su parte (Graells, 2011) sustenta que "como en los demás ámbitos de actividad humana, las TIC se convierten en un instrumento cada vez más indispensable en las instituciones educativas, en donde puede realizar múltiples funcionalidades:

- Fuente de información (hipermedia).
- Canal de comunicación interpersonal y para el trabajo colaborativo y para el intercambio de información e ideas (e-mail, foros telemáticos).
- Medio de expresión y para la creación (procesadores de textos y gráficos, editores de páginas web y presentaciones multimedia, cámara de vídeo).
- Instrumento cognitivo y para procesar la información: hojas de cálculo, gestores de bases de datos.
- Instrumento para la gestión, ya que automatizan diversos trabajos de la gestión de los centros: secretaría, acción tutorial, asistencias, bibliotecas.

- Recurso interactivo para el aprendizaje. Los materiales didácticos multimedia informan, entrenan, simulan guían aprendizajes, motivan.

- Medio lúdico y para el desarrollo psicomotor y cognitivo.

Hablando de las TIC como instrumento cognitivo y para el aprendizaje distribuido y colaborativo, (Graells, 2011) señala que cuando las TIC se utilizan como complemento de las clases presenciales (o como espacio virtual para el aprendizaje, como pasa en los cursos on-line) se puede considerar que se adentra en el ámbito del aprendizaje distribuido, planteamiento de la educación centrado en el estudiante que, con la ayuda de las TIC posibilita el desarrollo de actividades e interacción tanto en tiempo real como asíncronas. Los estudiantes utilizan las TIC cuando quieren y donde quieren (máxima flexibilidad) para acceder a la información, para comunicarse, para debatir temas entre ellos o con el profesor, para preguntar, para compartir e intercambiar información.

Es importante mencionar que un pilar en la evolución de las TIC es la Ingeniería de Software (IS), la cual (Sommerville, 2005) define como "... una disciplina de la ingeniería cuya meta es el desarrollo costeable de sistemas de software. Éste es abstracto e intangible. No está restringido por materiales, o gobernado por leyes físicas o por procesos de manufactura.".

1.2 Planteamiento Del Problema

Hoy en día hay una gran variedad de aplicativos que permiten realizar diagramas vectoriales para diversas necesidades, de igual manera ofrecen la posibilidad de poder compartirlos con un grupo de trabajo utilizando diversos canales de comunicación. Sin embargo estas plataformas en su mayoría web, están orientados a la compartición del documento en sí, y sobre todo a la disponibilidad en todo momento de la información generada.

En el 2011 surge en el Departamento de Ingeniería en Sistemas Computacionales del Instituto Tecnológico de Motul la necesidad de contar con una plataforma colaborativa que

permita como primera instancia la resolución de forma colaborativa de ejercicios de diagramas de Entidad - Relación de bases de datos, en diferentes variables de tiempo y espacio.

Así mismo debe de permitir la administración de los alumnos por grupos y asu vez por equipos de trabajo. De igual manera debe de poder administrarse los ejercicios que el o los maestros suban a la plataforma, los cuales a la hora de ser marcados como tareas tendrán fechas de entrega. El aplicativo debe de contar con un canal de comunicación que permita a los integrantes el poder intercambiar sus conocimientos y puntos de vista en tiempo real sobre el ejercicio que se esté realizando en el momento, así como una base de conocimientos y preguntas frecuentes. También debe de permitir al maestro la posibilidad de retroalimentar a los alumnos sobre las tareas que se marquen.

El Departamento de Ingeniería en Sistemas Computacionales del Instituto Tecnológico de Motul requiere de dicha plataforma que cumpla con los requisitos anteriores poder emplearla en sus aulas y proporcionar una nueva forma de generación de conocimientos entre sus alumnos y maestros.

1.3 Objetivos

1.3.1 General

Desarrollar una Plataforma Colaborativa que permita la Generación en tiempo real de Diagramas de tipo Entidad – Relación de bases de datos.

1.3.2 Específicos

1. Obtener los requerimientos de funcionalidad de la herramienta.
2. Analizar los requerimientos.
3. Realizar el diseño de clases, utilizando el lenguaje unificado de modelado (UML)
4. Diseñar la capa de Datos en C#.

5. Diseñar la capa de Negocios u Objetos en C#.
6. Diseñar las interfaces gráficas de la aplicación en C#.
7. Integrar las 3 capas de la aplicación.
8. Realizar pruebas de aplicación.
9. Analizar los resultados de las pruebas.

1.4 Justificación

De acuerdo con lo investigado hasta el momento en las diversas fuentes relacionadas con las plataformas colaborativas orientadas al proceso educativo, se puede decir que no existe una herramienta de software para satisfacer todas las necesidades que tiene El Departamento de Ingeniería en Sistemas Computacionales del Instituto Tecnológico de Motul, por lo que se justifica el desarrollo de una plataforma colaborativa que genere en tiempo real diagramas de Entidad-Relación, y que permita a los alumnos trabajar en la resolución de problemas de diseño de bases de datos dentro de equipos de trabajo permitiéndoles a los usuarios diferentes esquemas de colaboración en espacio y tiempo.

1.5 Delimitación

En este apartado se conocerán los alcances y limitantes del proyecto.

1.5.1 Alcances

- La plataforma colaborativa permitirá administrar a las personas que utilizan la aplicación, en tres clases de usuarios diferentes: Administrador General, Maestros y Alumnos.
- El software permitirá al maestro administrar cuales y cuantos ejercicios están disponibles en la plataforma.
- Los alumnos deberán poder ser agrupados en grupos de estudio y asu vez en equipos de trabajo.

- Los ejercicios deberán poder ser agrupados en tareas que serán asignados a equipos de trabajo en un determinado orden de ejecución y estas deberán de tener un orden de ejecución de ejercicios por equipo y una fecha de finalización.
- El maestro podrá en cualquier momento revisar avances de ejercicios en cualquier tarea asignada a cualquier equipo de su grupo de trabajo.
- El maestro podrá realizar una retroalimentación escrita en la versión final de cada ejercicio de cada tarea asignada.
- Los diagramas resultantes de los ejercicios podrán ser impresos.
- El software puede ejecutarse en una red local o por internet.

1.5.2 Limitaciones

- El único manejador de base de datos soportado en la plataforma será MySQL version 5.0.
- La plataforma es de tipo Escritorio.
- Es necesario instalar el software en cada una de las computadoras en la cuales se desee tener acceso a la plataforma.
- Es necesario contar con una conexión de red a la base de datos del servidor.
- El software no determinará si la solución del ejercicio es correcto o incorrecto.

CAPÍTULO 2: MARCO TEÓRICO

En este capítulo se presentan los fundamentos teóricos utilizados para el desarrollo del proyecto de software. Se muestra la metodología utilizada, así como también las diferentes herramientas de software usadas en el desarrollo (lenguaje de programación, sistema gestor de bases de datos, entre otros).

También se comenta acerca del aprendizaje colaborativo y su importancia para el proyecto. Por último, se incluye el estado del arte en el cual se mencionan algunas herramientas colaborativas para ver las características, las similitudes y diferencias que estas herramientas tienen con el software desarrollado.

CAPÍTULO 2: MARCO TEÓRICO

2.1 Aprendizaje Colaborativo

El aprendizaje colaborativo busca propiciar espacios en los cuales se dé el desarrollo de habilidades individuales y grupales a partir de la discusión entre los estudiantes al momento de explorar nuevos conceptos, siendo cada quien responsable de su propio aprendizaje. Se busca que estos ambientes sean ricos en posibilidades y más que organizadores de la información propicien el crecimiento del grupo. Diferentes teorías del aprendizaje encuentran aplicación en los ambientes colaborativos; entre éstas, los enfoques de Piaget y de Vygotsky basados en la interacción social (Lucero, 2003).

Muhlenbrock considera que el aprendizaje colaborativo es el compromiso mutuo establecido entre un grupo de personas, que se agrupan en un esfuerzo coordinado para dar respuestas a una tarea. Para él, este tipo de organización permite entender los procesos que se gestan al trabajar entre pares (Murcia, 2004).

Dillenbourg entre tanto, afirma que la clave para entender el aprendizaje colaborativo es reconocer las relaciones que se establecen entre la situación que se plantea, las interacciones que emergen y en consecuencia, los procesos y efectos que se generan en ella (Dillenbourg, 1999). Estos cuatro elementos que se describen a continuación se constituyen en los elementos clave que deben tenerse en cuenta al momento de evaluar un contexto de aprendizaje colaborativo (Dillenbourg, 1999):

- La situación, establecida a partir del grado de simetría de las acciones, el conocimiento y el estatus de los participantes para dar resolución a la tarea en forma conjunta.

- Las interacciones, enmarcadas dentro de la situación colaborativa que se ha establecido. Éstas pueden ser interactivas, sincrónicas y negociables. Dichas interacciones influyen en los procesos cognitivos de cada uno de los participantes.

- Los mecanismos de aprendizaje, obtenidos a partir de la interacción entre pares. Éstos pueden ser aquellos que operan en el caso de la cognición individual, como aquellos que operan a nivel grupal como la apropiación, el mutuo modelamiento y la internalización.

- Los efectos del aprendizaje colaborativo, generalmente medidos a partir de un pre-test o post-test con los cuales se pretende obtener una medición de las ganancias que han obtenido los estudiantes.

En definitiva, la clave para entender los procesos que se dan al interior del aprendizaje colaborativo son las relaciones bidireccionales que se establecen entre estos cuatro elementos. En la modalidad de aprendizaje colaborativo haciendo uso de herramientas tecnológicas (Computer-Supported Collaborative Learning - CSCL) se presentan los mismos elementos, pero la dinámica organizacional varía por el uso de una herramienta (computador) que media el aprendizaje colaborativo.

Al hablar de CSCL es importante hacer la diferencia entre el concepto del trabajo colaborativo y el aprendizaje colaborativo, ya que pueden causar confusión al momento de analizar este trabajo, a pesar de que ambas comparten las interacciones como base del éxito grupal. El primero (Computer-Supported Collaborative Learning) o CSCW, es usado a nivel organizacional en donde la división de las labores está definida de antemano y le facilitan al individuo alcanzar aprendizajes relacionados con los objetivos de la organización (Lucero, 2003). Por el contrario el CSCL se da en contextos de aprendizaje colaborativo escolar donde el desarrollo personal y grupal se constituye en el vector de su funcionamiento, donde la división de las labores no está predeterminada (Murcia, 2004).

A continuación se enlistan algunas de las ventajas que genera el aprendizaje colaborativo:

Con respecto a la ejecución de tareas grupales:

- Promueve el logro de objetivos cualitativamente más ricos en contenido, pues reúne propuestas y soluciones de varias personas del grupo.

- Se valora el conocimiento de los demás miembros del grupo.
- Incentiva el desarrollo del pensamiento crítico y la apertura mental.
- Permite conocer diferentes temas y adquirir nueva información.
- Fortalece el sentimiento de solidaridad y respeto mutuo, basado en los resultados del trabajo en grupo.

Aumenta:

- El aprendizaje de cada uno debido a que se enriquece la experiencia de aprender.
- La motivación por el trabajo individual y grupal.
- El compromiso de cada uno con todos.
- La cercanía y la apertura.
- Las relaciones interpersonales.
- La satisfacción por el propio trabajo.
- Las habilidades sociales, interacción y comunicación efectivas.
- La seguridad en sí mismo.
- La autoestima y la integración grupal.

Disminuye:

- Los sentimientos de aislamiento.
- El temor a la crítica y a la retroalimentación.

2.2 Fundamentos De Ingeniería De Software

Sommerville define a la Ingeniería de Software con las siguientes líneas:

"La Ingeniería del Software es una disciplina de la Ingeniería que comprende todos los aspectos de la producción de software desde las etapas iniciales de la especificación del sistema, hasta el momento de que este se utiliza" (Sommerville, 2005, p. 6).

En la definición anterior que presenta el autor existen dos palabras clave:

Ingeniería:

La ingeniería es el estudio y la aplicación de las distintas ramas de la tecnología. La actividad del ingeniero supone la concreción de una idea en la realidad. Esto quiere decir que, a través de técnicas, diseños y modelos, y con el conocimiento proveniente de las ciencias, la ingeniería puede resolver problemas y satisfacer necesidades humanas. La ingeniería también supone la aplicación de la inventiva y del ingenio para desarrollar una cierta actividad. Esto, por supuesto, no implica que no se utilice el método científico para llevar a cabo los planes (Otazu, 2007).

Software:

El software no son solo programas, sino todos los documentos asociados y la configuración de datos que se necesitan para hacer que estos programas operen de manera correcta. Un sistema de software consiste en diversos programas independientes, archivos de configuración que se utilizan para ejecutar estos programas, un sistema de documentación que describe la estructura del sistema, la documentación para el usuario que explica cómo utilizar el sistema y sitios web que permitan a los usuarios descargar la información de productos recientes (Sommerville, 2005).

El software de computadora es el producto que los ingenieros de software construyen y después mantienen en el largo plazo. El software se forma con (1) las instrucciones (programas de computadora) que al ejecutar se proporcionan las características, funciones y el grado de desempeño deseados; (2) las estructuras de datos que permiten que los programas

manipulen información de manera adecuada; y (3) los documentos que describen la operación y uso de los programas (Pressman, 2005).

Otras definiciones de Ingeniería del Software también son válidas, a continuación se enlistan algunas de ellas:

- La Ingeniería de Software es una disciplina que integra el proceso, los métodos, y las herramientas para el desarrollo de software de computadora (Pressman, 2005).
- El IEEE ha elaborado una definición más comprensible al establecer:

"Aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento del software, es decir, la aplicación de la ingeniería al software. 2) El estudio de enfoque como en 1)" (IEEE, 1993).

- La Ingeniería de Software es más que una disciplina o un cuerpo de conocimiento, la ingeniería es un verbo, una palabra de acción, una manera de abordar un problema (Pressman, 2005, p. 23).

Entre las principales áreas de estudio y/o investigación dela Ingeniería de Software se encuentran las siguientes:

- Métodos y Metodologías de Desarrollo de Software
- Procesos de Desarrollo de Software
- Gestión de Proyectos de Software
- Medición y Estimación de Software
- Ingeniería de Requisitos / Requerimientos
- Ingeniería de Software Empírica
- Gestión de Riesgos
- Usabilidad de Software

- Evaluación de Software
- Métricas de Software
- Calidad de Software
- Métodos Formales
- Ingeniería Web

2.3 Lenguaje Unificado De Modelado: UML

El Lenguaje de Modelado Unificado (UML) es un lenguaje estándar para la escritura de proyectos de software. El UML puede ser usado para visualizar, especificar, construir y documentar los componentes de un sistema de software extenso.

El UML es apropiado para una gran variedad de sistemas de modelado de sistemas de información de empresas para aplicaciones distribuidas basadas en Web aún más para sistemas empotrados de tiempo real. Este es un lenguaje muy expresivo, que abarca todos los panoramas necesarios para desarrollar y estructurar tales sistemas. Para aprender a usar UML adecuadamente se requiere aprender tres elementos principalmente: los bloques de construcción básicos, las reglas que dictan como esos bloques deben ser relacionados y algunos mecanismos que aplica todo el tiempo el lenguaje (Oktaba, 1999).

UML es un proceso independiente que óptimamente debe ser usado en un proceso que es un anejador de caso de uso, con arquitectura central, iterativa e incremental. En términos de estos conceptos, un sistema representa las cosas (u objetos) que se están desarrollando, visto desde diferentes perspectivas por diferentes modelos, con las vistas presentadas en forma de diagramas.

Según Oktaba normalmente, se verán las partes estáticas del sistema usando uno de los cuatro diagramas siguientes:

- Diagramas de clases
- Diagramas de objetos
- Diagramas de componentes
- Diagramas de implementación

Y para ver las partes dinámicas de un sistema, Oktaba afirma que se usarán los siguientes diagramas:

- Diagrama de casos de uso
- Diagrama de secuencia
- Diagrama de colaboración
- Diagrama de estados
- Diagrama de actividades

2.3.1 Casos De Uso

Los casos de uso se aplican para capturar el comportamiento propuesto del sistema que se está desarrollando, sin especificar como se implementó. Proporcionan una manera para que los desarrolladores establezcan un entendimiento común con los expertos del dominio y los usuarios finales. Sirven para ayudar a validar la arquitectura y verificar como el sistema evoluciona durante el desarrollo. Como se implementa el proyecto, los casos de uso se llevan a cabo mediante colaboraciones, en donde sus elementos trabajan para realizarlos (Oktaba, 1999). En UML, tales comportamientos se modelan como casos de uso que posiblemente se especifiquen independientemente de su realización. Un caso de uso es una descripción de un conjunto de secuencias de acciones, incluyendo variantes, que un sistema ejecuta, y así, producir un resultado importante para un actor.

Un caso de uso posiblemente tenga variantes. En todos los sistemas se encontrarán aquellos que son versiones especializadas de anteriores, casos de uso que son incluidos como parte de otros, y los que extienden el comportamiento a partir de otros. También se puede factorizar el comportamiento común y reutilizable de un conjunto de casos de uso organizándolos en tres clases de relaciones. Gráficamente, se representa mediante una elipse (Ver Figura 2.1).

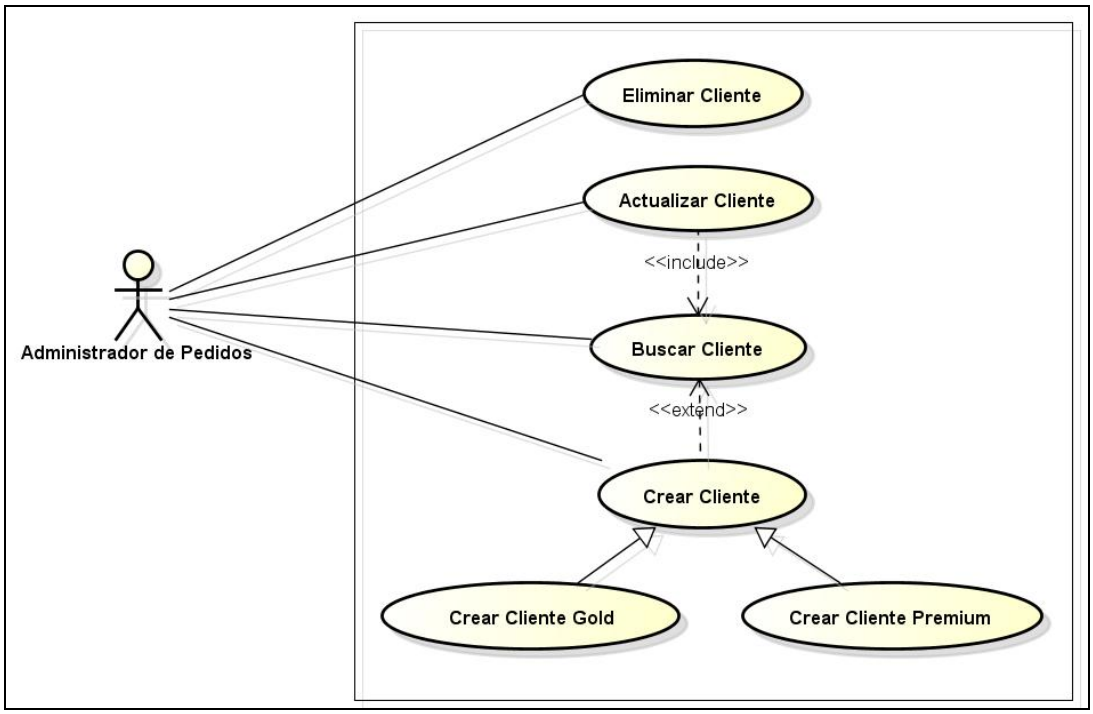

Figura 2.1 Ejemplo de un Caso de Uso

2.3.2 Clases

Las clases son el bloque constructor más importante de cualquier sistema orientado a objetos; es una descripción de un conjunto de objetos que comparten los mismos atributos, operaciones, relaciones y semántica, además, implementa una o más interfaces. Las clases se usan para representar el vocabulario del sistema que se está desarrollando, deben incluir abstracciones que forman parte del dominio del problema; también se usan para representar elementos de software, de hardware y hasta aquellos que son puramente conceptuales.

Las clases bien estructuradas son sencillas y forman parte de una distribución balanceada de responsabilidades dentro del sistema.

El modelado de un sistema involucra identificar los elementos que son importantes desde tu punto de vista, los cuales integrarán el vocabulario del sistema que estás modelando y cada uno de ellos será distinto de los otros y contará con un conjunto de propiedades. En UML todos estos elementos son modelados como clases. Una clase no es un objeto individual, más bien representa todo un conjunto de objetos.

En software, muchos lenguajes de programación soportan directamente el concepto de clase, lo cual es excelente, porque eso significa que las abstracciones que creas pueden, en muchas ocasiones, ser mapeadas directamente a uno de estos lenguajes, aún si estas son de elementos que no son software, por ejemplo un cliente, un negocio o una conversación.

UML proporciona una representación gráfica de una clase, como se pude apreciar en la Figura 2.2.

Esta notación permite visualizar una abstracción independiente de cualquier lenguaje de programación específico y de una manera que permite enfatizar sus partes más importantes: su nombre, atributos y operaciones.

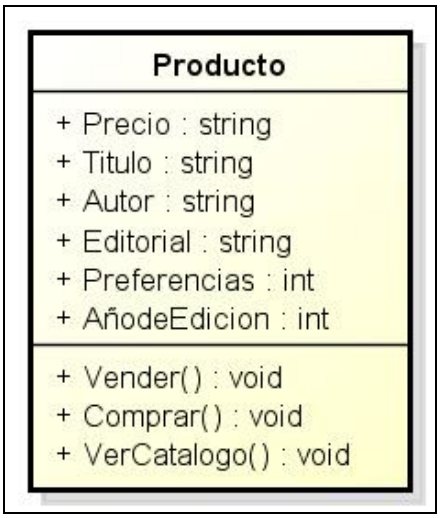

Figura 2.2 Ejemplo de una Clase.

2.3.3 Secuencias

Enfatiza el orden de tiempo de los mensajes. Gráficamente, este diagrama es una tabla que muestra objetos ordenados junto al eje de las X y los mensajes, son ordenados en incremento de tiempo junto al eje de las Y, como puede observarse en la Figura 2.3:

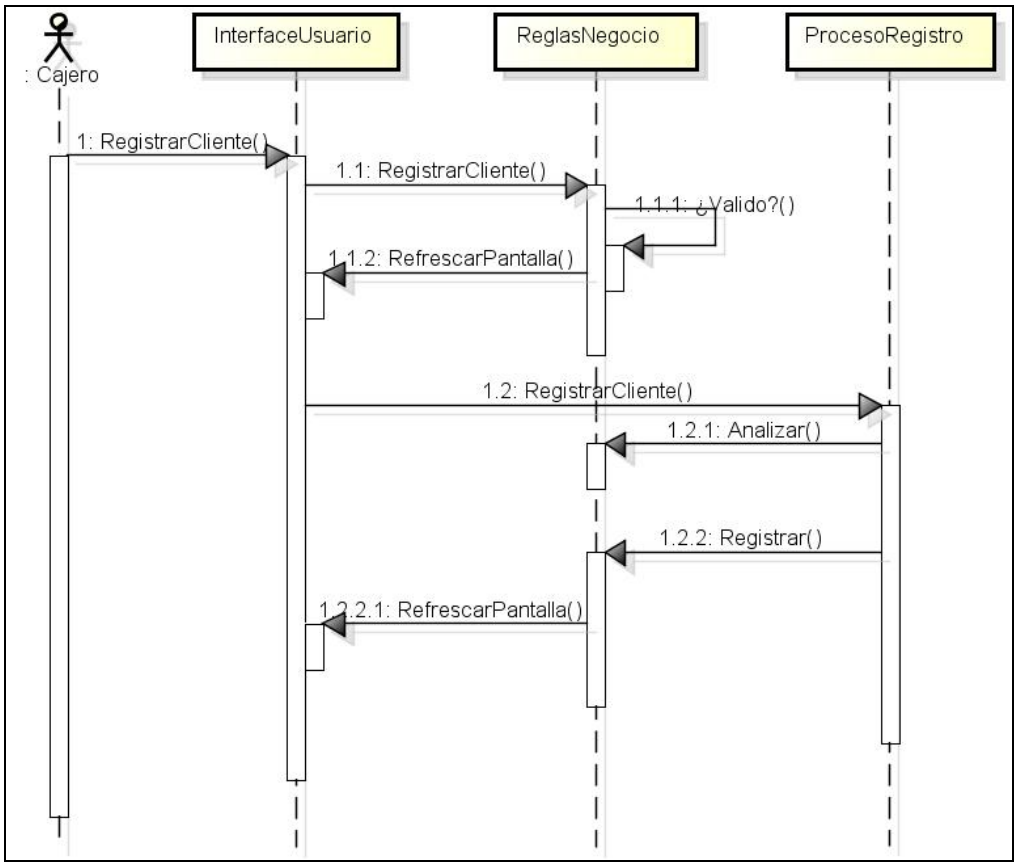

Figura 2.3 Ejemplo de un Diagrama de Secuencia.

Para hacer un diagrama de secuencia es recomendable tomar en cuenta los siguientes puntos:

- Colocar los objetos que participan en la interacción en la parte de arriba del diagrama, a través del eje de las X.

- Colocar los objetos que inician la interacción a la izquierda y los objetos más subordinados a la derecha.
- Colocar los mensajes que estos objetos envían y reciben junto al eje de las Y, en orden de incremento de tiempo de arriba hacia abajo.
- Existe la línea de vida del objeto, que representa la existencia de un objeto en un período de tiempo.
- Existe el foco de control, que muestra el período de tiempo en el que el objeto se encuentra representando una acción

2.3.4 Actividades

Los diagramas de actividades permiten describir como un sistema implementa su funcionalidad, modelan el comportamiento dinámico de un procedimiento, transacción o caso de uso haciendo énfasis en el proceso que se lleva a cabo. De igual forma son uno de los elementos de modelado que son mejor comprendidos por todos, ya que son herederos directos de los diagramas de flujo (Moyano)

Los diagramas de actividad son más expresivos que los diagramas de flujo. También heredan características de:

- Los diagramas de estados.
- Los diagramas de flujo de datos.

En un diagrama de actividad se pueden representar los objetos de datos que se generan, se consumen o se intercambian en un proceso y que son relevantes para su descripción. En la Figura 2.4 puede observarse un diagrama de actividades.

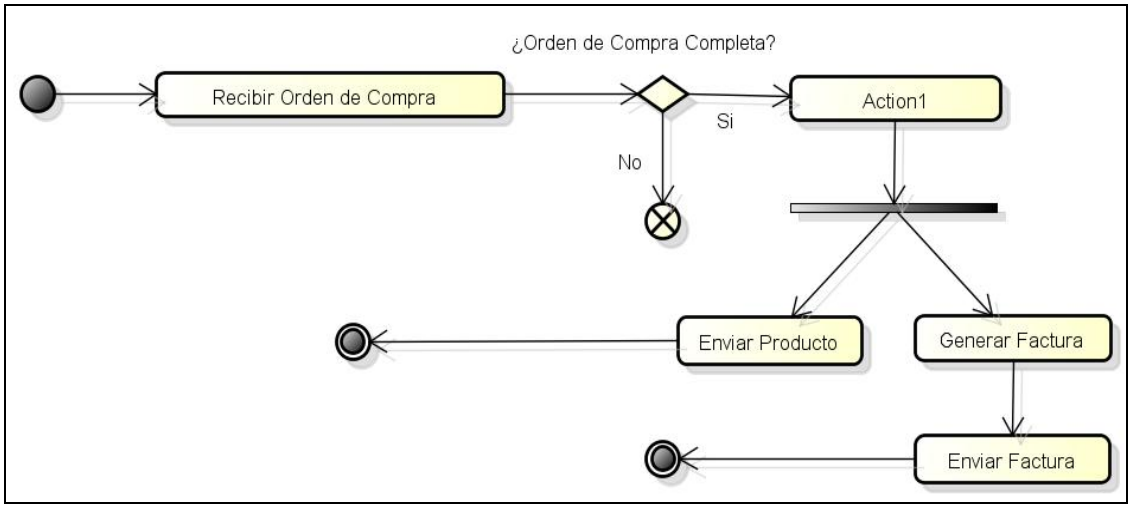

Figura 2.4 Ejemplo de un Diagrama de Actividades.

2.4 Arquitectura Del Software

En los inicios de la informática, la programación se consideraba un arte, debido a la dificultad que entrañaba para la mayoría de los mortales, pero con el tiempo se han ido desarrollando metodologías y fórmulas o trucos para conseguir los propósitos que se deseen. A todas estas técnicas se les ha llama Arquitectura del Software. Larman menciona la siguiente definición de Arquitectura del Software:

> "Una arquitectura es el conjunto de decisiones significativas sobre la organización del sistema de software, la selección de los elementos estructurales y sus interfaces, con los que se compone el sistema, junto con su comportamiento tal como se especifica en las colaboraciones entre esos, la composición de esos elementos estructurales y de comportamiento en subsistemas progresivamente más amplios, y el estilo de arquitectura que guía esta organización" (Larman, 2003).

La arquitectura de software debe de establecer los fundamentos para que analistas, diseñadores, programadores, etc. trabajen en una línea común que permita alcanzar los objetivos y necesidades del sistema de información.

Una arquitectura de software se selecciona y diseña con base en unos objetivos y restricciones. Los objetivos son aquellos prefijados para el sistema de información, pero no solamente los de tipo funcional, también otros objetivos como la mantenibilidad, auditabilidad, flexibilidad e interacción con otros sistemas de información.

Las restricciones son aquellas limitaciones derivadas de las tecnologías disponibles para implementar sistemas de información. Unas arquitecturas son más recomendables de implementar con ciertas tecnologías mientras que otras tecnologías no son aptas para determinadas arquitecturas. Por ejemplo, no es viable emplear una arquitectura software de tres capas para implementar sistemas en tiempo real.

2.4.1 Programación En 3 Capas

La programación por capas es un estilo de programación en la que el objetivo primordial es la separación de la lógica de negocios de la lógica de diseño, un ejemplo básico de esto es separar la capa de datos de la capa de presentación al usuario (Ver Figura 2.5) (Partners, 2009).

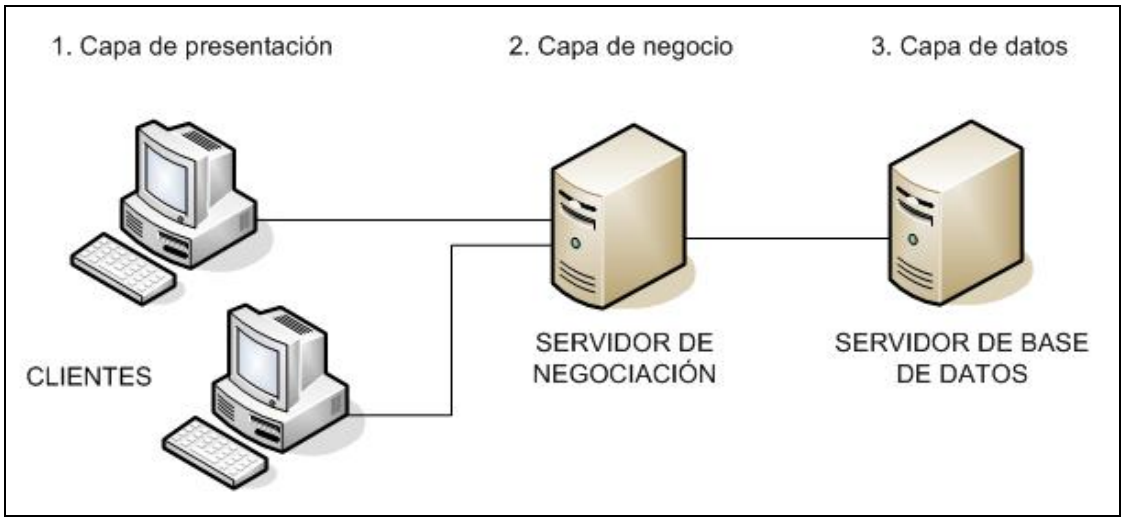

Figura 2.5 El modelo de aplicación 3 Capas.

La ventaja principal de este estilo, es que el desarrollo se puede llevar a cabo en varios niveles y en caso de algún cambio sólo se ataca al nivel requerido sin tener que revisar entre código mezclado. Un buen ejemplo de este método de programación seráa: Modelo de interconexión de sistemas abiertos. Además permite distribuir el trabajo de creación de una aplicación por niveles, de este modo, cada grupo de trabajo está totalmente abstraído del resto de niveles, simplemente es necesario conocer la API que existe entre niveles.

En el diseño de sistemas informáticos actual se suelen usar las arquitecturas multinivel o Programación por capas. En dichas arquitecturas a cada nivel se le confía una misión simple, lo que permite el diseño de arquitecturas escalables (que pueden ampliarse con facilidad en caso de que las necesidades aumenten). El diseño más en boga actualmente es el diseño en tres niveles (o en tres capas).

Capa de Presentación:

Es la que ve el usuario, presenta el sistema al usuario, le comunica la información y captura la información del usuario dando un mínimo de proceso (realiza un filtrado previo para comprobar que no hay errores de formato). Esta capa se comunica únicamente con la capa de negocio.

Capa de Negocio:

Es donde residen los programas que se ejecutan, recibiendo las peticiones del usuario y enviando las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica del negocio) debido a que es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos para almacenar o recuperar datos de él.

Capa de Datos:

Es donde residen los datos. Está formada por uno o más gestor de bases de datos que realiza todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

Todas estas capas pueden residir en un único ordenador (no sería lo normal), si bien lo más usual es que haya una multitud de ordenadores donde reside la capa de presentación (son los clientes de la arquitectura cliente/servidor). Las capas de negocio y de datos pueden residir en el mismo ordenador, y si el crecimiento de las necesidades lo aconseja se pueden separar en dos o más ordenadores. Así, si el tamaño o complejidad de la base de datos aumenta, se puede separar en varios ordenadores los cuales recibirán las peticiones del ordenador en que resida la capa de negocio.

Si por el contrario fuese la complejidad en la capa de negocio lo que obligase a la separación, esta capa de negocio podría residir en uno o más ordenadores que realizarían solicitudes a una única base de datos. En sistemas muy complejos se llega a tener una serie de ordenadores sobre los cuales corre la capa de datos, y otra serie de ordenadores sobre los cuales corre la base de datos.

En una arquitectura de tres niveles, los términos "capas" y "niveles" no significan lo mismo ni son similares. El término "capa" hace referencia a la forma como una solución es segmentada desde el punto de vista lógico: Presentación/ Lógica de Negocio/ Datos. En cambio, el término "nivel", corresponde a la forma en que las capas lógicas se encuentran distribuidas de forma física. Por ejemplo:

- Una solución de tres capas (presentación, lógica, datos) que residen en un solo ordenador (Presentación + lógica + datos). Se dice, que la arquitectura de la solución es de tres capas y un nivel.
- Una solución de tres capas (presentación, lógica, datos) que residen en dos ordenadores (presentación + lógica, lógica + datos). Se dice que la arquitectura de la solución es de tres capas y dos niveles.
- Una solución de tres capas (presentación, lógica, datos) que residen en tres ordenadores (presentación, lógica, datos). La arquitectura que la define es: solución de tres capas y tres niveles.

2.5 Herramientas de Software para el Desarrollo de la Aplicación

Para el desarrollo del proyecto se utilizaron diversas herramientas de software las cuales se mencionan a continuación y se describen en los apartados siguientes:

- Lenguaje de programación (Microsoft® C#)
- Sistema gestor de bases de datos (Mysql)
- Entorno de Desarrollo Integrado (Microsoft Visual Studio® 2010)
- Administrador de bases de datos (PhpMyAdmin)
- Aplicación servidor Apache
- Componentes dee stilos gráficos para Microsoft Visual Studio® DevComponents®
- SDK Microsoft Office Visio 2007®

Los únicos software que requieren licencia para su uso son: Microsoft Visual Studio 2010® y DevComponents®, todos los demás son software libre.

2.5.1 Microsoft® C#

C# (léase, en inglés C sharp) es un lenguaje de programación que permite el desarrollo de aplicaciones para Internet, para móviles y aplicaciones de propósito general. Inicialmente se desarrolló para programar en la plataforma .NET, pero dadas las características de ésta y la estandarización que se ha hecho de su estructura por parte de las principales entidades de estándares internacionales, se han desarrollado otras plataformas que cumplen con dicha estructura y por lo tanto C# puede ser utilizado como lenguaje de programación en ellas (Pedro, 2011).

El lenguaje C# es orientado a objetos y se ha creado basándose en la estructura de C y C++, especialmente su sintaxis y potencia, y adoptando el estilo y metodología de la programación en Visual Basic. Sin embargo es importante aclarar que C# no es el resultado de la evolución directa de ninguno de estos lenguajes, sino que ha sido creado desde cero, para programar sobre la plataforma .NET, la cual en esencia es una librería de clases que contienen o encapsulan una gran cantidad de funciones que trabajan sobre el sistema operativo.

La característica fundamental de este aspecto, es que dichas clases tienen una estructura común para todos los lenguajes que trabajen sobre esta plataforma. Esto trae como consecuencia que una clase que sea programada en C#, podrá ser heredada o utilizada en cualquier lenguaje de la plataforma, como pueden ser Visual Basic .NET o JScript, para comenzar.

Los programas de C# se ejecutan en .NET Framework, el cual le permite acceder a una infraestructura dotada con lenguajes de programación como C#, Visual Basic .NET, C++ y JScript, y con la posibilidad de acceder a infinidad de servicios útiles para desarrollar cualquier tipo de aplicación.

El proceso de compilación de C# es simple en comparación con el de C y C++, y es más flexible que en Java. No hay archivos de encabezado independientes, ni se requiere que los métodos y los tipos se declaren en un orden determinado. Un archivo de código fuente de C# puede definir cualquier número de clases, estructuras, interfaces y eventos.

2.5.2 Microsoft® Visual Studio 2010

Es una de las más poderosas herramientas para desarrollo construida por la humanidad y es fácil concluirlo si se analiza todo lo que ofrece: Desarrollo de aplicaciones de Consola, de Windows Forms, WPF, Silverlight, Web Sites, Web Applications, Web Services, Windows Services, Librerías de Clase, Windows Store Apps, Cloud Computing, Windows Phone, XBOX, DirectX, Office, Reporting Services, Sharepoint, Workflow y Lightswitch.

Permite el desarrollo nativo y administrado para todas las plataformas soportadas por Microsoft Windows, Windows Mobile, Windows CE, Windows Phone, .NET Famework, .NET Compact Framework, Microsoft Silverlight y WinRT.

El poderoso editor de código soporta IntelliSense y code refactoring para un desarrollo veloz y seguro. Además el debugger integrado puede funcionar a nivel de código y a nivel de máquina. Cuenta con soporte a diversos lenguajes: C/C++, VB.NET, C#, F#, motores de visualización web como HTML Simple, ASP.NET, ASP.NET MVC, y Razor y a través de instalaciones adicionales, M, Python, Ruby y muchos más. Además soporte a HTML, XHTML, HTML5, XML/XSLT, JavaScript, JSON y CSS3 (WarNov, 2012).

Por si fuera poco, es totalmente extensible, con más de 3300 extensiones y plugins disponibles para descargar en la galería online; de manera que cada vez más y más funcionalidades se agregan a este micro mundo de desarrollo. Estos plugins permiten que en cualquier momento se pueda descargar componentes con código pre-escrito, utilidades, herramientas y acceso por ejemplo a plataformas Open Source independientes como GitHub para administrar código fuente.

WarNov (WarNov, 2012) hace énfasis en que un servidor que se adapte al IDE puede perfectamente hacer una planeación y seguimiento de proyectos, así como administrar el control de código fuente, de ítems de trabajo y de evolución del proyecto, todo con una variedad de metodologías que incluyen tanto las ágiles como las convencionales.

2.5.3 PhpMyAdmin

PhpMyAdmin es una herramienta de software libre escrito en PHP, la intención de manejar la administración de MySQL a través de la World Wide Web. PhpMyAdmin es compatible con una amplia gama de operaciones con MySQL. Las operaciones más utilizadas

son compatibles con la interfaz de usuario (bases de datos de gestión, tablas, campos, relaciones, índices, usuarios, permisos, etc), mientras que usted todavía tiene la capacidad de ejecutar cualquier sentencia SQL directamente (PhpMyAdmin, 2008).

PhpMyAdmin viene con una amplia gama de documentación y los usuarios pueden actualizar nuestras páginas wiki para compartir ideas y HOWTOs para diversas operaciones. El equipo de PhpMyAdmin intentará ayudar si surge cualquier problema, se puede usar una variedad de canales de soporte para obtener ayuda. Para facilitar el uso de una amplia gama de personas, PhpMyAdmin está siendo traducido a 69 idiomas y es compatible con los dos idiomas LTR y RTL.

PhpMyAdmin ha ganado varios premios. Entre otros, fue elegida como la mejor aplicación PHP en varios premios y ha ganado todos los años los Premios de la Comunidad SourceForge.net Choice como "Mejor herramienta o utilidad para los administradores de sistemas". PhpMyAdmin es un proyecto de catorce años de edad, con una base de código estable y flexible. La versión de PhpMyAdmin que se decidió trabajar fue la 2.8.2.

Según documentación oficial (Team, 2012) afirma que PhpMyAdmin puede:

- Visualizar y borrar bases de datos, tablas, vistas, campos e índices
- Mostrar múltiples resultados a través de procedimientos almacenados o consultas
- Crear, copiar, borrar, renombrar y alterar bases de datos, tablas, campos e índices
- Realizar labores de mantenimiento de servidor, bases de datos y tablas, dando consejos acerca de la configuración del servidor
- Ejecutar, editar y marcar cualquier expresión SQL, incluyendo consultas en lote
- Carga tablas con el contenido de ficheros de texto
- Crea y lee volcados de tablas
- Exporta datos a varios formatos: CSV, XML, PDF, ISO/IEC 26300 - «OpenDocument Text and Spreadsheet», Microsoft Word 2000 y LATEX

- Importar datos y estructuras MySQL de planillas OpenDocument, así como también archivos XML, CSV y SQL
- Administrar múltiples servidores
- Gestionar privilegios y usuarios de MySQL
- Comprobar la integridad referencial en las tablas MyISAM
- Mediante Query-by-example (QBE), crear consultas complejas conectando automáticamente las tablas necesarias
- Crear gráficos PDF del diseño de su base de datos
- Buscar globalmente o solamente en una parte de una base de datos
- Transformar los datos almacenados a cualquier formato usando un conjunto de funciones predefinidas, como mostrar objetos binarios (BLOBs) como imágenes o enlaces de descarga
- Visualizar cambios en bases de datos, tablas y vistas
- Capacidad de trabajar con tablas InnoDB y claves foráneas
- Capacidad de utilizar mysqli, la extensión MySQL mejorada
- Crear, editar, ejecutar y eliminar funciones y procedimientos almacenados («stored procedures»)
- Crear, editar, exportar y eliminar eventos y disparadores
- Comunicarse en 62 idiomas distintos

2.5.4 MySQL

MySQL, el sistema de gestión de bases de datos SQL Open Source más popular, lo desarrolla, distribuye y soporta MySQL AB. MySQL AB es una compañía comercial, fundada por los desarrolladores de MySQL. Es una compañía Open Source de segunda generación que une los valores y metodología Open Source con un exitoso modelo de negocio (MySql, 2012). La versión con la que se trabajó durante el desarrollo fue la 5.0.22.

Entre las características más importantes y destacables que posee se encuentran los siguientes:

- Se encuentra escrito en C y en C++.
- Probado con un amplio rango de compiladores diferentes
- Funciona en diferentes plataformas.
- Usa GNU Automake, Autoconf, y Libtool para portabilidad.
- APIs disponibles para C, C++, Eiffel, Java, Perl, PHP, Python, Ruby, y Tcl.
- Uso completo de multi-threaded mediante threads del kernel. Pueden usarse fácilmente multiple CPUs si están disponibles.
- Proporciona sistemas de almacenamiento transaccional y no transaccional.
- Usa tablas con compresión de índice.
- Relativamente sencillo de añadir otro sistema de almacenamiento. Esto es útil si desea añadir una interfaz SQL para una base de datos propia.
- Un sistema de reserva de memoria muy rápido basado en hilos.
- Joins muy rápidos usando un multi-join de un paso optimizado.
- Tablas hash en memoria, que son usadas como tablas temporales.
- Las funciones SQL están implementadas usando una librería altamente optimizada y deben ser tan rápidas como sea posible. Normalmente no hay reserva de memoria tras toda la inicialización para consultas.
- El código MySQL se prueba con Purify (un detector de memoria perdida comercial) así como con Valgrind, una herramienta GPL (http://developer.kde.org/~sewardj/).
- El servidor está disponible como un programa separado para usar en un entorno de red cliente/servidor. También está disponible como biblioteca y puede ser incrustado en aplicaciones autónomas. Dichas aplicaciones pueden usarse por sí mismas o en entornos donde no hay red disponible.

2.5.5 Apache Web Server

El proyecto Apache HTTP Server es un esfuerzo de desarrollo de software de colaboración destinadas a crear una aplicación robusta, de código fuente de calidad comercial, con muchas características, y libremente disponible de un HTTP (Web) del servidor. El proyecto es administrado conjuntamente por un grupo de voluntarios ubicados en todo el mundo, el uso de Internet y la Web para comunicarse, planear y desarrollar el servidor y su documentación correspondiente. Este proyecto forma parte de la Fundación de Software Apache. Además, cientos de usuarios han contribuido con ideas, código y documentación del proyecto. Este archivo está destinado a describir brevemente la historia del Servidor Apache HTTP y reconocer los muchos colaboradores (Apache, 2012). La versión con la cual se trabajó en la aplicación fue la 2.2.2.

A continuación se mencionan algunas de las características de este servidor web:

- Corre en una multitud de Sistemas Operativos, lo que lo hace prácticamente universal.
- Es una tecnología gratuita de código fuente abierto. El hecho de ser gratuita es importante pero no tanto como que se trate de código fuente abierto. Esto le da una transparencia a este software de manera que si se quiere ver que es lo que se está instalando como servidor, se puede saber, sin ningún secreto, sin ninguna puerta trasera.
- Es un servidor altamente configurable de diseño modular. Es muy sencillo ampliar las capacidades del servidor Web Apache. Actualmente existen muchos módulos para Apache que son adaptables a éste, y están ahí para que se instalen cuando se necesiten. Otra cosa importante es que cualquiera que posea una experiencia decente en la programación de C o Perl puede escribir un módulo para realizar una función determinada.
- Trabaja con gran cantidad de Perl, PHP y otros lenguajes de script. Perl destaca en el mundo del script y Apache utiliza su parte del pastel de Perl tanto con soporte CGI

como con soporte mod perl. También trabaja con Java y páginas jsp. Teniendo todo el soporte que se necesita para tener páginas dinámicas.

- Permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. Es posible configurar Apache para que ejecute un determinado script cuando ocurra un error en concreto.
- Tiene una alta configurabilidad en la creación y gestión de logs. Apache permite la creación de ficheros de log a medida del administrador, de este modo se puede tener un mayor control sobre lo que sucede en el servidor.

2.5.6 DotNetBAR

La Suite DotNetBar para Visual Studio consta de 78 componentes para la creación de impresionantes interfaces de usuario de carácter profesional, facilitando la programación por medio del Visual Studio. Por más de 10 años DotNetBar está ayudando a los desarrolladores como a crear elegantes dichas interfaces. Es conveniente puntualizar que dicho software no es de tipo código abierto, sino que es propiedad de DevComponents.

El motivo por el cual se optó por esta suite fue para darle una interfaz de usuario más atractiva y elegante a la plataforma. DevComponents (DotNetBar, 2012) afirma que DotNetBar es el primer componente en el mundo en introducir todas las funciones de Office 2013, Office 2010, Windows 7 y Office 2007. En el proyecto se trabajó con la Version 10.0.0 de dicha Suite.

2.5.7 SDK Microsoft® Office Visio® 2003

En el desarrollo de la aplicación se utilizó el SDK (Software Development Kit) de Microsoft Office Visio 2003 el cual proporciona a los desarrolladores las herramientas,

muestras y documentación para apoyar el desarrollo de soluciones a medida en la plataforma de Visio 2003 con Microsoft Visual Studio 6.0 y Microsoft Visual Studio. NET 2003 o posteriores. Específicamente se empleó en la parte de la diagramación de los ejercicios de Entidad –Relación de bases de datos.

El SDK está pensado para programadores principiantes y avanzados de Office Visio, al igual que para quienes son integradores de sistemas y entornos de desarrollo corporativo. Este SDK contiene la primera versión del entorno de desarrollo ShapeStudio contiene aplicaciones de ejemplo de Microsoft Visual Basic. NET y Microsoft Visual C #. La versión del SDK con la que se trabajó fue la del 2003.

2.6 Aplicaciones Cliente-Servidor

Esta arquitectura consiste básicamente en que un programa, el cliente informático realiza peticiones a otro programa, el servidor, que les da respuesta. Aunque esta idea se puede aplicar a programas que se ejecutan sobre una sola computadora, es más ventajosa en un sistema multiusuario distribuido a través de una red de computadoras.

En esta arquitectura la capacidad de proceso está repartida entre los clientes y los servidores, aunque son más importantes las ventajas de tipo organizativo debidas a la centralización de la gestión de la información y la separación de responsabilidades, lo que facilita y clarifica el diseño del sistema. La separación entre cliente y servidor es una separación de tipo lógico, donde el servidor no se ejecuta necesariamente sobre una sola máquina ni es necesariamente un sólo programa.

Una disposición muy común son los sistemas multicapa en los que el servidor se descompone en diferentes programas que pueden ser ejecutados por diferentes computadoras aumentando así el grado de distribución del sistema. La arquitectura cliente-servidor sustituye a la arquitectura monolítica en la que no hay distribución, tanto a nivel físico como a nivel lógico. En la Figura 2.6 se puede ver el ejemplo de esta arquitectura.

Ventajas de la arquitectura cliente-servidor:

- Centralización del control: los accesos, recursos y la integridad de los datos son controlados por el servidor de forma que un programa cliente defectuoso o no autorizado no pueda dañar el sistema.
- Escalabilidad: se puede aumentar la capacidad de clientes y servidores por separado.
- Se reduce el tráfico de red considerablemente. Idealmente, el cliente se comunica con el servidor utilizando un protocolo de alto nivel de abstracción como por ejemplo SQL.

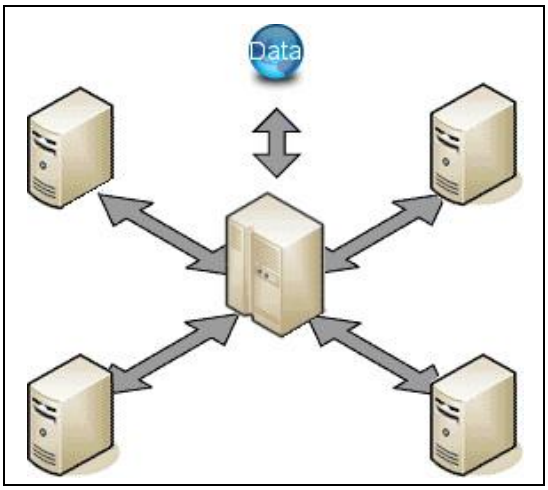

Figura 2.6 Ejemplo de una aplicación de tipo cliente/servidor.

2.7 Estado Del Arte

En este apartado se mencionan tres plataformas colaborativas. Las herramientas de diagramación CACOO y GLIFFY son de tipo Web, mientras que CREATELY tiene una versión en web y una en escritorio.

La tabla 2.1 visualiza una comparativa de opciones que ofrecen las plataformas anteriores versus (Herramienta Colaborativa para la Generación de Diagramas Entidad-Relación de Bases de Datos) HECODER.

Tabla 2.1 Comparativa de Plataformas Colaborativas.

COMPARATIVA DE PLATAFORMAS COLABORATIVAS				
CARACTERÍSTICA O FUNCIONALIDAD DEL SISTEMA	**GLIFFY**	**CREATELY**	**CACOO**	**HECODER**
Relativo a la Colaboracion				
1. Herramienta Colaborativa	✓	✓	✓	✓
1.1. Permite la colaboración en tiempo real	✓	✓	✓	✓
1.2. Permite crear Grupos de Trabajo con los usuarios de la aplicación	✓	✓	✓	✓
1.3. Permite crear Equipos de Trabajo dentro de los grupos trabajo	✗	✗	✗	✓
2. Permite la Generación de Diagramas	✓	✓	✓	✓
2.1. Los Diagramas son de Tipo Entidad – Relación de Bases de Datos	✓	✓	✓	✓
3. Cuenta con un Chat para conversación	✓	✓	✓	✓
3.1. El Chat es por equipo de trabajo	✗	✗	✗	✓
3. Cuenta con una sección de preguntas frecuentes: FAQ's	✓	✓	✓	✓
4. Permite Crear Ejercicios	✓	✓	✓	✓
4.1. Permite la Comparación automática de las soluciones de Ejercicios.	✗	✗	✗	✓
4.2. Permite una supervisión delos ejercicios en tiempo real	✓	✓	✓	✓
4.3. Permite Asignar Tareas que contienen ejercicios a equipos de Trabajo	✗	✗	✗	✓
Relativo a Precios de Licenciamiento				
1. Es gratuito	✓	✓	✓	✓
1.1. Límite de Usuarios en modalidad prueba	✓	✓	✓	✗
1.1. Límite de Días para utilizarlo en modalidad prueba	✓	✓	✓	✗
2. Puede tener versiones con licencia de pago	✓	✓	✓	✗
Relativo a la Forma de Ejecución y Comportamiento				
1. Permite su ejecución en Internet	✓	✓	✓	✓
2. Permite su ejecución en una red local sin internet	✗	✗	✗	✓
3. Permite agregar o eliminar elementos en las plantillas de componentes	✓	✓	✓	✓
4. Permite la impresión de los diagramas	✓	✓	✓	✓
Relativo a la Seguridad				
1. Permite diferentes tipos de Perfiles en la aplicación	✓	✓	✓	✓
1.1. Tiene el Perfil de Administrador General	✓	✓	✓	✓

1.2. Tiene el Perfil de Maestro	✗	✗	✗	✓
1.3. Tiene el Perfil de Alumno	✗	✗	✗	✓
2. Permite un Respaldo de la Base de datos desde la aplicación	✗	✗	✗	✓
3. Permite una Restauración de la Base de datos desde la aplicación	✗	✗	✗	✓
✓ = Aplica en algunos casos ✓ = Siempre aplica ✗ =No Aplica				

2.7.1 CACOO

Cacoo es una interesante herramienta 2.0 para crear diagramas de forma online con la particularidad de poder elaborarlos colaborativamente con otros usuarios. Fue desarrollado en Fukuoka, Japón, por Nulab Inc. Cacoo tiene una interfaz clara y sencilla que facilita mucho su uso. Además de los iconos correspondientes a las diversas opciones de edición, Cacoo cuenta con un chat para poder hablar y comunicarse con los usuarios durante los trabajos colaborativos, los cuales pueden ser en tiempo real.

Un diagrama creado con Cacoo puede ser editado por múltiples personas al mismo tiempo. Los cambios se reflejan en tiempo real. Trabajar con Cacoo parece como si todos estuvieran trabajando juntos en una misma habitación. En los diagramas elaborados con esta herramienta se puede dibujar, incluir texto, imágenes (prediseñadas o subirlas desde nuestro ordenador) y todo tipo de figuras geométricas, bocadillos para texto, iconos y dibujos decorativos.

En la figura 2.7 se presenta la interfaz de esta herramienta.También permite exportar los trabajos a formato PNG, compartirlos a través de enlaces y en las redes sociales (EducaconTIC, 2012).

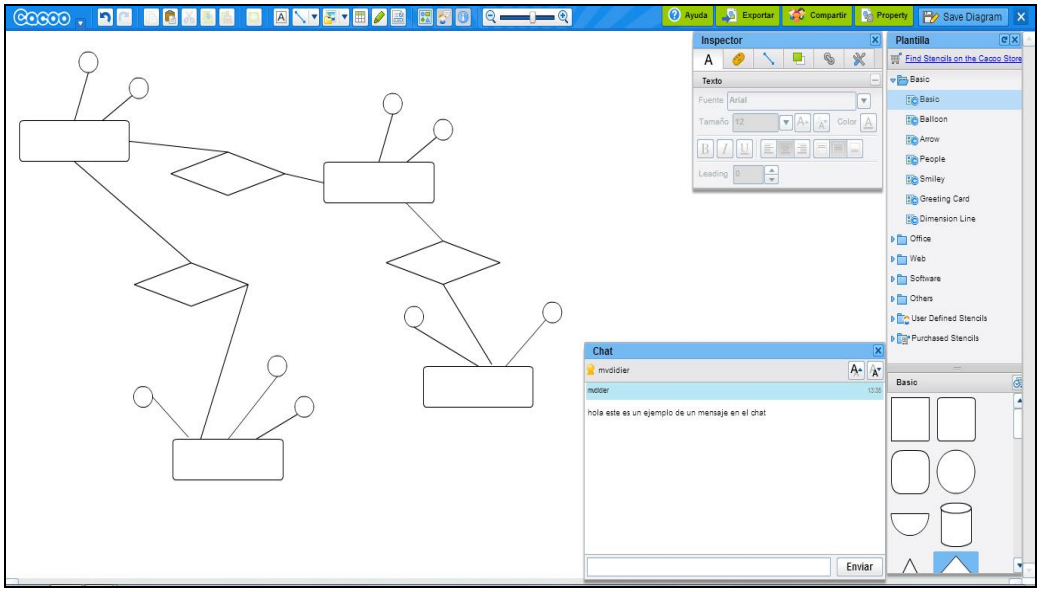

Figura 2.7 Interfaz de CACOO.

A continuación se enuncian algunas de las características principales que posee esta plataforma colaborativa:

- Es una Herramienta en Línea
- Se pueden crear Equipos de trabajo
- Se puede exportar a diversos formatos como PDF, JPG, PNG, SVG
- Se puede chatear en línea con los miembros de un equipo de trabajo
- Los miembros del equipo pueden modificar en tiempo real los diagramas
- Cuenta con una amplia gama de pinceles, además de que se puede crear nuevos pinceles personalizados a partir de imágenes de vectores.
- Contiene una traducción al Español

2.7.2 CREATELY

Creately es un software de diagramación y diseño operado por Cinergix Ltd., Pty Está basado en la nube Flex de Adobe / tecnologías Flash y proporciona una plataforma de comunicación visual para los equipos virtuales. Se puede utilizar para crear diagramas de flujo, diagramas de Gantt, organigramas, diseños UML, mapas mentales, los diseños de placas de circuitos, arte garabato y muchos otros tipos de diagramas (Cinergix, 2012).

Cuenta con una interfaz de arrastrar y soltar WYSIWYG y fácil de utilizar funciones de colaboración. Su principal producto es una aplicación basada en un navegador con soporte para todos los navegadores modernos como Firefox, Safari, Chrome y Opera navegador (Ver Figura 2.8).

Para las personas que prefieren offline de diagramas que proporcionan un software de escritorio que se ejecuta en Microsoft Windows, Mac OS y Linux.

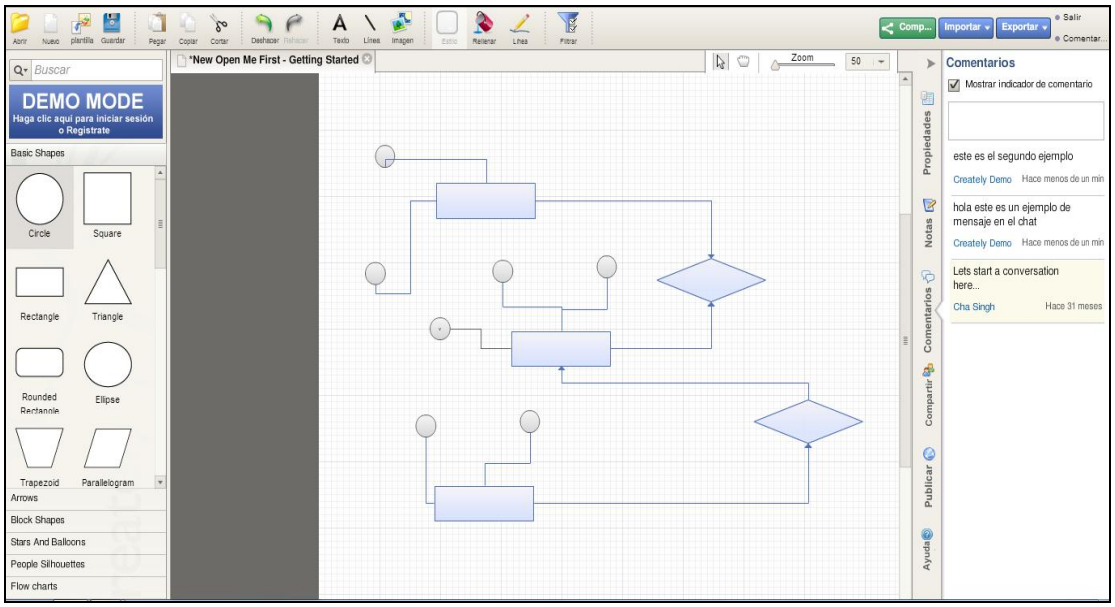

Figura 2.8 Interfaz web de CREATELY.

A continuación se enuncian algunas de las características principales que posee esta plataforma colaborativa:

- Potente colaboración en tiempo real
- Permite crear múltiples proyectos, comparte con los clientes y el equipo de
- Permite un historial de versiones completo para todos los diagramas
- Post-it estilo de comentarios de revisión
- Puede publicar diagramas, empotrado en sitios web / blogs / wikis

2.7.3 GLIFFY

Se trata de una solución en formato cloud computing (usa el software directamente de la web como un servicio) desarrollado por Gliffy, Inc., que permite en unos pocos minutos realizar gráficos de muchos tipos. Entre otros, es posible realizar diagramas de software, interfaces gráficas, diagramas de Venn, organigramas, diagramas de flujo, diagramas de red y mapas de construcción. Los trabajos realizados pueden ser exportados en formatos como JPG y PNG (Bortnik, 2010).

> "Aunque Gliffy.com no logra superar a Visio (especialmente en la galería gráfica), sí lo hace ante otros productos como Creately y brinda un excelente equilibrio en la relación prestaciones / costo / requerimientos" (Bortnik, 2010).

Al igual que otros servicios de este tipo, Bortnik comenta que es posible contratar extensiones pagas que ofrecen más (y mejores) funcionalidades, pero la versión gratuita es de excelente calidad.

Existen aplicaciones de escritorio que realizan la misma tarea y que quizás algunos usuarios las prefieran, especialmente Microsoft Visio, la aplicación predilecta para esta tarea por la gran mayoría de los usuarios (también es posible exportar el trabajo a formato de Visio para continuarlo allí) (Ver Figura 2.9).

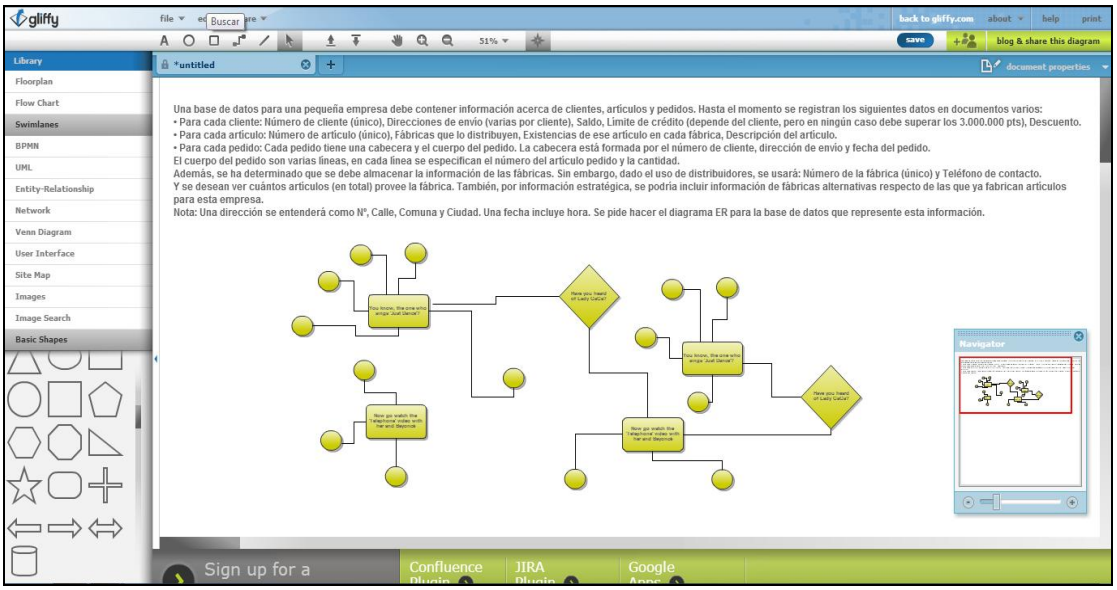

Figura 2.9 Interfaz web de GLIFFY.

El editor con todas las funciones Gliffy diagrama Online ofrece:

- Herramientas robustas y capacidades únicas de colaboración.
- Galerías de plantilla para poner en marcha la creación de diagramas.
- Bibliotecas de formas extensas con características de uso de importación de imágenes
- Ahorro automático y revisión de control para el seguimiento de los cambios entre los usuarios.
- Con un solo clic permite la publicación web.
- Lo mejor en su clase de seguridad.

CAPÍTULO 3: DESARROLLO

En este capítulo se muestran las actividades que se realizaron para el desarrollo de la plataforma colaborativa. A partir de este momento se utilizará HECODER para referirse a la plataforma colaborativa de software o simplemente “el software”, o “la plataforma”.

Para explicar de una manera clara y concisa el desarrollo de la plataforma es necesario comentar que se utilizó una programación en 3 Capas: Datos, Negocios y Presentación. Se hablarán de los casos de uso, clases, diagramas de secuencia y actividades que se realizaron para este proyecto.

Y por último, se describen algunas interfaces principales del sistema y algunas de las pruebas realizadas al mismo.

CAPÍTULO 3: DESARROLLO

3.1 Listado De Requerimientos

Un desarrollo de software tiene éxito si se han comprendido perfectamente los requisitos del software. Si un programa se analiza y especifica pobremente, decepcionará al usuario y desprestigiará a quien lo ha desarrollado. No importa lo bien diseñado o codificado que esté un programa, si no se ha analizado correctamente, defraudará al cliente y frustrará al desarrollador. La obtención de requisitos se enfoca en la descripción del propósito del sistema y es la que implica el reto mayor.

Larman define a los requerimientos como una descripción de las necesidades o deseos de un producto. La meta primaria de la fase de requerimientos es identificar y documentar lo que en realidad se necesita, en una forma que claramente se lo comunique al cliente y a los miembros de desarrollo (Larman, 2003).

En el caso de este proyecto, el proceso de análisis y de obtención de requerimientos se lleva a cabo a través de trabajar conjuntamente con el personal del Departamento de Ingeniería en Sistemas Computacionales del Instituto Tecnológico de Motul, quienes proporcionan los parámetros bajo los cuales la aplicación debe desarrollarse, para poder de esta manera cumplir con los objetivos de este proyecto.

3.1.1 Tipos de Requerimientos

Los requerimientos obtenidos se pueden clasificar de dos maneras: Funcionales y No Funcionales.

- **Requisitos Funcionales:** Describen las interacciones entre el sistema y su ambiente, en forma independiente a su implementación. El ambiente incluye al usuario y cualquier otro sistema externo con el cual interactúe el sistema

- **Requisitos No Funcionales:** Describen atributos sólo del sistema o del ambiente del sistema que no están relacionados directamente con los requisitos funcionales. Los requisitos no funcionales incluyen restricciones cuantitativas, como el tiempo de respuesta o precisión, tipo de plataforma (lenguajes de programación y/o sistemas operativos, etc.) (Espinoza, 2013).

En la Tabla 3.1 se enlista una muestra representativa de los requerimientos que tiene que cumplir la plataforma colaborativa para cumplir con los objetivos de este proyecto.

Tabla 3.1 Listado de Requerimientos de HECODER.

REQUERIMIENTOS DE HECODER		
Requerimiento	**Funcional**	**No Funcional**
[1] Contar con una herramienta colaborativa que permita generar diagramas de E-R de una base de datos en tiempo real	✓	
[2] La aplicación no deberá tener que adquirir licencias de software para su uso.		✓
[3] La aplicación deberá funcionar sobre MySQL.		✓
[4] La aplicación deberá contar con un perfil de tipo administrador general, de maestro y uno de alumno.	✓	
[5] El usuario con perfil Administrador General deberá tener la opción para poder respaldar y restaurar la base de datos del sistema.	✓	
[6] El usuario con perfil Administrador General deberá tener la opción de administrar el alta, baja y cambio de usuarios con cuenta de tipo Maestro y Alumno.	✓	
[7] El usuario con perfil Administrador General deberá tener la opción de administrar el alta, baja y cambio de Grupos de Trabajo.	✓	
[8] El usuario con perfil Maestro deberá tener la opción de administrar el	✓	

alta, baja y cambio de Equipos de Trabajo		
[9] El usuario con perfil Maestro deberá tener la opción de administrar el alta, baja y cambio de Ejercicios de Entidad - Relación	✓	
[10] El usuario con perfil Maestro deberá tener la opción de administrar el alta, baja y cambio de Tareas	✓	
[11] El usuario con perfil Alumno y Maestro deberá tener la opción de intercambiar puntos de vista por medio de un chat online.	✓	
[12] El usuario con perfil Alumno y Maestro deberá tener la opción de poder hacer y responder preguntas de la base de datos de preguntas frecuentes (FAQ's)	✓	

3.2 Funciones

Larman afirma que las funciones del sistema son las que este habrá de hacer. "Con el objetivo de verificar que algún X es verdad una función del sistema, la siguiente oración deberá tener sentido: El sistema deberá hacer <X>" (Larman, 2003).

3.2.1 Tipos de Funciones

Las funciones obtenidas se pueden clasificar de tres maneras: Evidentes, Ocultas y Superflua.

- **Funciones Evidentes:** Debe de realizarse y el usuario debería saber que se ha hecho
- **Funciones Ocultas:** Debe de realizarse aunque no es visible para los usuarios. Esto se aplica a muchos servicios técnicos subyacentes, como guardar información en un mecanismo persistente de almacenamiento
- **Funciones Superflua:** Opcionales, su inclusión no repercute significativamente en el costo ni en otras funciones

En la Tabla 3.2 se enlista una muestra representativa de las funciones que tiene que cumplir la plataforma colaborativa para satisfacer los objetivos de este proyecto.

Tabla 3.2 Listado de Funciones de HECODER.

FUNCIONES DE HECODER		
Ref.	**Función**	**Categoría**
1	Desplegar pantalla de inicio de sesión.	Evidente
1	Verificar lista de permisos que tiene asignado el usuario autenticado para cargar opciones habilitadas de almacenamiento persistente.	Oculta
1	Desplegar lista de selección de Tareas recuperadas de almacenamiento persistente.	Evidente
1	Desplegar ejercicio para diagramar en línea recuperado de almacenamiento persistente.	Evidente
1	Desplegar datos del usuario autenticado recuperado de almacenamiento persistente.	Evidente
1	Desplegar la barra de herramientas de selección de componentes de diagramas Entidad – Relación: Entidad, Atributo, Relación y Unión.	Evidente
1	Desplegar listado de botones con el listado de palabras que conforman el ejercicio desde almacenamiento persistente.	Evidente
1	Registrar los cambios en el nombramiento de cada objeto del diagrama en almacenamiento persistente.	Oculta
1	Contabilizar los elementos que conforman el diagrama de Entidad – Relación desde almacenamiento persistente.	Oculta
1	Desplegar en un grid las cantidades de cada objeto que se encuentra en el diagrama de Entidad - Relación	Evidente
1	Registrar los cambios de los ejercicios de cada equipo en almacenamiento persistente.	Oculta
1	Desplegar los cambios realizados por otro miembro del equipo sobre el mismo ejercicio desde almacenamiento persistente.	Evidente
1	Registrar la finalización del ejercicio en almacenamiento persistente.	Oculta
1	Validar si existente nuevos ejercicios por realizar desde almacenamiento persistente.	Oculta
3	Validar existencia del archivo de configuración de la base de datos.	Oculta
3	Desplegar pantalla de configuración de conexión a la base de datos del sistema.	Evidente
3	Validar conexión a la base de datos con los datos del archivo de configuración.	Oculta
3	Desplegar mensaje de error de conexión a la base de datos.	Evidente

3.3 Casos de Uso

El desarrollo del software HECODER incluyó la elaboración de los diagramas de casos de uso. En esta sección se presentan los 4 casos de uso elaborados para el desarrollo del software. En la Figura 3.1 se muestra el caso de uso general del sistema con sus respectivos actores, los cuales son 3: el Administrador General, el Maestro y por último, el Alumno.

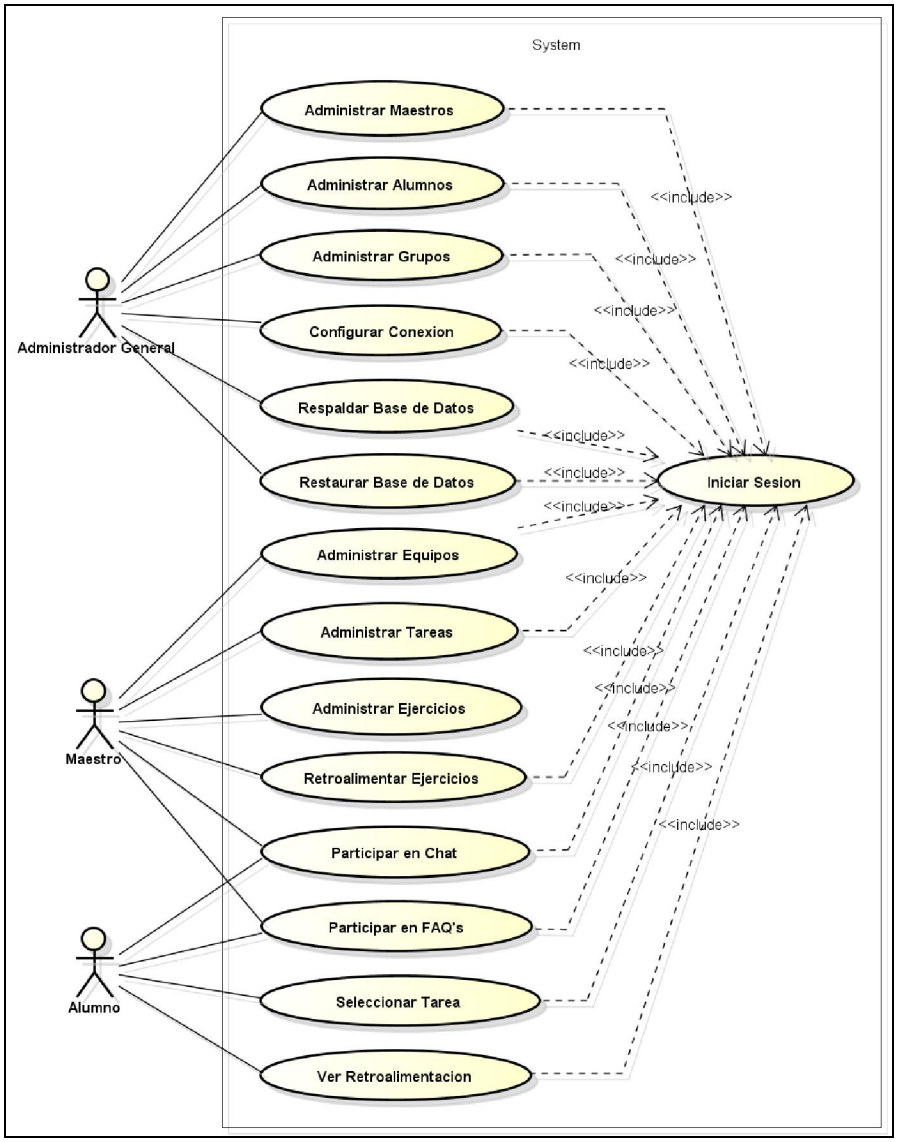

Figura 3.1 Caso de Uso: General.

3.3.1 Caso de Uso: Administrador General

El diagrama de la Figura 3.2 corresponde al caso de uso del "Administrador General". Este caso de uso involucra al actor Administrador General. El sistema representa al software HECODER. Durante la interacción con el sistema el Administrador General puede elegir una de las cinco opciones, las cuales son:

- **Administrar Maestros:** Engloba lo referente a la administración de los usuarios con perfil de Maestro. Este caso de uso está relacionado con 3 casos de uso extendido, los cuales son:
  - Alta de Maestros.
  - Editar Maestros.
  - Activar/Desactivar Maestros.

- **Administrar Grupos:** Este caso de uso engloba la gestión relacionada con los grupos de trabajo con los que se cuente en la institución educativa. Este caso de uso está relacionado con 3 casos de uso extendido, los cuales son:
  - Alta de Grupos.
  - Editar Grupos.
  - Activar/Desactivar Grupos.

- **Administrar Alumnos:** Este caso de uso engloba la administración de los Alumnos. Este caso está relacionado con 3 casos de uso extendido, los cuales son:
  - Alta de Alumnos.
  - Editar Alumnos.
  - Activar/Desactivar Alumnos.

- **Respaldar la Base de Datos:** Este caso de uso hace referencia al proceso de respaldar la base de datos del sistema.

- **Restaurar Base de Datos:** Este caso de uso trata sobre el proceso de restaurar la base de datos del sistema.

- **Configurar la Conexión:** Este caso de uso es sobre el proceso de configurar el archivo que utiliza el sistema para poder conectarse con la base de datos y establecer la comunicación.

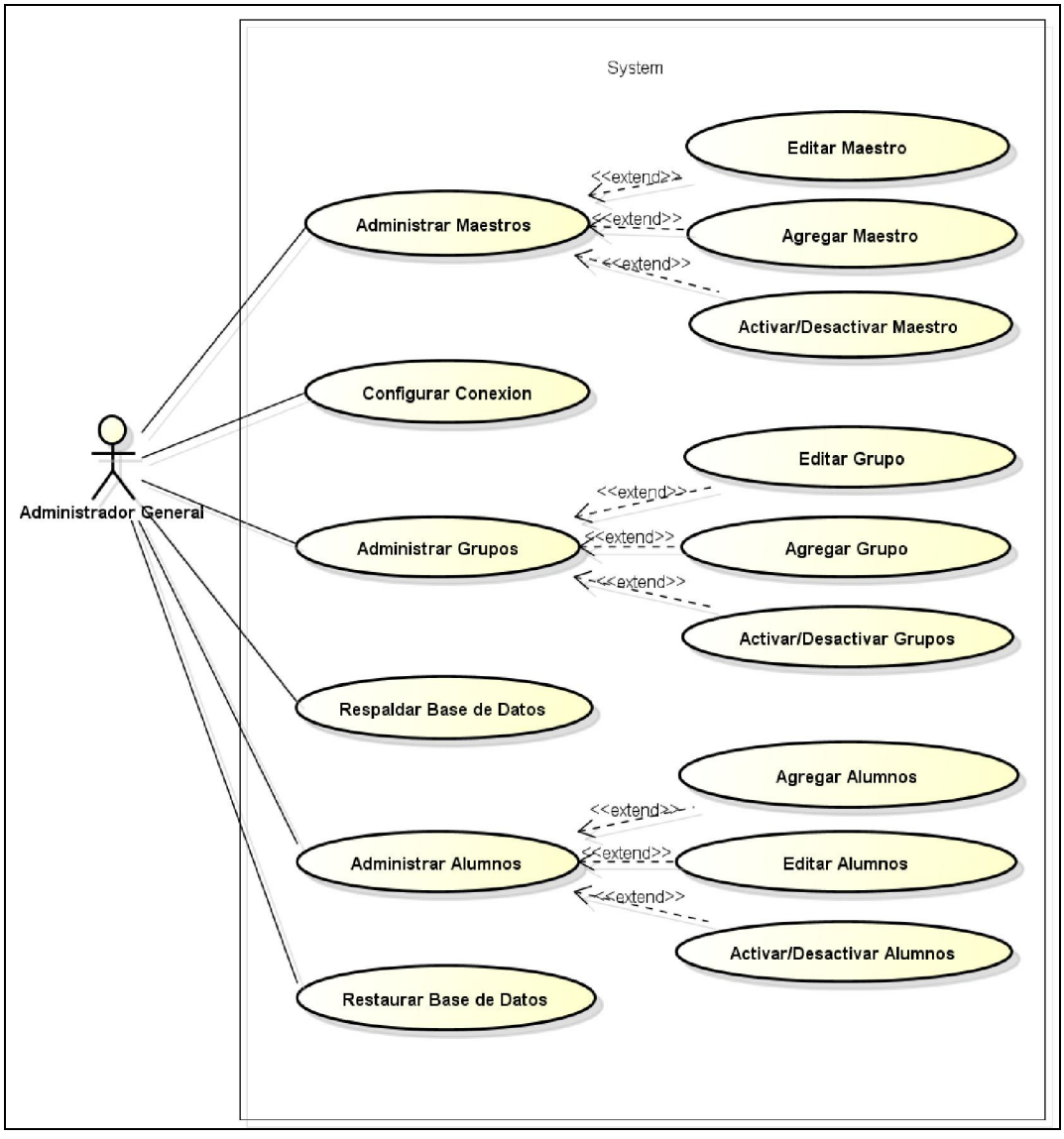

Figura 3.2 Caso de Uso: Administrador General.

3.3.2 Caso de Uso: Maestro

El diagrama de la Figura 3.3 corresponde al caso de uso del "Maestro". Este caso de uso involucra al actor Maestro. El sistema representa al software HECODER. Durante la interacción con el sistema el Maestro puede elegir una de las seis opciones, las cuales son:

- **Administrar Equipos:** Engloba lo referente a la administración de como los Maestros agruparan a los alumnos para conformar los equipos de trabajo colaborativos. Este caso de uso está relacionado con 3 casos de uso extendido, los cuales son:
  - Alta de Equipos.
  - Editar Equipos.
  - Activar/Desactivar Equipos.

- **Administrar Tareas:** Este caso de uso hace referencia al proceso en el que se seleccionan los ejercicios que se incluirán en una Tarea, así como el orden en que los equipos de trabajo están realizándolos. Este caso de uso está relacionado con 3 casos de uso extendido, los cuales son:
  - Alta de Tareas.
  - Editar Tareas.
  - Activar/Desactivar Tareas.

- **Administrar Ejercicios:** Este caso de uso engloba la administración de los Ejercicios que serán incluidos en las distintas tareas que se tengan que realizar por parte de los alumnos. Este caso de uso está relacionado con 3 casos de uso extendido, los cuales son:
  - Alta de Ejercicios.
  - Editar Ejercicios.
  - Activar/Desactivar Ejercicios.

- **Participar en las FAQ's:** Caso de uso en el que el maestro puede añadir preguntas a la base de datos de Preguntas Frecuentes o FAQ's, así como la posibilidad de realizar búsquedas de preguntas o respuestas en esta misma y por último, la opción de responder a cualquier pregunta que se encuentre en esta base. Este caso de uso está relacionado con 3 casos de uso extendido, los cuales son:
  - Agregar Nueva Pregunta.
  - Buscar Preguntas.
  - Responder Preguntas.

- **Retroalimentar Ejercicios:** Caso de uso en el que el Maestro puede anexar comentarios, recomendaciones o sugerencias, a una solución de un ejercicio de una tarea de un equipo sobre la solución del mismo.

- **Participar en el Chat:** Caso de uso en el que el Maestro puede participar en la conversación con los diferentes equipos para darles comentarios, sugerencias o ayuda en la solución de algún ejercicio en particular.

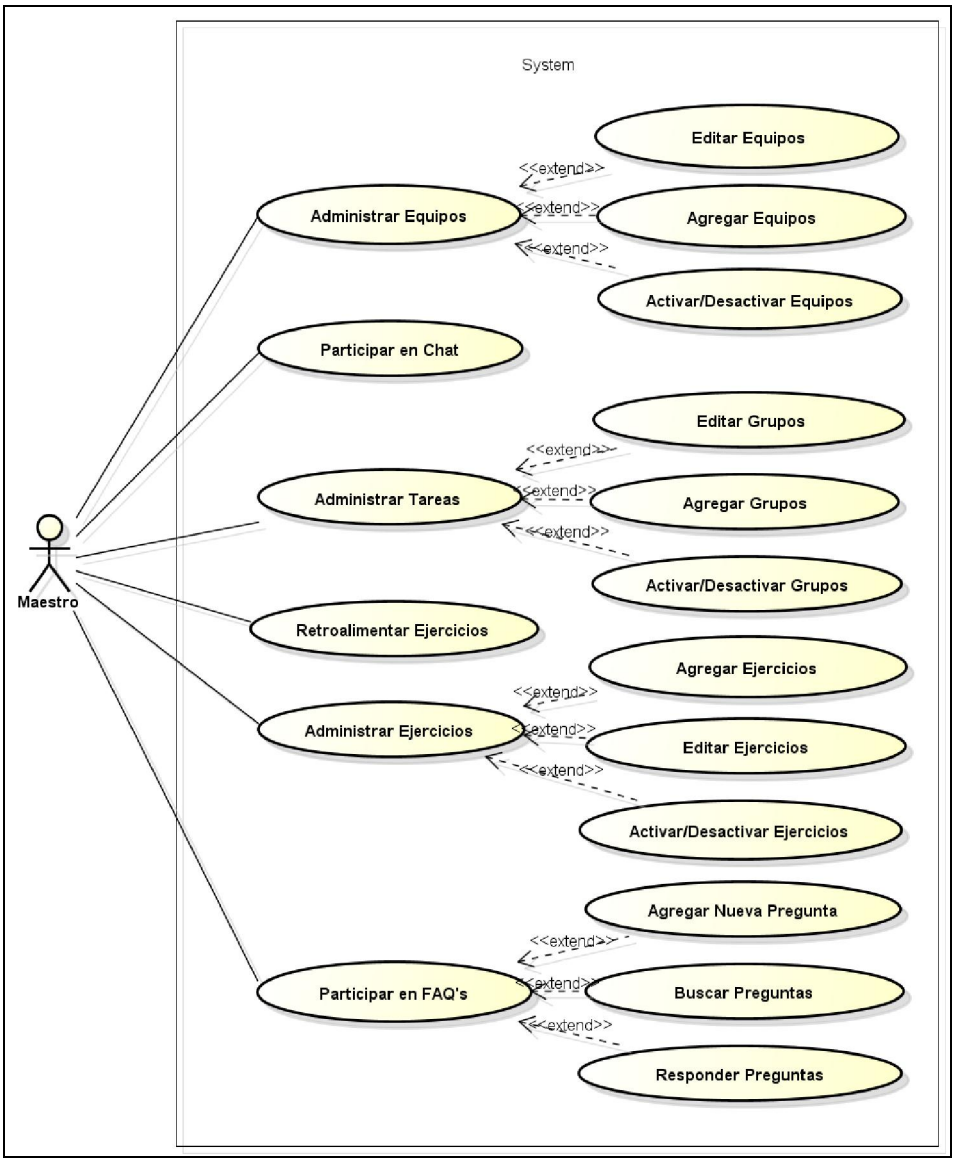

Figura 3.3 Caso de Uso: Maestro.

3.3.3 Caso de Uso: Alumno

El diagrama de la Figura 3.4 corresponde al caso de uso del "Alumno". Este caso de uso involucra al actor Alumno. El sistema representa al software HECODER. Durante la interacción con el sistema el Alumno puede elegir una de las cuatro opciones, las cuales son:

- **Seleccionar Tarea:** Caso de uso en el que se representa el proceso en el que un alumno selecciona que tarea realizar, así como los procesos de guardar los avances de la solución de un ejercicio y por último el de dar por finalizado un ejercicio. Este caso de uso está relacionado con 3 casos de uso extendido, los cuales son:
  - Resolver Ejercicio.
  - Guardar Avance.
  - Finalizar Ejercicio.

- **Visualizar Retroalimentación:** Este caso de uso representa el proceso que sigue un alumno al querer revisar los comentarios que realice el Maestro sobre la solución de un ejercicio finalizado que el equipo resolvió.

- **Participar en las FAQ's:** Caso de uso en el que el alumno puede añadir preguntas a la base de datos de Preguntas Frecuentes o (Frequently Asked Questions) FAQ's, así como la posibilidad de realizar búsquedas de preguntas o respuestas en esta misma y por último, la opción de responder a cualquier pregunta que se encuentre en la base de datos. Este caso de uso está relacionado con 3 casos de uso extendido:
  - Agregar Nueva Pregunta.
  - Buscar Preguntas.
  - Responder Preguntas.

- **Participar en el Chat:** Caso de uso en el que el Alumno puede participar en la conversación con los diferentes integrantes de su equipo y con el Maestro para poder aportar algún tipo de comentario, sugerencia, ayuda o alguna idea de cómo poder solucionar el ejercicio que esté tratando de resolver.

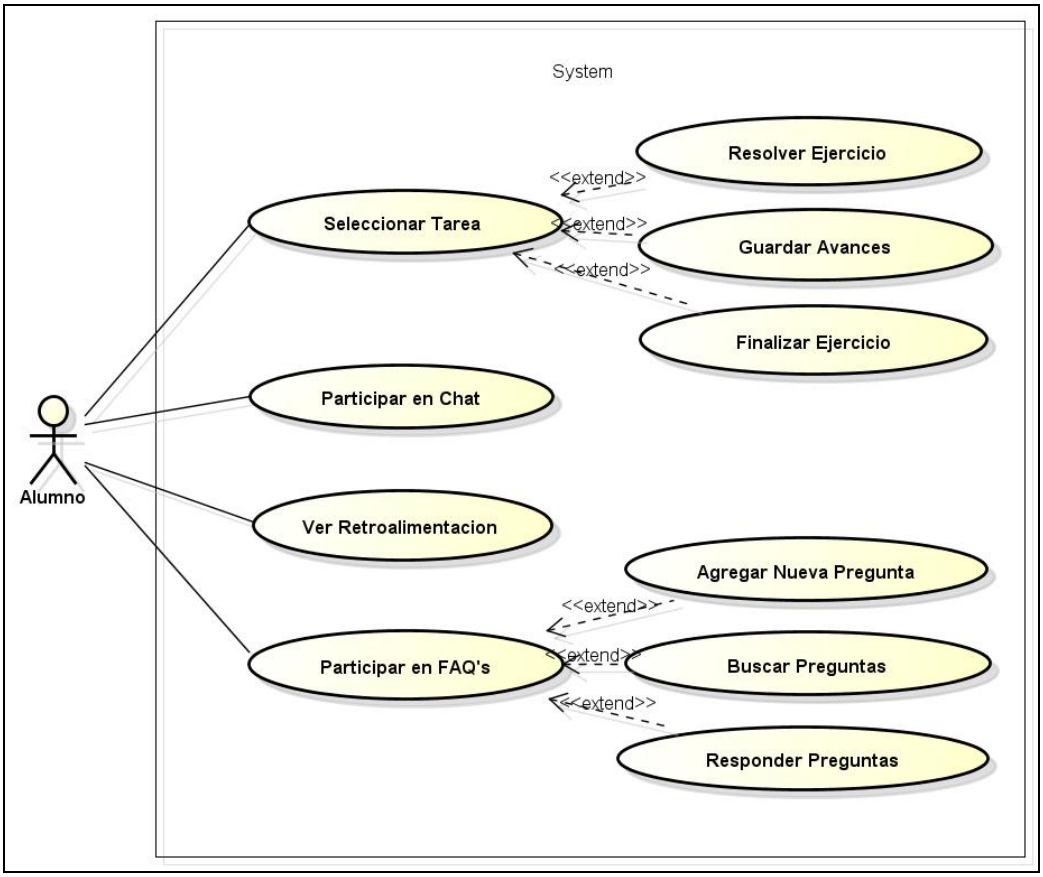

Figura 3.4 Caso de Uso: Alumno.

3.4 Diseño

En esta etapa se describen las actividades realizadas en la fase de diseño, tomando como guía el Lenguaje Unificado de Modelado: UML.

3.4.1 Diagramas De Clases

La aplicación fue desarrollada bajo una arquitectura en Tres Capas: Presentación, Objetos y Datos, lo cual involucra un número extenso de clases. En las figuras 3.5 y 3.6 se

muestran de manera general los diagramas de clases entre la Capa de Datos y de Objetos y entre la Capa de Objetos y de Presentación, respectivamente.

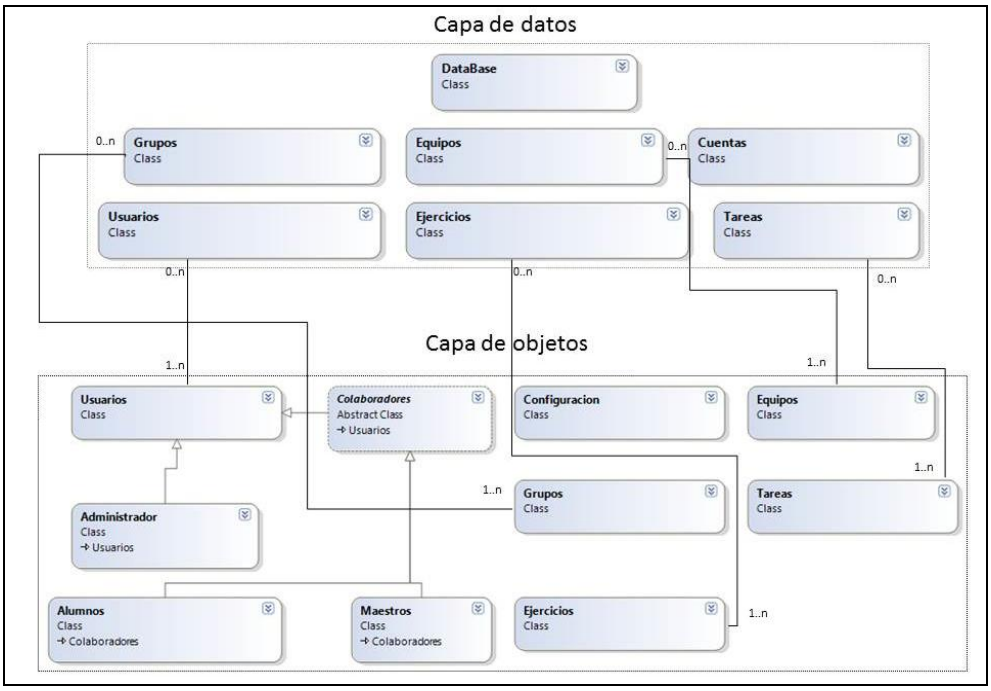

Figura 3.5 Diagrama de Clases de la Capa de Datos y de Objetos.

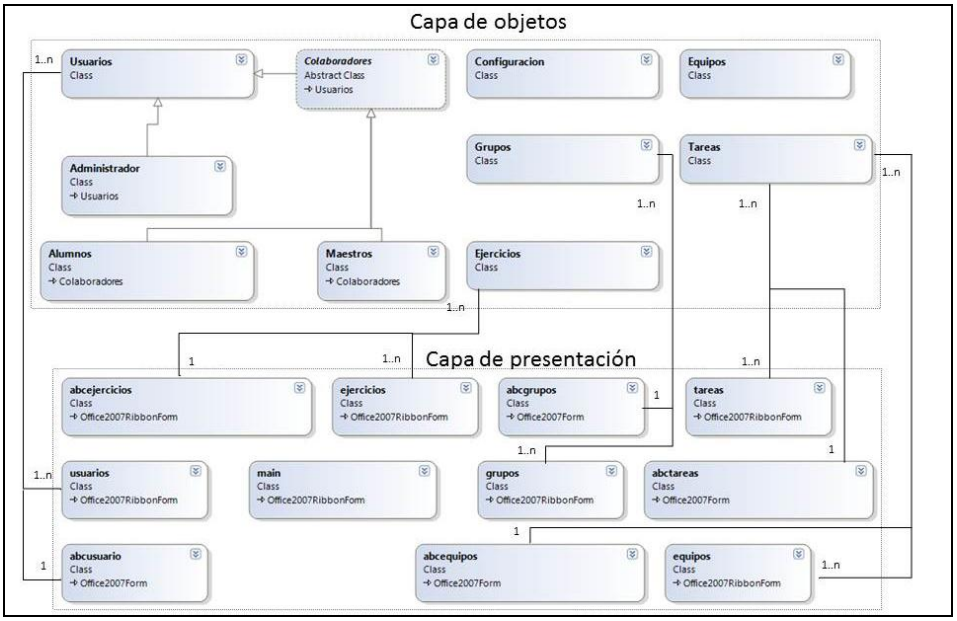

Figura 3.6 Diagrama de Clases de la Capa de Objetos y la de Presentación.

La figura 3.7 muestra el diagrama de Clases de la Capa de Negocios para HECODER, ya refinado. Se hace notar que las clases que aparecen en el diagrama son las relacionadas de manera directa con el dominio del problema, es decir, no se muestran clases de servicios ni de utilerías. En la figura 3.7 se omiten los métodos debido a que ocuparían un espacio considerable.

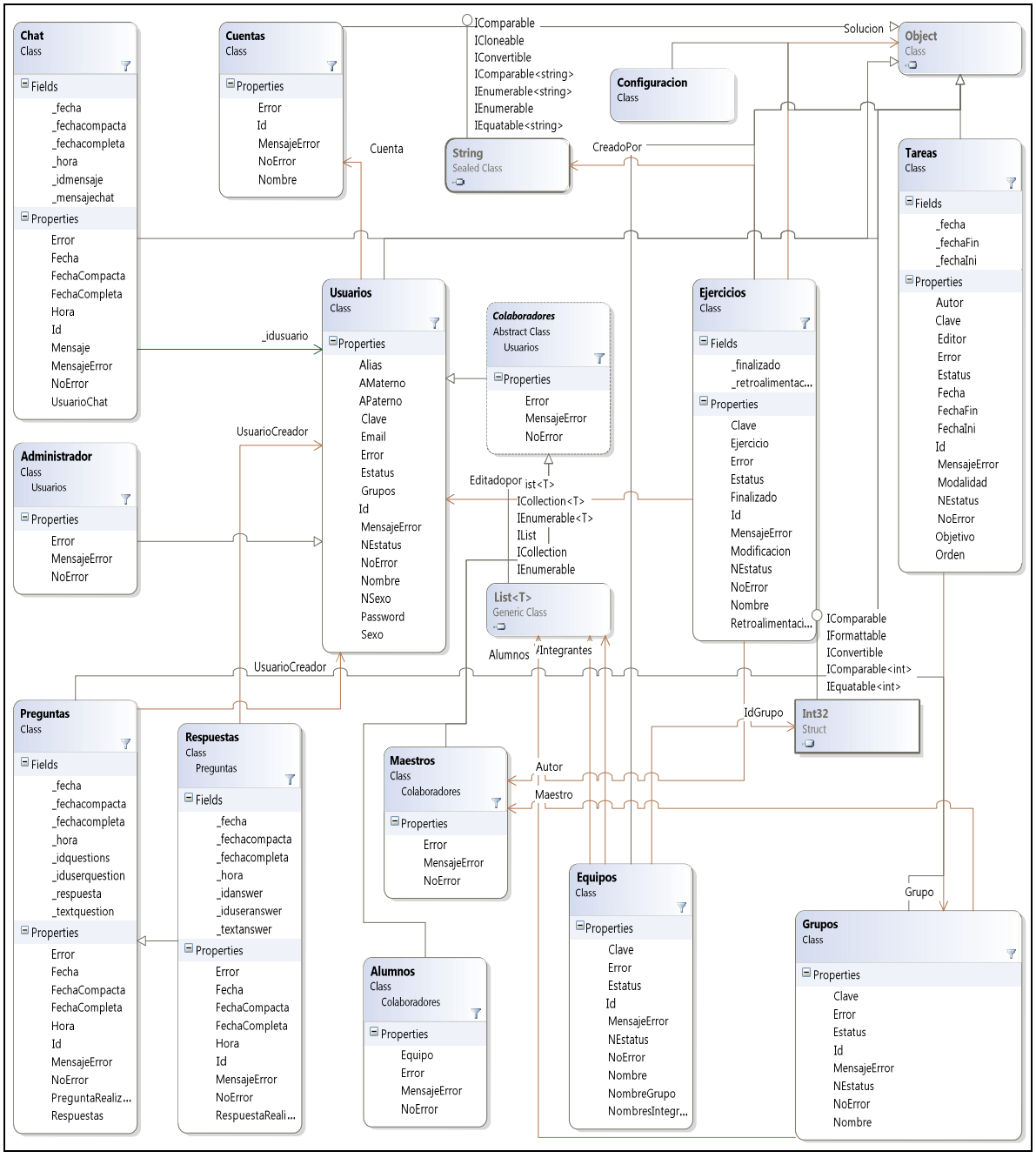

Figura 3.7 Diagrama de clases de la Capa de Negocios.

Obsérvese el diagrama completo de la clase Preguntas y Respuestas con todos sus atributos y métodos en la figura 3.8.

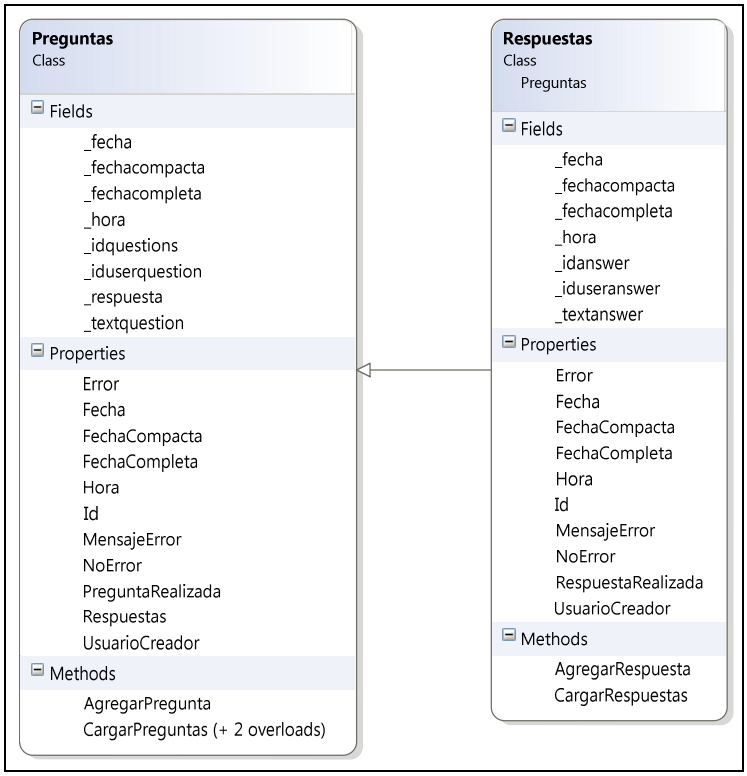

Figura 3.8 Diagrama de clase para las clases de Preguntas y Respuestas.

3.4.2 Diagramas de Secuencia

Los diagramas de secuencia ilustran la interacción entre objetos y el orden secuencial en el que ocurren dichas interacciones, es decir cómo se comunican los objetos entre sí. Los objetos se comunican mediante interfaces, para poder invocar a una operación. En los Casos de Uso se modelan las características del sistema y se desarrollan escenarios. El diagrama de secuencias proporciona un camino a partir de los escenarios para describir las operaciones en una forma más detallada.

En las figuras 3.9 hasta la 3.12 se aprecian los diagramas de secuencia de los procesos más importantes del sistema. Es importante aclarar que debido a que la arquitectura es en capas, algunas tareas llevan un proceso similar por lo que se seleccionó las más representativas para fines de este trabajo.

En la figura 3.9 se aprecia el diagrama de secuencia para el caso de uso de alta de usuarios; como se observa en el diagrama el único actor involucrado es el Administrador General, la interfaz es la de Presentacion.abcusuario, el objeto empleado es el de Objetos.Usuarios y la en la clase de datos es la Datos.Usuarios..

El administrador general proporciona los datos del usuario que quiere dar de alta, presiona el botón de aceptar, se dispara el evento del botón aceptar y se procede a llenar el objeto de Objetos.Usuarios con los datos capturados; inmediatamente se envía a la capa de datos en el método de NuevoUsuarioDB, en donde hace la operación con la base de datos y se almacena la información.

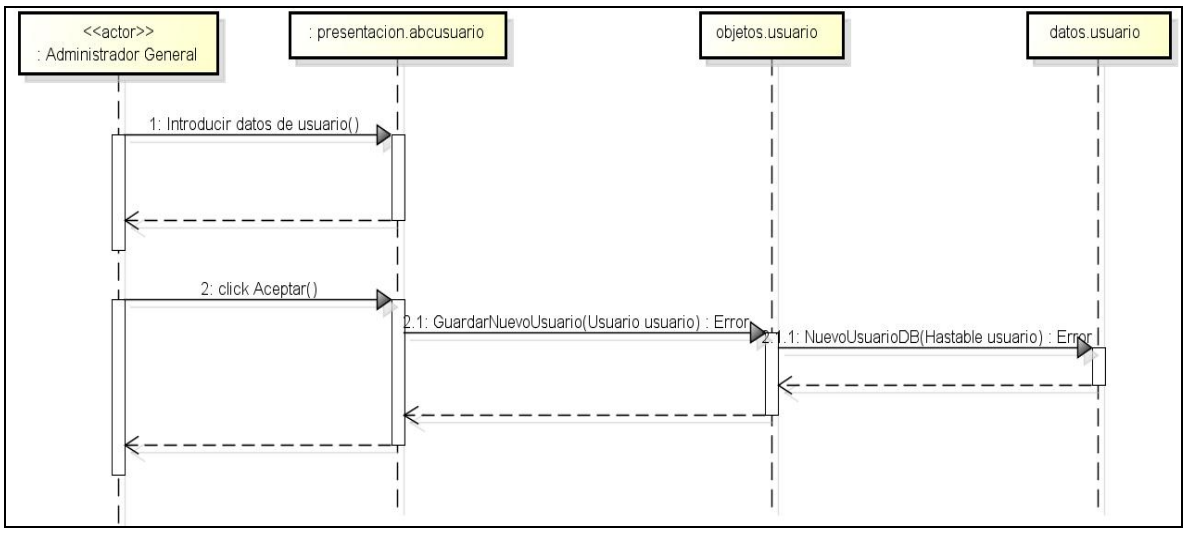

Figura 3.9 Diagrama de Secuencia para Alta de usuarios.

Siguiendo con el catálogo de usuarios, la figura 3.10 muestra el diagrama de secuencia para el caso de uso de activar o desactivar usuarios. En este caso el actor es nuevamente el

administrador general, la interfaz de captura es presentación.abcusuario, los objetos relacionados son Maestro o Alumno que heredan de un objeto superior llamado Colaborador y la capa de datos: datos. Usuario.

El administrador general selecciona del catálogo de maestro o alumnos al registro que desee activar o desactivar, presiona el botón de aceptar, se dispara el evento del botón aceptar y se invoca el método activar/desactivar de la clase Colaborador; inmediatamente se envía a la capa de datos en el método de ActivarDeactivarDB, en donde hace la operación con la base de datos y se actualiza la información.

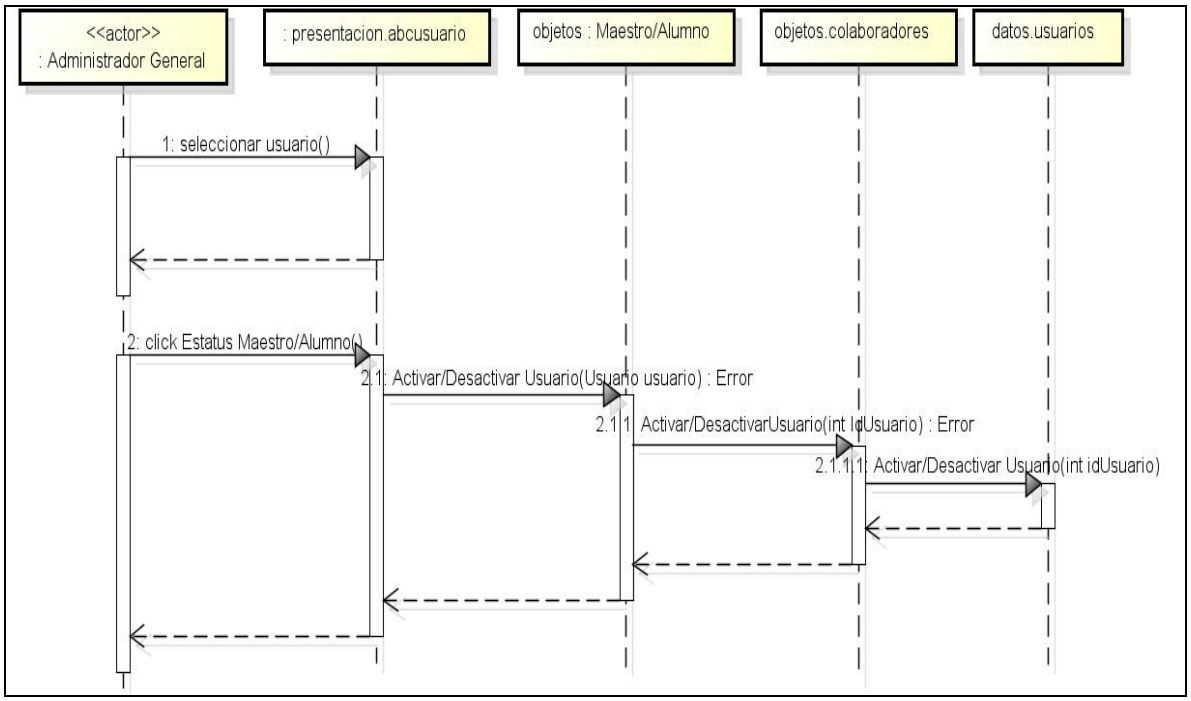

Figura 3.10 Diagrama de secuencia para Activar/Desactivar usuarios.

Por otro lado, la figura 3.11, muestra el diagrama de secuencia para el caso de uso de agregar una pregunta a la base de conocimientos o FAQ's. Los actores que intervienen son el Alumno o el Maestro, la interfaz de captura es presentación.abcpregunta, los objetos relacionados son Objetos.Pregunta y la capa de datos: datos. Pregunta.

El flujo de operación de este caso es el siguiente: el Maestro o Alumno selecciona la pestaña de las FAQ's la opción de agregar pregunta, escribe la pregunta en la sección de las FAQ's, presiona el botón de guardar, se dispara el evento del botón guardar y se invoca el método AgregarPregunta de la clase Pregunta; inmediatamente se envía a la capa de datos en el método de AgregarPreguntaDB, en donde hace la operación con la base de datos y se agrega la información.

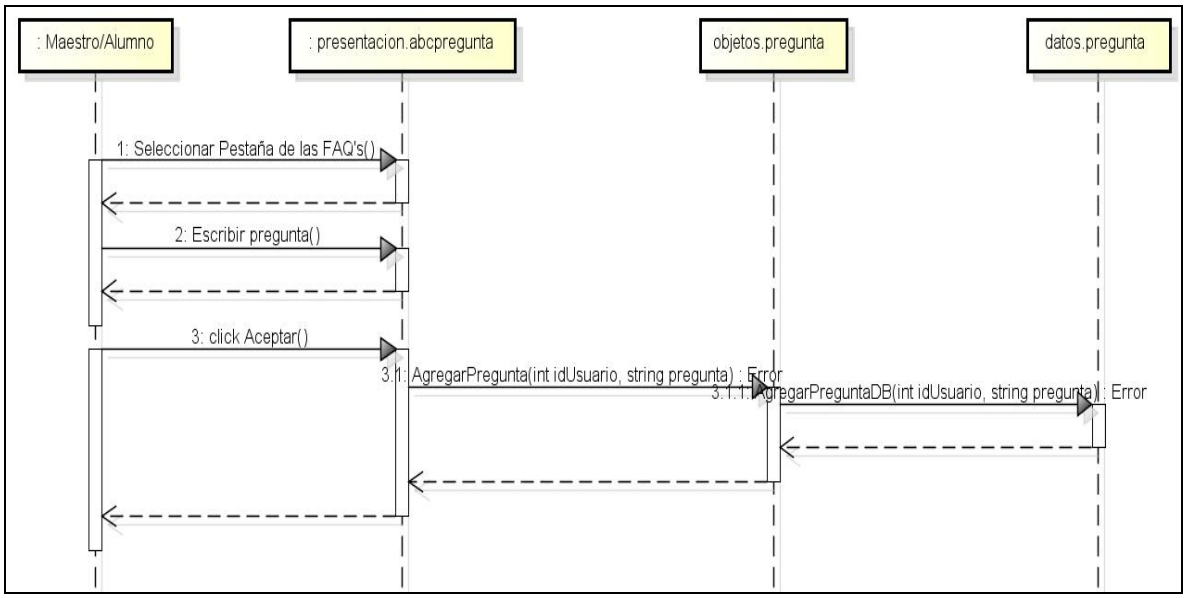

Figura 3.11 Diagrama de secuencia para Agregar una Pregunta a la Base de Conocimientos.

Finalmente, en la figura 3.12, se muestra el diagrama de secuencia para el caso de uso de responder una pregunta de la base de conocimientos o FAQ's. Los actores que intervienen son el Alumno o el Maestro; las interfaces involucradas son las de Presentación.main y Presentación.abcrespuesta; los objetos relacionados son Objetos.Respuesta y la capa de datos: Datos.Respuesta.

El flujo de operación de este caso es el siguiente: el Maestro o Alumno selecciona de la pestaña de las FAQ's, la pregunta de la cual desee agregar una respuesta. En ese momento se visualiza la interfaz de captura de responder pregunta (presentación.abcrespuesta); posteriormente de escribir la respuesta el actor presiona el botón de aceptar, por lo que se

ejecuta el método de c de la clase Objetos.Respuesta, el cual invoca a la clase de datos, específicamente al método RespuestaDB, el cual realiza la operación en la base de datos y agrega la respuesta a ésta.

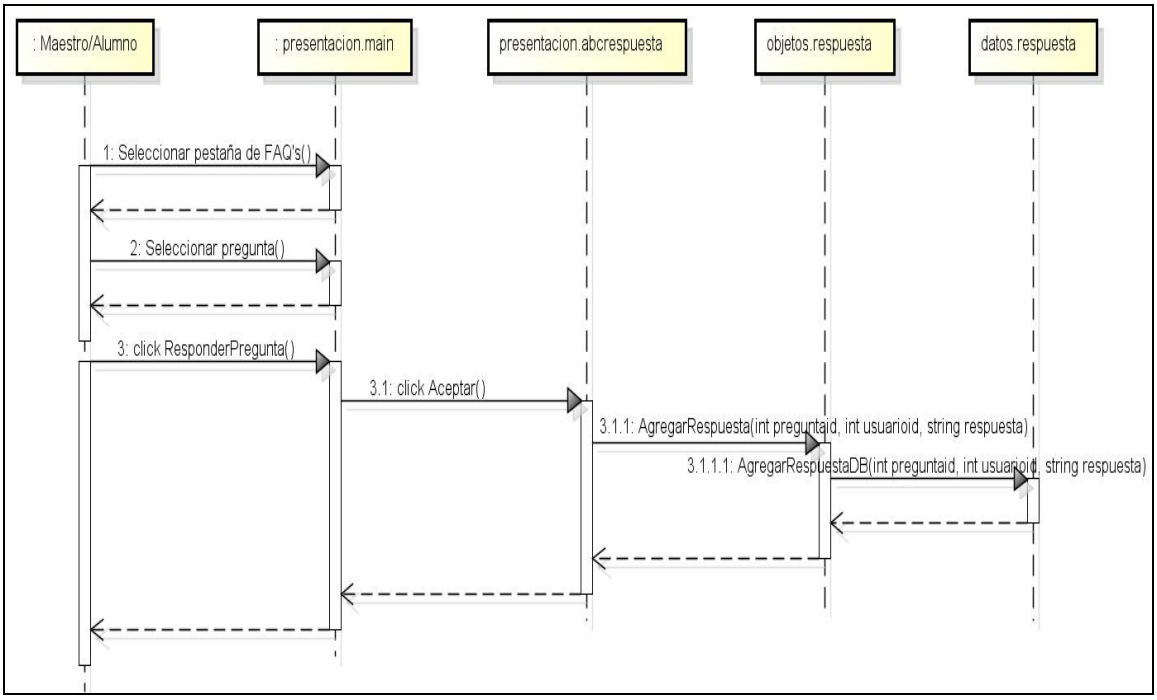

Figura 3.12 Diagrama de secuencia para responder una Pregunta.

3.4.3 Diagramas de Actividades

Con los diagramas de actividades se puede apreciar la secuencia de actividades por las que se pasa al momento de llevar un proceso. A continuación se presenta el diagrama de actividades para el proceso “Alta de usuario” (ver Figura 3.13).

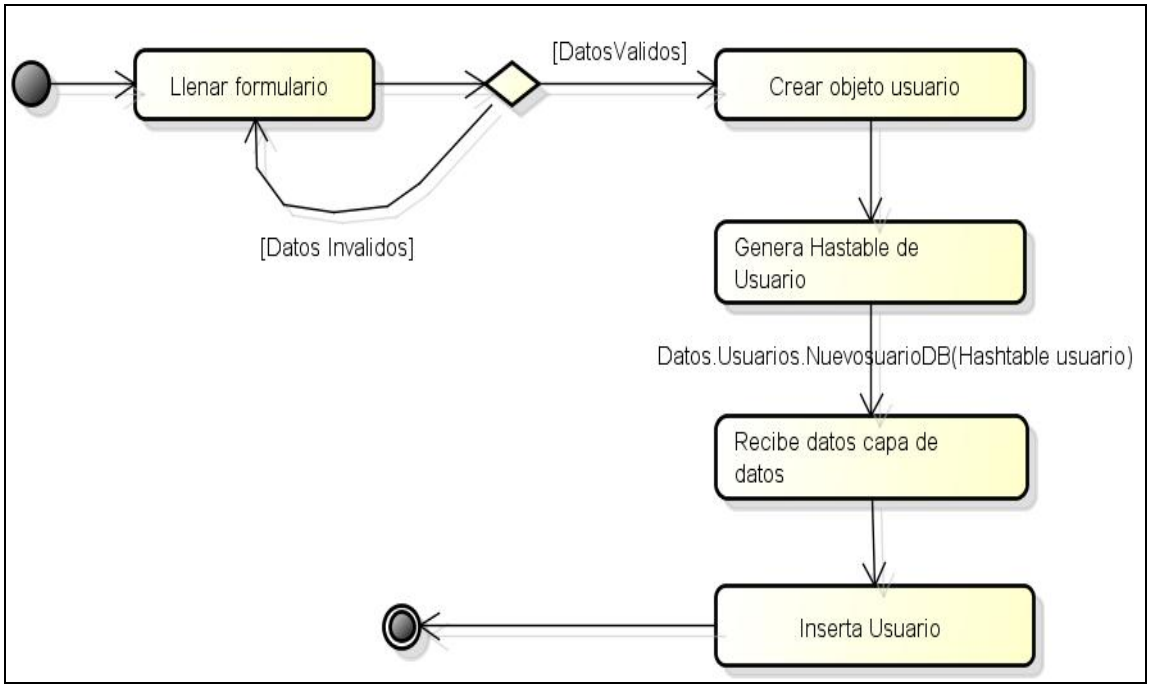

Figura 3.13 Diagrama de actividades para Alta de usuario.

A continuación se presenta el Diagrama de actividades para el proceso "Alta de pregunta" (ver Figura 3.14).

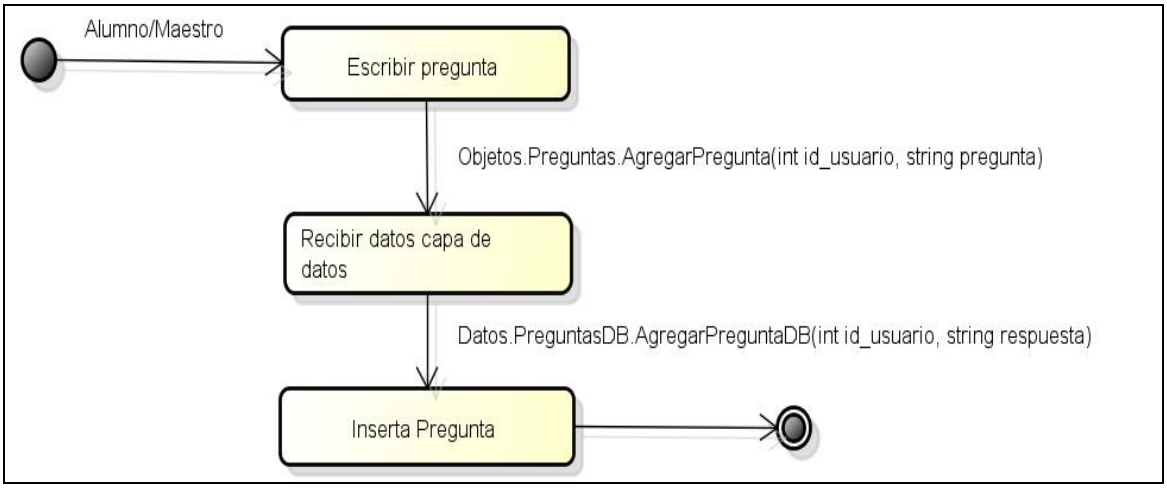

Figura 3.14 Diagrama de actividades para Alta de pregunta.

A continuación se presenta el Diagrama de actividades para el proceso "Activar o Desactivar Usuario" (ver Figura 3.15).

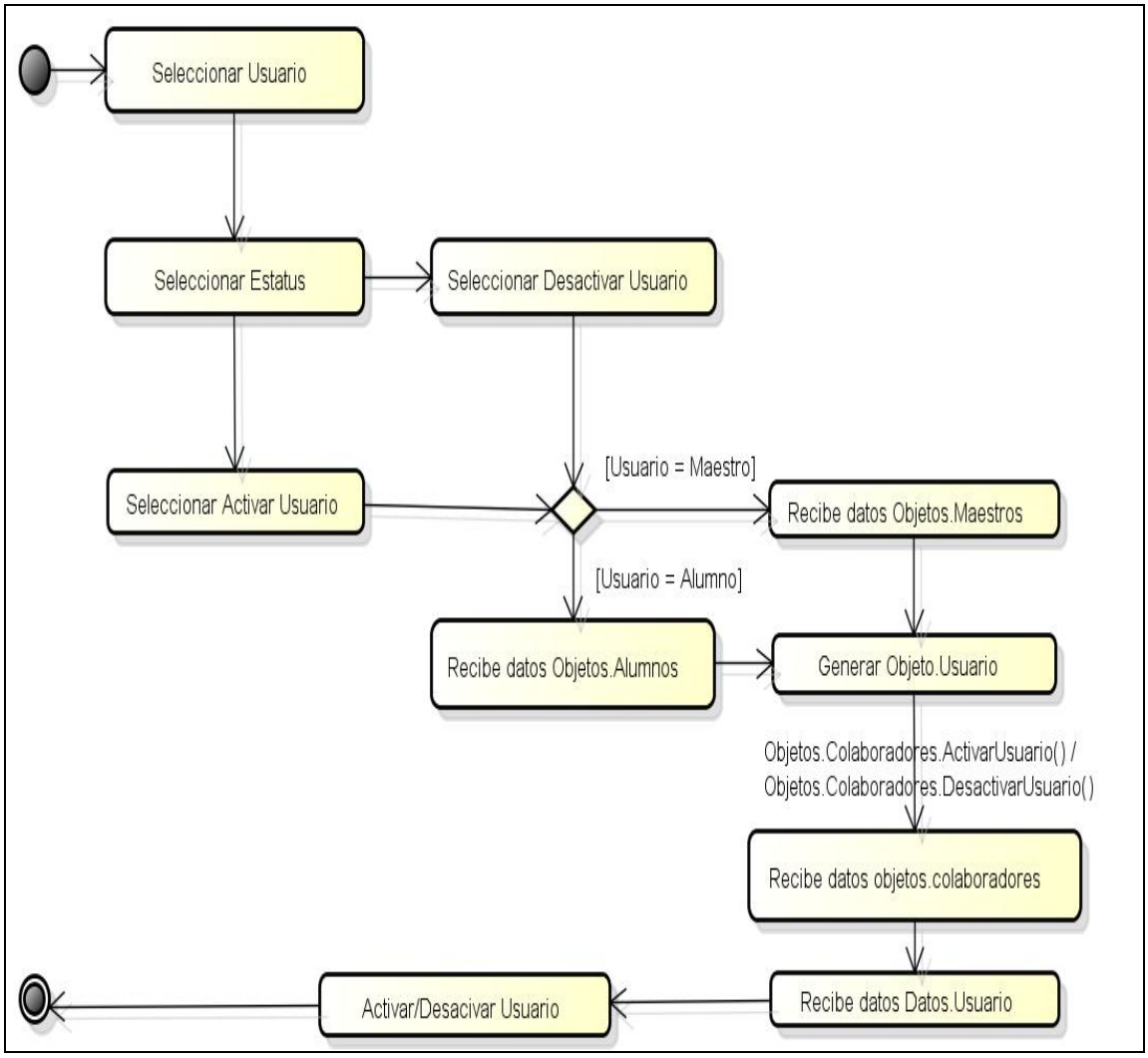

Figura 3.15 Diagrama de actividades para Activar/Desactivar Usuario.

Cabe aclarar que los diagramas de actividades presentados son los más significativos del sistema, que debido a la arquitectura que utiliza, contienen actividades con secuencias similares.

3.5 Descripción de las Interfaces

A continuación se muestran algunas de las interfaces más representativas de la plataforma; las demás se omitieron por motivo de no hacer más extenso este trabajo y debido a que presentan comportamientos y funciones parecidas a las que se presentan a continuación.

3.5.1 Inicio de Sesión

HECODER tiene como pantalla inicial la de inicio de sesión, en la cual un usuario previamente registrado y autorizado por el Administrador General puede autenticarse para poder acceder al sistema. Como se puede visualizar en la figura 3.16, los datos requeridos para tal evento son la clave de usuario y su respectiva contraseña.

Figura 3.16 Pantalla de Inicio de sesión.

3.5.2 Pantalla Principal

Posteriormente al proceso de verificación de autentificación se muestra la pantalla principal, la cual tiene diferentes opciones en su barra de menú según sea el tipo de cuenta con la que el usuario autenticado inicio su sesión (véase Figura 3.17).

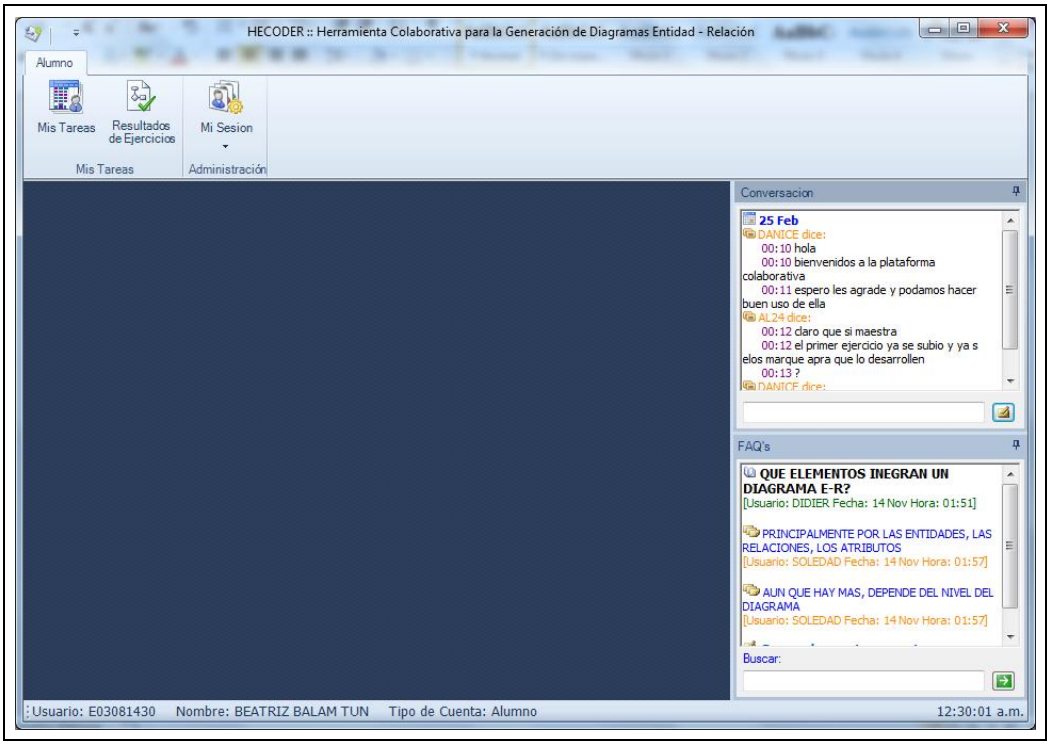

Figura 3.17 Pantalla principal de HECODER.

Como se mencionó anteriormente, la plataforma cuenta con tres tipos de cuenta: Administrador General, Maestro y Alumno. En la figura 3.18 se observa la barra de menú que se despliega cuando un usuario con cuenta de tipo Administrador General inicia su sesión.

Figura 3.18 Menú de Opciones que tiene la cuenta de tipo Administrador General.

El Administrador General tiene la facultad de poder realizar las siguientes actividades:

- Administrar el catálogo de Grupos de Trabajo.
- Administrar el catálogo de Maestros.
- Administrar el catálogo de Alumnos.
- Establecer la configuración de la platilla de la barra de herramientas en el diseñador de ejercicios.
- Respaldar la Base de datos del sistema.
- Restaurar la Base de datos del sistema

Por otro lado, cuando un usuario con cuenta de tipo Maestro ingresa al sistema la barra de menú (véase figura 3.19) cambia, y le despliega otras opciones tales como:

- Administrar el catálogo de Equipos de trabajo.
- Administrar el catálogo de Ejercicios.
- Administrar el catálogo de Tareas.
- Ver avances de ejercicios.
- Retroalimentar ejercicios finalizados

Figura 3.19 Menú de Opciones que tiene la cuenta de tipo Maestro.

Y por último, cuando un usuario con cuenta de tipo Alumno inicia sesión la barra de menú (véase figura 3.20) se despliega con la siguiente lista de opciones de actividades que puede realizar:

- Realizar los ejercicios de las tareas asignadas.
- Ver la retroalimentación de ejercicios.

Figura 3.20 Menú de Opciones que tiene la cuenta de tipo Alumno.

3.5.3 Ejercicios

A continuación se muestra la interfaz para dar de alta un nuevo Ejercicio. En dicha ventana se puede visualizar que los necesarios son: la clave, nombre, el cuerpo del ejercicio y la solución de este por parte del Maestro. Así como los botones de acción para guardar el ejercicio y cerrar la ventana. Se aprecia una hoja donde se puede diseñar el diagrama entidad relación correspondiente al problema a resolver y también la barra de herramientas con los componentes que integran a un diagrama Entidad-Relación básico, como lo son: Entidad, Relación, Atributo y un Conector de Unión.

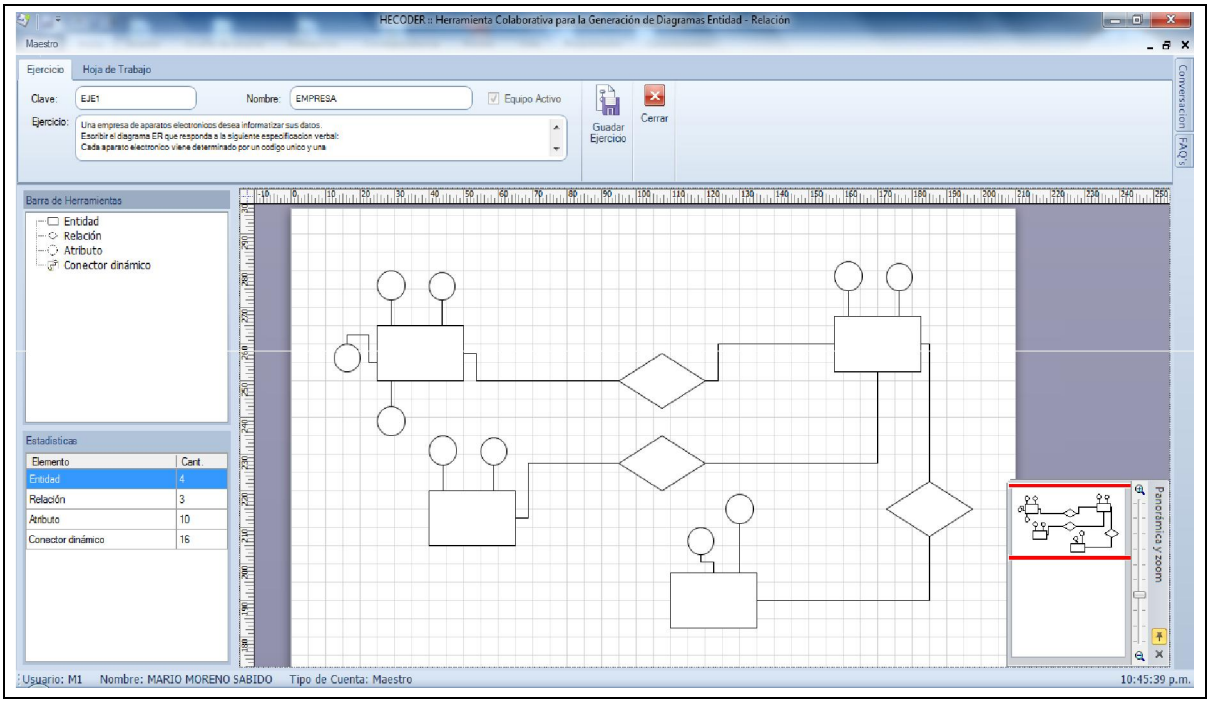

Figura 3.21 Ventana para dar de alta un Ejercicio.

3.5.4 Equipos de Trabajo

Los equipos de trabajo están conformados por alumnos que el maestro agrupa para poder trabajar con ellos. Las figuras 3.22 y 3.23 se puede observar la ventana para dar de alta un nuevo equipo de trabajo, donde se permite ingresar los datos generales del equipo como su nombre y grupo, así como escoger los alumnos que integraran el mismo.

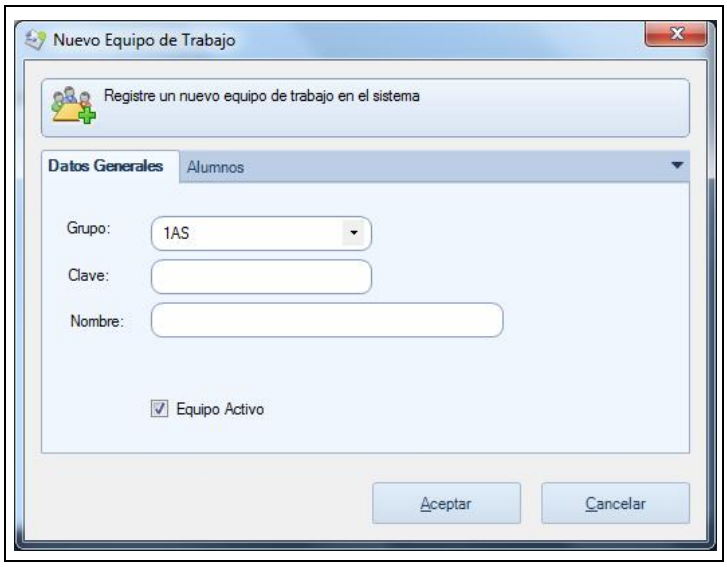

Figura 3.22 Ventana alta de equipo pestaña datos.

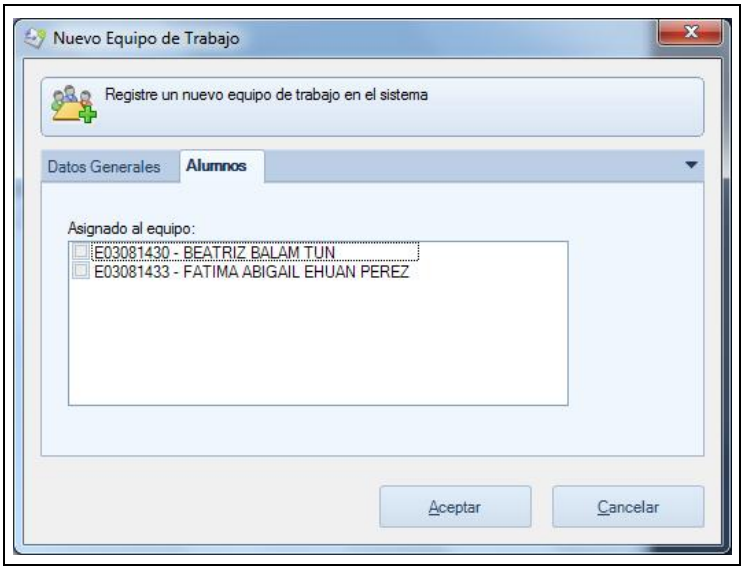

Figura 3.23 Ventana alta de equipo pestaña alumnos.

3.5.5 Usuarios del Sistema

En las figuras 3.24, 3.25 y 3.26 se puede observar la ventana para dar de alta un nuevo usuario del sistema, que puede ser de tipo Maestro o Alumno. Consta de tres pestañas:

- Datos generales
- Grupos
- Seguridad.

La ventana en general tiene los campos necesarios para ingresar los datos personales de un usuario, el grupo al que pertenecerá el usuario en caso de ser alumno y en caso de ser maestro los grupos que tendrá a su cargo.

De igual manera ofrece los campos necesarios para que el usuario tenga acceso sistema, como su Alias y su contraseña.

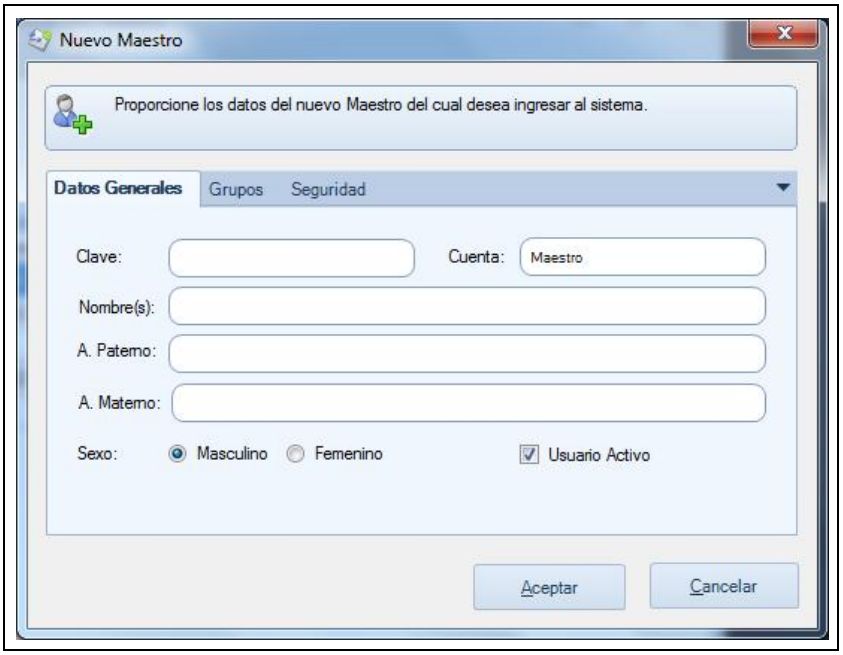

Figura 3.24 Alta de usuario pestaña de datos generales.

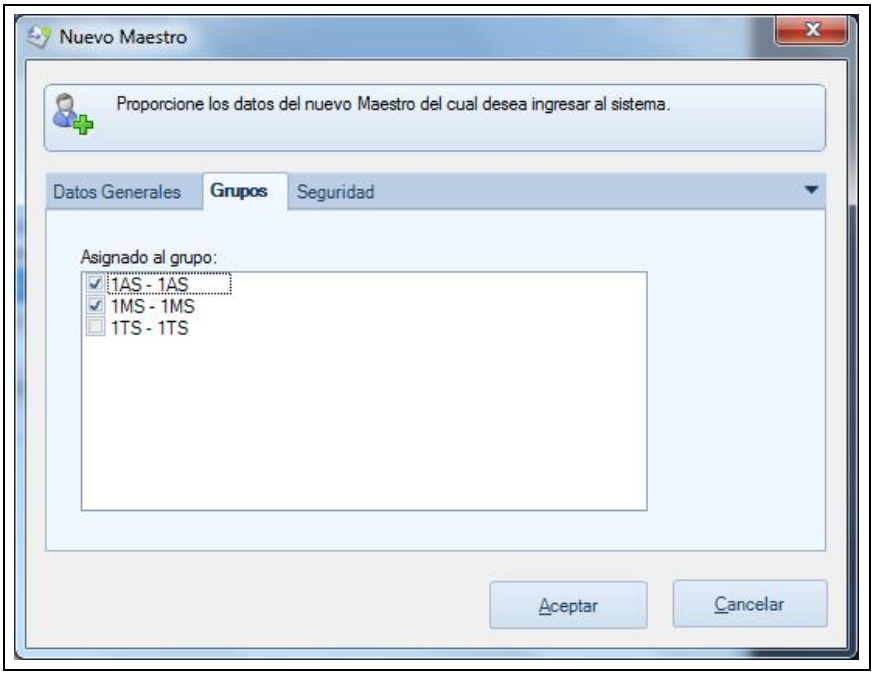

Figura 3.25 Alta de usuario pestaña de Grupos.

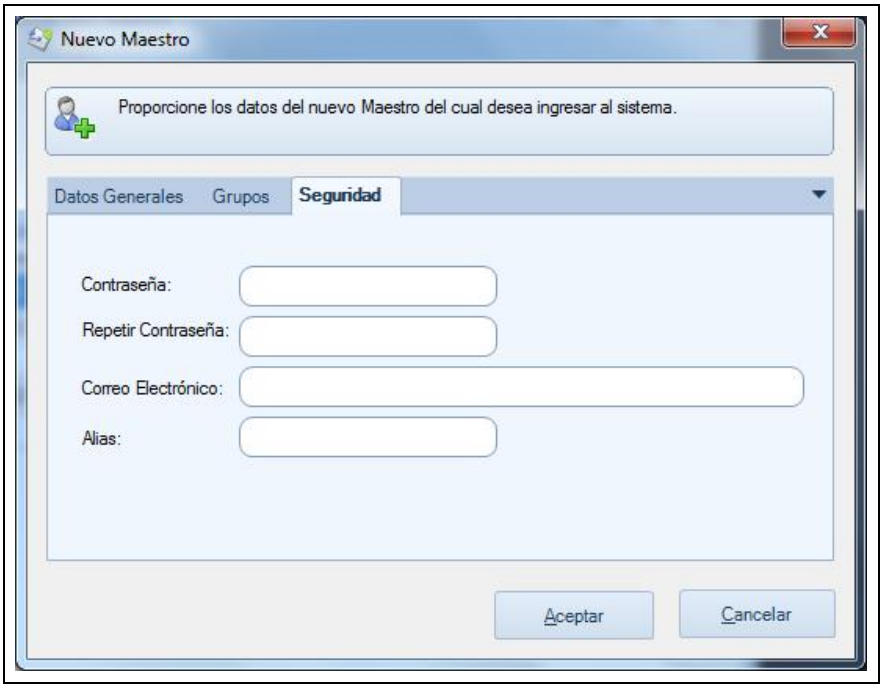

Figura 3.26 Alta de usuario pestaña de seguridad.

3.5.5 Chat

En la figura 3.27 se puede observar la ventana de chat, la cual es para entablar la conversación e intercambio de información entre el maestro con cada uno de los equipos por separado, por lo que ayuda a los integrantes de un mismo equipo a intercambiar puntos sobre la solución de un ejercicio que se esté realizando, o preguntarle al maestro sobre pistas para poder terminarlo correctamente.

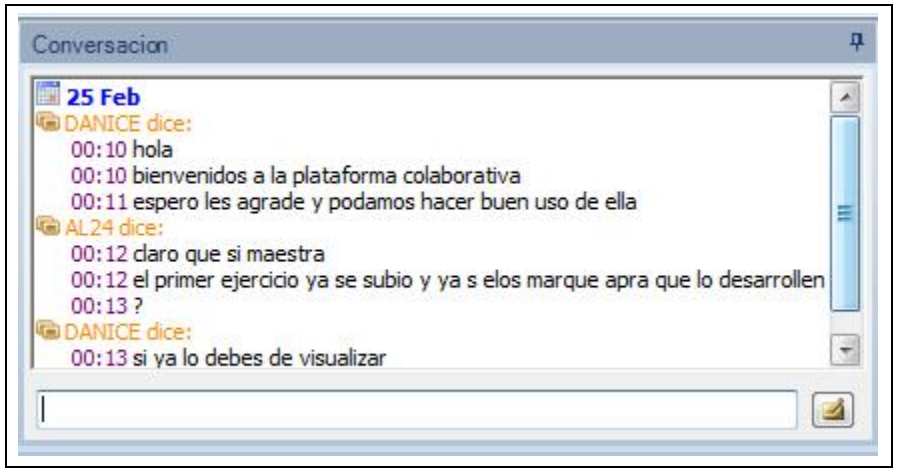

Figura 3.27 Chat.

3.5.6 FAQ's

En la figura 3.28 se puede observar la ventana relacionada con la base de datos del conocimiento o preguntas frecuentes, también conocida como FAQ's. En dicha sección se presenta la opción de realizar búsquedas en el conjunto de preguntas y respuestas sobre algún tema o duda que se tenga por parte del alumno o del maestro; es importante mencionar que los dos usuarios pueden agregar información a dicha base e ir enriqueciéndola.

Por cada pregunta pueden desprenderse infinidad de respuestas. Cada pregunta cómo cada respuesta guarda la fecha y el pseudónimo de quien la creó, esto con la idea de poder identificar quien la realizó.

Figura 3.28 FAQ's.

3.6 Pruebas: Formato de Caja Negra.

Esta estrategia de testing se centra en la verificación de las funcionalidades de la aplicación: Datos que entran, resultados que se obtienen, interacción con los actores, funcionamiento de la interfaz de usuario y en general todo aquello que suponga estudiar el correcto comportamiento que se espera del sistema.

Por tanto, la diferencia fundamental respecto al testing de caja blanca es que en este caso no se trabaja con el código fuente sino con el programa; no importa en este caso cómo se consigue la funcionalidad de un módulo sino que el mismo responda de la manera en que debería según las especificaciones del sistema.

A continuación se presentan algunas de las pruebas que se le aplicaron a la plataforma, que en realidad consiste en una tabla en donde se puede apreciar que caso de uso se está revisando, cual es el objetivo de ese caso, cuales son las entradas y los resultados esperados y los reales.

La tabla 3.3 muestra las pruebas realizadas al proceso de inserción de un Maestro al sistema.

Tabla 3.3 Prueba Inserción de Maestro.

PRUEBA: INSERCIÓN MAESTRO				
Objetivo: Verificar validación de valores de entrada				**Observaciones**
Iteración	**Entrada**	**Resultado Esperado**	**Resultado Real**	
1	Datos correctos	Mensaje guardado	Mensaje guardado	
2	Sin Clave	Mensaje Error	Mensaje Error	
3	Sin Nombre	Mensaje Error	Mensaje Error	
4	Sin Grupos	Mensaje Guardado	Maestro Guardado	
5	Sin Contraseña	Mensaje Error	Mensaje Error	
6	Sin Alias	Mensaje Error	Mensaje Error	

La tabla 3.4 muestra las pruebas realizadas al proceso de inserción de un Alumno al sistema.

Tabla 3.4 Prueba Inserción de Alumno.

PRUEBA: INSERCIÓN ALUMNO				
Objetivo: Verificar validación de valores de entrada				
Iteración	**Entrada**	**Resultado Esperado**	**Resultado Real**	**Observaciones**
1	Sin Clave	Mensaje Error	Mensaje Error	En la iteración número 5. Se imprimió el mensaje incorrecto
2	Sin Nombre	Mensaje Error	Mensaje Error	
3	Sin Contraseña	Mensaje Error	Mensaje Error	
4	Datos completos	Mensaje éxito	Mensaje éxito	
5	Números en campos tipo texto	Mensaje Error	Mensaje éxito	

La tabla 3.5 muestra las pruebas realizadas al proceso de inserción de un Grupo al sistema.

Tabla 3.5 Prueba Inserción de Grupos.

PRUEBA: INSERCIÓN GRUPOS				
Objetivo: Verificar validación de valores de entrada				**Observaciones**
Iteración	**Entrada**	**Resultado Esperado**	**Resultado Real**	
1	Datos correctos	Mensaje guardado	Mensaje guardado	-
2	Clave vacía	Mensaje Error	Mensaje Error	
3	Nombre vacío	Mensaje Error	Mensaje Error	
4	Datos vacíos	Mensaje Error	Mensaje Error	

La tabla 3.6 muestra las pruebas realizadas al proceso de inserción de un Equipo al sistema.

Tabla 3.6 Prueba Inserción de Equipos.

PRUEBA: INSERCIÓN EQUIPOS				
Objetivo: Verificar la inserción de equipos con sus parámetros correctos				**Observaciones**
Iteración	**Entrada**	**Resultado Esperado**	**Resultado Real**	
1	Datos correctos	Mensaje éxito	Mensaje éxito	Es posible insertar equipos sin alumnos
2	Clave vacía	Mensaje Error	Mensaje Error	
3	sin alumnos	Mensaje Error	Mensaje Guardado	
4	Sin nombre	Mensaje Error	Mensaje Error	

CAPÍTULO 4: CONCLUSIONES Y RECOMENDACIONES

En este capítulo se presentan las conclusiones y las recomendaciones sobre el proyecto HECODER. También se mencionan los beneficios obtenidos y los posibles trabajos a futuro para la mejora continua de la herramienta.

CAPÍTULO 4: CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

El producto obtenido de esta tesis fue la herramienta colaborativa para la generación de diagramas Entidad – Relación de Bases de Datos llamada HECODER. Esta plataforma dota al Maestro de una innovadora técnica de enseñanza para temas de bases de datos, ya que brinda la posibilidad de crear ejercicios, de observar los avances que tienen los estudiantes en las tareas marcadas en tiempo real, así como poder tener un contacto en diferente tiempo y modo con los alumnos, eliminando la barrera del aula escolar por un periodo de tiempo determinado, diversificando la forma, lugar y tiempo de como un estudiante participe en la solución de un ejercicio en forma grupal logrando de esta forma la generación de conocimiento.

Los beneficios que proporciona esta herramienta colaborativa son los siguientes:

- **Innovación:** HECODER es una herramienta que moderniza las prácticas de antaño de cómo explicar o impartir clases de los Maestros en las aulas de las instituciones educativas, ya que permite ir más allá de un lugar físico y tiempo establecido para intercambiar conocimientos, puntos de opinión o ideas con sus compañeros para la solución de ejercicios que tengan que realizar.

- **Solución Colaborativa:** HECODER es un software completamente colaborativo, diseñado para trabar de forma colaborativa y no sólo de forma compartida. Se puede acceder a un ejercicio al mismo tiempo y manipularlo o en distinto horario y editarlo.

- **Perfiles de Usuario:** HECODER cuenta con tres tipos de perfiles o cuentas, las cuales son: Administrador General, Maestro y Estudiante, lo que encaja perfectamente en la

administración y emulación de un aula en una institución educativa, en donde cada perfil cuenta con determinadas actividades que puede llegar a realizar.

- **Analizador Automatizado de Soluciones:** HECODER le permite al Maestro realizar un análisis de cada ejercicio resulto por cada equipo de trabajo y compararlo con la solución propuesta resuelta por el Maestro. Dicha comparativa arroja en un resumen las entidades, atributos y relaciones que están presentes en ambas soluciones y son idénticas y señala las que no lo están o están de más. Adicionalmente el Maestro puede agregar comentarios a cada ejercicio resuelto para dar una retroalimentación a los equipos.

- **Tiempo Real:** HECODER es una plataforma que trabaja en tiempo real, las modificaciones que un alumno realice en su hoja de trabajo desde la computadora en la que se encuentre trabajando lo puede visualizar otro integrante de su equipo inmediatamente.

- **Gratuito**: HECODER es un software completamente gratuito que surge por la necesidad del Departamento de Ingeniería en Sistemas Computacionales del Instituto Tecnológico de Motul de contar con una plataforma colaborativa que permita como primera instancia la resolución de forma colaborativa de ejercicios de diagramas de Entidad - Relación de bases de datos, en diferentes variables de tiempo y espacio; sin embargo no cuenta con ningún tipo de licenciamiento, restricción ni exclusividad para su instalación como para su uso.

4.2 Recomendaciones

A partir del trabajo descrito en esta tesis, se proponen una serie de recomendaciones que pueden verse a su vez como mejoras sobre el trabajo presentado, estando algunas de estas ampliaciones dirigidas a suplir las limitaciones presentadas y comentadas en anteriores apartados de este capítulo.

Algunas de estas ampliaciones y mejoras pueden encauzar nuevas líneas de investigación a seguir en el área de aplicación del trabajo.

Sin ningún orden en concreto, las propuestas de trabajo futuro son las que se describen a continuación:

- **Herramienta Multiplataforma:** Proporcionarle al software la posibilidad de poder ser ejecutada en distintas plataformas tecnológicas, tales como celulares, laptops, PC de escritorio, tabletas, etc. Esto con la finalidad de que sea lo más portable y de fácil acceso, y no encasillarla en una sola plataforma o sistema operativo.

- **Plataforma desde la WEB:** Desarrollar una interfaz web que permita acceder a la herramienta desde cualquier lugar. Esto se lograría reutilizando las capas de negocio y datos ya existentes, modificando únicamente la capa de presentación. Y esto propiciaría un acceso 100% online y permitiría a todos los usuarios independientemente del lugar en donde se encuentren puedan acceder a ella.

- **Incluir Reportes e Indicadores de Colaboración:** Desarrollar el módulo de reportes e indicadores para poder explotar la información que se genere por medio de la plataforma, con el propósito de poder tener métricas y poder medir en alguna medida el porcentaje de colaboración que se esté suscitando en los trabajos marcados dentro y fuera del aula, para tratar de identificar factores que ayuden o en caso contrario influyan en los procesos colaborativos.

- **Optimizar el rendimiento:** Mejorar el rendimiento de la plataforma a la hora de estar trabajando por medio de nuevas tecnologías que se puedan aplicar a la capa de datos y que mejoren de manera notable el desempeño de las consultas, búsquedas, transacciones, etc., para lograr una mejor experiencia al usuario a la hora de su interacción.

REFERENCIAS

Apache. (2012). *About*. Recuperado el 04 de Febrero de 2013, de Apache: HTTP Server Project: http://httpd.apache.org/ABOUT_APACHE.html

Bortnik, S. (04 de Junio de 2010). *Gliffy.com: la mejor herramienta de diagramas online*. Recuperado el 04 de Febrero de 2013, de bitelia: http://bitelia.com/2010/06/gliffycom-la-mejor-herramienta-de-diagramas-online

Cinergix. (12 de Noviembre de 2012). *Creately - Online Diagramming*. Recuperado el 04 de Febrero de 2013, de Chrome Web Store: https://chrome.google.com/webstore/detail/creately-online-diagrammi/figjjaggcjcojopflaabmebmocabdglm/details

DevComponents. (2012). *DotNetBar = Professional Applications*. Recuperado el 04 de Febrero de 2013, de DevComponents: http://www.devcomponents.com/dotnetbar/

Dillenbourg, P. (1999). *Collaborative Learning Cognitive and Computational Approaches.* New York: Pergamon Earli.

EducaconTIC, L. (23 de Enero de 2012). *Cacoo, diagramas en colaboración*. Recuperado el 04 de Febrero de 2013, de EducaconTIC: http://www.educacontic.es/blog/cacoo-diagramas-en-colaboracion

IEEE, T. I. (1990). IEE Standard 610.12-1990. *IEE Standars Collection: Software Engineering*.

Larman, C. (2003). *Uml y patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado.* (Segunda ed.). Madrid, España: Prentice Hall.

Lucero, M. M. (2003). Entre el Trabajo colaborativo y el Aprendizaje Colaborativo. *Revista Iberoamericana de Educación*.

Moyano, J. M. (s.f.). *Diagramas de actividad y diagramas de estados.* Recuperado el 04 de Febrero de 2013, de CTR - Computadores y Tiempo Real: http://www.ctr.unican.es/asignaturas/procodis_3_II/Doc/stateDiagram.pdf

MySQL. (2012). *1.4. Panorámica del sistema de gestión de base de datos MySQL*. Recuperado el 04 de Febrero de 2013, de MySQL 5.0 Reference Manual :: 1 Información general: http://dev.mysql.com/doc/refman/5.0/es/what-is.html

Oktaba, D. H. (1999). *Tecnológia Orientada a Objetos*. Recuperado el 03 de Febrero de 2013, de Capítulo 2: Introduciendo a el UML: http://uxmcc1.iimas.unam.mx/~cursos/Objetos/Cap2/cap2.html

Otazu, R. Q. (13 de Julio de 2007). *¿Que es la ingenieria de Software?* Recuperado el 2013 de Febrero de 03, de Blog de Rodolfo Quispe-Otazu: http://www.rodolfoquispe.org/blog/que-es-la-ingenieria-de-software.php

Partners, M. S. (11 de Noviembre de 2009). *Tecnologias Microsoft.* Recuperado el 03 de Febrero de 2013, de Día 3: Introducción a la programación en 3 Capas: http://mredison.files.wordpress.com/2009/11/viernes13noviembre2009.pdf

Pedro. (2011). *Capítulo 1: Fundamentos de programación.* Recuperado el 03 de Febrero de 2013, de La Página de Pedro V: http://www.pedrov.info/imprimir/ProgramacionCSharp_01.pdf

PhpMyAdmin. (Septiembre de 2008). *About*. Recuperado el 04 de Febrero de 2013, de PhpMyAdmin: http://www.phpmyadmin.net/home_page/index.php

Pressman, R. S. (2005). *Ingenieria del Software: Un Enfoque Practico* (Sexta ed.). Mc Graw Hill.

Team, T. P. (2012). *Introducción*. Recuperado el 04 de Febrero de 2013, de Documentación de phpMyAdmin: https://phpmyadmin-spanish.readthedocs.org/en/latest/intro.html

WarNov. (7 de Septiembre de 2012). *Visual Studio 2012 – El Lanzamiento!* Recuperado el 03 de Febrero de 2013, de MSDN Blogs: http://blogs.msdn.com/b/warnov/archive/2012/09/07/visual-studio-2012-el-lanzamiento.aspx