Metodología para desarrollo de un Sitio *Web*

Rosalba Juárez & Felipe Lara-Rosano

CCADET/UNAM
Circuito exterior s/n cp. 04510, Ciudad de México, México

rosalba.juarez@ccadet.unam.mx;flararosano@gmail.com

RESUMEN

En este trabajo se presenta una metodología para desarrollar un sitio *web, el cual* comprende páginas para presentar información en Internet. Se han desarrollado dos grupos de páginas *web:* una parte frontal (*Front end*) que despliega la información visible para todo el público, y un grupo que se encuentra detrás (*back end*), en donde se ubican las paginas en las que se puede cambiar la información. La herramienta para la realización de diagramas es el *Unifed Modeling Languaje (UML)*. Este trabajo se enfoca principalmente en la parte que se encuentra detrás, aunque también se involucra parte de la frontal. Se presenta una forma de obtener la información, la realización y análisis de los diferentes diagramas que conforman su diseño. Se toma como ejemplo el Seminario de Teorías, Métodos y Modelos de la Complejidad Social(Seminario TMMCS) impartido por Felipe Lara Rosano.

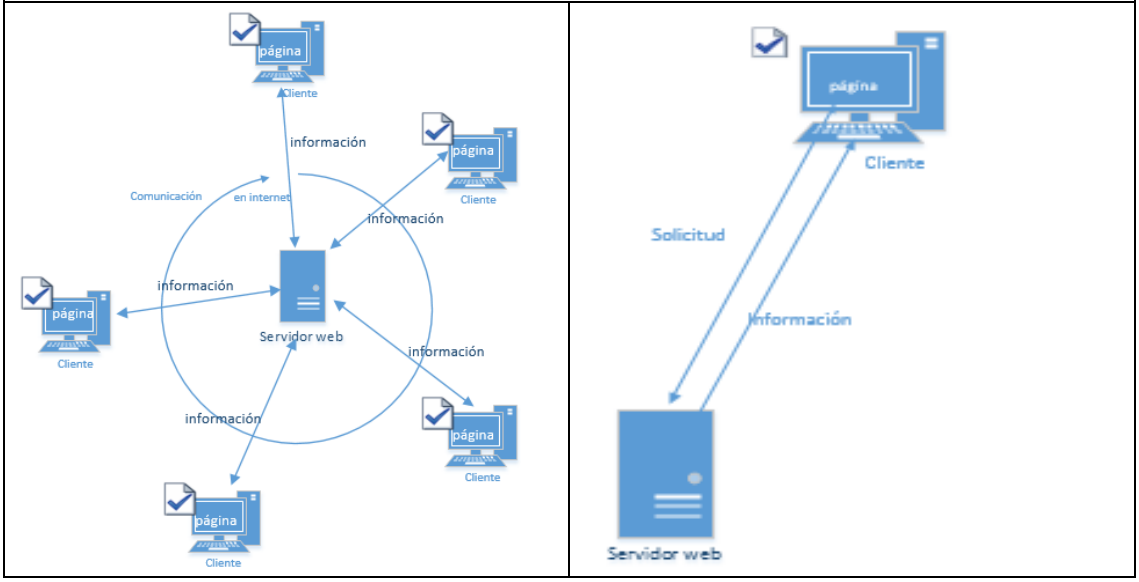
PALABRAS CLAVE: desarrollo, sitio *web*, *unifed modeling languaje*, diagramas.

INTRODUCCIÓN

La *World Wide Web* es un sistema de información y comunicación con base en *hipertextos* (enlazar una palabra con otra página), usado de manera popular en la *Internet*, con comunicaciones de datos que operan de acuerdo con un modelo cliente/servidor (Figura 1. cliente servidor)[1].

La función de un sitio *web* es cubrir las expectativas del usuario (persona que tiene información de lo que se quiere presentar en el sitio *web* y pueda tomar decisiones). Para desarrollar un sitio web es necesario definir el problema y su alcance. Ello implica realizar entrevistas y/u observación continua con el usuario; definir la tecnología que se tiene para su desarrollo, las herramientas con las que se cuentan y el conocimiento que se tiene de ellas.

La Figura 1 esquematiza modelo cliente servidor.



OBJETIVO.
El objetivo general del proyecto en el que se inserta este trabajo es desarrollar un sitio web que facilite el funcionamiento de un grupo heterogéneo de alumnos en el contexto del (Seminario TMMCS). Los objetivos particulares del sitio web son proporcionar un espacio donde el alumno encuentre información del seminario, por ejemplo tareas con la fecha que se dejó y fecha de entrega; proyectos que están manejando cada uno de los integrantes del grupo, herramientas de software que se manejan en el (Seminario TMMCS), etc.

El objetivo particular de este trabajo es presentar el método de diagramación dentro del proceso de desarrollo de dicho sitio.

MÉTODO
El método para desarrollar nuestro sitio web consiste en lo siguiente:

1. Recopilación de información del usuario, para definir el problema y su alcance, mediante entrevistas y/u observación continua.
2. Definición de la tecnología para el desarrollo.
3. Diseño de diferentes diagramas para analizar la solución.
4. Maquetación web (distribución de los elementos en la página web), que no se presentará en este artículo.

1. Recopilación de información del usuario, para definir el problema y su alcance, mediante entrevistas y/u observación continua.
Esta etapa consiste en entender las necesidades del usuario, aterrizar sus ideas y definir sus objetivos, así como saber qué información quiere transmitir en el sitio *web*. Es importante aclararle todas y cada una de sus dudas y mostrarle alternativas de soluciones posibles, lo cual requiere de habilidad de comunicación para transmitir ideas en forma clara y oportuna. Se pueden usar varias formas de obtener la información, desde entrevistas con el usuario, cuestionario previo, con lluvia

de ideas, por observación o el método *Team Kawakita Jiro (TKJ),* un método sistémico, creativo y participativo en el que se busca la solución de un problema.
En nuestro ejemplo las necesidades del usuario se detectaron por observación: se asistió al seminario para ver cómo estaban organizados los grupos y cuáles eran las tareas realizadas dentro y fuera del salón de clase.
Este seminario es bastante heterogéneo, cuenta con personas que trabajan con las siguientes especialidades; música, educación, economía, filosofía, derecho, innovación, entre otras. Por esta razón es indispensable que el sitio *web* cuente con información casi personalizada para las diferentes áreas que se manejan; las tareas son generales pero se personalizan dependiendo del área que trabaje el alumno, también los alumnos son de diferente nivel ya sea licenciatura, maestría o doctorado. A cada alumno --para permanecer en el **(Seminario TMMCS)**-- se le pide que aporte un proyecto referente a un problema real donde aplique la teoría que vaya aprendiendo en la clase. Asimismo se desarrollan talleres tanto de las teorías cubiertas en el Seminario como de herramientas de software.
Al analizar la información que tenemos del Seminario llegamos a la definición del objetivo del sitio *web*: será un espacio donde el alumno encuentre información de la clase, tareas con la fecha que se dejó y la fecha de entrega, qué proyectos están manejando cada uno de los integrantes del grupo y herramientas de software que se manejan en el seminario.

2. Definición de la tecnología para el desarrollo.
Se usara la herramienta *Unifed Modeling Languaje (UML)***.**
En lo que respecta a la arquitectura del servidor y su sistema operativo, se puede utilizar una pc de escritorio con caracteristicas suficientes para que responda rápido, pero no se recomienda esta opción porque, aunque tengamos el hardware robusto, no se tiene la misma arquitectura que un servidor. El sistema operativo que se recomienda es Linux en versiones actualizadas y estables.
El protocolo de comunicación del sitio es el *CGI (Common Gateway Interface)*, con el servidor web Apache. El manejador de base de datos puede ser ***mysql*** y ***postgress.*** Por ahora *mysql* se ha posicionado como el favorito.
Los lenguajes de programación *html, javascript, php, java*, son los más usados y generalmente se elige una combinación de éstos.

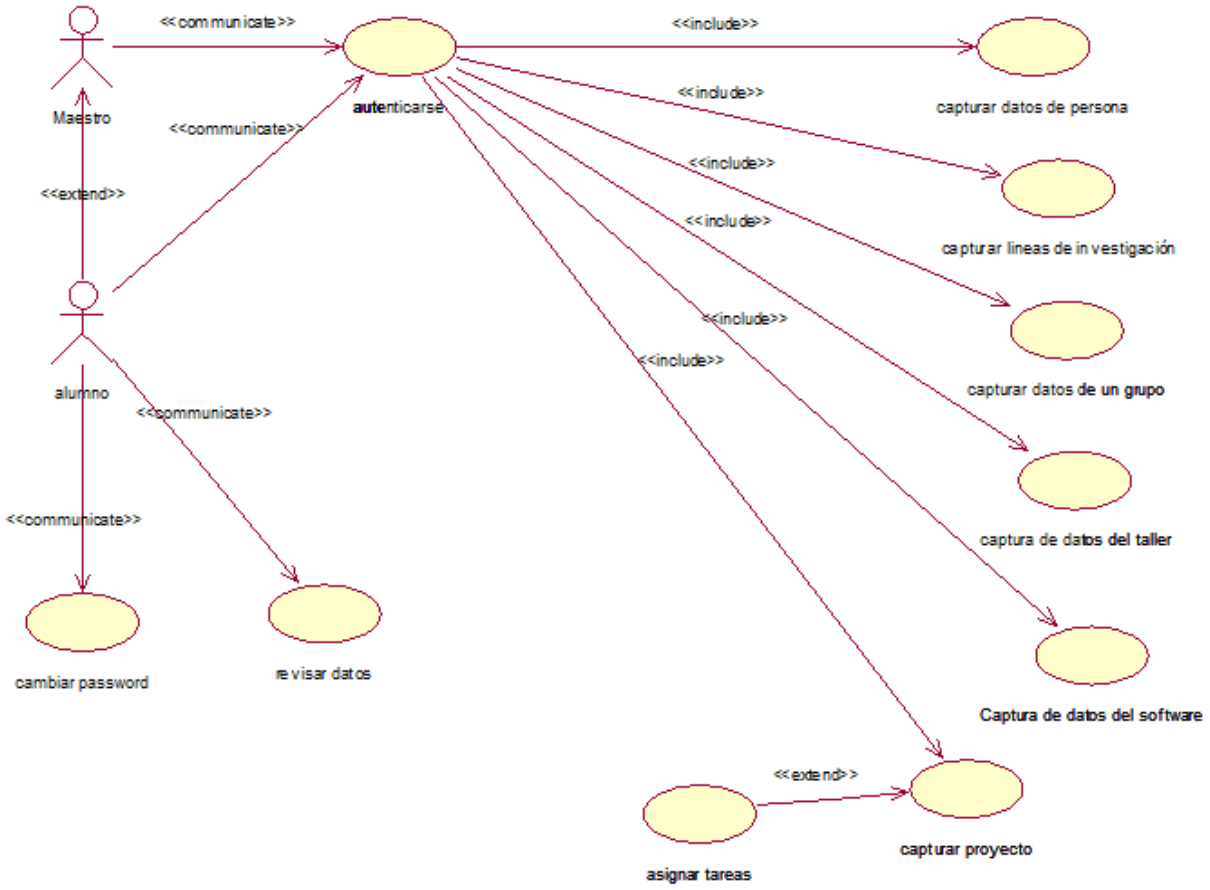
3. Diseño de diferentes diagramas para analizar la solución.
Para poder diseñar un sitio web se utilizan herramientas *Unifed Modeling Languaje (UML)*, también llamadas herramientas *Computer Aidet Sftware Engineerin (CASE)*. Este es un "lenguaje de modelado visual" que utiliza diagramas o modelos, para describir métodos o procesos de un sistema, algunas alternativas gratuitas son: *StarUML*, *ArgoUML*, *BoUML*, *FrameUML* y *Tiny UML*.
Para este ejemplo se utilizó *Rational Rose* basado en cuatro tipos modelos: *casos de uso, dinámico, lógico y físico*.

El modelo de casos de uso describe la funcionalidad del sistema desde la pespectiva del usuario y cómo se relaciona con su entorno. Contiene representaciones de los actores (usuarios del sistema) y las acciones que pueden llevar a cabo en el sitio web. Dentro de los elementos que posee este diagrama se definen: *casos de uso* que es una operación completa desarrollada por los autores y el sistema mediante un dialogo. En el diagrama se puede poner un nombre significativo, como algún caso de uso donde asociarlo. Se pueden presentar cuatro tipos de relaciones: *comunicación* (relación entre un actor y un caso de uso), *extensión* (un caso de uso dado puede extenderse a otro caso de uso) , *inclusión* (un caso de uso puede incluirse en otro caso de uso) y *generalización* (se define la relacion entre un caso o varios casos de uso) .

En nuestro ejemplo elaboró un diagama de casos de uso, con dos actores (Figura 2), el maestro y el alumno, que tienen una relacion extendida de comunicación para autenticarse. Al autenticarse, el maestro puede registrar a los alumnos, capturar lineas de investigacion, crear grupos, dar de alta

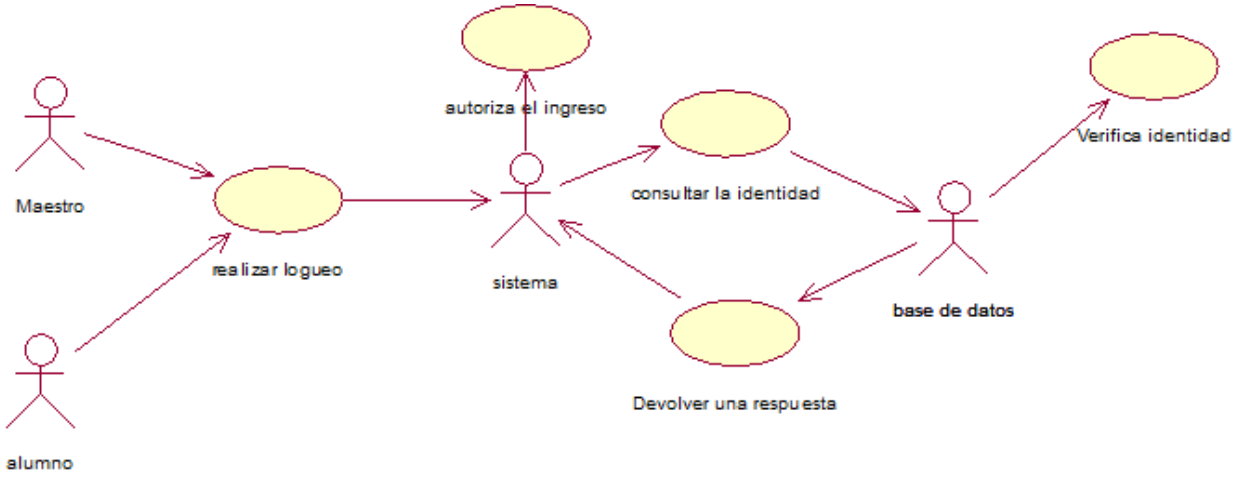
un taller, capturar datos de *software* que tienen relaciones de generalización. En los proyectos hay una relacion de extension entre proyectos y se puede asignar tareas al proyecto.

Figura 2.- Casos de uso.



La figura 3 representa el diagrama de autenticación. Es un nuevo diagrama de casos de uso. En *rational rose* seleccionamos *Use case view*; con el clic derecho del ratón se abre un submenú y seleccionamos *new;* damos un clic a *Use case Diagram*, dando doble clic con el ratón en el botón izquierdo podemos cambiar el nombre a nuestro diagrama. En este caso vamos a crear cuatro actores: el maestro, el alumno, el sistema y la base de datos. Creamos un caso de uso para realizar la autenticación o *logueo.* Tiene que haber una relacion tanto del maestro como del alumno. Se realiza el *logueo* al sistema por lo que ponemos otra relación, se crea otro caso de uso para que el sistema autorice el ingreso y lo relacionamos. Creamos dos caso de uso: para que consulte la identidad y para devolver una respuesta. Las relaciones quedarían entre el sistema a consultar identidad y de consultar identidad a base de datos; de base de datos a devolver una respuesta y de ésta al sistema, se crea otro caso de uso llamado verifica identidad que tiene relación con la base de datos.

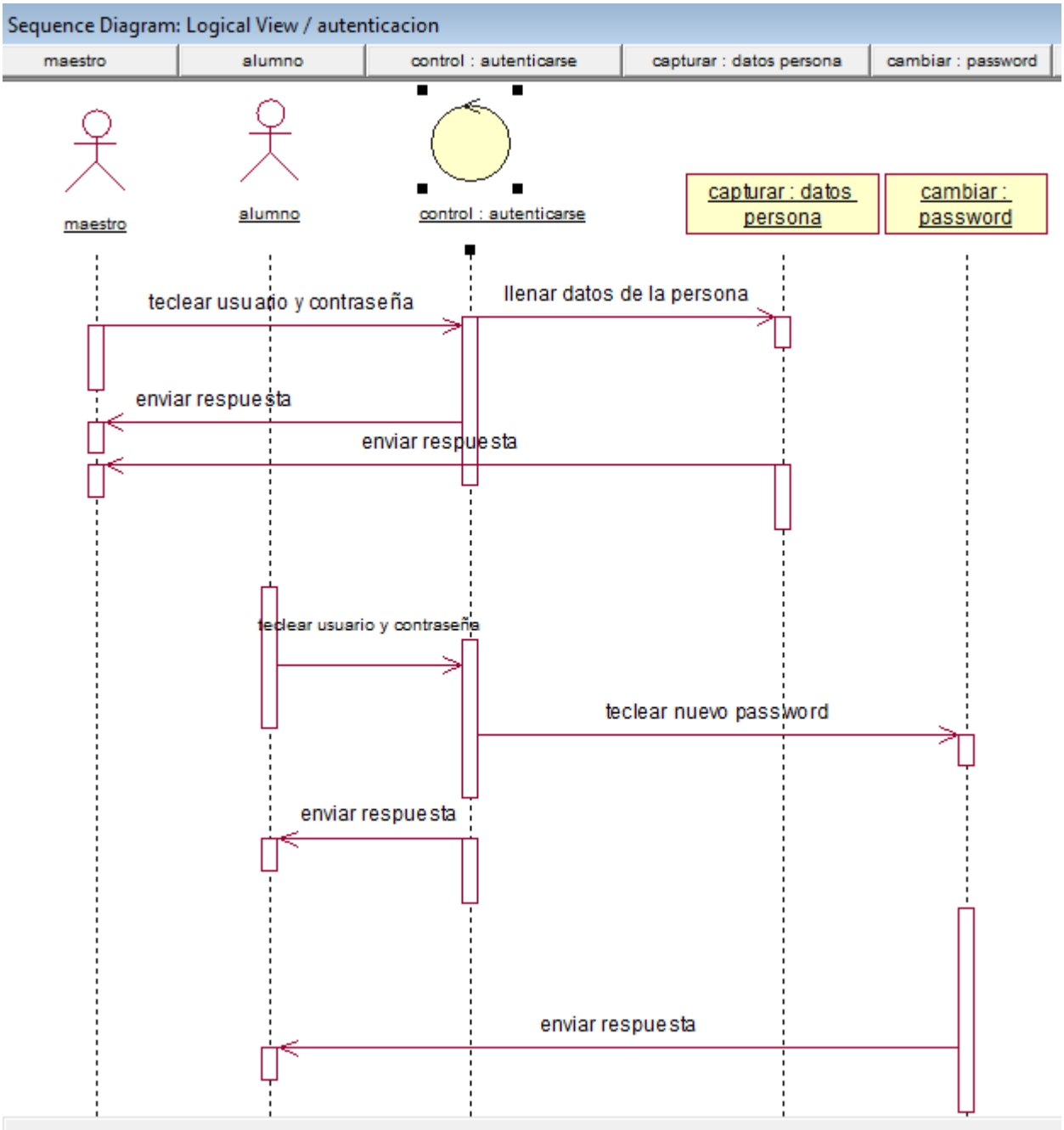
Figura 3.- Diagrama Autenticación



Así se realizarían todos los diagramas de casos de uso, con esta herramienta.

El modelo dinámico se usa para modelar el comportamiento del sistema a lo largo del tiempo. Se pueden realizar diagramas de secuencia, colaboración y de actividades. El diagrama de secuencias muestra la interacción de un conjunto de objetos a través del tiempo, es importante porque puede dar detalles del diagrama de casos de uso. Los elementos que lo compone son los objetos, las líneas de vida de un objeto, las activaciones y los mensajes. Un objeto se representa con un rectángulo que contiene el nombre y la clase del objeto, se distinguen tres tipos, objeto entidad, objeto relación y objeto control.
Las líneas de vida se representada por una línea punteada con rectángulos que representan la ejecución de métodos.
Las activaciones muestran el periodo de tiempo en el cual el objeto se encuentra desarrollando alguna activación y se denotan por un rectángulo delgado sobre la línea de vida del objeto; por último se presentan los mensajes representados por una línea sólida desde el objeto que emite el mensaje hasta el objeto que lo ejecuta.
En nuestro ejemplo vamos a crear el diagrama de secuencias (4.- Diagrama de secuencias) para el alumno y el maestro relacionado con la autenticación. Lo primero es crear el diagrama en *rational rose*, seleccionamos *logical view*, con el *clic* derecho del ratón *new*, elegimos *sequence digram*, lo nombramos autenticación. Se muestra el área de trabajo en la parte derecha, en la izquierda tenemos los elementos que vamos creando, se selecciona el elemento maestro se arrastra hacia la parte derecha y lo mismo hacemos con alumno. Creamos el tercer objeto que va a ser de tipo control (transitorios) lo nombramos *autenticarse*. Los siguientes objetos que creamos son de tipo *entidad* -- se presentan cuando el sistema necesita almacenar datos o vincular la aplicación con el entorno – Les asignaremos el nombre de: *capturar datos de la persona* y *cambiar password*. El siguiente objeto es de tipo *mensaje* – seleccionando la flecha de la barra de herramientas que se encuentra en medio de la parte derecha e izquierda –. Se crean la *activación* que se estará realizando entre los diferentes objetos, por ejemplo el maestro al autenticarse se crea un mensaje y se describe la actividad a realizar. En nuestro caso será *teclea usuario y contraseña*; se crea otro mensaje de autenticarse al capturar datos de la persona y se describe la actividad a realizar que es *llenar datos de la persona*, para terminar una línea de vida se pone una X.

4.- Diagrama de secuencias



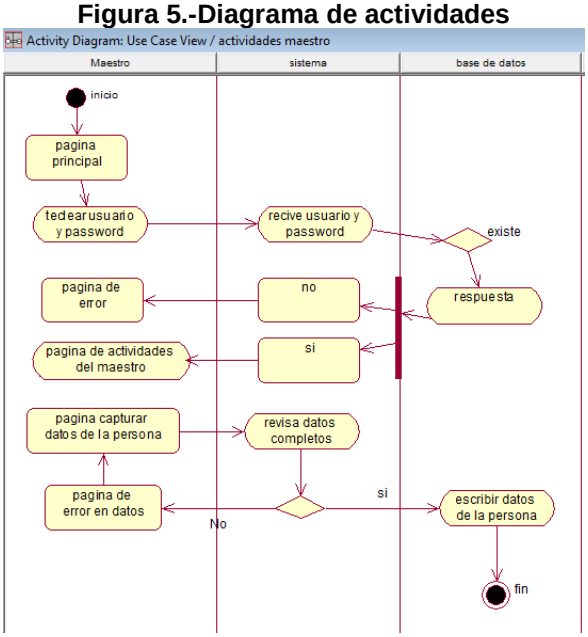
Una vez realizado el diagrama de secuencias la herramienta *rational rose* crea el diagrama de colaboración, seleccionamos de la barra de menú *Browse* y damos clic a créate *collaboration diagram*, lo presenta en forma desorganizada así que seleccionando cada objeto lo movemos donde se pueda ver bien, por medio de la numeración y los mensajes se sigue la secuencia. Así estaríamos realizando todos los diagramas para cada uno de los casos de uso.

Diagrama de actividades (Figura 5.-Diagrama de actividades) [2]. Se usa para mostrar cómo se produce el flujo de trabajo o los procesos dentro del sistema, los caminos que se pueden tomar desde el inicio al final y se puede especificar quién realiza las acciones. Los elementos que componen el diagrama de actividades son los siguientes: a) *Las actividades* –representadas por rectángulos con bordes redondeados y dentro se puede poner una descripción breve de la actividad— b) *Los flujos* se representan con flechas que indican el sentido de su dirección, las *bifurcaciones condicionales* –se especifican con un rombo, al interior se puede poner una descripción—c) las *acciones concurrentes* o *barras de sincronización* – se dibujan con barras gruesas—

Para nuestro ejemplo vamos a crear el diagrama de actividades que describe la autenticación, por parte del maestro y cómo registra los datos de las personas.

En *rational rose* seleccionamos *use case view* damos un *clic* a *new* y seleccionamos *activity diagram*. Le asignamos el nombre de actividades del maestro. En nuestra ventana de trabajo, seleccionamos el ultimo icono de la barra que es el *swimlane* – se hace una separación, dando doble *clic* en la parte superior podemos poner un nombre – creamos tres que, para nuestro ejemplo, son *maestro, sistema* y *base de datos*. Seleccionamos el punto de color negro para poner el *estado inicial* en maestro, seleccionamos un estado y le ponemos *página principal*, unimos *estado inicial* con *página principal* con una flecha, creamos una *actividad* que se llama *teclear usuario y password* y lo unimos con un *flujo* desde de *página principal*. En la parte del sistema agregamos una *actividad* y unimos el *flujo* con la *actividad* creada. En la parte de *base de datos* creamos una *bifurcación*; dando doble clic le ponemos *existe*, la base manda respuesta de si o no, para esto vamos a usar una sincronización vertical entre sistema y base de datos, unimos los *flujos* correspondientes, cuando la respuesta es no al maestro se le envía una *página de error* y unimos el flujo desde *no* a *página de error* Por el contrario si la respuesta es sí enviamos *página de actividades del maestro* y unimos el *flujo*. El maestro puede estar en el estado *página capturar datos de la persona*, el sistema revisa que los datos sean adecuados, creamos una *bifurcación*. Si están bien los datos escribe a la base de datos la información, si están mal envía un error al maestro, agregamos el final en la base de datos.

Figura 5.-Diagrama de actividades



Modelo Lógico (Figura 6.-Modelo lógico) [3] es una vista estática de los objetos y las clases que cubren el espacio de análisis y diseño del sitio *web*, mostrando sus clases y atributos entre ellos. Contiene como elementos las clases y las relaciones. La *clase* es una unidad básica que encapsula la información de un objeto representada por un rectángulo que posee tres divisiones: parte superior –*nombre de la clase*--, en la de en medio – los *atributos que son las características de la clase* y pueden ser de tres tipos: públicas, privadas y protegidas--
En el tercero contiene los *métodos u operaciones*—pueden ser públicos, privados o protegidos--los cuales se usan para interactuar con su entorno. Otros elementos que contiene son *las relaciones* –herencia, relación y asociación-- que poseen *cardinalidad* (cómo se relacionan): de 1 a 1, de 1 a muchos, de muchos a muchos.
En nuestro ejemplo vamos a crear la parte del maestro que ingresa datos de una persona Entonces, seleccionamos en *rational rose use case view*→ *new* → *class Diagram*, le damos el nombre de clase *maestro*, para ello, en nuestra área de trabajo con la barra intermedia seleccionamos una clase y damos clic en el área de trabajo, le asignamos el nombre de *maestro*, otra con el nombre de *conexión* y una con *sistema*. Seleccionamos la clase *maestro* con un clic derecho y se despliega un submenú donde seleccionamos *new attribute.* Le asignamos el nombre de *usuario*, agregamos otro atributo y le damos el nombre de *password* estos van a ser privados. Con el *clic* derecho sobre la clase ahora seleccionamos *new operation* el maestro realiza la operación *autenticarse* y el registro de la persona. De la misma manera le damos sus atributos y operaciones a las clases conexión que se encarga de la base de datos y a la de sistema que revisa si los datos existen. Las cardinalidades van a ser de uno a muchos entre las clases maestro-sistema y sistema-conexión.

6.-Modelo lógico

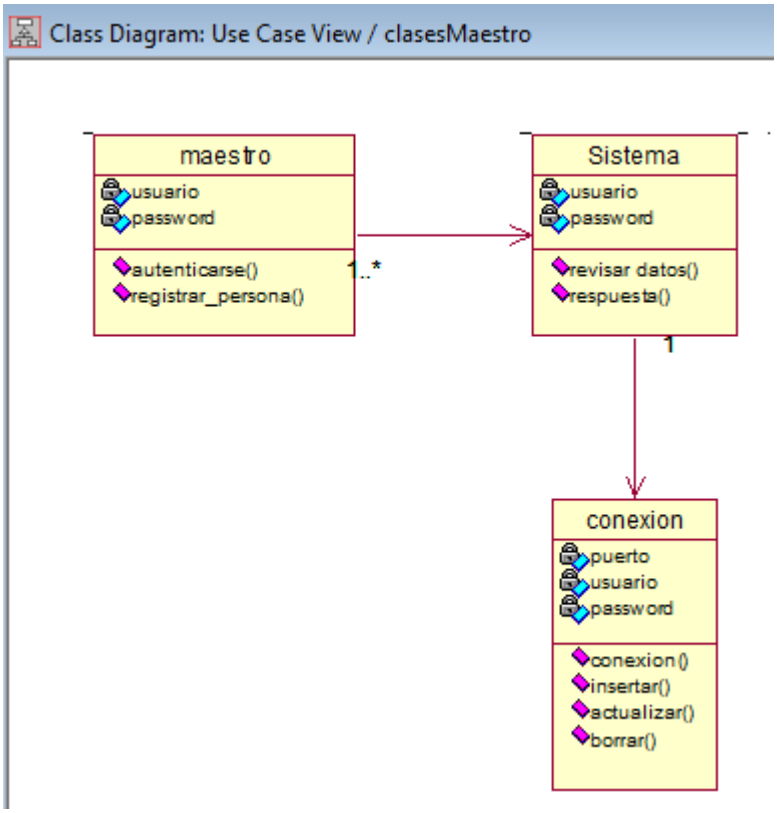


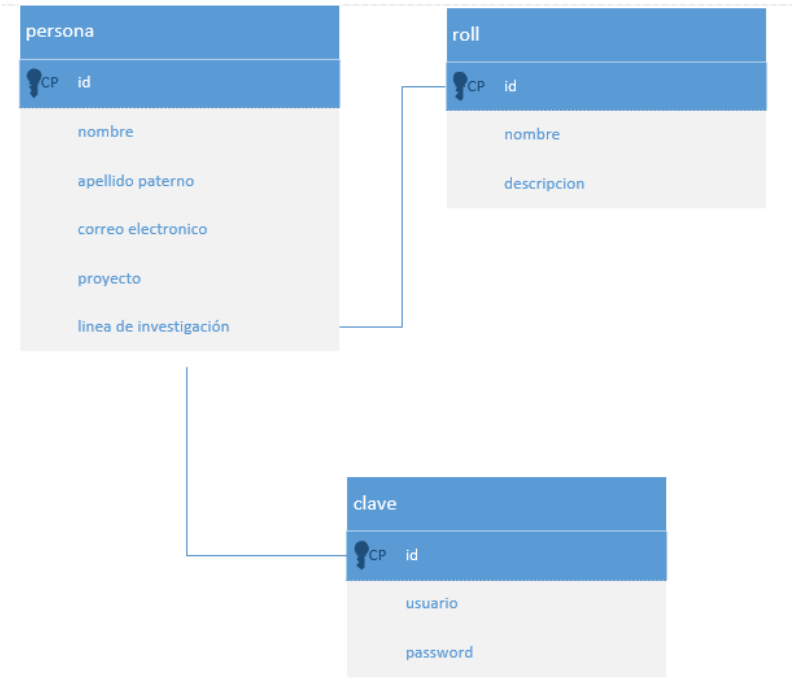
Diagrama de base de datos relacional (Figura 7.- Base de datos relacional**)** [4] es un conjunto de tablas o entidades –las columnas son el nombre de los campos y los renglones son donde se guardan los datos en su campo correspondiente-- se van a guardar los datos dentro del servidor por medio de un manejador de base de datos—intermediario—. Entre los gratuitos más conocidos en sistema operativo *Linux* son: *postgres* y *mysql*, para sistema operativo *Windows* está *open office* con *Access* como manejador de base de datos, el que incluye una parte grafica para realizar la base de datos.

Para nuestro ejemplo del maestro vamos a crear la base de datos relacional con una herramienta que se llama *Visio* que se usa en sistema operativo *windows,* es muy amigable y se pueden realizar varios tipos de diagramas. Solo vamos a usar tres tablas que son la tabla *personas* –donde se van a incluir todos los datos que se quiere guardar de la persona—*rol* que es el papel que juega cada actor dentro del sitio web y *claves* que es donde se guardaran los datos de usuario y *password*.

Entramos a *Visio* → bases de datos→notación de base de datos UML→ crear.

Tiene dos partes la parte izquierda donde se muestran los elementos que tenemos y la parte derecha que es nuestro espacio de trabajo. En la parte izquierda tomamos el elemento *entidad* con un *clic* sin dejar de oprimir lo pasamos a el espacio de trabajo. Dando doble *clic* a esta entidad podemos cambiar su nombre para este va a ser *persona*; seleccionamos el siguiente renglón donde se crea un identificador único, en la siguiente parte se crean los datos que se quieran guardar de esa persona, en nuestro caso van a ser *nombre, apellido paterno, apellido materno, correo electrónico, proyecto, línea de investigación*. Para escribir cada uno en *Visio* seleccionamos en la entidad el último atributo damos *clic* derecho y seleccionamos *insertar atributo*, le damos un nombre, así agregamos todos los atributos que necesitemos. De la misma manera agregamos nuestra siguiente entidad que sería *roll*, con el *id* que es un identificador único, el nombre y la descripción. Agregamos la entidad de *clave* con los campos de id, usuario y *password*. Las relaciones se realizan entre persona y *roll*—donde *roll* le cede su *id* a persona--, persona y clave –donde clave cede su *id* a persona.

Figura 7.- Base de datos relacional



Hasta este momento terminamos una parte del diseño, del ejemplo del maestro que si recordamos es donde se presenta la solución por medio de un conjunto de diagramas, utilizando la metodología de *UML* y ayudándonos de la herramienta *Rational rose* y *Visio*, Falta la parte de maquetación donde crearíamos la plantilla(s) que van a verse en el sitio, tanto para el *font end* como para el *back end.*

CONCLUSIONES

El intercambio de información en comunidades heterogéneas es un desafío que adquiere importancia en el contexto académico de corte interdisciplinario. En este trabajo se presenta una adecuación metodoloógica para contender con este problema en el desarrollo de sitios web. La metodología ha sido probada para implementando algunos sitios *web* como la página del Journal of Applied Research and Technology .

REFERENCIAS

[1] Jon December. Introducción a Java . Prentice Hall 1996.
[2] Kenneth E. Kendall. Systems Analysis and desing, 9th ed. Prentice Hall 2006.
[3] James Martin. Análisis y Diseño Orientado a Objetos, ed. Prentice Hall 1992.
[4] Stephanie Jones. Introduction to Oracle: SQL and PL/SQL. Vol I-II-III. ed. Oracle corporation 1996