Benemérita Universidad Autónoma de Puebla

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

PROYECTO DE TESIS

PORTAL DE BOLSA DE TRABAJO Y PRÁCTICAS PROFESIONALES

ROBERTO MUÑOZ CASTILLO

ASESORA: ALMA DELIA AMBROSIO VAZQUEZ

OCTUBRE 2012

Índice

Capítulo 1

INTRODUCCIÓN

1.1 Ámbito

En la Facultad de Ciencias de la Computación de la BUAP existe la coordinación de prácticas profesionales y bolsa de trabajo, esta coordinación se encarga de recolectar y difundir toda convocatoria de prácticas profesionales o bolsa de trabajo, así mismo lleva el control de la supervisión de todo practicante registrado. Adicionalmente ésta coordinación tiene que levantar reportes de toda aquella información útil en beneficio del sector productivo y de los estudiantes.

1.2 Justificación

Sin embargo esta información aunque es llevada electrónicamente hace falta concentrarla en un portal que vuelva los datos ágiles, oportunos consistentes y actualizados.

1.3 Metodología

Este trabajo se realizó bajo la metodología RUP, siguiendo un *Lenguaje Unificado de Modelado* (UML) y utilizando como herramienta de trabajo el programa "StarUML".

1.4 Referencias del sistema

La información con que se cuenta son:

1) Convocatorias impresas de prácticas profesionales y bolsa de trabajo así como expedientes impresos de practicantes.
2) Alguna reglamentación oportuna para las convocatorias y para los alumnos interesados en las convocatorias.

1.5 Objetivos

El objetivo de este trabajo es implementar un portal que administre toda información relacionada con la práctica profesional y bolsa de trabajo, facilitando los procesos que estas tareas conllevan.

1.6 Objetivos específicos

- Llevar la administración de todas las vacantes publicadas.
- Difundir las propuestas a los alumnos de la FCC por un medio electrónico y así hacer más visibles las convocatorias.
- Llevar un registro de las empresas que ofertan propuestas.

1.7 Restricciones

No aplica.

Capítulo 2

Marco Teórico

2.1 Seguridad en WEB

La web presenta nuevo retos que generalmente no se aprecian en el contexto de la seguridad de los computadores ni de la red: [1]

- Internet es bidireccional. Al contrario que los entornos de publicación tradicionales, incluso los sistemas de publicación electrónica que hacen uso de teletexto, respuesta de voz o respuesta de fax, la web es vulnerable a los ataques a los servidores web desde internet.
- La web se emplea cada vez más para presentar información de empresas y de productos, y como plataforma para transacciones de negocios. Se puede perjudicar la imagen y ocasionar pérdidas económicas, si se manipulan los servidores web.
- Aunque los navegadores web son muy fáciles de usar, los servidores web relativamente sencillos de configurar y gestionar y gestionar los contenidos web cada vez más fáciles de desarrollar, el *software* subyacente es extraordinariamente complejo.

En determinadas circunstancias, es interesante poder limitar el acceso a documentos reservados o útiles para un conjunto restringido de personas. En éste caso se restringirá el acceso con la siguiente restricción:
Limitación de acceso por nombres de usuario y clave de acceso. Sólo los usuarios que conozcan una clave de acceso válida pueden acceder a la información. [2]

Los usuarios o visitantes de su sitio no deben poder ver o entrar en los directorios de su aplicación, coloque un *index* con un contenido "mensaje" que indique que no está en un área permitida y lo redireccione al *index* principal de su aplicación.

Usar la función *mysql_real_escape_string* para el tratamiento de datos en la base de datos. De esta forma se minimiza un posible ataque por sql injection. [3]

Manipulación de URL
Los *forms* HTML envían sus resultados usando uno de dos métodos posibles: GET o POST. Si el método es GET, todos los parámetros del *form* y sus valores correspondientes aparecen en cadena de búsqueda del URL que el usuario ve. Esta cadena puede ser fácilmente manipulable.
En este caso su usará el método POST para no hacer visible el paso de variables. [4]

2.2 Bases de Datos

¿Por qué es importante el diseño de una base de datos?
Una buena base de datos no es algo que simplemente suceda; la estructura de su contenido debe diseñarse con cuidado.

Una base de datos bien diseñada facilita la administración de datos y se convierte en un valioso generador de información; mientras una que este mal diseñada probablemente se convertirá en tierra de cultivo de datos redundantes, es decir datos innecesariamente duplicados. En muchas ocasiones los datos son los causantes de errores de información difíciles de rastrear. [5]

¿Qué son las Bases de Datos?
Una base de datos es un "almacén" que nos permite guardar grandes cantidades de información de forma organizada para que luego podamos encontrar y utilizar fácilmente. [6]
Desde el punto de vista informático, la base de datos es un sistema formado por un conjunto de datos almacenados en discos que permiten el acceso directo a ellos y un conjunto de programas que manipulen ese conjunto de datos.
Cada base de datos se compone de una o más tablas que guarda un conjunto de datos. Cada tabla tiene una o más *columnas* y *filas*. Las columnas guardan una parte de la información sobre cada elemento que queramos guardar en la tabla, cada fila de la tabla conforma un registro.

Definición de Base de Datos
Se define una base de datos como una serie de datos organizados y relacionados entre sí, los cuales son recolectados y explotados por los sistemas de información de una empresa o negocio en particular.
Características
Entre las principales características de los sistemas de base de datos podemos mencionar:

- Independencia lógica y física de los datos.
- Redundancia mínima.
- Acceso concurrente por parte de múltiples usuarios.
- Integridad de los datos.
- Consultas complejas optimizadas.
- Seguridad de acceso y auditoría.
- Respaldo y recuperación.
- Acceso a través de lenguajes de programación estándar.

- Sistemas de Gestión de Base de Datos (SGDB).

Los Sistemas de Gestión de Base de Datos (en inglés DataBase Management System) son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta.

Ventajas de las Bases de Datos

- Control Sobre la Redundancia de Datos:

Los sistemas de ficheros almacenan varias copias de los mismos datos en ficheros distintos. Esto hace que se desperdicie espacio de almacenamiento, además de provocar la falta de consistencia de datos.
En los sistemas de bases de datos todos estos ficheros están integrados, por lo que no se almacenan varias copias de los mismos datos. Sin embargo, en una base de datos no se puede eliminar la redundancia completamente, ya que en ocasiones es necesaria para modelar las relaciones entre los datos.

- Consistencia de Datos:

Eliminando o controlando las redundancias de datos se reduce en gran medida el riesgo de que haya inconsistencias. Si un dato está almacenado una sola vez, cualquier actualización se debe realizar sólo una vez, y está disponible para todos los usuarios inmediatamente. Si un dato está duplicado y el sistema conoce esta redundancia, el propio sistema puede encargarse de garantizar que todas las copias se mantienen consistentes.

- Compartición de Datos:

En los sistemas de ficheros, los ficheros pertenecen a las personas o a los departamentos que los utilizan. Pero en los sistemas de bases de datos, la base de datos pertenece a la empresa y puede ser compartida por todos los usuarios que estén autorizados.

- Mantenimiento de Estándares:

Gracias a la integración es más fácil respetar los estándares necesarios, tanto los establecidos a nivel de la empresa como los nacionales e internacionales. Estos estándares pueden establecerse sobre el formato de los datos para facilitar su intercambio, pueden ser estándares de documentación, procedimientos de actualización y también reglas de acceso.

- Mejora en la Integridad de Datos:

La integridad de la base de datos se refiere a la validez y la consistencia de los datos almacenados. Normalmente, la integridad se expresa mediante restricciones o reglas que no se pueden violar. Estas restricciones se pueden aplicar tanto a los datos, como a sus relaciones, y es el SGBD quien se debe encargar de mantenerlas.

- Mejora en la Seguridad:

La seguridad de la base de datos es la protección de la base de datos frente a usuarios no autorizados. Sin unas buenas medidas de seguridad, la integración de datos en los sistemas de bases de datos hace que éstos sean más vulnerables que en los sistemas de ficheros.

- Mejora en la Accesibilidad a los Datos:

Muchos SGBD proporcionan lenguajes de consultas o generadores de informes que permiten al usuario hacer cualquier tipo de consulta sobre los datos, sin que sea necesario que un programador escriba una aplicación que realice tal tarea.

- Mejora en la Productividad:

El SGBD proporciona muchas de las funciones estándar que el programador necesita escribir en un sistema de ficheros. A nivel básico, el SGBD proporciona todas las rutinas de manejo de ficheros típicas de los programas de aplicación.
El hecho de disponer de estas funciones permite al programador centrarse mejor en la función específica requerida por los usuarios, sin tener que preocuparse de los detalles de implementación de bajo nivel.

- Mejora en el Mantenimiento:

En los sistemas de ficheros, las descripciones de los datos se encuentran inmersas en los programas de aplicación que los manejan.
Esto hace que los programas sean dependientes de los datos, de modo que un cambio en su estructura, o un cambio en el modo en que se almacena en disco, requiere cambios importantes en los programas cuyos datos se ven afectados.
Sin embargo, los SGBD separan las descripciones de los datos de las aplicaciones. Esto es lo que se conoce como independencia de datos, gracias a la cual se simplifica el mantenimiento de las aplicaciones que acceden a la base de datos.

- Aumento de la Concurrencia:

En algunos sistemas de ficheros, si hay varios usuarios que pueden acceder simultáneamente a un mismo fichero, es posible que el acceso interfiera entre ellos de modo que se pierda información o se pierda la integridad. La mayoría de los SGBD gestionan el acceso concurrente a la base de datos y garantizan que no ocurran problemas de este tipo.

- Mejora en los Servicios de Copias de Seguridad:

Muchos sistemas de ficheros dejan que sea el usuario quien proporcione las medidas necesarias para proteger los datos ante fallos en el sistema o en las aplicaciones. Los usuarios tienen que hacer copias de seguridad cada día, y si se produce algún fallo, utilizar estas copias para restaurarlos.
En este caso, todo el trabajo realizado sobre los datos desde que se hizo la última copia de seguridad se pierde y se tiene que volver a realizar. Sin embargo, los SGBD actuales funcionan de modo que se minimiza la cantidad de trabajo perdido cuando se produce un fallo.

Desventajas de las Bases de Datos

- Complejidad:

Los SGBD son conjuntos de programas que pueden llegar a ser complejos con una gran funcionalidad. Es preciso comprender muy bien esta funcionalidad para poder realizar un buen uso de ellos.

- Coste del Equipamiento Adicional:

Tanto el SGBD, como la propia base de datos, pueden hacer que sea necesario adquirir más espacio de almacenamiento. Además, para alcanzar las prestaciones deseadas, es posible que sea necesario adquirir una máquina más grande o una máquina que se

dedique solamente al SGBD. Todo esto hará que la implantación de un sistema de bases de datos sea más cara.

- Vulnerable a los Fallos:

El hecho de que todo esté centralizado en el SGBD hace que el sistema sea más vulnerable ante los fallos que puedan producirse. Es por ello que deben tenerse copias de seguridad (Backup).

2.3 Normalización

Concepto.

La normalización es un concepto que hace referencia a las relaciones. Básicamente, el principio de normalización indica que las tablas de las bases de datos eliminarán las incoherencias y redundancias, y minimizaran la ineficacia. [7]

Descripción.

La normalización funciona mediante una serie de etapas llamadas formas normalizadas. Las primeras de estas etapas se describen como primera forma normalizada (1NF), segunda forma normalizada (2NF) y tercera forma normalizada (3NF). Desde un punto de vista estructural, 2NF es mejor que 1NF, y 3NF es mejor que 2 NF. Para la mayoría de los propósitos de diseño de base de datos de negocios, 3NF es el nivel al que se tiene que llegar en el proceso de normalización.

Además, algunas aplicaciones muy especializadas pueden requerir más allá de 4NF. Estas aplicaciones, en general, son promovidas por requerimientos de investigación estadística que se salen del ámbito de la mayoría de las operaciones de negocios. [8]

Fundamentos de la normalización.

La normalización es el proceso de organizar los datos de una base de datos. Se incluye la creación de tablas y el establecimiento de relaciones entre ellas según reglas diseñadas tanto para proteger los datos como para hacer que la base de datos sea más flexible al eliminar la redundancia y las dependencias incoherentes. [9]

Los datos redundantes desperdician el espacio de disco y crean problemas de mantenimiento. Si hay que cambiar datos que existen en más de un lugar, se deben cambiar de la misma forma exactamente en todas sus ubicaciones. Un cambio en la dirección de un cliente es mucho más fácil de implementar si los datos sólo se almacenan en la tabla Clientes y no en algún otro lugar de la base de datos.

Hay algunas reglas en la normalización de una base de datos. Cada regla se denomina una "forma normal". Si se cumple la primera regla, se dice que la base de datos está en la "primera forma normal". Si se cumplen las tres primeras reglas, la base de datos se considera que está en la "tercera forma normal". Aunque son posibles otros niveles de normalización, la tercera forma normal se considera el máximo nivel necesario para la mayor parte de las aplicaciones.

Al igual que con otras muchas reglas y especificaciones formales, en los escenarios reales no siempre se cumplen los estándares de forma perfecta.

En las descripciones siguientes se incluyen ejemplos.

- Primera Forma Normal

Elimine los grupos repetidos de las tablas individuales.
Cree una tabla independiente para cada conjunto de datos relacionados.
Identifique cada conjunto de datos relacionados con una clave principal.
No use varios campos en una sola tabla para almacenar datos similares. Por ejemplo, para realizar el seguimiento de un elemento del inventario que proviene de dos orígenes posibles, un registro del inventario puede contener campos para el Código de proveedor 1 y para el Código de proveedor 2.

¿Qué ocurre cuando se agrega un tercer proveedor? Agregar un campo no es la respuesta, requiere modificaciones en las tablas y el programa, y no admite fácilmente un número variable de proveedores. En su lugar, coloque toda la información de los proveedores en una tabla independiente denominada Proveedores y después vincule el inventario a los proveedores con el número de elemento como clave, o los proveedores al inventario con el código de proveedor como clave.

- Segunda Forma Normal

Cree tablas independientes para conjuntos de valores que se apliquen a varios registros.
Relacione estas tablas con una clave externa.
Los registros no deben depender de nada que no sea una clave principal de una tabla, una clave compuesta si es necesario. Por ejemplo, considere la dirección de un cliente en un sistema de contabilidad. La dirección se necesita en la tabla Clientes, pero también en las tablas Pedidos, Envíos, Facturas, Cuentas por cobrar y Colecciones. En lugar de almacenar la dirección de un cliente como una entrada independiente en cada una de estas tablas, almacénela en un lugar, ya sea en la tabla Clientes o en una tabla Direcciones independiente.

- Tercera Forma Normal

Elimine los campos que no dependan de la clave.
Los valores de un registro que no sean parte de la clave de ese registro no pertenecen a la tabla. En general, siempre que el contenido de un grupo de campos pueda aplicarse a más de un único registro de la tabla, considere colocar estos campos en una tabla independiente. Por ejemplo, en una tabla Contratación de empleados, puede incluirse el nombre de la universidad y la dirección de un candidato. Pero necesita una lista completa de universidades para enviar mensajes de correo electrónico en grupo. Si la información de las universidades se almacena en la tabla Candidatos, no hay forma de enumerar las universidades que no tengan candidatos en ese momento. Cree una tabla Universidades independiente y vincúlela a la tabla Candidatos con el código de universidad como clave.
EXCEPCIÓN: cumplir la tercera forma normal, aunque en teoría es deseable, no siempre es práctico. Si tiene una tabla Clientes y desea eliminar todas las dependencias posibles entre los campos, debe crear tablas independientes para las ciudades, códigos postales, representantes de venta, clases de clientes y cualquier otro factor que pueda estar duplicado en varios registros. En teoría, la normalización merece el trabajo que

supone. Sin embargo, muchas tablas pequeñas pueden degradar el rendimiento o superar la capacidad de memoria o de archivos abiertos.
Puede ser más factible aplicar la tercera forma normal sólo a los datos que cambian con frecuencia. Si quedan algunos campos dependientes, diseñe la aplicación para que pida al usuario que compruebe todos los campos relacionados cuando cambie alguno.

- Cuarta Forma Normal

La cuarta forma normal, también llamada Forma normal de Boyce Codd (BCNF, Boyce Codd Normal Form), y la quinta forma normal existen, pero rara vez se consideran en un diseño real. Si no se aplican estas reglas, el diseño de la base de datos puede ser menos perfecto, pero no debería afectar a la funcionalidad.

Tabla de ejemplo

Estos pasos demuestran el proceso de normalización de una tabla de alumnos ficticia.

1. Tabla sin normalizar:

Nº alumno	Tutor	Despacho-Tut	Clase1	Clase2	Clase3
1022	García	412	101-07	143-01	159-02
4123	Díaz	216	201-01	211-02	214-01

2. Primera forma normal: no hay grupos repetidos

   Las tablas sólo deben tener dos dimensiones. Puesto que un alumno tiene varias clases, estas clases deben aparecer en una tabla independiente. Los campos Clase1, Clase2 y Clase3 de los registros anteriores son indicativos de un problema de diseño.

   Las hojas de cálculo suelen usar la tercera dimensión, pero las tablas no deberían hacerlo. Otra forma de considerar ese problema es con una relación de uno a varios y poner el lado de uno y el lado de varios en tablas distintas. En su lugar, cree otra tabla en la primera forma normal eliminando el grupo repetido (Nº clase), según se muestra a continuación:

Nº alumno	Tutor	Despacho-Tut	Nº clase
1022	García	412	101-07
1022	García	412	143-01
1022	García	412	159-02
4123	Díaz	216	201-01
4123	Díaz	216	211-02

4123	Díaz	216	214-01

3. Segunda forma normal: eliminar los datos redundantes

   Observe los diversos valores de Nº clase para cada valor de Nº alumno en la tabla anterior. Nº clase no depende funcionalmente de Nº alumno (la clave principal), de modo que la relación no cumple la segunda forma normal.

   Las dos tablas siguientes demuestran la segunda forma normal:

   Alumnos:

Nº alumno	Tutor	Despacho-Tut
1022	García	412
4123	Díaz	216

   Registro:

Nº alumno	Nº clase
1022	101-07
1022	143-01
1022	159-02
4123	201-01
4123	211-02
4123	214-01

4. Tercera forma normal: eliminar los datos no dependientes de la clave.

   En el último ejemplo, Despacho-Tut (el número de despacho del tutor) es funcionalmente dependiente del atributo Tutor. La solución es pasar ese atributo de la tabla Alumnos a la tabla Personal, según se muestra a continuación:

   Alumnos:

Nº alumno	Tutor
1022	García
4123	Díaz

Personal:

Nombre	Habitación	Dept
García	412	42
Díaz	216	42

2.4 Ingeniería de Software

La ingeniería de software es una disciplina formada por un conjunto de métodos, herramientas y técnicas que se utilizan en el desarrollo de los programas informáticos (software). [10]

Esta disciplina trasciende la actividad de programación, que es la actividad principal a la hora de crear un software. El ingeniero de software se encarga de toda la gestión del proyecto para que éste se pueda desarrollar en un plazo determinado y con el presupuesto previsto.

La ingeniería de software, por lo tanto, incluye el análisis previo de la situación, el diseño del proyecto, el desarrollo del software, las pruebas necesarias para confirmar su correcto funcionamiento y la implementación del sistema.
Cabe destacar que el proceso de desarrollo de software implica lo que se conoce como ciclo de vida del software, que está formado por cuatro etapas: concepción, elaboración, construcción y transición.
La concepción fija el alcance del proyecto y desarrolla el modelo de negocio; la elaboración define el plan del proyecto, detalla las características y fundamenta la arquitectura; la construcción es el desarrollo del producto; y la transición es la transferencia del producto terminado a los usuarios.

Una vez que se completa este ciclo, entra en juego el mantenimiento del software. Se trata de una fase de esta ingeniería donde se solucionan los errores descubiertos (muchas veces advertidos por los propios usuarios) y se incorporan actualizaciones para hacer frente a los nuevos requisitos. El proceso de mantenimiento incorpora además nuevos desarrollos, para permitir que el software pueda cumplir con una mayor cantidad de tareas.

¿Cuál es la diferencia entre Ingeniería de software y Ciencia de la computación?
La ciencia de la computación comprende la teoría y los fundamentos. Y la ingeniería de software comprende las formas prácticas para desarrollar y entregar un software útil. [11]

¿Cuál es la diferencia entre Ingeniería de software y la Ingeniería de sistemas?
La ingeniería de sistemas se refiere a todos los aspectos del desarrollo de sistemas informáticos, incluyendo hardware, software e ingeniería de procesos. La ingeniería de software es parte de este proceso.

2.5 Programación Orientada a Objetos

Programación Orientada a Objetos (POO)
Tecnología Orientada a Objetos
Hoy en día la tecnología orientada a objetos ya no se aplica solamente a los lenguajes de programación, además se viene aplicando en el análisis y diseño con mucho éxito, al igual que en las bases de datos. Es que para hacer una buena programación orientada a objetos hay que desarrollar todo el sistema aplicando esta tecnología, de ahí la importancia del análisis y el diseño orientado a objetos. [12]

La programación orientada a objetos es una de las formas más populares de programar y viene teniendo gran acogida en el desarrollo de proyectos de software desde los últimos años. Esta acogida se debe a sus grandes capacidades y ventajas frente a las antiguas formas de programar.

¿Cuáles son las ventajas de un lenguaje orientado a objetos?

- Fomenta la reutilización y extensión del código.
- Permite crear sistemas más complejos.
- Relacionar el sistema al mundo real.
- Facilita la creación de programas visuales.
- Construcción de prototipos
- Agiliza el desarrollo de software
- Facilita el trabajo en equipo
- Facilita el mantenimiento del software

Lo interesante de la POO es que proporciona conceptos y herramientas con las cuales se modela y representa el mundo real tan fielmente como sea posible.

El modelo Orientado a Objetos
Para entender este modelo vamos a revisar 4 conceptos básicos:

- Objetos
- Clases
- Herencia
- Envío de mensajes

1. Objetos
Entender que es un objeto es la clave para entender cualquier lenguaje orientado a objetos.
Existen muchas definiciones que se le ha dado al Objeto. Primero empecemos entendiendo que es un objeto del mundo real. Un objeto del mundo real es cualquier cosa que vemos a nuestro alrededor. Digamos que para leer este artículo lo hacemos a

través del monitor y una computadora, ambos son objetos, al igual que nuestro teléfono celular, un árbol o un automóvil.
Analicemos un poco más a un objeto del mundo real, como la computadora. No necesitamos ser expertos en hardware para saber que una computadora está compuesta internamente por varios componentes: la tarjeta madre, el chip del procesador, un disco duro, una tarjeta de video, y otras partes más. El trabajo en conjunto de todos estos componentes hace operar a una computadora.
Internamente, cada uno de estos componentes puede ser sumamente complicado y puede ser fabricado por diversas compañías con diversos métodos de diseño. Pero nosotros no necesitamos saber cómo trabajan cada uno de estos componentes, como saber qué hace cada uno de los chips de la tarjeta madre, o cómo funciona internamente el procesador. Cada componente es una unidad autónoma, y todo lo que necesitamos saber de adentro es cómo interactúan entre sí los componentes, saber por ejemplo si el procesador y las memorias son compatibles con la tarjeta madre, o conocer donde se coloca la tarjeta de video. Cuando conocemos como interaccionan los componentes entre sí, podremos armar fácilmente una computadora.

¿Qué tiene que ver esto con la programación? La programación orientada a objetos trabaja de esta manera. Todo el programa está construido en base a diferentes componentes (Objetos), cada uno tiene un rol específico en el programa y todos los componentes pueden comunicarse entre ellos de formas predefinidas.
Todo objeto del mundo real tiene 2 componentes: características y comportamiento.
Por ejemplo, los automóviles tienen características (marca, modelo, color, velocidad máxima, etc.) y comportamiento (frenar, acelerar, retroceder, llenar combustible, cambiar llantas, etc.).

Los Objetos de Software, al igual que los objetos del mundo real, también tienen características y comportamientos. Un objeto de software mantiene sus características en una o más "variables", e implementa su comportamiento con "métodos". Un método es una función o subrutina asociada a un objeto.

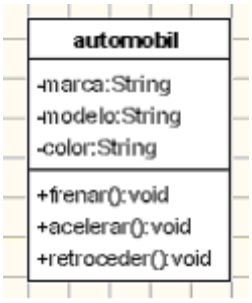

Ilustración 1. Ej. Objetos

Definición teórica: Un objeto es una unidad de código compuesto de variables y métodos relacionados.

2. Las Clases
En el mundo real, normalmente tenemos muchos objetos del mismo tipo. Por ejemplo, nuestro teléfono celular es sólo uno de los miles que hay en el mundo. Si hablamos en términos de la programación orientada a objetos, podemos decir que nuestro objeto celular es una instancia de una clase conocida como "celular". Los celulares tienen

características (marca, modelo, sistema operativo, pantalla, teclado, etc.) y comportamientos (hacer y recibir llamadas, enviar mensajes multimedia, transmisión de datos, etc.).

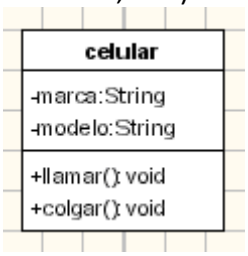

Ilustración 2. Ej. Clases

Cuando se fabrican los celulares, los fabricantes aprovechan el hecho de que los celulares comparten esas características comunes y construyen modelos o plantillas comunes, para que a partir de esas se puedan crear muchos equipos celulares del mismo modelo. A ese modelo o plantilla le llamamos CLASE, y a los equipos que sacamos a partir de ella la llamamos OBJETOS.

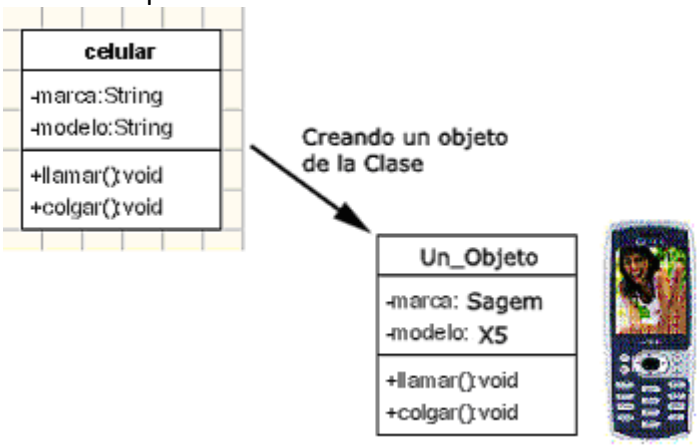

Ilustración 3. Ej. De clase

Esto mismo se aplica a los objetos de software, se puede tener muchos objetos del mismo tipo y mismas características.
Definición teórica: La clase es un modelo o prototipo que define las variables y métodos comunes a todos los objetos de cierta clase. También se puede decir que una clase es una plantilla genérica para un conjunto de objetos de similares características.
Por otro lado, una instancia de una clase es otra forma de llamar a un objeto. En realidad no existe diferencia entre un objeto y una instancia. Sólo que el objeto es un término más general, pero los objetos y las instancias son ambas representación de una clase.
Definición Teórica: Una instancia es un objeto de una clase en particular.

3. Herencia
La herencia es uno de los conceptos más cruciales en la POO. La herencia básicamente consiste en que una clase puede heredar sus variables y métodos a varias subclases (la clase que hereda es llamada superclase o clase padre). Esto significa que una subclase, aparte de los atributos y métodos propios, tiene incorporados los atributos y métodos heredados de la superclase. De esta manera se crea una jerarquía de herencia.
Por ejemplo, imaginemos que estamos haciendo el análisis de un Sistema para una tienda que vende y repara equipos celulares.

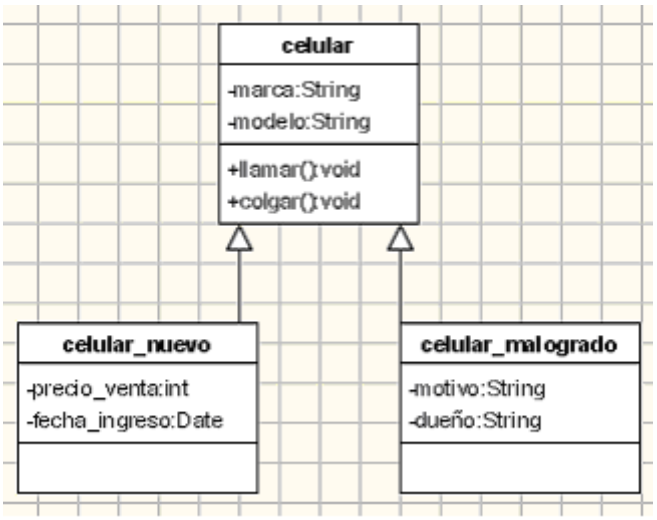

Ilustración 4. Ej. Herencia

En el gráfico vemos 2 Clases más que posiblemente necesitemos para crear nuestro Sistema. Esas 2 Clases nuevas se construirán a partir de la Clase Celular existente. De esa forma utilizamos el comportamiento de la Súper Clase.
En general, podemos tener una gran jerarquía de Clases tal y como vemos en el siguiente gráfico:

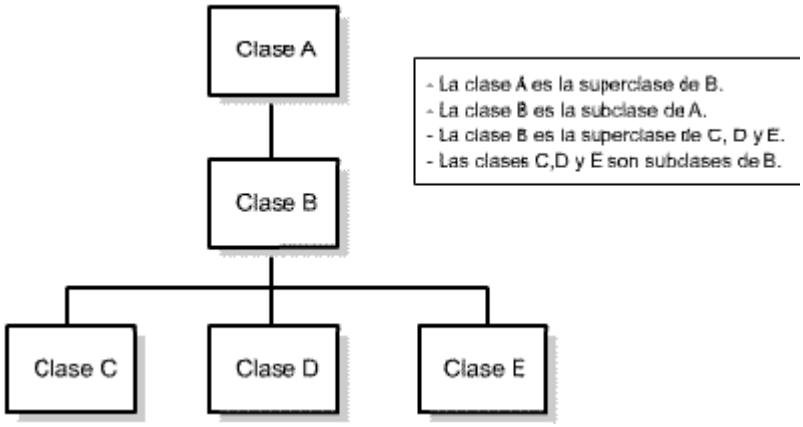

Ilustración 5. Diagrama de herencia

4. Envío de Mensajes
Un objeto es inútil si está aislado. El medio empleado para que un objeto interactúe con otro son los mensajes. Hablando en términos un poco más técnicos, los mensajes son invocaciones a los métodos de los objetos.

Lenguajes de Programación Orientado a Objetos
En 1985, E. Stroustrup extendió el lenguaje de programación C a C++, es decir C con conceptos de clases y objetos, también por esas fechas se creó desde sus bases el lenguaje EIFFEL.
En 1995 apareció el más reciente lenguaje OO, Java desarrollado por SUN, que hereda conceptos de C++.
El lenguaje de desarrollo más extendido para aplicaciones Web, el PHP 5, trae todas las características necesarias para desarrollar software orientado a objetos.
Además de otros lenguajes que fueron evolucionando, como el Pascal a Delphi.

Finalmente también otros lenguajes script como el *ActionScript* que si bien no es totalmente orientado a objetos pero sí posee las características.

Análisis y diseño Orientado a Objetos

Para el desarrollo de software orientado a objetos no basta usar un lenguaje orientado a objetos. También se necesitará realizar un análisis y diseño orientado a objetos.
El modelamiento visual es la clave para realizar el análisis OO. Desde los inicios del desarrollo de software OO han existido diferentes metodologías para hacer esto del modelamiento, pero sin lugar a duda, el Lenguaje de Modelamiento Unificado (UML) puso fin a la guerra de metodologías.
Según los mismos diseñadores del lenguaje UML, éste tiene como fin modelar cualquier tipo de sistemas (no solamente de software) usando los conceptos de la orientación a objetos. Y además, este lenguaje debe ser entendible para los humanos y máquinas.
Actualmente en la industria del desarrollo de software tenemos al UML como un estándar para el modelamiento de sistemas OO. Fue la empresa Racional que creó estas definiciones y especificaciones del estándar UML, y lo abrió al mercado. La misma empresa creó uno de los programas más conocidos hoy en día para este fin; el Racional Rose, pero también existen otros programas como el *Poseidon* que trae licencias del tipo *community edition* que permiten su uso libremente.
El UML consta de todos los elementos y diagramas que permiten modelar los sistemas en base al paradigma orientado a objetos. Los modelos orientados a objetos cuando se construyen en forma correcta, son fáciles de comunicar, cambiar, expandir, validar y verificar. Este modelamiento en UML es flexible al cambio y permite crear componentes plenamente reutilizables.

2.6 Metodología RUP (Rational Unified Process)

RUP Es un proceso de desarrollo de software que junto con el Lenguaje Unificado de Modelado (UML), constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. [13]

El RUP no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización.

RUP se divide en 4 fases, dentro de las cuales se realizan varias iteraciones según el proyecto y en las que se hace mayor o menos esfuerzo en las distintas actividades. En las iteraciones de cada fase se hacen diferentes esfuerzos en diferentes actividades: [14]

- Fase de Inicio

(Inspección y Concepción) Se hace un plan de fases, donde se identifican los principales casos de uso y se identifican los riesgos. Se concreta la idea, la visión del producto, como se enmarca en el negocio, el alcance del proyecto.

- Fase de Elaboración:

Se realiza el plan de proyecto, donde se completan los casos de uso y se mitigan los riesgos. Planificar las actividades necesarias y los recursos requeridos, especificando las características y el diseño de la arquitectura.

- Fase de Construcción:

Se basa en la elaboración de un producto totalmente operativo y en la elaboración del manual de usuario. Construir el producto, la arquitectura y los planes, hasta que el producto está listo para ser enviado a la comunidad de usuarios.

- Fase de Transición:

Se realiza la instalación del producto en el cliente y se procede al entrenamiento de los usuarios. Realizar la transición del producto a los usuarios, lo cual incluye: manufactura, envío, entrenamiento, soporte y mantenimiento del producto, hasta que el cliente quede satisfecho, por tanto en esta fase suelen ocurrir cambios. Con estas fases se logra ejecutar un conjunto de mejores prácticas, como lo son:

- Desarrollar Software Iterativamente
- Modelar el software visualmente
- Controlar los Requerimientos
- Usar arquitecturas basadas en componentes
- Verificación continua de la calidad
- Controlar los cambios

Beneficios de la Metodología Orientada a Objetos.

- Promueve la reusabilidad.
- Reduce la complejidad del mantenimiento (extensibilidad y facilidad de cambios).
- Riqueza semántica.
- Disminuye la brecha semántica entre la visión interna y la visión externa del sistema.
- Facilita la construcción de prototipos.

Ventajas de la Metodología Orientada a Objetos.

- Reutilización
- El diseñador piensa en términos del comportamiento de objetos y no en detalles de bajo nivel
- Confiabilidad, Integridad y Estabilidad.
- Mantenimiento más sencillo. Modificaciones locales.
- Modelado más realista.
- Modelos empresariales inteligentes.
- Independencia del diseño.
- Mejores herramientas CASE.
- Bibliotecas de clases para las empresas.

- Se construyen clases cada vez más complejas.
- Nuevos mercados para el software.
- Diseño de mayor calidad.
- Programación más sencilla.
- Mejor comunicación entre los profesionales de los Sistemas de Información y los empresarios.
- Mayor nivel de automatización de las bases de datos.
- La comprensión del sistema es más fácil porque la semántica entre el sistema y la realidad son similares.

2.7 UML

El Lenguaje de Modelado Unificado (UML)

Una exigencia de la gran mayoría de instituciones dentro de su Plan Informático estratégico, es que los desarrollos de software bajo una arquitectura en Capas, se formalicen con un lenguaje estándar y unificado. [15]

Es decir, se requiere que cada una de las partes que comprende el desarrollo de todo software de diseño orientado a objetos, se visualice, especifique y documente con lenguaje común.

Este lenguaje unificado que cumple con estos requerimientos, es ciertamente UML, el cual cuenta con una notación estándar y semánticas esenciales para el modelado de un sistema orientado a objetos.

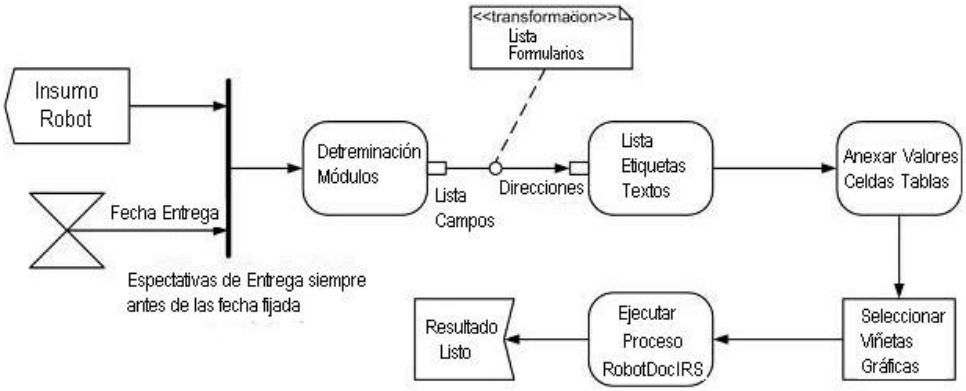

Ilustración 6. Ej. Diagrama UML

¿Qué es UML?

El Lenguaje de Modelado Unificado (UML: Unified Modeling Language) es la sucesión de una serie de métodos de análisis y diseño orientadas a objetos que aparecen a fines de los 80's y principios de los 90s.UML es llamado un lenguaje de modelado, no un método. Los métodos consisten de ambos de un lenguaje de modelado y de un proceso. El UML, fusiona los conceptos de la orientación a objetos aportados por Booch, OMT y OOSE (Booch, G. et al., 1999). UML incrementa la capacidad de lo que se

puede hacer con otros métodos de análisis y diseño orientados a objetos. Los autores de UML apuntaron también al modelado de sistemas distribuidos y concurrentes para asegurar que el lenguaje maneje adecuadamente estos dominios.

El lenguaje de modelado es la notación (principalmente gráfica) que usan los métodos para expresar un diseño. El proceso indica los pasos que se deben seguir para llegar a un diseño.

La estandarización de un lenguaje de modelado es invaluable, ya que es la parte principal del proceso de comunicación que requieren todos los agentes involucrados en un proyecto informático. Si se quiere discutir un diseño con alguien más, ambos deben conocer el lenguaje de modelado y no así el proceso que se siguió para obtenerlo.

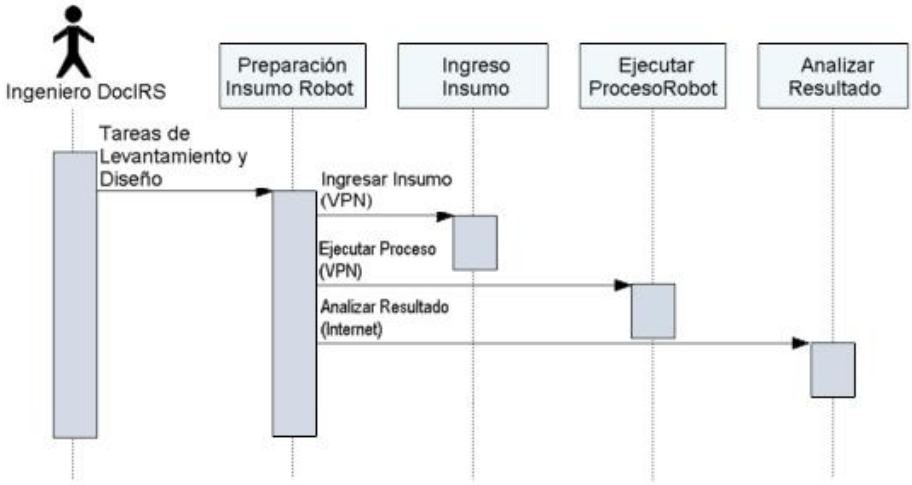

Ilustración 7. Lenguaje de modelado

Una de las metas principales de UML es avanzar en el estado de la integración institucional proporcionando herramientas de interoperabilidad para el modelado visual de objetos. Sin embargo para lograr un intercambio exitoso de modelos de información entre herramientas, se requirió definir a UML una semántica y una notación.

La notación es la parte gráfica que se ve en los modelos y representa la sintaxis del lenguaje de modelado. Por ejemplo, la notación del diagrama de clases define como se representan los elementos y conceptos como son: una clase, una asociación y una multiplicidad. ¿Y qué significa exactamente una asociación o multiplicidad en una clase? Un meta modelo es la manera de definir esto (un diagrama, usualmente de clases, que define la notación).

Para que un proveedor diga que cumple con UML debe cubrir con la semántica y con la notación.

Una herramienta de UML debe mantener la consistencia entre los diagramas en un mismo modelo. Bajo esta definición una herramienta que solo dibuje, no puede cumplir con la notación de UML.

El lenguaje está dotado de múltiples herramientas para lograr la especificación determinante del modelo, se especificarán algunas formas:

- Modelamiento de Clases
- Casos de Uso
- Diagrama de Interacción

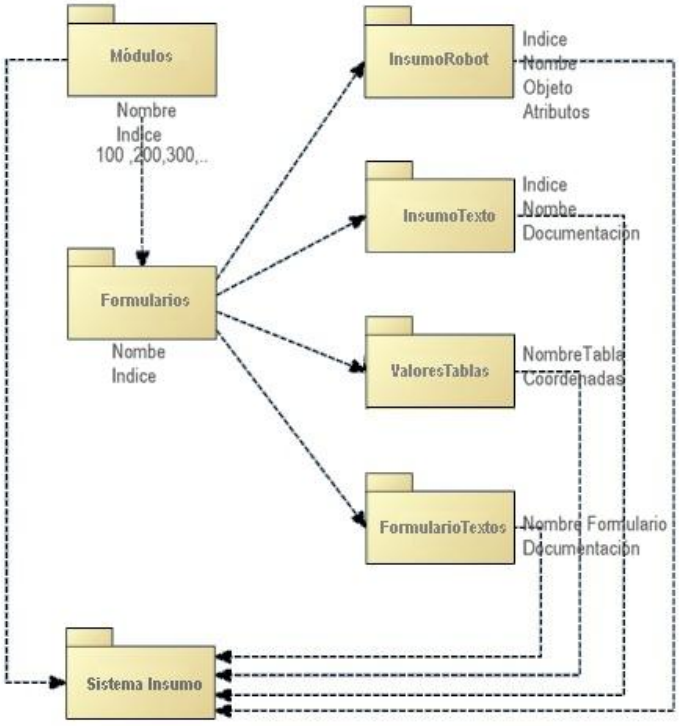

Ilustración 8. Herramientas de modelado

- *Los diagramas de clases de UML forman la vista lógica.*
- *Los diagramas de interacción de UML constituyen la vista de proceso.*
- *La vista de desarrollo captura el software en su entorno de desarrollo.*
- *Los diagramas de despliegue integran la vista física.*
- *Los escenarios: el modelo de casos de uso.*

Modelamiento de Clases

Un diagrama de clases sirve para visualizar las relaciones entre las clases que involucran el sistema, las cuales pueden ser asociativas, de herencia o de uso.

Un diagrama de clases está compuesto por los siguientes elementos:

- Clase: atributos, métodos y visibilidad.

- Relaciones: Herencia, Composición, Agregación, Asociación y Uso.

Elementos

Clase

Es la unidad básica que encapsula toda la información de un Objeto (un objeto es una instancia de una clase). A través de ella podemos modelar el entorno en estudio (una Casa, un Auto, una Cuenta Corriente, etc.).

En UML, una clase es representada por un rectángulo que posee tres divisiones:

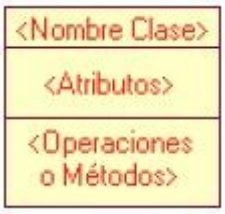

Ilustración 9. Ej. Clase

En donde:

- **Superior**: Contiene el nombre de la Clase
- **Intermedio**: Contiene los atributos (o variables de instancia) que caracterizan a la Clase (pueden ser private, protected o public).
- **Inferior**: Contiene los métodos u operaciones, los cuales son la forma como interactúa el objeto con su entorno (dependiendo de la visibilidad: private, protected o public).

Atributos y Métodos:

Atributos:

Los atributos o características de una Clase pueden ser de tres tipos, los que definen el grado de comunicación y visibilidad de ellos con el entorno, estos son:

- **public** (+): Indica que el atributo será visible tanto dentro como fuera de la clase, es decir, es accesible desde todos lados.
- **private** (-): Indica que el atributo sólo será accesible desde dentro de la clase (sólo sus métodos lo pueden accesar).
- **protected** (#): Indica que el atributo no será accesible desde fuera de la clase, pero si podrá ser accesado por métodos de la clase además de las subclases que se deriven (ver herencia).

Métodos:

Los métodos u operaciones de una clase son la forma en como ésta interactúa con su entorno, éstos pueden tener las características:

- **public** (+): Indica que el método será visible tanto dentro como fuera de la clase, es decir, es accesible desde todos lados.
- **private** (-): Indica que el método sólo será accesible desde dentro de la clase (sólo otros métodos de la clase lo pueden accesar).
- **protected** (#): Indica que el método no será accesible desde fuera de la clase, pero si podrá ser accesado por métodos de la clase además de métodos de las subclases que se deriven (ver herencia).

Relaciones entre Clases:

Ahora ya definido el concepto de Clase, es necesario explicar cómo se pueden interrelacionar dos o más clases (cada uno con características y objetivos diferentes).

Antes es necesario explicar el concepto de cardinalidad de relaciones: En UML, la cardinalidad de las relaciones indica el grado y nivel de dependencia, se anotan en cada extremo de la relación y éstas pueden ser:

- **uno o muchos**: 1..* (1..n)
- **0 o muchos**: 0..* (0..n)
- **número fijo**: m (m denota el número).

Herencia (Especialización/Generalización):

Indica que una subclase hereda los métodos y atributos especificados por una Súper Clase, por ende la Subclase además de poseer sus propios métodos y atributos, poseerá las características y atributos visibles de la Súper Clase (public y protected), ejemplo:

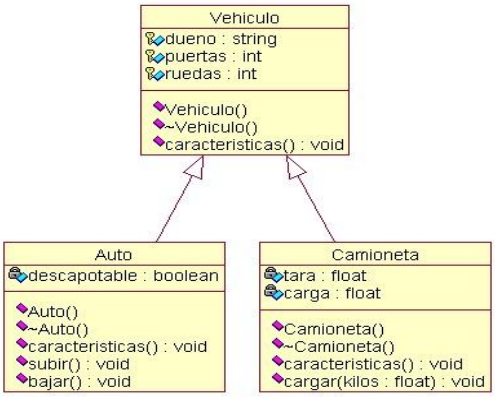

Ilustración 10. Relación entre clases

En la figura se especifica que Auto y Camión heredan de Vehículo, es decir, Auto posee las Características de Vehículo (Precio, VelMax, etc.) además posee algo particular que

es Descapotable, en cambio Camión también hereda las características de Vehículo (Precio, VelMax, etc.) pero posee como particularidad propia Acoplado, Tara y Carga.

Cabe destacar que fuera de este entorno, lo único "visible" es el método Características aplicable a instancias de Vehículo, Auto y Camión, pues tiene definición pública, en cambio atributos como Descapotable no son visibles por ser privados.

Agregación:

Para modelar objetos complejos, bastan los tipos de datos básicos que proveen los lenguajes: enteros, reales y secuencias de caracteres. Cuando se requiere componer objetos que son instancias de clases definidas por el desarrollador de la aplicación, tenemos dos posibilidades:

- **Por Valor**: Es un tipo de relación estática, en donde el tiempo de vida del objeto incluido está condicionado por el tiempo de vida del que lo incluye. Este tipo de relación es comúnmente llamada **Composición** (el Objeto base se construye a partir del objeto incluido, es decir, es "parte/todo").
- **Por Referencia**: Es un tipo de relación dinámica, en donde el tiempo de vida del objeto incluido es independiente del que lo incluye. Este tipo de relación es comúnmente llamada **Agregación** (el objeto base utiliza al incluido para su funcionamiento).

Asociación:

La relación entre clases conocida como Asociación, permite asociar objetos que colaboran entre sí. Cabe destacar que no es una relación fuerte, es decir, el tiempo de vida de un objeto no depende del otro.

Ejemplo:

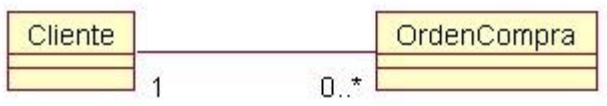

Ilustración 11. Ej. Asociación

Un cliente puede tener asociadas muchas Órdenes de Compra, en cambio una orden de compra solo puede tener asociado un cliente.

Dependencia o Instanciación (uso):

Representa un tipo de relación muy particular, en la que una clase es instanciada (su instanciación es dependiente de otro objeto/clase). Se denota por una flecha punteada.

El uso más particular de este tipo de relación es para denotar la dependencia que tiene una clase de otra, como por ejemplo una aplicación grafica que instancia una ventana (la creación del Objeto Ventana está condicionado a la instanciación proveniente desde el objeto Aplicación):

Ilustración 12. Ej. Dependencia

Cabe destacar que el objeto creado (en este caso la Ventana gráfica) no se almacena dentro del objeto que lo crea (en este caso la Aplicación).

Casos Particulares:

Clase Abstracta:

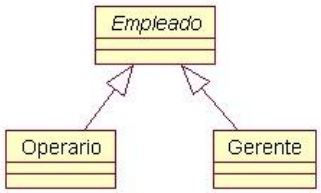

Ilustración 13. Clase abstracta

Una clase abstracta se denota con el nombre de la clase y de los métodos con letra "itálica". Esto indica que la clase definida no puede ser instanciada pues posee métodos abstractos (aún no han sido definidos, es decir, sin implementación). La única forma de utilizarla es definiendo subclases, que implementan los métodos abstractos definidos.

Clase parametrizada:

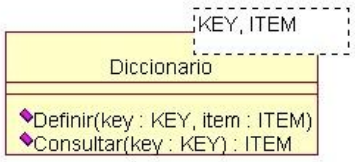

Ilustración 14. Ej. Clase parametrizada

Una clase parametrizada se denota con un sub-cuadro en el extremo superior de la clase, en donde se especifican los parámetros que deben ser pasados a la clase para que esta pueda ser instanciada. El ejemplo más típico es el caso de un Diccionario en donde una llave o palabra tiene asociado un significado, pero en este caso las llaves y elementos pueden ser genéricos. La genericidad puede venir dada de un *Template*

(como en el caso de C++) o bien de alguna estructura predefinida (especialización a través de clases).

En el ejemplo no se especificaron los atributos del Diccionario, pues ellos dependerán exclusivamente de la implementación que se le quiera dar.

Casos de Uso

Introducción

El diagrama de casos de uso representa la forma en como un Cliente (Actor) opera con el sistema en desarrollo, además de la forma, tipo y orden en como los elementos interactúan (operaciones o casos de uso).

Un diagrama de casos de uso consta de los siguientes elementos:

- Actor.
- Casos de Uso.
- Relaciones de Uso, Herencia y Comunicación.

Elementos

Actor:

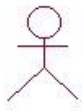

Ilustración 15. Ej. Actor

Una definición previa, es que un *Actor* es un rol que un usuario juega con respecto al sistema. Es importante destacar el uso de la palabra rol, pues con esto se especifica que un Actor no necesariamente representa a una persona en particular, sino más bien la labor que realiza frente al sistema.

Como ejemplo a la definición anterior, tenemos el caso de un sistema de ventas en que el rol de Vendedor con respecto al sistema puede ser realizado por un Vendedor o bien por el Jefe de Local.

Caso de Uso:

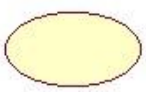

Ilustración 16. Ej. Caso de Uso

Es una operación/tarea específica que se realiza tras una orden de algún agente externo, sea desde una petición de un actor o bien desde la invocación desde otro caso de uso.

Relaciones:

Ilustración 17. Ej. Relación

Asociación

Es el tipo de relación más básica que indica la invocación desde un actor o caso de uso a otra operación (caso de uso). Dicha relación se denota con una flecha simple.

Ilustración 18. Ej. Asociación

Dependencia o Instanciación

Es una forma muy particular de relación entre clases, en la cual una clase depende de otra, es decir, se instancia (se crea). Dicha relación se denota con una flecha punteada.

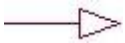

Ilustración 19. Ej. Dependencia

Generalización

Este tipo de relación es uno de los más utilizados, cumple una doble función dependiendo de su estereotipo, que puede ser de **Uso** (<<uses>>) o de **Herencia** (<<extends>>).

Este tipo de relación está orientado exclusivamente para casos de uso (y no para actores).

extends: Se recomienda utilizar cuando un caso de uso es similar a otro (características).

uses: Se recomienda utilizar cuando se tiene un conjunto de características que son similares en más de un caso de uso y no se desea mantener copiada la descripción de la característica.

De lo anterior cabe mencionar que tiene el mismo paradigma en diseño y modelamiento de clases, en donde está la duda clásica de **usar** o **heredar**.

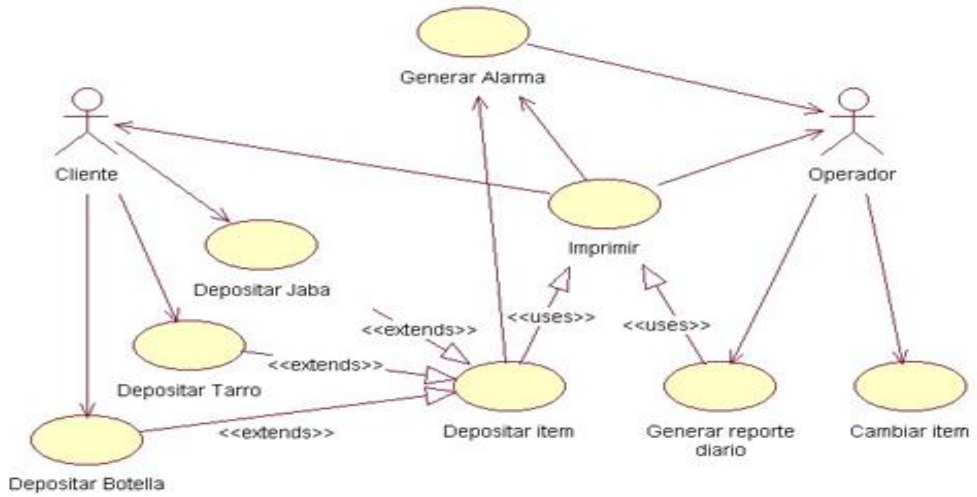

Ilustración 20. Ej. Casos de uso

Diagrama de Interacción

Introducción

El diagrama de interacción, representa la forma en como un Cliente (Actor) u Objetos (Clases) se comunican entre sí en petición a un evento. Esto implica recorrer toda la secuencia de llamadas, de donde se obtienen las responsabilidades claramente.

Dicho diagrama puede ser obtenido de dos partes, desde el Diagrama Estático de Clases o el de Casos de Uso (son diferentes).

Los componentes de un diagrama de interacción son:

- Un Objeto o Actor.
- Mensaje de un objeto a otro objeto.
- Mensaje de un objeto a sí mismo.

Elementos

Objeto/Actor:

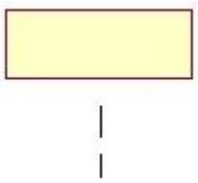

Ilustración 21. Ej. Elemento objeto/actor

El rectángulo representa una instancia de un Objeto en particular, y la línea punteada representa las llamadas a métodos del objeto.

Mensaje a Otro Objeto:

Se representa por una flecha entre un objeto y otro, representa la llamada de un método (operación) de un objeto en particular.

Mensaje al Mismo Objeto:

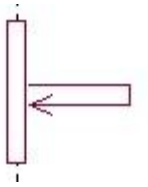

Ilustración 22.Ej. Mensaje al mismo objeto

No solo llamadas a métodos de objetos externos pueden realizarse, también es posible visualizar llamadas a métodos desde el mismo objeto en estudio.

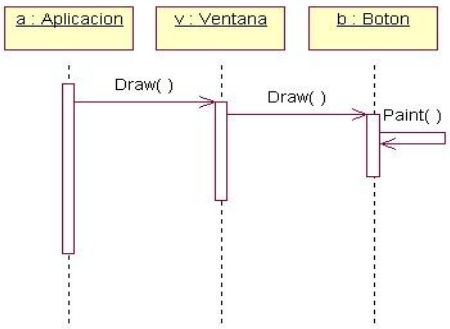

Ilustración 23. Ej. Mensaje a otros objetos

2.8 Herramienta StarUML

Es una herramienta para el modelamiento de software basado en los estándares UML (Unified Modeling Language) y MDA (Model Driven Arquitecture). [16]

Sus principales características son:

- Soporte completo al diseño UML mediante el uso de:
  - – Diagrama de casos de uso
  - – Diagrama de clase
  - – Diagrama de secuencia
  - – Diagrama de colaboración.
  - – Diagrama de estados
  - – Diagrama de actividad.
  - – Diagrama de componentes
  - – Diagrama de despliegue.
  - – Diagrama de composición estructural (UML 2.0)

- Definir elementos propios para los diagramas, que no necesariamente pertenezcan al estándar de UML.
- La capacidad de generar código a partir de los diagramas y viceversa, actualmente funcionando para los lenguajes c++, c# y java.
- Generar documentación en formatos Word, Excel y PowerPoint sobre los diagramas.
- Patrones *GoF (Gang of Four)*, *EJB (Enterprise JavaBeans)* y personalizados.
- Plantillas de proyectos.
- Posibilidad de crear *plugins* para el programa.

En definitiva esta es una de las mejores alternativas gratis que hay en Internet para el modelamiento de software.

Capítulo 3

Diseño del sistema

3.1 Descripción de la información

La información publicada para las convocatorias de prácticas profesionales, bolsa de trabajo, academia, etc., es validada por un encargado del departamento de bolsa de trabajo y prácticas profesionales, así como los expedientes y supervisión de los practicantes.
También llevará un control de información oportuna y eventos relacionados.

Descripción del entorno
La implementación del portal es de gran importancia, ya que no se cuenta con periodos para la recepción de propuestas, y de ésta forma se agilizará el proceso de publicación.
Administración de vacantes
Una vez publicada la propuesta se realiza un reclutamiento para las vacantes de bolsa de trabajo, prácticas profesionales y academia.

Administración Empresarial
La información empresarial es controlada con formatos establecidos para la bolsa de trabajo y prácticas profesionales.

Diagrama de Casos de Uso

A continuación se muestra el diagrama de casos de uso.

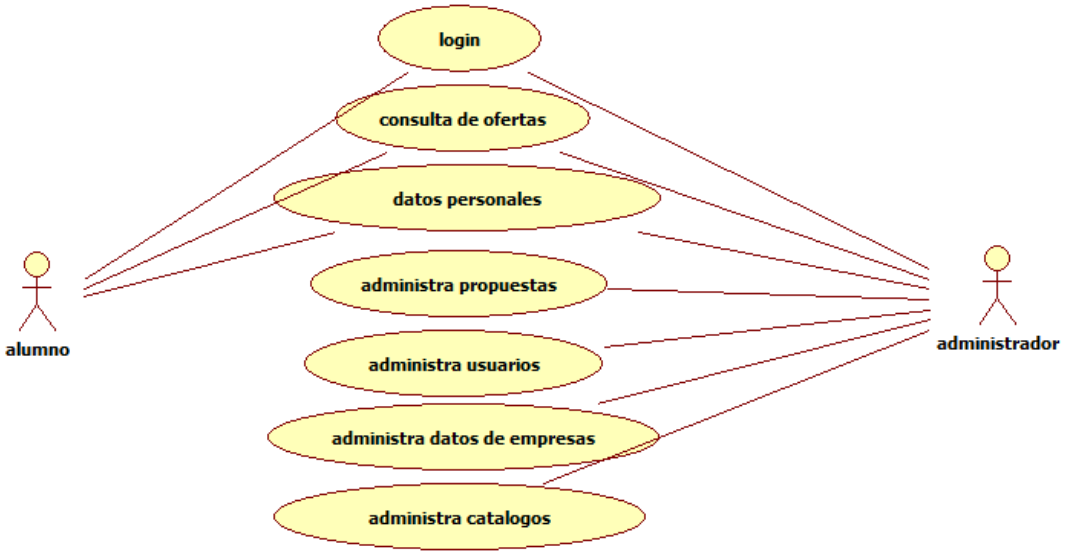

Ilustración 24. Diagrama de Casos de Uso

Documentación de Casos de Uso

Nombre del CU: Login
1 Descripción breve
Método con el cual se podrá ingresar al sistema.

2 Flujo de eventos
2.1 Pre-condiciones
Haberse dado de alta en el sistema.

2.2 Flujo Principal
2.2.1 “Login”

2.2.1.1 El usuario tendrá que ingresar su *usuario* y *contraseña* para poder acceder al sistema, de otra forma tendrá que llevar a cabo el registro (en el caso de los alumnos).
Nombre del CU: Consulta de ofertas
3 Descripción breve
El sistema le permite al alumno y al administrador revisar las propuestas de bolsa de trabajo, prácticas profesionales, etc.

4 Flujo de Eventos
4.1 Pre-condiciones
Haber completado el “login”.

4.2 Flujo Principal
4.2.1 “Consulta de ofertas”
4.2.1.1 El usuario escoge la opción “Bolsa de Trabajo” y el sistema le muestra todas las propuestas que hay vigentes ordenadas de la más reciente, a la más antigua.

4.2.1.2 El usuario escoge la opción “Prácticas Profesionales” y el sistema le muestra todas las opciones que hay para realizar prácticas profesionales, ordenadas de la más reciente, a la más antigua.

4.2.1.3 El usuario escoge la opción “Academia” y el sistema le muestra todas las opciones que hay para academia, ordenadas de la más reciente, a la más antigua.

4.2.1.4 El usuario escoge la opción “Curso” y el sistema le muestra todas las opciones que hay, ordenadas de la más reciente, a la más antigua.

Nombre del CU: Datos personales

5 Descripción breve

El sistema le permitirá tanto al alumno como al administrador ingresar, modificar o eliminar información personal.

6 Flujo de Eventos

6.1 Pre-condición

Haber completado el “login”.

6.2 Flujo Principal

6.2.1 “Datos personales”

6.2.1.1 El alumno podrá ingresar su información académica o personal, por si no ha completado la información necesaria a la hora del registro, y así tener más completa su información.

6.2.1.2 El administrador podrá modificar la información de acceso al sistema.

6.2.1.3 El alumno escoge “Datos personales” para el flujo principal, si la actividad seleccionada es:

Modificar, se ejecuta 6.3.1 “Modificar”

6.2.1.4 El administrador escoge “Datos personales” para el flujo principal, si la actividad seleccionada es:

MODIFICAR, se ejecuta 6.3.2: “Modificar”

Nombre del CU: Administra propuestas

7 Descripción breve

El administrador podrá agregar, modificar o eliminar alguna propuesta.

8 Flujo de Eventos

8.1 Pre-condición

Haber completado el “login”.

8.2 Flujo Principal

8.2.1 “Administra propuestas”

8.2.1.1 El administrador tendrá la opción de ingresar las propuestas que desee y aparecerá un formulario en donde tenga que ingresar los datos como: título, fecha, empresa, puesto/vacante, vigencia, descripción de la empresa, requisitos, lo que ofrece la empresa, etc., de la propuesta que es enviada por la empresa o lugar que ofrece la vacante. Una vez hecho esto, debe guardar los cambios para que la propuesta sea publicada.

8.2.1.2 El administrador escoge “Administra propuestas” para el flujo principal, si la actividad seleccionada es:
Modificar, se ejecuta 8.3.1: “Modificar”
Eliminar, se ejecuta 8.3.2: “Eliminar”

Nombre del CU: Administra usuarios
9 Descripción breve

El administrador podrá agregar, modificar o eliminar a algún alimno, el cual pueda hacer uso del sistema.

10 Flujo de Eventos

10.1 Pre-condición

El administrador debe haber completado el “login”.

10.2 Flujo Principal

10.2.1 “Administra usuarios”

10.2.1.1 El administrador tendrá la opción de ingresar a un nuevo alumno, el cual pueda hacer del sistema, y así pueda consultar las propuestas publicadas.

10.2.1.2 El administrador escoge “Administra usuarios” para el flujo principal. Si la actividad seleccionada es:
Modificar, se ejecuta 10.3.1: “Modificar”
Eliminar, se ejecuta 10.3.2: “Eliminar”

Nombre del CU: Administra datos de empresas
11 Descripción breve
El administrador podrá agregar, modificar o eliminar alguna empresa, escuela o lugar donde tengan vacantes para prácticas profesionales, bolsa de trabajo, academia, etc.

12 Flujo de Eventos

12.1 Pre-condiciones

El administrador debe haber completado el “login”.

12.2 Flujo Principal

12.2.1 “Administra datos de empresas”

12.2.1.1 El administrador escoge la opción “ingresar” y el sistema le permite agregar una nueva empresa, escuela o lugar, el cual ofrezca vacantes para prácticas profesionales, bolsa de trabajo, academia, etc. En la cual deberá de llenar los campos necesarios de información, tales como: nombre, dirección, teléfonos, contacto, etc.

12.2.1.2 El administrador escoge “Administra datos de empresas” para el flujo principal, si la actividad seleccionada es:
Modificar, se ejecuta 12.3.1: “Modificar”
Eliminar, se ejecuta 12.3.2: “Eliminar”

Nombre del CU: Administra catálogos
13 Descripción breve
El administrador podrá agregar, modificar o eliminar alguna *área* para clasificar las propuestas, o algún *tipo* para la publicación de propuestas.

14 Flujo de Eventos

14.1 Pre-condición

Haber completado el "login".

14.2 Flujo Principal

14.2.1 "Administra catálogos"

14.2.1.1 El administrador tendrá la opción de ingresar un *área* cómo: redes, base de datos, programación, etc. para que se puedan clasificar mejor las propuestas publicadas.

O en dado caso podrá agregar un nuevo *tipo* de propuesta, para que sea publicada.

14.2.1.2 El administrador escoge "Administra propuestas" para el flujo principal, si la actividad seleccionada es:
Modificar, se ejecuta 14.3.1: "Modificar"
Eliminar, se ejecuta 14.3.2: "Eliminar"

Flujos alternos

6.3.1 El alumno tendrá la opción de "modificar" su información tanto personal como académica, por si algún campo es erróneo o no es ingresado.
6.3.2 El administrador escoge la opción "modificar" para corregir algún campo que esté erróneo o quiera ser modificado.

8.3.1 El administrador escoge la opción de "Modificar" en la cual tiene que buscar la propuesta a modificar (eligiendo uno de los criterios de búsqueda), y una vez seleccionada se procede a realizar las modificaciones en el campo que se requiera como: título, fecha, empresa, puesto/vacante, vigencia, descripción de la empresa, requisitos, lo que ofrece la empresa, etc. Una vez hechos los cambios correspondientes basta con guardar, para que queden realizados los cambios.
8.3.2 El administrador escoge la opción "Eliminar" en la cual busca la propuesta (eligiendo uno de los criterios de búsqueda) y procede a hacer clic en el botón de "borrar", y el sistema le mandará una notificación para comprobar si se quiere eliminar la propuesta. Una vez que se confirmó, la propuesta quedará eliminada.

10.3.1 El administrador selecciona la opción de "Modificar" para poder cambiar los datos personales del alumno, que son llenados de forma errónea o están vacíos.
10.3.2 El administrador selecciona la opción de "Eliminar" si quiere quitar algún alumno del sistema.

12.3.1 El administrador escoge "Modificar" sí requiere corregir algún campo de información de la empresa, lugar o de algún contacto.
12.3.2 El administrador selecciona "Eliminar" si desea quitar del sistema alguna empresa dada de alta.

14.3.1 El administrador escoge “Modificar” si requiere cambiar el nombre del *área* o *tipo* que ingresó anteriormente.

3.2 Representación del flujo de información

Diagramas de secuencia

Alumno

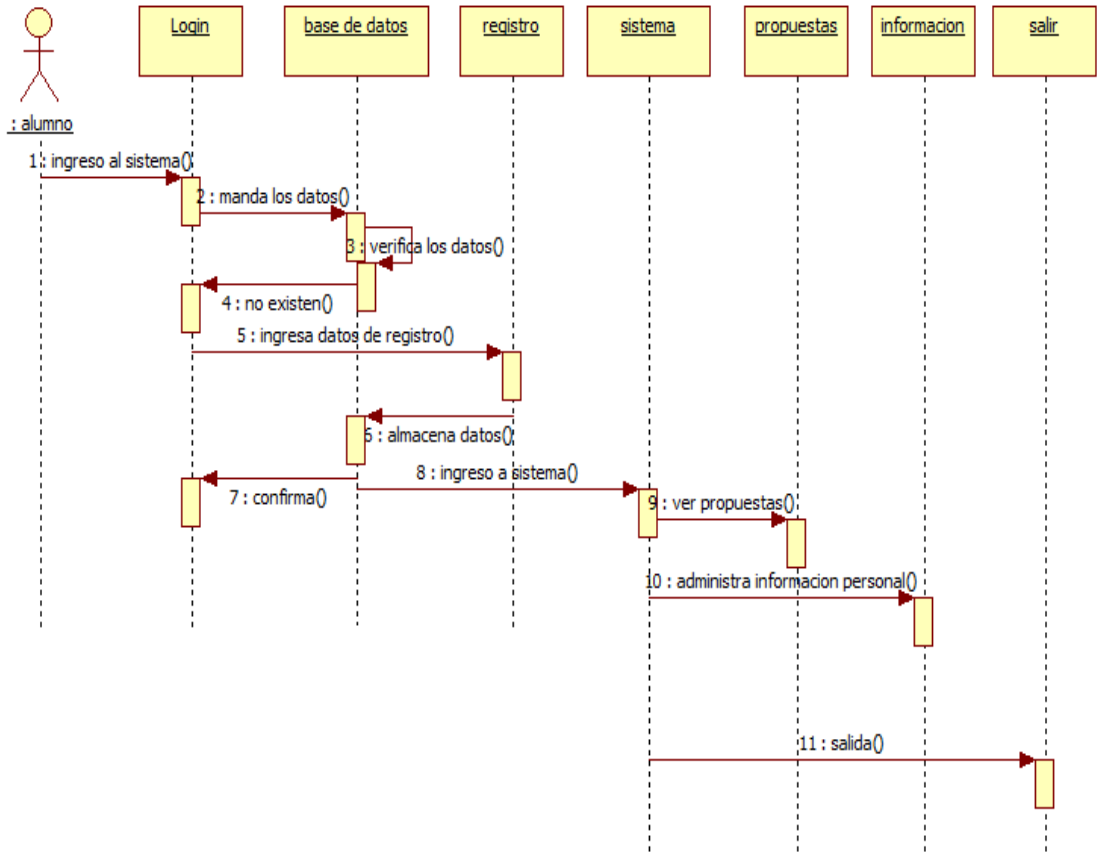

Ilustración 25. Diagrama de secuencia de alumno

Administrador

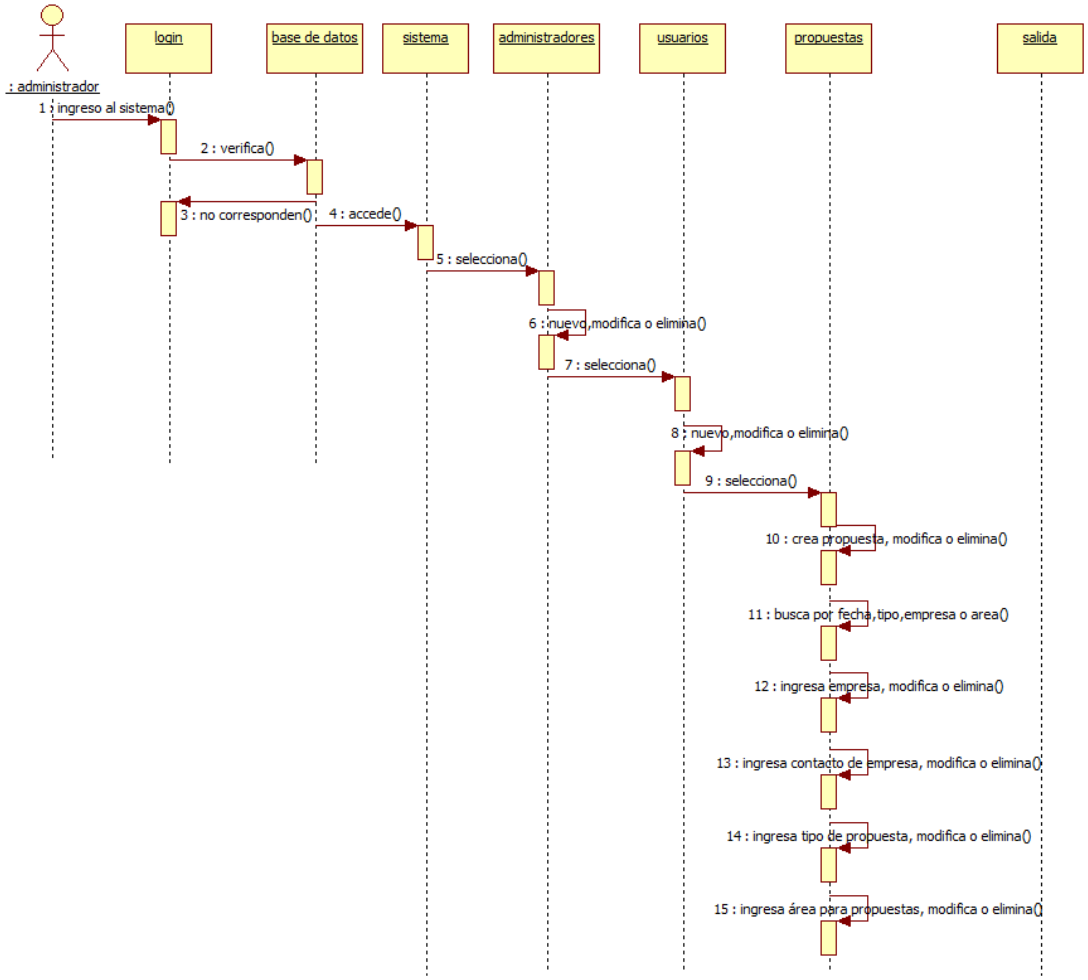

Ilustración 26. Diagrama de secuencia de administrador

Diagramas de colaboración

Alumno

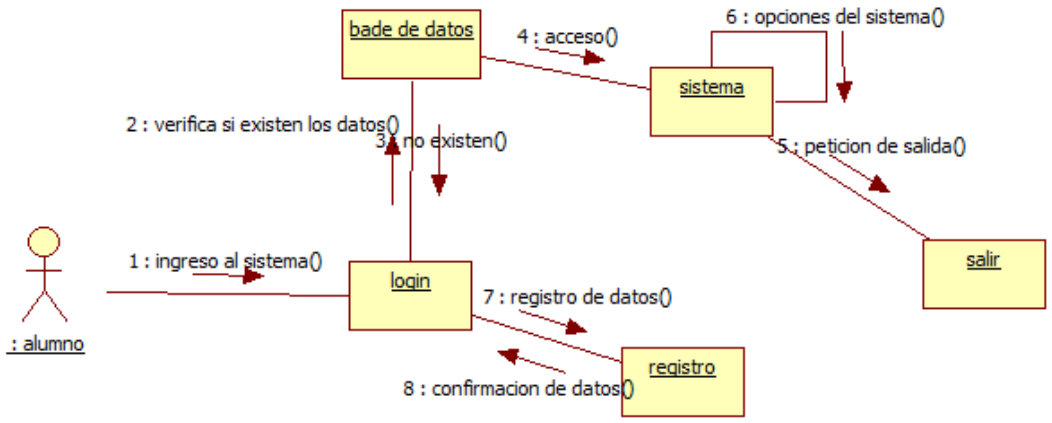

Ilustración 27. Diagrama de colaboración Alumnos

Administrador

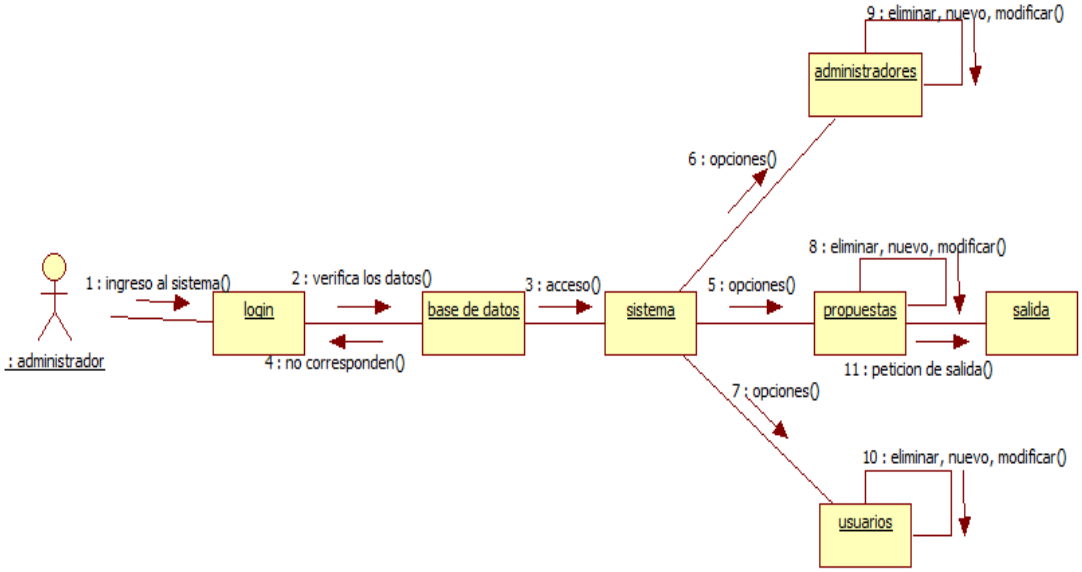

Ilustración 28. Diagrama de colaboración Administrador

Diagramas actividad

Alumnos

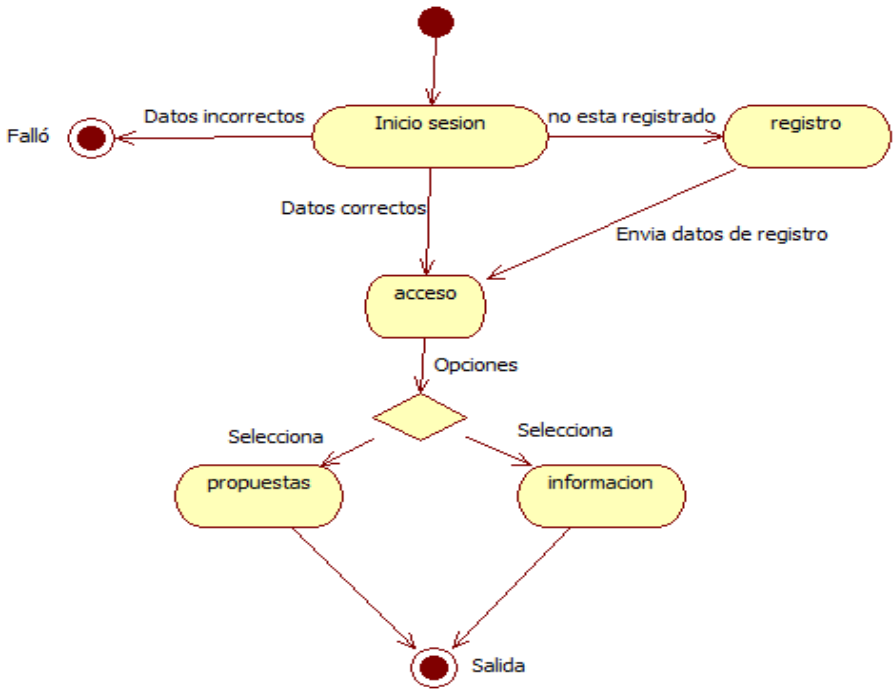

Ilustración 29. Diagrama actividad Alumnos

Administrador

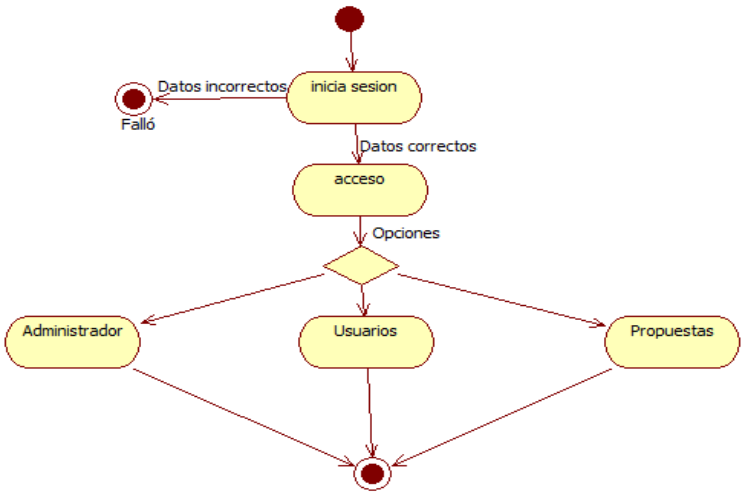

Ilustración 30. Diagrama actividad Administrador

3.3 Representación del contenido de información

Diagrama de paquetes

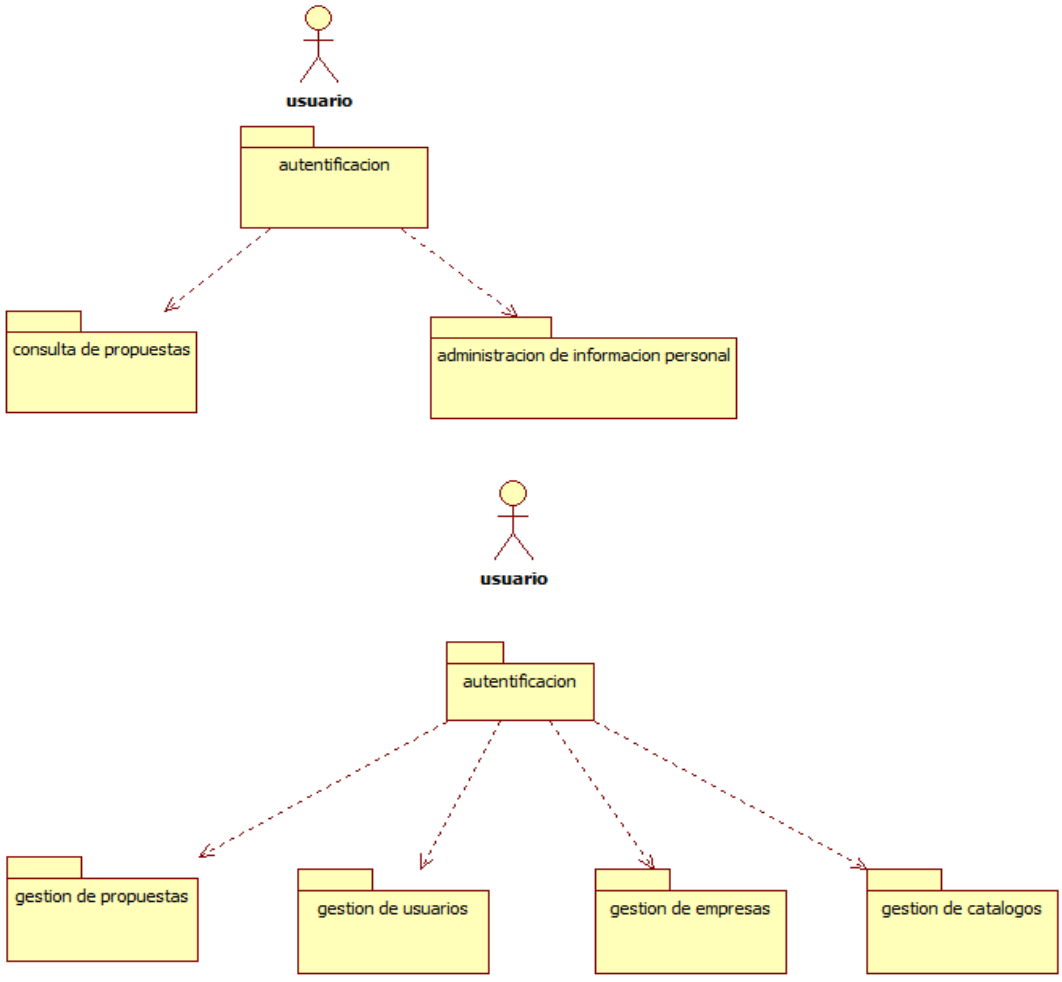

Ilustración 31. Diagrama de paquetes de usuario y administrador

3.4 Representación de la estructura de información

Diagrama E-R

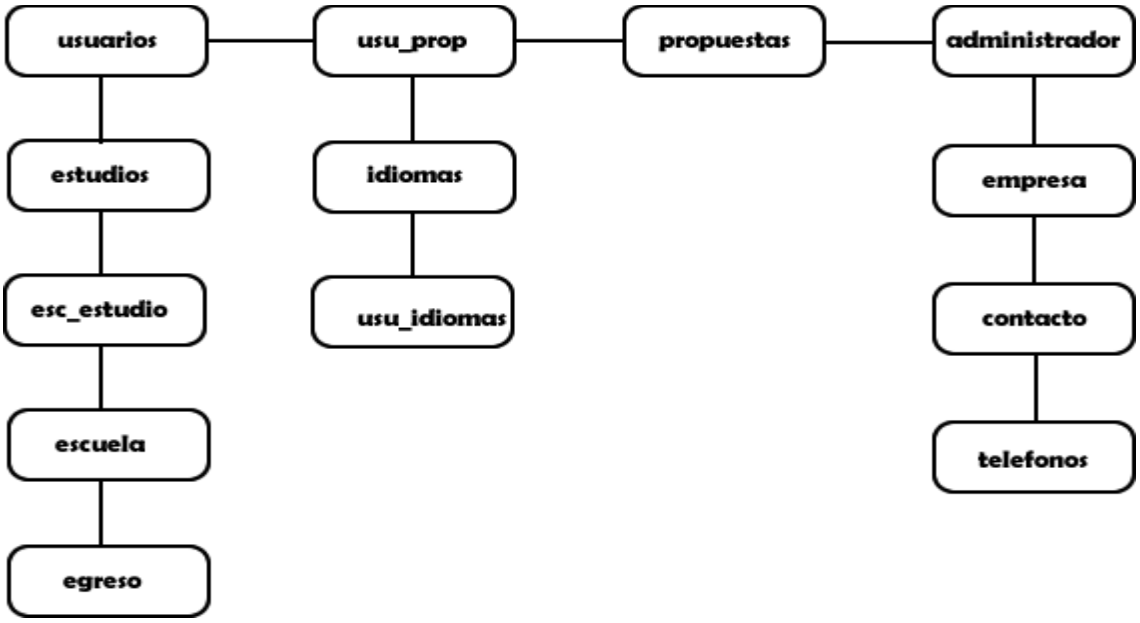

Ilustración 32. Diagrama Entidad Relación

Diagrama de clases

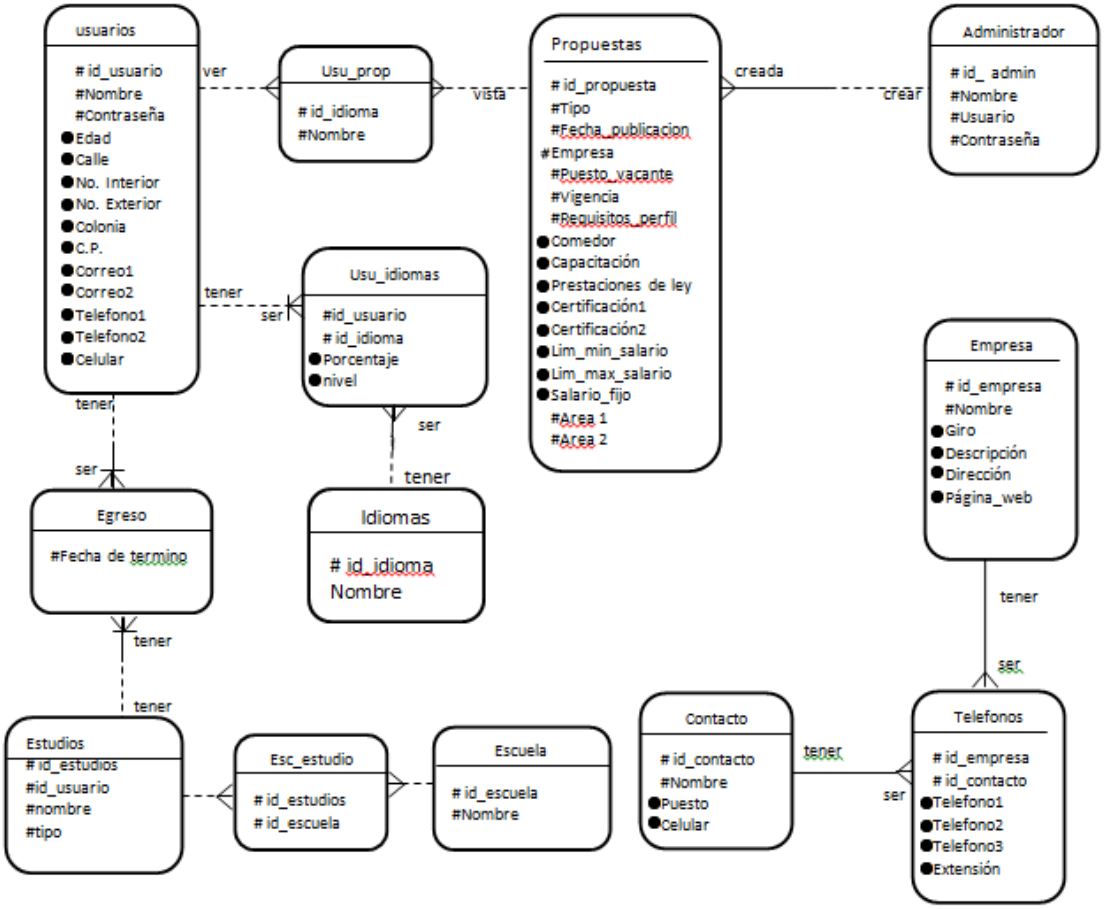

Ilustración 33. Diagrama de Clases

3.5 Descripción de la interfaz del sistema

El administrador del sistema contará con un menú el cual le permitirá realizar todas las acciones necesarias.
El menú cuenta con opciones como:
Propuestas.- aquí se podrán ingresar todas las propuestas necesarias, así como realizar los cambios necesarios o eliminar alguna. Teniendo la opción de poder consultar todas las propuestas por algún método de búsqueda.
Alumnos.- de igual forma aquí se podrá agregar cualquier número de alumnos para el uso del sistema, así como la modificación de información o baja de alguno.
Administrador.- aquí se podrá dar de alta algún usuario con privilegios de administrador, el cual podrá hacer todas las opciones posibles en el sistema.
Empresas.- Aquí se reúne toda la información necesaria de las empresas que ofrecen vacantes, así como sus respectivos datos de contacto para poder hacer comunicación directa.

Los usuarios del sistema podrán revisar todas aquellas propuestas publicadas ya sea de prácticas profesionales, bolsa de trabajo, academia, etc.

3.6 Descripción funcional

El desarrollo del software presentado en este documento fue desarrollado bajo la metodología RUP, se cuidó la reciprocidad entre la ingeniería de requerimientos y el software desarrollado.

Partición funcional

El software denominado: portal de bolsa de trabajo y prácticas profesionales, se encuentra dividido funcionalmente en 4 partes:

1) Registro
2) Administración de propuestas
3) Administración de usuarios
4) Administración de privilegios

1 Descripción funcional

Módulo Registro (usuario)

1.1 Narración del procesamiento
El módulo de registro tiene la función de permitir a un usuario registrar información personal por vez primera, donde este usuario espera la aprobación de una cuenta de usuario, para después ingresar al sistema cada vez que lo requiera.

Se muestra el flujo de trabajo del módulo *registro*

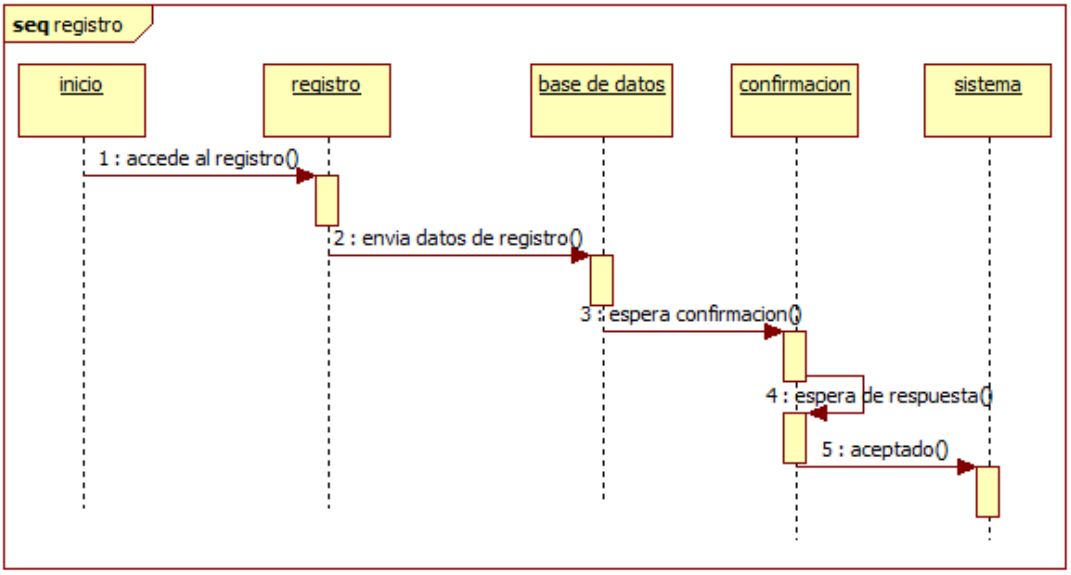

Ilustración 34. Diagrama de secuencia del módulo registro

1.2 Restricciones y limitaciones

El alumno no podrá ingresar al sistema hasta que los campos obligatorios sean llenados, y sea aceptado por algún administrador.

1.3 Requerimientos de funcionamiento

Haber completado el registro.

1.4 Restricciones en diseño

El alumno tiene que esperar hasta que un administrador lo valide para poder hacer uso del sistema.

1.5 Diagrama

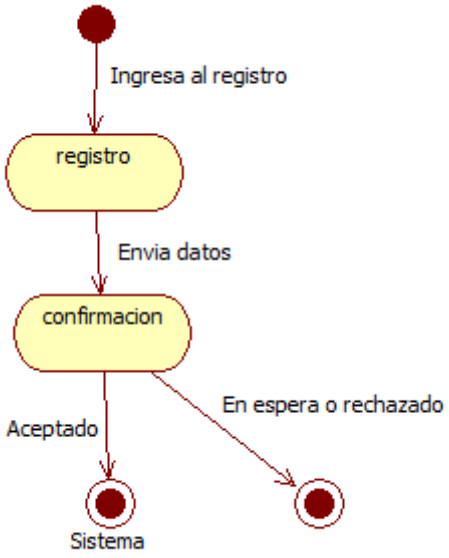

Ilustración 35. Diagrama de actividad del módulo registro

2 Descripción funcional

Módulo Administración de propuestas (usuario)

2.1 Narración del procesamiento

Se muestra el flujo de trabajo del módulo *administración de propuestas*

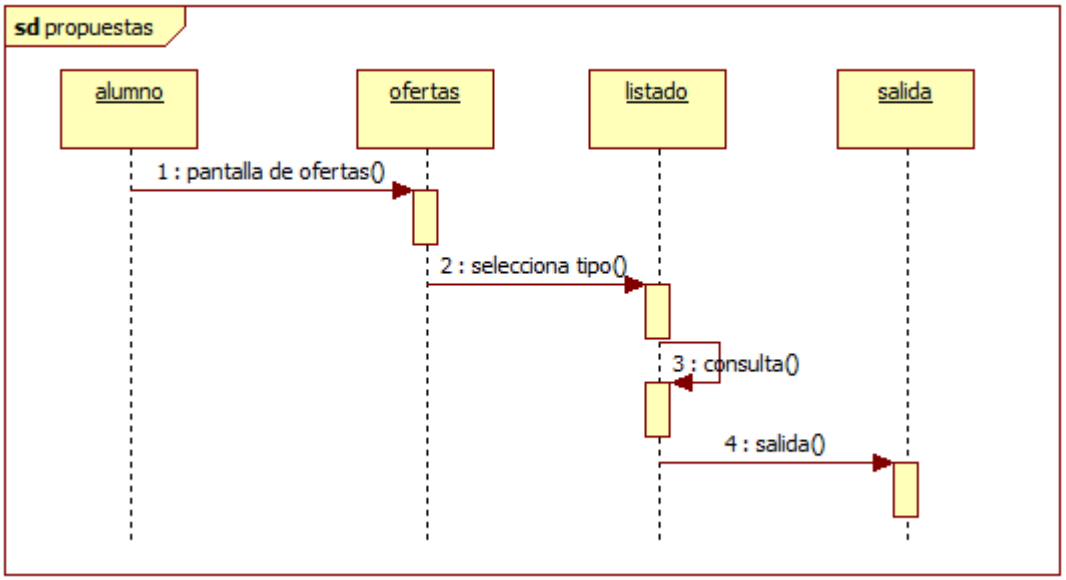

Ilustración 36. Diagrama de secuencia del módulo administración de propuestas

2.2 Restricciones y limitaciones

El alumno no podrá consultar las propuestas hasta que haya seleccionado un tipo de propuesta.

2.3 Requerimientos de funcionamiento

Haber completado el login.

2.4 Restricciones en diseño

El alumno sólo tiene la opción de poder consultar las propuestas ofertadas

2.5 Diagrama

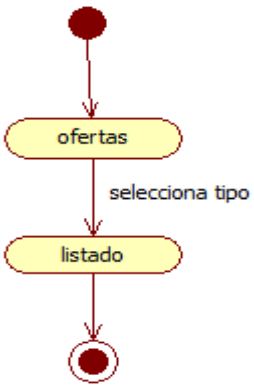

Ilustración 37. Diagrama de actividad del módulo administración de propuestas

3 Descripción funcional

Módulo Administración de usuarios (usuario)

3.1 Narración del procesamiento

Se muestra el flujo de trabajo del módulo *Información*

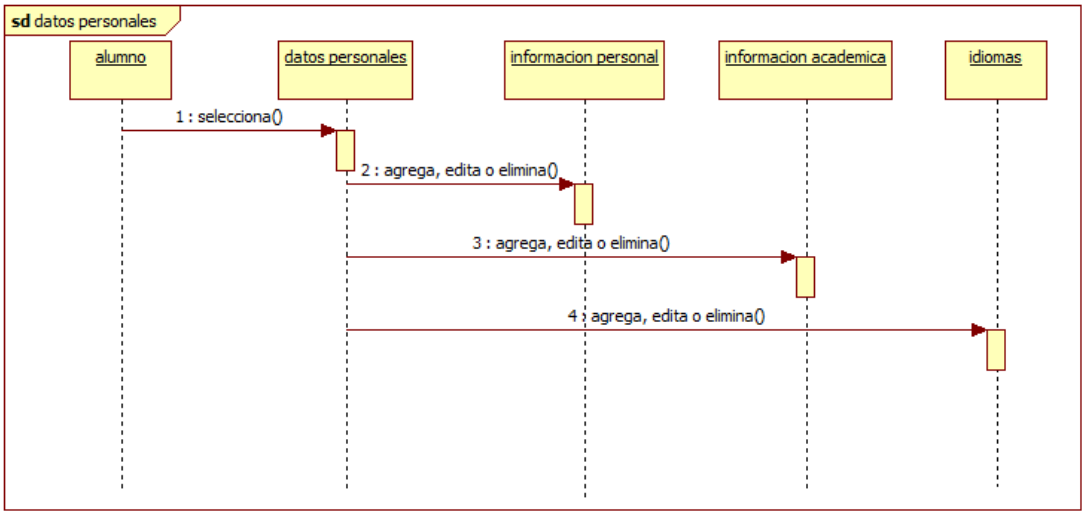

Ilustración 38. Diagrama de secuencia del módulo Información

3.2 Restricciones y limitaciones

El alumno no podrá editar su información hasta que haya realizado el login.

3.3 Requerimientos de funcionamiento

Haber completado el registro y el login.

3.4 Restricciones en diseño

Los datos de información académica e idiomas se pueden llenar hasta ser validados en el sistema.

3.5 Diagrama

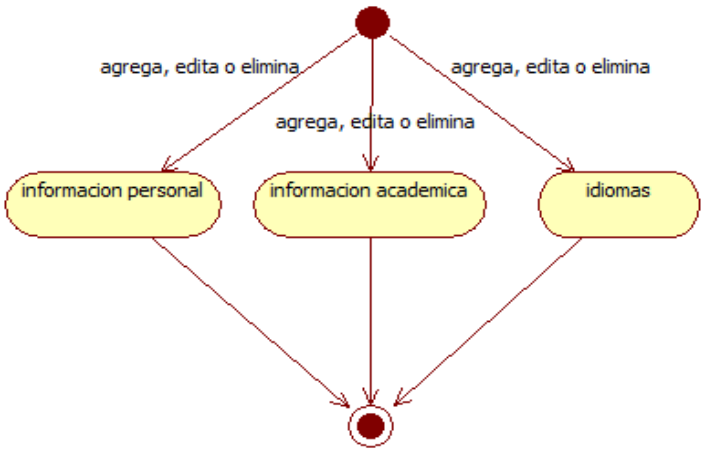

Ilustración 39. Diagrama de actividad del módulo Información

4 Descripción funcional

Módulo Administración de privilegios (administrador)

4.1 Narración del procesamiento

Se muestra el flujo de trabajo del módulo *administración de privilegios*

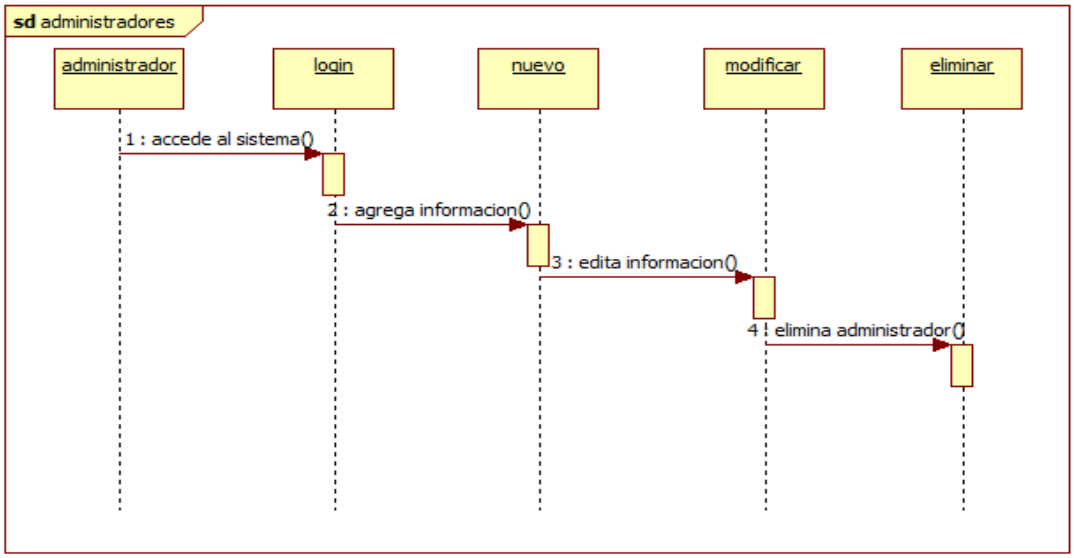

Ilustración 40. Diagrama de secuencia del módulo administración de privilegios

4.2 Restricciones y limitaciones

Si no se llenan los campos obligatorios no se puede continuar.
Haber aceptado la confirmación para eliminar.

4.3 Requerimientos de funcionamiento

Haber sido validado como administrador y haber realizado el login.

4.4 Restricciones en diseño

Si no cuenta con privilegios de administrador no se podrá ingresar a ésta área.
Para eliminar un administrador se tiene que confirmar la operación.

4.5 Diagrama

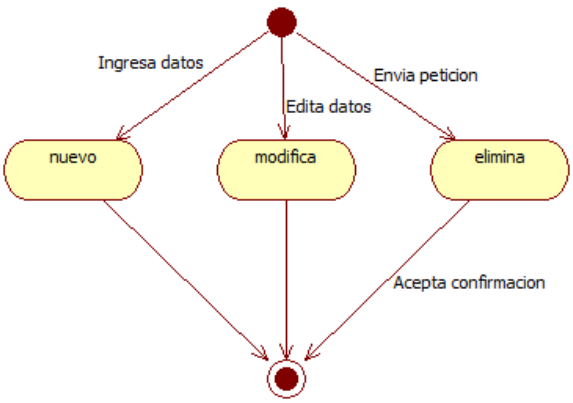

Ilustración 41. Diagrama de actividad del módulo administración de privilegios

5 Descripción funcional

Módulo Administración de usuarios (administrador)

5.1 Narración del procesamiento

Se muestra el flujo de trabajo del módulo Administración de *usuarios*

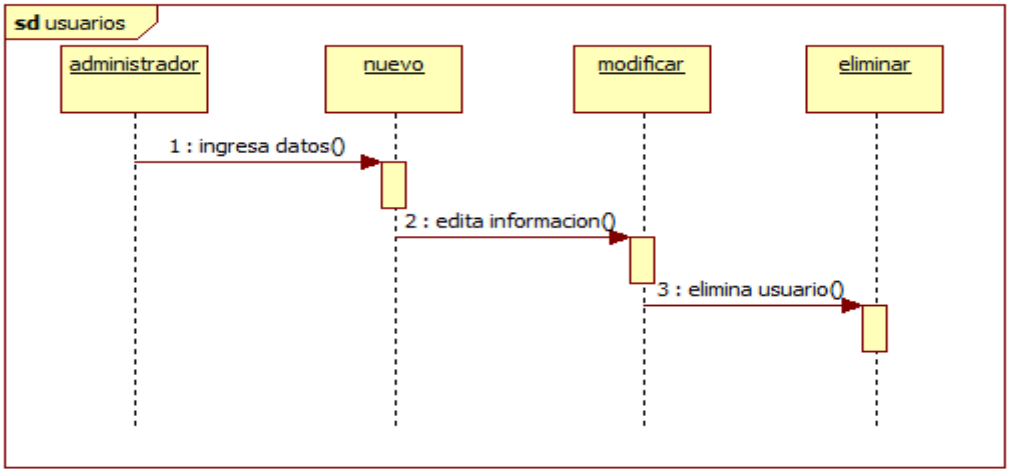

Ilustración 32. Diagrama de secuencia del módulo administración de usuarios

5.2 Restricciones y limitaciones

Si no se llenan los campos obligatorios no se puede continuar.
Haber aceptado la confirmación para eliminar.

5.3 Requerimientos de funcionamiento

Haber sido validado como administrador y haber realizado el login.

5.4 Restricciones en diseño

Contar con los permisos necesarios para poder realizar éstas acciones.
Para eliminar un usuario se tiene que confirmar la operación.

5.5 Diagrama

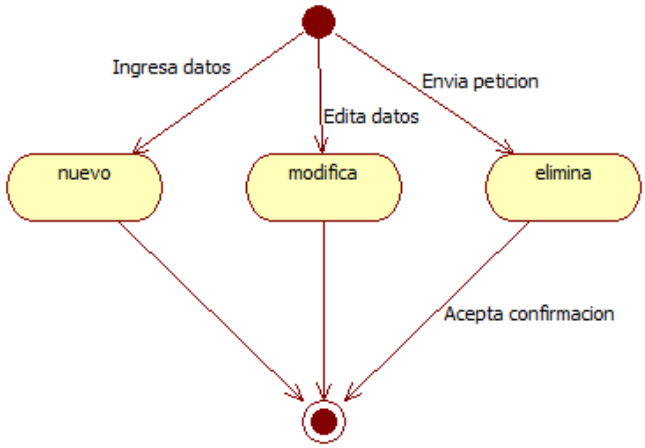

Ilustración 43. Diagrama de actividad del módulo administración de privilegios

6 Descripción funcional

Módulo Administración de propuestas (administrador)

6.1 Narración del procesamiento

Se muestra el flujo de trabajo del módulo Administración de *propuestas.*

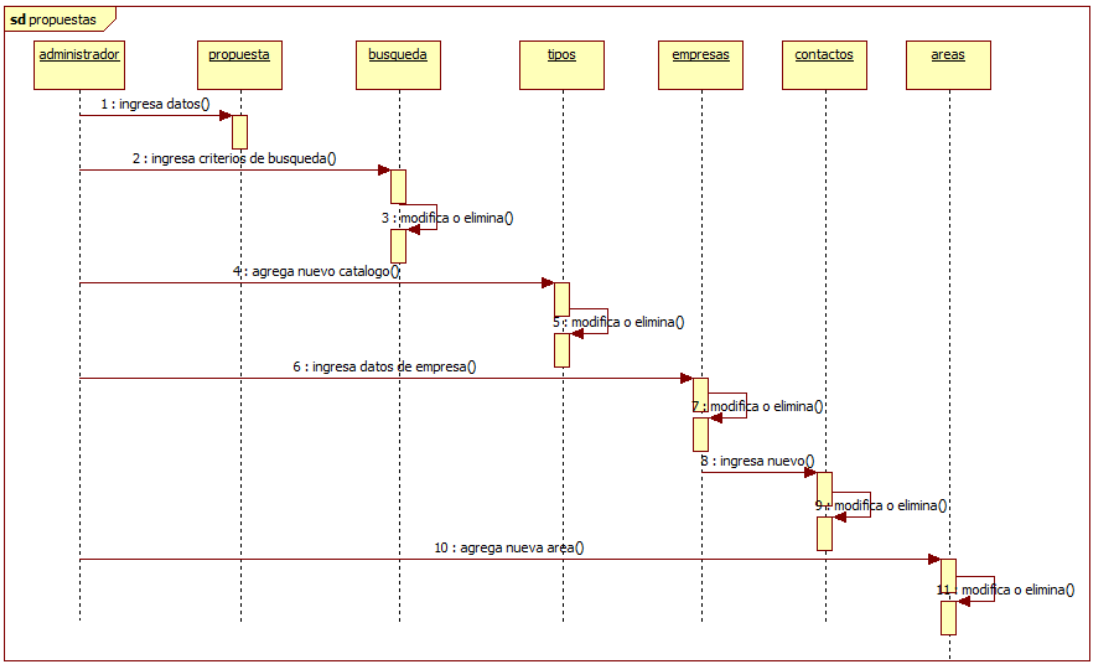

Ilustración 44. Diagrama de secuencia del módulo administración de propuestas

6.2 Restricciones y limitaciones

Si no se llenan los campos obligatorios no se puede continuar.
Haber aceptado la confirmación para eliminar.

6.3 Requerimientos de funcionamiento

Haber sido validado como administrador y haber realizado el login.

6.4 Restricciones en diseño

Contar con los suficientes privilegios para realizar las opciones.

6.5 Diagrama

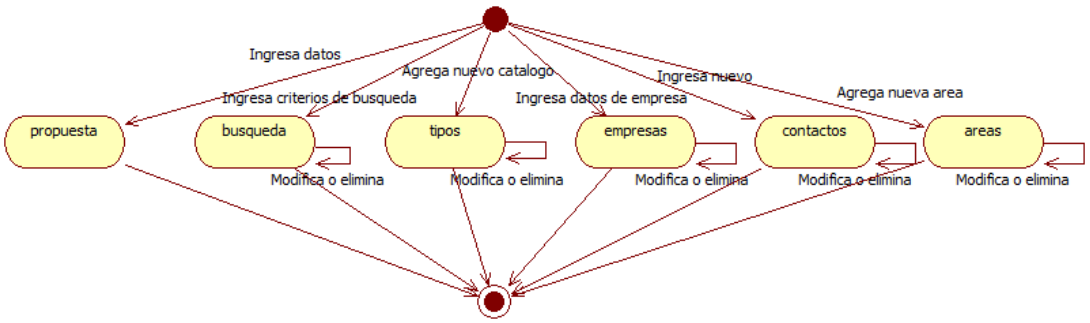

Ilustración 45. Diagrama de actividad del módulo administración de propuestas

3.7 Criterios de Validación

A continuación se describen los criterios de validación que se aplicaron al sistema.

1.1 Límites de funcionamiento

1.2 Clases de pruebas
Pruebas funcionales: como su propio nombre indica, prueban una funcionalidad completa, donde pueden estar implicadas una o varias clases, la propia interfaz de usuario.

Pruebas de aceptación: son pruebas funcionales, pero vistas directamente desde el cliente. Digamos que son aquellas pruebas que demuestran al cliente que la funcionalidad está terminada y funciona correctamente.

Pruebas de integración: conjunto de pruebas unitarias, funcionales, de regresión y/o de aceptación que se realizan las probar el software. Incluye también comprobar que lo programado por los diferentes desarrollados no "choca" entre sí y que funcionará en un entorno real.

1.3 Respuesta esperada
Con la realización de éste sistema se pretende agilizar la gestión de prácticas profesionales y bolsa de trabajo, y llevar un mejor control de las propuestas publicadas a los alumnos.

1.4 Restricciones especiales
Ninguna

Capítulo 4

Implementación del Sistema

4.1 Implementación

Inicio (Login)
Pantalla de bienvenida al usuario, en la cual debe d ingresar un usuario y contraseña para ingresar al sistema.

La página de inicio consta de 5 zonas:
Logo de la Facultad de Ciencias de la Computación.
Slider con fotos de la facultad.
Área de **login.**
Botón de **registro.**
Contacto.- provee información con los datos del coordinador(a) el área de Servicio Social y Prácticas Profesionales.

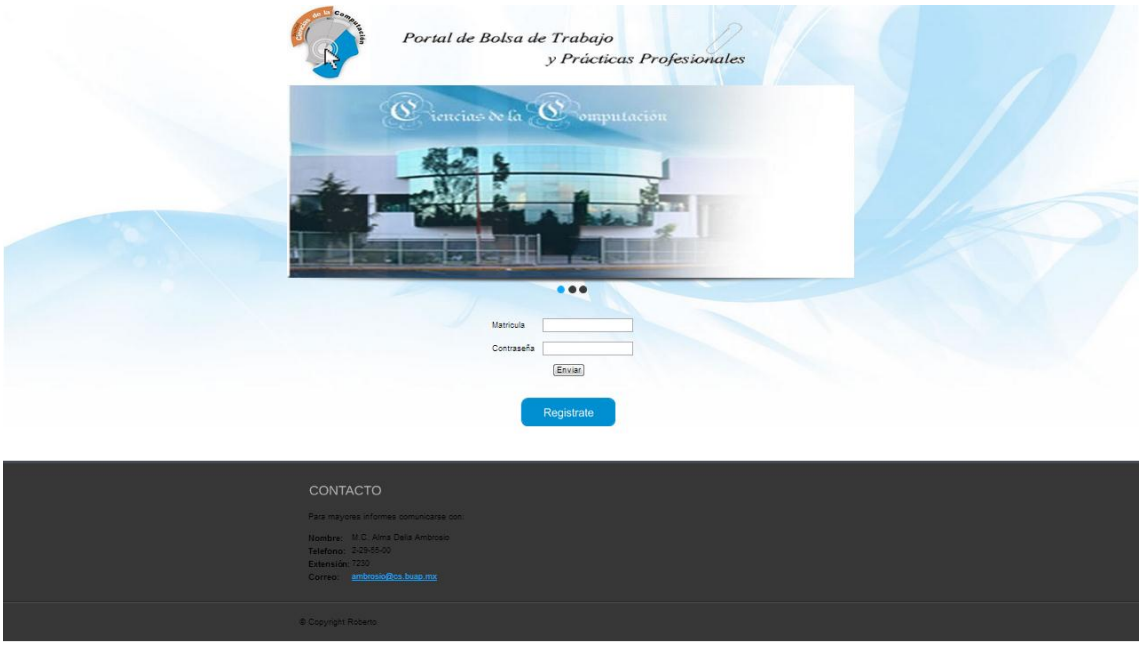

Ilustración 46. Login

Registro

Ésta página le permite al usuario poder registrarse al sistema, para poder hacer uso del mismo.
Se compone de:
Logo del sistema.
Un **menú** para poder regresar a la página de login.
Área de **registro**, donde debe llenar la información personal como: matricula, nombre, apellidos, contraseña, etc.
Habiendo campos obligatorios que deberán ser llenados para completar el registro.
Contacto.- provee información con los datos del coordinador(a) el área de Servicio Social y Prácticas Profesionales.

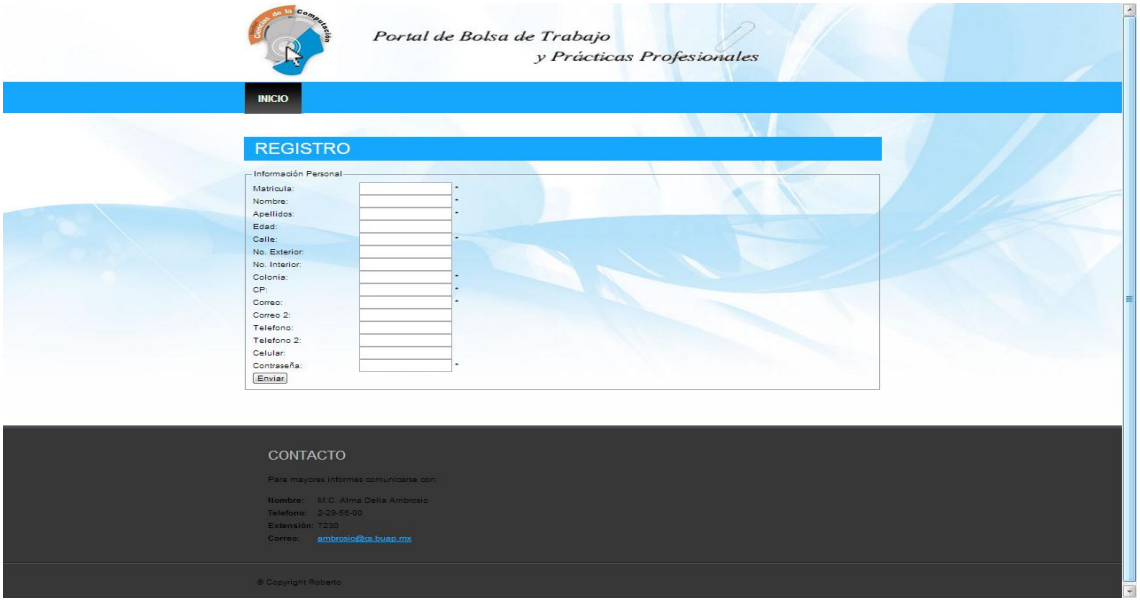

Ilustración 47. Registro

Inicio (Usuarios)

Ésta sección le permite al usuario observar la pantalla de inicio del sistema que contiene:
Logo del sistema.
Menú.- permite navegar las opciones del sistema.
Slider con fotografías de la facultad.
Mensaje de **Bienvenida** personalizado.
Contacto.- provee información con los datos del coordinador(a) el área de Servicio Social y Prácticas Profesionales.

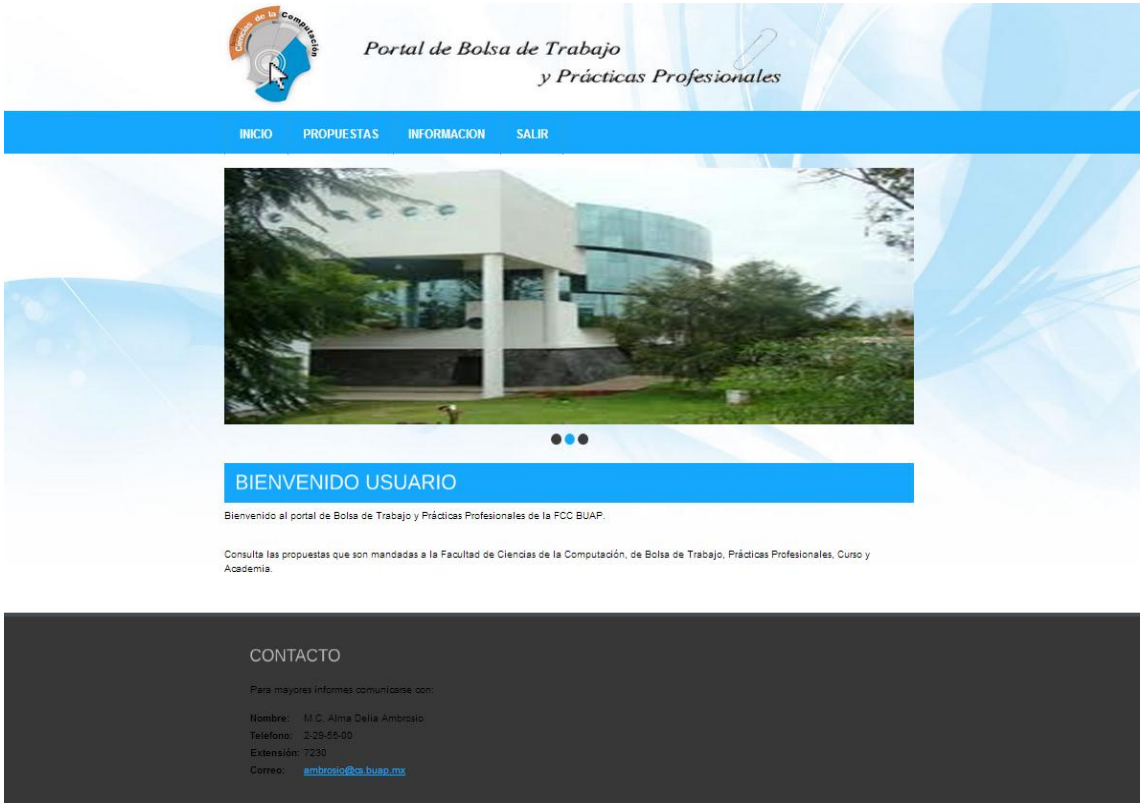

Ilustración 48. Inicio (Usuarios)

Propuestas

En ésta sección el usuario puede observar las propuestas que son publicadas de tipo: academia, bolsa de trabajo, curso o práctica profesional.
Y está compuesta por:
Logo del sistema.
Menú.- permite navegar las opciones del sistema.
Tipos de **propuestas** publicadas.
Contacto.- provee información con los datos del coordinador(a) el área de Servicio Social y Prácticas Profesionales.

Ilustración 49. Propuestas

Dentro de cada tipo de propuesta podemos observar los siguientes campos:
Tipo de propuesta.
Título de la propuesta.
Información complementaria de la propuesta.
Paginación del total de resultados.

Ilustración 50. Contenido de Propuestas

Información

En éste apartado el usuario puede modificar la información que almacenó al momento del registro.

Ilustración 51. Información personal

También cuenta con otras secciones de estudio como:
Información académica.- donde se almacenará la información correspondiente a los estudios obtenidos.
Idiomas.- información de las lenguas extranjeras que tiene estudios.

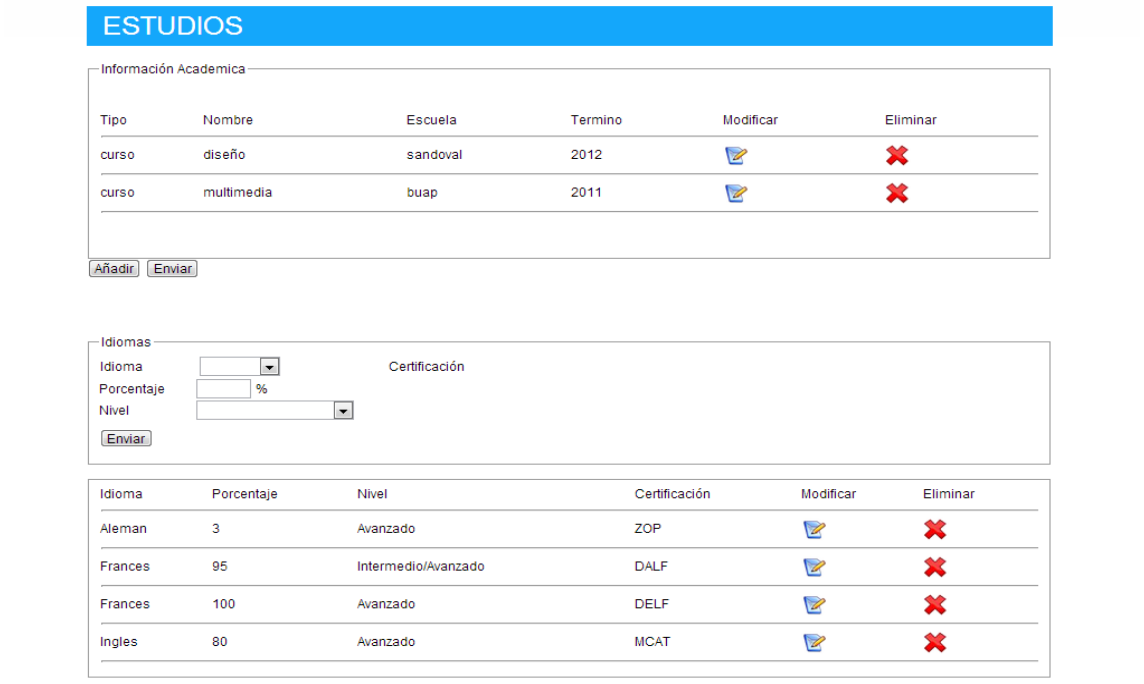

Ilustración 52. Información académica

Salir
Cierra la sesión y regresa a la página de Inicio (Login)

Ilustración 54. Cerrar sesión (Salir)

Inicio (Administrador)

Ésta sección le permite al administrador observar la pantalla de inicio del sistema que contiene:
Logo del sistema.
Menú.- permite navegar las opciones del sistema.
Slider con fotografías de la facultad.
Mensaje de **Bienvenida** personalizado.
Menú secundario permite navegar las opciones del sistema.

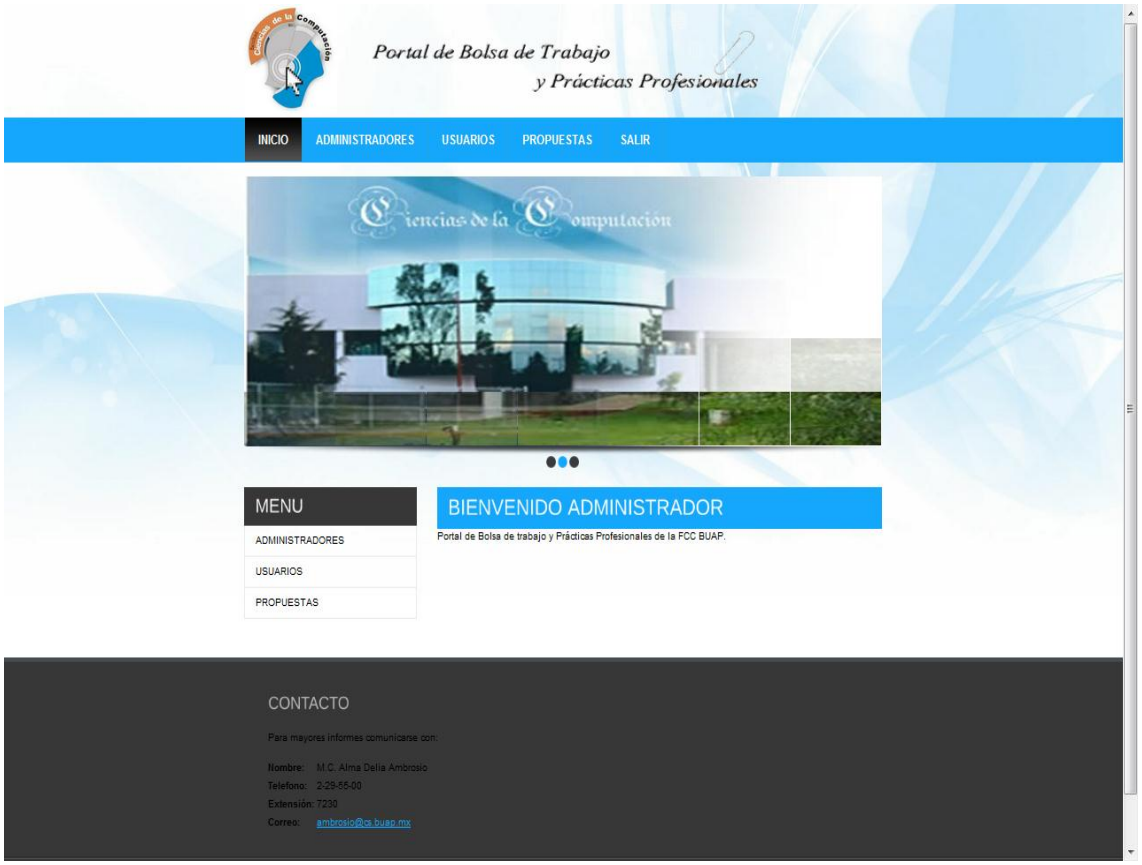

Ilustración 55. Inicio administrador

Administradores
Ésta sección permite agregar al sistema un nuevo administrador, modificar o eliminar.

Se compone de:
Logo del sistema.
Menú.- permite navegar las opciones del sistema.
Área con una descripción de las operaciones que se pueden realizar.
Menú secundario con las opciones que se pueden ejecutar en ésta sección.

Menú secundario de Administradores
Nuevo.- permite agregar un nuevo administrador al sistema.
Modificar/Eliminar.- permite hacer alguna corrección a algún administrador o en caso contrario, eliminarlo del sistema.

Ilustración 56. Sección *Administradores*

Ilustración 57. Submenú administradores (nuevo)

Ilustración 58. Submenú administradores (modificar/eliminar)

Usuarios

Ésta sección permite agregar al sistema un nuevo usuario, modificar o eliminar.

Se compone de:
Logo del sistema.
Menú.- permite navegar las opciones del sistema.
Área con una descripción de las operaciones que se pueden realizar.
Menú secundario con las opciones que se pueden ejecutar en ésta sección.

Menú secundario de Usuarios
Nuevo.- permite agregar un nuevo usuario al sistema.
Modificar/Eliminar.- permite hacer alguna corrección a algún usuario o en caso contrario, eliminarlo del sistema.
Pendientes de revisión.- permite ver los usuarios que se registraron y están en espera de ser validados

Ilustración 59. Sección *Usuarios*

Ilustración 60. Submenú Usuarios (nuevo)

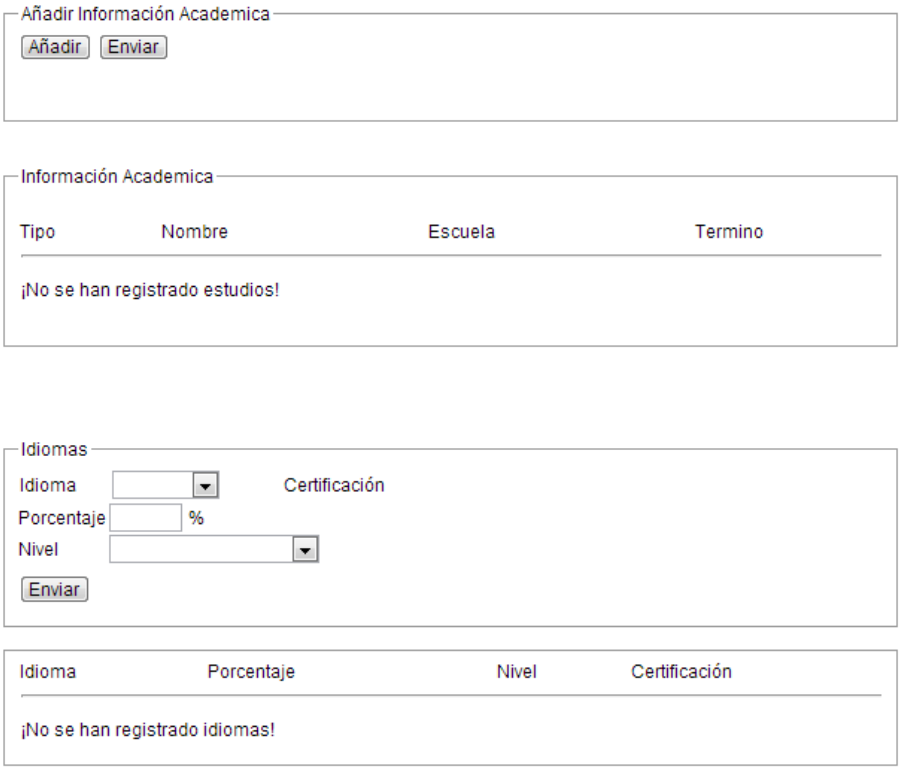

Ilustración 61. Submenú Usuarios (nuevo) 'complemento'

Ilustración 62. Submenú Usuarios (modificar/eliminar)

Ilustración 63. Submenú Usuarios (pendientes de revisión)

Propuestas

Ésta sección permite agregar al sistema una nueva propuesta, empresa, contacto de empresa, así como modificar o eliminar.

Se compone de:
Logo del sistema.
Menú.- permite navegar las opciones del sistema.
Área con una descripción de las operaciones que se pueden realizar.
Menú secundario con las opciones que se pueden ejecutar en ésta sección.

Menú secundario de Propuestas
Nueva.- permite agregar una nueva propuesta al sistema, de tipo academia, curso, bolsa de trabajo, práctica profesional o cualquier tipo que esté dado de alta.
Búsqueda.- permite encontrar alguna propuesta por diferentes criterios: empresa, tipo, área o fecha.
Tipos.- permite añadir al sistema un nuevo tipo de propuesta para que sea publicada, por otra parte permite modificar o eliminar algún *tipo.*
Empresas.- permite añadir empresas, con sus respectivos contactos para poder aparecer en las propuestas añadidas al sistema. Contando con la opción de modificar o eliminar.
Área.- permite añadir al sistema una nueva área, la cual servirá para clasificar las propuestas. Contando con la opción de modificar o eliminar.

Ilustración 64. Sección Propuestas

Ilustración 65. Submenú propuestas (nueva)

Ilustración 66. Submenú propuestas (búsqueda)

Ilustración 67. Submenú propuestas (tipos)

Ilustración 68. Submenú propuestas (empresas)

Ilustración 69. Submenú propuestas (área)

4.2 Conclusiones y perspectivas

Conclusiones

Por medio de metodologías (que son el conjunto de métodos, técnicas y herramientas) se construyó el software "Portal de bolsa de trabajo y prácticas profesionales" con su respectiva documentación, para que sea aprovechado, estudiado y sobre todo usado. En cualquier medio que pueda ser útil.

De cualquier manera la enseñanza que dejó el estudio e implementación, para la elaboración del software, tiene un valor invaluable y sirve de gran experiencia para la vida laboral.

Este trabajo se realizó bajo la metodología RUP, siguiendo un *Lenguaje Unificado de Modelado* (UML) y utilizando como herramienta de trabajo el programa "StarUML", junto con programas como: *Dreamweaver* y *Xampp*.

El uso de éstas herramientas junto con la metodología (RUP) fueron usadas para cumplir con una buena calidad para el diseño e implementación del sistema, ya que RUP es la metodología más usada, y no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada proyecto. Esto permite ir modelando el sistema según sean las necesidades.

El uso de UML sirvió para que cada una de las partes que comprende el desarrollo del software de diseño orientado a objetos, se visualice, especifique y documente con un lenguaje común.

El uso de "StartUML" sirvió para la elaboración de los diagramas, ya que ésta herramienta el de las más usadas y además es de uso gratuito.

Las herramientas *Dreamweaver* y *Xampp* fueron usadas para la creación del código y gestión de base de datos respectivamente.

Perspectiva

Desde el punto de vista personal el software es *bueno* y de gran utilidad, considerando que puede ser modificado para un mejor funcionamiento o para hacer la estructura interna (código) más ordenado o práctico.

Una mejora puede ser el uso de programación con *Ajax* para que las consultas e inserción de datos sean más rápidas y seguras.

Otra modificación puede ser que los usuarios sean notificados vía correo electrónico, de las nuevas propuestas que son publicadas.

El apartado de “Casos de éxito” sería muy bueno ya que así las demás personas pueden ver cómo sus compañeros o conocidos formaron parte de una empresa por medio de éste portal, y cuál es la experiencia que tienen en ese trabajo.

4.3 Apéndice

Requisitos para prácticas profesionales

Jóvenes se les informa lo siguiente:

1) A partir de ahora la información oficial respectiva a Prácticas Profesionales y Bolsa de Trabajo que se les había enviado desde vinculacion@cs.buap.mx, será enviada desde esta misma cuenta profesionales@cs.buap.mx.

2) También se les recuerda que los requisitos para realizar Prácticas Profesionales son:
a) Haber Liberado el Servicio Social.
b) Con el objetivo de ver la Práctica Profesional como una forma de prepararse para empleo, de preferencia estar cursando máximo tres materias.
c) Traer su Kardex simple donde se indique el porcentaje de materias cursado y el promedio de la carrera.
d) Traer currículo con los correspondientes comprobantes.
e) Traer sus datos personales para poder contactarlos durante su Práctica Profesional
f) Y después de la entrevista en la empresa donde realizarían sus Prácticas Profesionales, traer con firma y sello de la empresa el Plan de Trabajo con las actividades a realizar dentro de las Prácticas Profesionales y que correspondan con su carrera.

3) Datos adicionales sobre el procedimiento de Prácticas Profesionales:
a) Las Prácticas Profesionales tienen una duración de 480 horas.

b) Una vez que entreguen toda la documentación indicada en los requisitos (punto 2), se les extiende una carta de Presentación de Practicante.

c) Con el objetivo de que las Prácticas Profesionales sean realizadas en apego al reglamento y en beneficio tanto del Practicante como del Sector Productivo, éstas son supervisadas durante la vigencia de todo este proceso.

d) Es responsabilidad del Practicante actuar con profesionalismo con respecto a la confidencialidad de los datos de la empresa donde realicen sus Prácticas Profesionales.

e) Una vez que concluyan sus Práctica Profesionales, traer la liberación respectiva con sello y firma de la empresa, así como un reporte final donde se indiquen las actividades que fueron realizadas de igual manera con sello y firma de la empresa.

f) Con la liberación de la empresa se les extiende una Carta de Liberación de Prácticas Profesionales.

4) Ante un entorno globalizado en el que estamos ya inmersos y con el objetivo de que cada día te prepares más para el mercado laboral actual, te hacemos llegar las siguientes recomendaciones:

a) Con base en la experiencia de trabajo con distintas empresas del Sector Productivo es relevantemente prioritario tu actitud: Ser proactivo, trabajo en equipo, tus habilidades de comunicación y respeto, habilidades para socializarte, etc. (soft skills).

b) También se te recuerda que en la mayoría de las empresas para los egresados de nuestras carreras es determinante para poderse insertar o no, un nivel del idioma inglés hablado, escrito y leído, en un porcentaje mínimo del 80%.

c) Tu promedio en la carrera es importante, pero ahora en el Sector Productivo les aplican examen de conocimientos dependiendo del área donde pretendas insertarte, así que es recomendable te especialices en el área de tu preferencia.

d) Es recomendable que en cuanto cubran el 70% de sus créditos, realicen el Servicio Social, para liberarlo a tiempo y ya estar listos cuando una oferta de Práctica Profesional que sea de tu interés sea publicada.

En caso de cualquier duda acudir a la Coordinación de Práctica Profesional y Bolsa de Trabajo.

Requisitos para una convocatoria

Tipo de oferta: Práctica Profesional, empleo, academia, etc.

Datos completos de la empresa:

a) Nombre completo
b) Giro
c) Dirección
d) Teléfonos
e) Año de fundación

Datos completos del contacto dentro de la empresa:

a) Nombre completo
b) Teléfonos
c) Emails
d) Puesto

Perfil buscado:

a) Nombre de la vacante o puesto
b) Perfil académico
c) Perfil de desarrollo personal y /o datos adicionales.

Lo que la empresa ofrece:

a) Prestaciones
b) Salario o en su caso rango de salario.
c) Capacitación
d) Certificación
e) Servicios (transporte, comedor, etc.)

M.C. ALMA DELIA AMBROSIO VÁZQUEZ
COORDINADORA DE PRÁCTICAS PROFESIONALES Y BOLSA DE TRABAJO
FAC. DE CS. DE LA COMPUTACIÓN
BUAP

4.4 Bibliografía

[1] Fundamentos de seguridad en redes aplicaciones y estándares | 2° Edición | William Stallings | Prentice Hall
[2] http://www.webtaller.com/maletin/articulos/seguridad-web.php
[3] http://www.xombra.com/go_articulo.php?nota=131
[4]http://www.monografias.com/trabajos75/seguridad-desarrollo-aplicaciones-web/seguridad-desarrollo-aplicaciones-web2.shtml#mitossobra
[5] Sistemas de bases de datos. Diseño, implementación y administración | Peter Rob, Carlos Coronel | Cengage Learning Editores
[6]http://www.maestrosdelweb.com/principiantes/%C2%BFque-son-las-bases-de-datos/
[7] Base de datos relacionales | Fray León Osorio Rivera | Instituto tecnológico metropolitano | 1ª edición: Diciembre de 2008
[8] Sistemas de bases de datos. Diseño, implementación y administración | Peter Rob, Carlos Coronel | Cengage Learning Editores
[9] http://support.microsoft.com/kb/283878/es
[10] http://definicion.de/ingenieria-de-software/
[11] Ingenieria del Software, Ian Sommerville (Prentice Hall)
Prentice Hall | Ian Sommerville | 7ma Edicion
[12] http://www.ciberaula.com/articulo/tecnologia_orientada_objetos/
[13] http://es.wikipedia.org/wiki/Proceso_Unificado_de_Rational
[14] http://es.scribd.com/doc/31440864/Metodologia-RUP
[15] http://www.docirs.cl/uml.htm
[16] http://black-byte.com/review/staruml/