45

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES
ARAGÓN

"SISTEMA DE CONTROL PARA EL PROGRAMA DE SEGUIMIENTO DE EGRESADOS DE LA ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES CAMPUS ARAGÓN (Pro-SE-A)"

T E S I S

QUE PARA OBTENER EL TÍTULO DE :

INGENIERO EN COMPUTACIÓN

P R E S E N T A :

KARINA JANETTE OLVERA ALVAREZ

ASESOR:
M. en I. JESÚS DÍAZ BARRIGA

MÉXICO 2002

DEDICATORIA

A DIOS

Por haberme cubierto de bendiciones cada día de mi vida.

A MI PADRE

Por enseñarme a ganar y a perder, a aprender de los errores, a valorar lo que tenemos.

A MI MADRE

Por ser la fuerza que me impulsa a seguir viviendo cada día.

A MIS HERMANOS JORGE, PAOLA Y VANESSA

Por compartir conmigo momentos de alegría y dolor, siempre juntos.

A MI ABUELITA AMPARO

Por ser mi amiga y cómplice, y por todo el cariño que me ha brindado.

A MIS TIAS Y TIOS

Porque con cada uno de ustedes no perdí un papá, sino gane ocho parejas de ellos, ayudándome en todo momento, brindándome su apoyo y comprensión.

A MIS AMIGAS KARLA, VASTHY Y GLORIA

Por ser un apoyo, por todo lo hemos vivido y viviremos juntas.

A MIS COMPAÑEROS DE BECAS JAVA

A CLAUDIA, SILVIA, MIGUEL, VICTOR, JUAN Y ALEJANDRA

Por haber compartido conmigo conocimiento y amistad.

A MIS AMIGAS Y AMIGOS

INCLUYENDO A MIS PRIMOS, PRIMAS, SOBRINOS Y SOBRINAS

Porque siempre han estado a mi lado en buenos y malos tiempos, por toda la alegría que compartimos.

A MIS MAESTROS

AL M. EN I. JESUS DIAZ, M. EN C. LUIS RAMIREZ, E ING. RICARDO OROZCO

Por su paciencia, ayuda e impulso para la realización de este trabajo.

Índice

Karina Janette Olvera Alvarez
Ingeniería en Computación
Generación 95 - 99

INTRODUCCION

OBJETIVO GENERAL .-

El objetivo principal del presente trabajo es realizar el Sistema de Control para el Programa de Seguimiento de Egresados de la Escuela Nacional de Estudios Profesionales Campus Aragón (ProSE-A) dando a conocer algunas de las posibilidades de desarrollo del Lenguaje Java a través de un sistema sencillo y amigable al usuario que se pueda visualizar en web.

OBJETIVOS PARTICULARES .-

Dar a conocer el lenguaje Java, sus conceptos básicos, ventajas y desventajas con respecto a otros lenguajes; para ampliar su uso como lenguaje de programación sencillo, eficaz y robusto.

Desarrollar el Sistema de Control para el Programa de Seguimiento de Egresados de la Escuela Nacional de Estudios Profesionales Campus Aragón (ProSE-A) como una herramienta para web para ser utilizada por alumnos, exalumnos y egresados de la escuela así como de los Jefes de Carrera o diversos funcionarios de la misma a fin de mantener un lazo de unión entre ellos; y sobre todo, de apoyar los programas de regularización, titulación y postgrado.

JUSTIFICACION .-

La importancia de contar con un sistema automatizado a través de Internet que cumpla la misión de atraer a los alumnos que por diversas circunstancias han abandonado su carrera o su proceso de titulación; es la razón por la cual se construye el Sistema de Control para el Programa de Seguimiento de Egresados de la Escuela Nacional de Estudios Profesionales Campus Aragón (ProSE-A); como un punto de enlace entre la Universidad y sus alumnos y egresados.

HIPÓTESIS .-

Con este trabajo se pretende describir la realización del Sistema de Control para el Programa de Seguimiento de Egresados de la Escuela Nacional de Estudios Profesionales Campus Aragón (ProSE-A); dando a conocer el lenguaje Java utilizándolo conforme la programación orientada a objetos, además de el análisis y el diseño orientado a objetos a través de la notación UML.

CAPITULADO .-

En el Primer Capitulo se revisa la historia del Lenguaje Java y la Programación Orientada a Objetos; y se dan a conocer sus conceptos básicos así como sus ventajas y desventajas con respecto a otros lenguajes.

En el Segundo Capítulo se muestra el análisis y diseño del Sistema de Control para el Programa de Seguimiento de Egresados de la Escuela Nacional de Estudios Profesionales Campus Aragón (ProSE-A) como una herramienta para web; utilizando la notación UML.

En el Tercer Capitulo especifica el desarrollo de la construcción del Sistema de Control para el ProSE-A; así como las limitaciones que existieron para su construcción.

A continuación se presentan las Conclusiones a las que se llego en base al desarrollo del trabajo; además de un Glosario de términos técnicos en donde se podrán encontrar algunas expresiones utilizadas a lo largo de la Tesis.

También podemos encontrar, como material Anexo, el Cuestionario de información requerida en los desayunos de egresados organizados por la ENEP – Aragón; así como el Manual de Usuario del Sistema. Al final se encontrará la referencia Bibliográfica utilizada en la realización de este trabajo de tesis.

CAPITULO I

"Historia y Conceptos Básicos del Lenguaje Java."

Introducción a la Programación Orientada a Objetos y Java.

Historia de la Orientación a Objetos

La Orientación a Objetos (O.O.) surge en Noruega en 1967 con un lenguaje llamado Simula 67, desarrollado por Krinsten Nygaard y Ole-Johan Dahl, en el centro de cálculo noruego.

Simula 67 introdujo por primera vez los conceptos de clases, co-rutinas y subclases (conceptos muy similares a los lenguajes Orientados a Objetos de hoy en día).

El nacimiento de la Orientación a Objetos en Europa pasó inadvertido para gran parte de los programadores. Hoy tenemos la Orientación a Objetos como un niño de mas de 33 años al que todos quieren bautizar.

Uno de los problemas de inicio de los años setentas era que pocos sistemas lograban terminarse, pocos se terminaban con los requisitos iniciales y no todos los que se terminaban cumpliendo con los requerimientos se usaban según lo planificado. El problema consistía en cómo adaptar el software a nuevos requerimientos imposibles de haber sido planificados inicialmente.

Un alto grado de planificación y previsión es contrario a la propia realidad. El hombre aprende y crea a través de la experimentación, no de la planeación. La Orientación a Objetos brinda estos métodos de experimentación, no exige la planificación de un proyecto por completo antes de escribir la primer línea de código.

En los 70's científicos del centro de investigación en Palo Alto Xerox (Xerox park) inventaron el lenguaje Small talk que dio soporte al hecho anterior (investigar no planificar).

Small talk fue el primer lenguaje Orientado a Objetos puro de los lenguajes Orientados a Objetos, es decir, únicamente utiliza clases y objetos (Java usa tipos de datos primitivos, o bien los Wrappers que son clases que encapsulan tipos de datos primitivos).

Hasta ese momento uno de los defectos más graves de la programación era que las variables eran visibles desde cualquier parte del código y podían ser modificadas

incluyendo la posibilidad de cambiar su contenido (no existen niveles de usuarios o de seguridad, o lo que se conoce como visibilidad).

Quien tuvo la idea fue D. Parnas cuando propuso la disciplina de ocultar la información. Su idea era encapsular cada una de las variables globales de la aplicación en un solo módulo junto con sus operaciones asociadas, sólo mediante las cuales se podía tener acceso a esas variables.

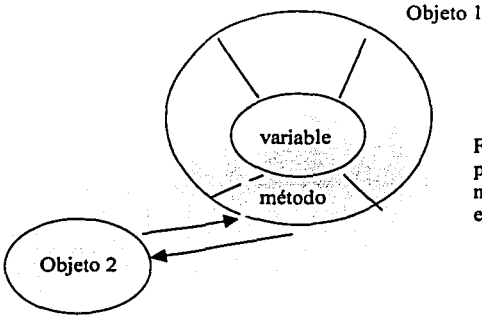


Fig. 1 Otros objetos sólo pueden acceder a los métodos para obtener el estado de las variables

El resto de los módulos (objetos) podían acceder a las variables sólo de forma indirecta mediante las operaciones diseñadas para tal efecto.
En los años 80's Bjarne Stroustrup de AT&T Labs., amplió el lenguaje C para crear C++ que soporta la programación Orientada a Objetos.

En esta misma década se desarrollaron otros lenguajes Orientados a Objetos como Objective C, Common Lisp Object System (CIOS), Object Pascal, Ada y otros.
Posteriores mejoras en herramientas y lanzamientos comerciales de C++ por distintos fabricantes, justificaron la mayor atención hacia la programación Orientada a Objetos en la comunidad de desarrollo de software. El desarrollo técnico del hardware y su disminución de costo fue el detonante final. Con más computadoras al alcance de más personas, más programadores, más problemas y más algoritmos surgieron.

En el inicio de los 90's se consolida la Orientación a Objetos como una de las mejores maneras para resolver problemas. Aumenta la necesidad de generar prototipos más rápidamente (concepto RAD Rapid Aplication Developments o Desarrollo de

Aplicación Rápida) sin esperar a que los requerimientos iniciales estén totalmente precisos. La programación orientada a eventos (POE) es un RAD (un evento, en programación, es el click del mouse, el doble click, arrastrar, soltar, entrar, salir del mouse, la digitación de una tecla, la minimización de una ventana, etc.) El 95% de las aplicaciones de programación orientada a eventos utiliza una GUI (interfaz gráfica de usuario). A bajo nivel están basadas en Programación Orientada a Objetos (POO). Por ejemplo, Delphi es un lenguaje con POE que utiliza POO (Pascal 7.0).

En 1996 surge un desarrollo llamado JAVA (extensión de C++). Su filosofía es aprovechar el software existente. Facilitar la adaptación del mismo a otros usos diferentes a los originales sin necesidad de modificar el código ya existente.
En 1997-98 se desarrollan herramientas 'CASE' orientadas a objetos.

Del 98 a la fecha se desarrolla la arquitectura de objetos distribuidos RMI, Corba, COM, DCOM.

Actualmente la orientación a objetos parece ser el mejor paradigma, no obstante, no es una solución a todos los problemas. Trata de eliminar la crisis del software.
Entre los creadores de metodologías orientadas a objetos se encuentran: G. Booch, Rambaught, Ivar Jacobson y Peter Cheng.

¿Qué es la Orientación a Objetos?
Es una manera de pensar, otra manera de resolver un problema; lo más reciente en metodologías de desarrollo de software. Es un proceso mental humano aterrizado en una computadora.

"Si nos detenemos a pensar en un determinado problema que intentamos resolver podremos identificar entidades de interés, las cuales pueden ser objetos potenciales que poseen un conjunto de propiedades o atributos, y un conjunto de métodos mediante los cuales muestran su comportamiento. Y no solo eso, también podemos ver, poniendo un poco más de atención, un conjunto de interrelaciones entre ellos conducidas por los mensajes a los que responden mediante métodos."[1]

Antes se adecuaba el usuario al entendimiento de la computadora. Actualmente, se le enseña a la computadora a entender el problema.

[1] Java 2 Curso de Programación; Fco. Javier Cevallos; Ed. Alfaomega Ra-Ma; México; 2000.

La Orientación a Objetos es un paradigma, es decir, es un modelo para aclarar algo o para explicarlo. La Orientación a Objetos es el paradigma que mejora el diseño, desarrollo y mantenimiento del software ofreciendo una solución a largo plazo a los problemas y preocupaciones que han existido desde el comienzo del desarrollo del software: La falta de portabilidad del código, su reusabilidad, la modificación (que antes era difícil de lograr), ciclos de desarrollo largo, técnicas de programación no intuitivas.

La Orientación a Objetos está basada en los tres métodos de organización que utilizamos desde la infancia; entre un objeto y sus atributos (automóvil > marca, color, número de llantas, etc.); Entre un objeto y sus componentes donde incluso otros objetos pueden formar parte de otros objetos (agregación) (camión > motor, parabrisas, llantas); entre un objeto y su relación con otros objetos (camión > vehículos automotores; una bicicleta no entraría en esta relación).

La metodología del software orientado a objetos consiste en:
- Saber el espacio del problema
- Realizar una abstracción
- Crear los objetos (espacio de la solución)
- Instanciarlos (esto es, traerlos a la vida)
- Dejarlos vivir (ellos ya saben lo que tienen que hacer)

Crisis del software.

La velocidad de desarrollo del hardware supera al software, de manera que los productos que se desarrollan de éste son creados para usarse con un procesador dos modelos anteriores al más reciente.

De la misma forma, en la curva del software se ha llegado a obtener una curva distinta a la ideal, debido a la aparición constante de nuevos fallos después de haber sido lanzado un nuevo software.

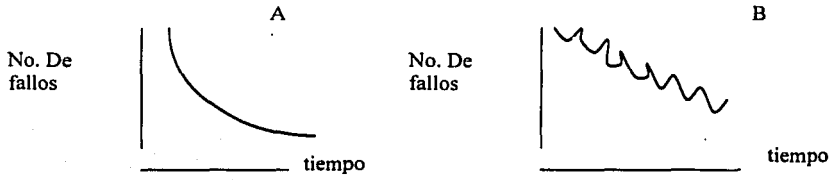


Fig. 2 La grafica A representa la curva ideal del desarrollo de software y la gráfica B la curva real.

Con lenguajes orientados a objetos sí se cubre la curva ideal y se puede alcanzar el siguiente procesador.

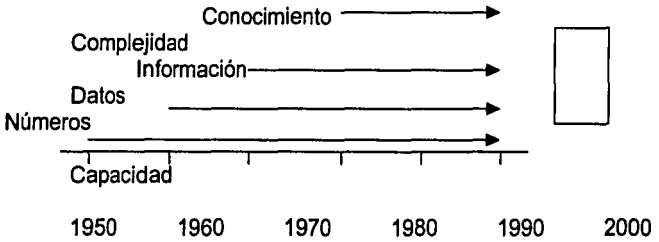


Fig. 3 Representa el avance que se ha tenido en software a través de los años para alcanzar la curva ideal.

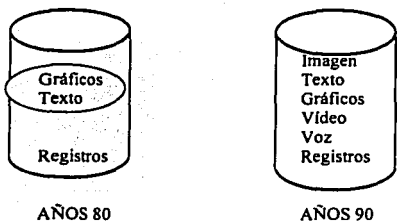


Fig. 4 Representa la diferencia entre una década y otra con respecto a la información que se puede almacenar.

¿Qué es la programación Orientada a Objetos?
Es la programación que utiliza los términos basados en clases, objetos, métodos, etc.

"Un estilo de programación en el que un programa se contempla como un conjunto de objetos limitados que, a su vez, son colecciones independientes de estructuras de datos y rutinas que interactúan con otros objetos. Una clase define las estructuras de datos y rutinas de un objeto. Un objeto es una instancia de una clase, que se puede usar como una variable en un programa. En algunos lenguajes orientados a objetos, éste responde a mensajes, que son el principal medio de comunicación. En otros lenguajes orientados a objeto se conserva el mecanismo tradicional de llamadas a procedimientos. "[2]

"La **Programación Orientada a Objetos** (POO) es un modelo de programación que utiliza objetos, ligados mediante mensajes, para la solución de problemas. Puede considerarse como una extensión natural de la programación estructurada en un intento de potenciar los conceptos de modularidad y reutilización del código."[3]

Los **Mecanismos Básicos de la Programación Orientada a Objetos** son:

- Objetos: un objeto es una entidad que tiene unos atributos particulares, las propiedades, y unas formas de operar sobre ellos, los métodos.
- Mensajes: un mensaje está asociado con un método, de tal forma que cuando un objeto recibe un mensaje, la respuesta a este mensaje es ejecutar el método asociado.
- Métodos: un método se implementa en una clase de objetos y determina cómo tiene que actuar el objeto cuando recibe el mensaje vinculado con ese método. A su vez, un método puede también enviar mensajes a otros objetos solicitando una acción o información. En adición, las propiedades (atributos) definidas en la clase permitirán almacenar información para dicho objeto.
- Clases: una clase es un tipo de objetos definido por el usuario. Una clase equivale a la generalización de un tipo específico de objetos. Un objeto de una determinada clase se crea en el momento en que se define una variable de dicha clase. Algunos autores emplean el término **instancia** (traducción directa

[2] *Enciclopedia Microsoft® Encarta® 98* © 1993-1997 Microsoft Corporation.
[3] Java 2 Curso de Programación; Fco. Javier Cevallos; Ed. Alfaomega Ra-Ma; México; 2000.

de instance), en el sentido de que una instancia es la representación concreta y especifica de una clase. Cuando se escribe un programa utilizando un lenguaje orientado a objetos, no se definen objetos verdaderos, se definen clases de objetos, donde una clase se ve como una plantilla para múltiples objetos con características similares. Afortunadamente no se tendrán que escribir todas las clases que necesite en su programa, porque Java proporciona una biblioteca de clases estándar para realizar las operaciones más habituales que podamos requerir.

Las **características fundamentales de la Programación Orientada a Objetos** son:

- **Abstracción**: por medio de la abstracción conseguimos no detenernos en los detalles concretos de las cosas que no interesen en cada momento, sino generalizar y centrarse en los aspectos que permitan tener una visión global del problema.
- **Encapsulamiento**: esta característica permite ver un objeto como una caja negra en la que se ha introducido de alguna manera toda la información relacionada con dicho objeto. Esto nos permitirá manipular los objetos como unidades básicas, permaneciendo oculta su estructura interna. La abstracción y el encapsulamiento están representados por la clase. La clase es una abstracción, porque en ella se definen las propiedades o atributos de un determinado conjunto de objetos con características comunes, y es un ejemplo de encapsulamiento porque constituye una caja negra que encierra tanto los datos que almacena cada objeto como los métodos que permiten manipularlos.
- **Herencia**: la herencia permite el acceso automático a la información contenida en otras clases. De esta forma, la reutilización del código está garantizada. Con la herencia todas las clases están clasificadas en una jerarquía estricta. Cada clase tiene su superclase (la clase superior en la jerarquía), y cada clase puede tener una o más subclases (las clases inferiores en la jerarquía). Las clases que están en la parte inferior en la jerarquía se dice que heredan de las clases que están en la parte superior en la jerarquía. El término heredar significa que las subclases disponen de todos los métodos y propiedades de su superclase. Este mecanismo proporciona una forma rápida y cómoda de extender la funcionalidad de una clase. En Java cada clase sólo puede tener una superclase, lo que se denomina **herencia simple**. En otros lenguajes orientados a objetos, como C++, las clases pueden tener más de una superclase, lo que se conoce como **herencia múltiple**. Java soluciona lo

complicado de este concepto y su comportamiento compartido utilizando interfaces. Una **interfaz** es una colección de nombres de métodos, sin incluir sus definiciones, que puede ser añadida a cualquier clase para proporcionarle comportamientos adicionales no incluidos en los métodos propios o heredados.

- **Polimorfismo**: Esta característica permite implementar múltiples formas de un mismo método, dependiendo cada una de ellas de la clase sobre la que se realice la implementación. Esto hace que se pueda acceder a una variedad de métodos distintos (todos con el mismo nombre) utilizando exactamente el mismo medio de acceso.[4]

Qué es un Objeto y sus partes.

Un objeto es dinámico, tiene un ciclo de vida; una clase al pasar por un procesador se convierte en objeto, toma vida. Un sustantivo es candidato fuerte a ser objeto, y un verbo es candidato fuerte a ser un método.

Un objeto es un Tipo de Dato Abstracto (TDA). Tiene características llamadas atributos y tiene un comportamiento denominado como método. Es una abstracción que puede representar algo tangible o algo no tangible. En el lenguaje C los métodos son funciones, en Java son métodos. Todo objeto tiene por lo menos un atributo y como mínimos dos métodos: set y get (asignar y obtener un valor).

```
x.setvalor(1);
y.setvalor(2)
z.setvalor(x.getvalor()+y.getvalor());
```

Los objetos son creados a partir de Clases, una clase de objetos es un TDA. Todo objeto necesita ser construido. El código de una clase se estructura así:

```
class Clase{
        //Atributos
        tipoDato nombre;
        //Métodos
        tipoRetorno nombreMetodo([parámetros]);
}
```

[4] Java 2 Curso de Programación; Fco. Javier Cevallos; Ed. Alfaomega Ra-Ma; México; 2000.

Hay dos cosas que nos interesan al crear un objeto:
- Inspeccionar sus variables (atributos y estados) y
- Llamar sus métodos (para utilizarlos en tiempo de ejecución)

Ciclo de vida de un objeto:
Declaración, Instancia, Inicialización y alternativamente Destrucción.

Equivalencia de términos entre Lenguaje Estructurado y Orientado a Objetos.

Procedimiento, función → Método
Datos → Variables de Clase, o Variables de Instancia
Llamadas a Procedimientos → Mensajes
TDA (Tipo de dato abstracto) → Clase
(Sin Equivalencia) → Herencia

Características de los lenguajes Orientados a Objetos.

		Small talk	Object Pascal	C++	CLOS	ADA	Eiffel	Java
Abstracción	Variables de instancia	Si	Si	Si	Si	Si	Si	Si
	Métodos de instancia	Si	Si	Si	Si	Si	Si	Si
	Variables de clase	Si	No	Si	Si	No	No	Si
	Métodos de clase	Si	No	Si	Si	No	No	Si
Encapsulamiento	De variables	Privado	Pub.	Pub,priv, protegido	Rw y accesos publico	Pub,priv	Priv	Pub,priv, prot,amigable
	De métodos	Público	Pub.	Pub,priv, protegido		Pub,priv	Pub,priv	Pub,priv. prot,amigable
Modularidad	Tipos de módulos	Ninguno	Unit	Archivo	Paquete	Paquete	Unit	Paquete
Jerarquía	Herencia	Simple	Simple	Múltiple	Múltiple	No	Múltiple	Simple

Tipos	Verificación fuerte	No	No	Si	Opcional múltiple	Si	Si	Híbrido
	Polimorfismo	Simple	Simple	Simple		No	Si	3 tipos
Concurrencia	Multitarea	No	No	Indirecta (mediant e clases)	Si	Si	No	Si
Persistencia (post. Mortem)	Objetos persistentes	no	no	No	no	no	no	si

Compiladores.

Para traducir un programa escrito en un lenguaje de alto nivel (programa fuente) a lenguaje máquina se utiliza un programa llamado compilador. Este programa tomará como datos nuestro programa escrito en el lenguaje de alto nivel y dará como resultado el mismo programa pero escrito en lenguaje máquina, programa que ya puede ejecutar directa o indirectamente la computadora.

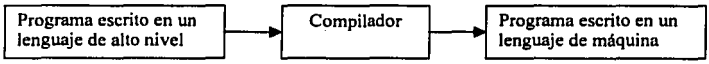


Fig. 5 Conversión de Lenguaje de Alto nivel a lenguaje de Maquina a través de un compilador.

Un programa escrito en el lenguaje C necesita del compilador C para poder ser traducido. Posteriormente el programa traducido podrá ser ejecutado directamente por el ordenador. En cambio, para traducir un programa escrito en el lenguaje Java necesita del compilador Java; en este caso, el lenguaje máquina no corresponde al de la computadora sino al de la máquina ficticia, denominada máquina virtual de Java, que será puesta en marcha por la computadora para ejecutar el programa.

¿Qué es una Máquina Virtual? Una máquina que no existe físicamente sino que es simulada en una computadora por un programa. ¿Por qué utilizar una máquina virtual? Porque, por tratarse de un programa, es muy fácil instalarla en cualquier computadora, basta con copiar ese programa en su disco duro, por ejemplo. Y ¿qué ventajas reporta? Pues, en el caso de Java, que es un programa escrito en este lenguaje y compilado, puede ser ejecutado en cualquier computadora del mundo que

tenga instalada esa máquina virtual. Esta solución hace posible que cualquier computadora pueda ejecutar un programa escrito en Java independientemente de la plataforma que utilice, lo que se conoce como transportabilidad de programas.

Interpretes.

A diferencia de un compilador, un intérprete no genera un programa escrito en lenguaje máquina a partir del programa fuente, sino que efectúa la traducción y ejecución simultáneamente para cada una de las sentencias del programa. Durante la ejecución de cada una de las sentencias del programa, ocurre simultáneamente la traducción.

A diferencia de un compilador, un intérprete verifica cada línea del programa cuando se escribe, lo que facilita la puesta a punto del programa. En cambio la ejecución resulta más lenta ya que acarrea una traducción simultánea.

¿Qué es Java?

Java es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems, en donde describen al lenguaje de la siguiente manera:

Es simple, orientado a objetos, distribuido (generalmente se usa en ambientes de red), interpretado, sólido (ante las adversidades se manejan excepciones), seguro (existen objetos que verifican la seguridad, -security manager, -class loader), de arquitectura neutral (corre en cualquier ambiente), portable (el bytecode se puede llevar donde sea y correrá), de alto desempeño (las tareas complejas se pueden delegar entre varios objetos), multiejecución y dinámico (los objetos existen sólo cuando se necesitan y viven el tiempo necesario).

Java es un lenguaje de programación de alto nivel con el que se puede escribir tanto programas convencionales como para Internet.

Java incluye dos elementos: un compilador y un intérprete. El compilador produce un código de bytes que se almacena en un fichero para ser ejecutado por el intérprete Java denominado máquina virtual de Java.

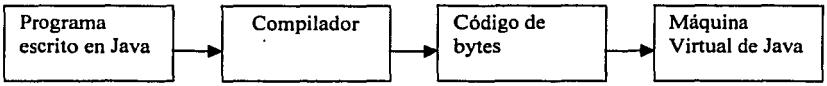


Fig. 6 Conversión de Lenguaje de Alto nivel a lenguaje de Maquina en Java.

Los codigos de bytes de Java son un conjunto de instrucciones correspondientes a un lenguaje máquina que no es específico de ningún procesador, sino de la máquina virtual de Java.

La idea de los creadores de Java era diseñar un compilador que produjera un lenguaje que pudiera ser interpretado a velocidades, si no iguales, sí cercanas a las de los programas nativos (programas en código máquina propio de cada computadora), logro conseguido mediante la máquina virtual de Java.

Un compilador JIT (Just In Time – Compilación al Instante) interacciona con la máquina virtual para convertir el código de bytes en código de máquina nativo. Como consecuencia, se mejora la velocidad durante la ejecución.

Gracias a Java ya es posible, con el navegador adecuado, transferir al usuario pequeñas aplicaciones, los llamados Applets de Java. Estos programas forman parte integrante de la página web y el usuario puede ejecutarlos por medio del navegador correspondiente.

Un **Applet** es un programa escrito en Java que se ejecuta en el contexto de una página web en cualquier computadora, independientemente de su sistema operativo y de la arquitectura de su procesador.

Además de ser transportable y orientado a objetos, **Java** es un lenguaje fácil de aprender. Tiene un tamaño pequeño que favorece el desarrollo y reduce las posibilidades de cometer errores; a la vez es potente y flexible.[5]

¿Cómo surgió Java?

Java tiene sus inicios en 1990 cuando Sun Microsystems pretende introducir un sistema para controlar electrodomésticos, calculadoras, relojes, e incluso PDAs o

[5] Java 2 Curso de Programación; Fco. Javier Cevallos; Ed. Alfaomega Ra-Ma; México; 2000.

Asistentes Personales (pequeñas computadoras) que, además, permitieran la conexión a redes.

Para ello James Gosling, líder del grupo de desarrollo, trató de crear un tipo de hardware polivalente, creando a su vez el lenguaje OAK (roble), al depender el lenguaje de un circuito implicaba que se tenía que modificar el lenguaje. Las modificaciones resultaban complicadas debido a la modificación del circuito y el proyecto resultó demasiado costoso. Se decidió recopilar de todos los lenguajes lo mejor de ellos, esto lo hacían en un café llamado Java Café, de ahí el nombre del lenguaje y el logotipo del mismo.

"El nombre "Java" surgió en una de las sesiones de "brainstorming" celebradas por el equipo de desarrollo del lenguaje. Buscaban un nombre que evocara la esencia de la tecnología (viveza, animación, rapidez, interactividad ...). Java fue elegido de entre muchísimas propuestas. No es un acrónimo, sino únicamente algo humeante, caliente y que a muchos programadores les gusta beber en grandes cantidades: una taza de café (Java en argot Inglés americano). De esta forma, Sun lanzó las primeras versiones de Java a principios de 1995.
Casualmente, la pronunciación en inglés de este término es "yava", que puede entenderse fuera de contexto como ¡ya va!."[6]

Las dos características para crear Java fueron, la independencia de ejecución y la sencillez; al crearse Java, este lenguaje fue introducido en los aparatos anteriores, y otros más como tostadores, televisores, refrigeradores, etc. Con esto surgió el primer edificio inteligente, llamado First Person. Al tener una independencia de arquitectura se decidió introducir el lenguaje en software, y se hizo en un navegador Web, el cual funcionó, y fue llamado HotJava.

Java aunque es un lenguaje sencillo, recopila todas las funciones de los lenguajes existentes y puede hacer todo tipo de tareas. Java es un lenguaje tanto interpretado como compilado. Surge formalmente el 23 de Mayo de 1994 con la versión JDK 1.0

¿Cómo funciona Java?

Un lenguaje de programación interpretado debe verificar instrucción por instrucción, después genera un código de máquina y después se ejecuta. El lenguaje compilado

[6] *El lenguaje de Programación Java™*

utiliza el procesador de la plataforma y crea código ejecutable justo para esa plataforma.

Java primero se compila pero su resultado no está dirigido al procesador sino a una máquina virtual, después esta máquina interpreta el código; cada arquitectura tiene una máquina virtual, de ahí la independencia de Java.

La sencillez permite que Java tenga una curva de aprendizaje rápida, no tiene conceptos inútiles o poco usados, ni manera alguna de dañar el ambiente de ejecución, es decir no tiene apuntadores, no hay goto (ir hacia), structs (estructuras), ni malloc (asignación y desasignación de memoria). El entorno de ejecución es reducido (por eso se puede ver en un browser o navegador). La multitarea se hace en una línea. Se preocupa por la solución del problema, no por la sintaxis del lenguaje.

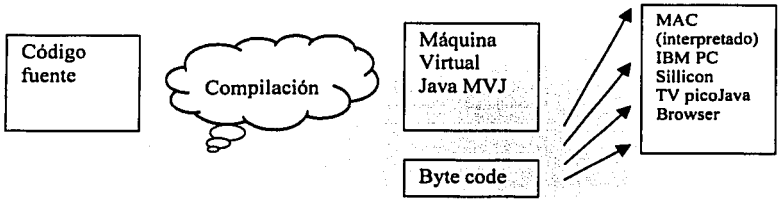


Fig. 7 El compilador entrega un código de máquina, el cual es interpretado por el elemento final que lo necesita.

Algunas de las plataformas que soportan Java son: Solaris, Windows NT, Windows 95-98, Windows CE 2.0, Linux, Macintosh, etc.

Requisitos para correr Java: Tener instalado JDK o tener un WebBrowser que cuente con el ambiente de ejecución (Netscape 6.0, HotJava, Internet Explorer 5.0), actualmente ningún browser soporta JDK 1.2 de manera nativa, se tienen que utilizar otros programas "plugins" para poder soportarlo.

El lenguaje JAVA (extensión de C++) tiene como filosofía aprovechar el software existente. Facilitar la adaptación del mismo a otros usos diferentes a los originales sin necesidad de modificar el código ya existente.

Java permite hacer cosas excitantes con las páginas web que antes no eran posibles. De manera que en este momento la gran interactividad que proporciona Java marca la diferencia en las páginas web. Tal es así como una aplicación en donde el usuario pueda hacer transacciones y estas se actualicen en tiempo real; u ofrecer pequeñas aplicaciones como hojas de cálculo o calculadoras para los visitantes de la web; o mostrar figuras 3D, tales como moléculas o dinosaurios que pueden ser rotados con un click de mouse.

Java también aumenta el contenido multimedia de un sitio, ofreciendo animaciones fluidas, gráficos mejorados, sonido y vídeo, fuera de lo necesario para enganchar aplicaciones de ayuda dentro de los browsers.

Otra de las diferencias de Java con los demás lenguajes es que cuando se escribe en la mayor parte de los lenguajes de programación, es necesario decidir el procesador y sistema operativo en los que se va a ejecutar el programa determinado. Entonces se incluyen llamadas especificas de función a una biblioteca asociada al sistema operativo de aquella plataforma destino.

Cuando se está preparado para probar el programa, se envía el código fuente a un compilador que lo transforma en un conjunto de instrucciones propias del procesador que utiliza la computadora destino.

Cuando se escriben programas en Java no se necesita pensar en llamadas a bibliotecas de sistema operativo. Java contiene sus propias bibliotecas, llamadas paquetes (packages), que son independientes de la plataforma.

De la misma forma, no es necesario preocuparse de que el producto terminado se ejecutará en tal o cual procesador. El compilador de Java no genera instrucciones nativas. En su lugar escribe código de byte para una máquina que no existe realmente, la Java Virtual Machine (Máquina Virtual de Java), o JVM.

Sun ha implantado una versión de software de la JVM para la mayor parte de las plataformas normales. Cuando se carga el archivo de código de byte (llamado class) en la máquina destino, se ejecuta en la JVM de aquella. La JVM lee el archivo de la clase y realiza el trabajo especificado por el Java original.

Los programas de Java autosuficientes son conocidos como aplicaciones, mientras que los que se ejecutan con la ayuda de otro programa (generalmente un navegador web), son conocidos como applets.

En la mayoría de los lenguajes el "Producto Final" es un archivo ejecutable de instrucciones binarias locales que ejecuta el programa pulsando doblemente el ratón sobre el icono de la aplicación. Java es algo diferente. Como los archivos de clase contienen código de byte para la JVM, para ejecutar la aplicación es necesario lanzar una implantación de la JVM (con una sola línea de comando). El Java Development Kit (JDK) incluye un intérprete de Java, llamado simplemente java, que implanta la JVM.

Las primeras versiones del JDK soportaban una interfaz gráfica de usuario (GUI), conocida como Abstract Windowing Toolkit (AWT). En las versiones más recientes, Sun ha introducido el paquete Swing, que incluye y amplía el AWT. Swing contiene muchos más componentes que AWT, de modo que se pueden construir interfaces más sofisticadas. Y lo que es mas importante, Swing implanta el Lightweight User Interface Framework (Marco Ligero de Interfaz de Usuario), que incluye "aspecto y gusto personal" (LookAndFeel). Esta nueva característica significa que un usuario final que prefiera el aspecto de Motiv Interface de Sun lo puede tener, a pesar de que el programador puede preferir la interfaz de Java Básica.

Esta es solo una muestra de las opciones que ofrece Java sobre otros lenguajes de programación estructurados u orientados a objetos, además de la ventaja de ser fácil de comprender para el mismo programador.

La ejecución de las aplicaciones no tiene necesariamente que quedar limitada al navegador. Una vez cargados los programas Java, pueden ejecutarse con las herramientas apropiadas desde el disco duro. Estas herramientas son, generalmente, programas de línea de comandos que no dependen del entorno gráfico de un navegador. Con ello, Java ofrece al otro lado de la red la posibilidad de componer programas independientes de las plataformas para la línea de comandos (**standalone**).

Para ejecutar las aplicaciones de Java es necesario que el software las interprete.

Para crear aplicaciones ejecutables de programas Java se necesita un entorno de desarrollo que disponga por lo menos de un compilador. Con un programa Java se suministra al compilador el texto, a partir del cual se genera el código que luego se podrá ejecutar en un navegador.

¿Que se puede hacer con Java?.-

- Applets.- Programas gráficos que viven en un browser. No tienen un metodo main. Su comportamiento lo da el propio browser. (al cargar la página, éste aparece; y se puede minimizar, ocultar, etc.).
- Aplicaciones Stand-Alone. Necesitan un método main. Necesitan el intérprete de java para ejecutarse; por lo que puede decirse que se ejecutan por si solos.
- Manipuladores de Contenido. Plugs-ins. Se cargan y están latentes en un browser. Cuando se quiere abrir una aplicación el manipulador de contenido debe interpretarla y ejecutarla.
- Manipuladores de Protocolos. Son programas tipo "demonio", es decir que atienden peticiones en un esquema especifico.
- Llamadas a métodos nativos. Teniendo "rutinas, módulos y/u otros programas" en otros lenguajes, enlazarlos en un solo código Java.
- Servlets. Programas destinados a ejecutarse en un servidor. Formularios tales como los CGI's, pero mas poderosos por ser independientes de la plataforma.
- Dispositivos embebidos. Tarjetas inteligentes y circuitos.

¿Que hace diferente a Java de otros lenguajes?

Java:

- Es un lenguaje de programación diseñado para uso de los programadores profesionales (experimentados).
- Utiliza las técnicas orientadas a objetos.
- Admite el modelo Cliente/Servidor.
- Admite la concurrencia.
- Posee un fuerte modelo de seguridad.

Ventajas y desventajas de Java.

Ventajas.-
Multinivel, manera de aprendizaje rápida, es público.
Con Java no se paga por todo el software cuando no se va a usar todo.
Un servidor puede contener la aplicación y proporcionar este recurso a máquinas con escasos recursos.
Es independiente de la plataforma, los archivos .class se pueden transportar en Internet, basta con tener un ambiente de ejecución y un archivo de clases en el servidor, para correr los applets.

Desventajas.-
Se compila e interpreta (doble tiempo).
El tiempo de compilación es más lento que compilaciones en otros lenguajes (aunque la ventaja es que hace una verificación de datos fuerte, revisa por dentro a las estructuras y "prueba el código" evitando generar peticiones, castings u operaciones no permitidas). Checa la jerarquía de clases asegurando la integridad de cada uno.
Tecnología nueva que no está estandarizada. Es público.

Ambientes de desarrollo para Java.

Visual J++ Microsoft, Semantec Café, Jbuilder (Borland), Jdeveloper, Visual Age (IBM), Java Workshop (SUN), JESS.

Herramientas de desarrollo: el Java SDK

El Java SDK con sus herramientas constituye una vía de introducción gratuita al desarrollar programas con Java. Contiene los siguientes programas y componentes:

Javac	El compilador de Java. De un archivo ASCII con el programa fuente de Java se crea un archivo de paquetes de código que puede ejecutarse con el interprete de Java o como parte de una página web.
Java	Intérprete de Java que ejecuta un programa Java traducido ya en paquetes de código y presente en la computadora local.
Jdb	Depurador de errores inspirado en los depuradores dbx y gdb de Unix.
Visor de Aplicaciones	Carga una página HTML de la red, la muestra y ejecuta las aplicaciones Java que contiene.
Javadoc	Herramienta de documentación que, a partir del código fuente de Java,

	genera un archivo HTML con la documentación de las rutinas del código.
Javap	Decompilador de código de bytes que lee un archivo de paquetes de código de Java y muestra las informaciones contenidas sobre clases, métodos y objetos.
Javah	Crea archivos de encabezado, Archivos C, que hacen posible acceder a módulos de encabezado (módulos C), desde el código Java, por lo que se pierde la independencia de plataforma del correspondiente programa Java.

Java no es un lenguaje trivial. Se basa en gran parte en los lenguajes C y C++, lo que significa que está totalmente orientado a objetos. Los programadores de C++ encontraran muchos conceptos familiares aunque, por otro lado, Java no tiene reparos en eliminar algunas "vacas sagradas" del mundo C y C++.

Los **punteros**, por ejemplo, no existen en Java porque son los instrumentos con los que los programas mal dirigidos llevan a cabo los daños. En lugar del puntero se introduce un concepto "Array" (matriz o arreglo) revalorizado, al que se acostumbra el programador enseguida haciendo olvidar pronto la aritmética de punteros. A partir del estado actual de C++ aparecen dispositivos multitarea incluidos directamente en Java.

En el desarrollo de Internet, donde los paquetes de datos no se hacen esperar mucho, ya tiene sentido realizar varios procesos de comunicación al mismo tiempo, lo que no representa ningún problema con subprocesos (**"threads"**) múltiples.

API

La biblioteca de tiempos de ejecución se denomina API de Java, y abarca un amplio espectro de servicios: esto abarca desde funciones matemáticas y de cadenas básicas, como las que se conocen de otros lenguajes, hasta elementos visuales (cuadros y máscaras de dialogo), funciones para diseñar objetos gráficos o para reproducir sonidos, o funciones para acceder a otros servicios de Internet y comunicación entre aplicaciones a través de la misma.

De acuerdo con la filosofía del lenguaje orientado a objetos, la API de Java se desarrolla como una jerarquía de clases donde éstas ofrecen una parte diferente de la

funcionalidad total. Para iniciarse en el desarrollo de Java es preciso, en primer lugar, familiarizarse con el vocabulario, la sintaxis y los conceptos del lenguaje.

Seguridad.

Java protege las aplicaciones en una especie de red de seguridad de la que es muy difícil, si no imposible, escapar. Así, por ejemplo, Java no permite ningún acceso a los archivos de la máquina local, para lo que se bloquean los servicios correspondientes de la biblioteca de tiempos de ejecución.

Para evitar ser manipuladas, las aplicaciones contienen además sumas de verificación que garantizan la integridad de los paquetes de código. Java limita la posibilidad de accesos ilegales al no reconocer ningún puntero, con el que se podría acceder a cualquier lugar de la memoria. Y las matrices (arrays) que Java ofrece en lugar del puntero son supervisadas exactamente para que una aplicación no pueda escribir en la memoria a través del final de una "matriz".[7]

Herencia.

Es una relación del tipo "tipo de" entre elementos en común entre clases. Surgé de la vida cotidiana y la necesidad de clasificación, de donde se obtiene de algo muy general diferentes diversificaciones que lo particularizan cada una a su manera. Así un hombre es un tipo de primate que a su vez es un tipo de mamífero.

La composición es la creación de objetos a partir de objetos ya existentes, se dice que la relación es de tipo "parte de" así un motor es parte de un camión, pero el motor en sí ya tiene un funcionamiento propio.

Herencia en Java.

Todas las clases creadas en Java son herederas de la clase Object. Si una clase no tiene definida otra de la cual va a heredar, entonces es por default heredera directa de la clase Objects que tiene como métodos equals y clone que comparan dos objetos y crean objetos de la misma clase, respectivamente.

[7] "Java, El lenguaje de Programación de Internet"; Varios Autores; Editorial Marcombo; 1996; Barcelona, España.

Cuando una clase hereda de otra, esta va a poseer todos los atributos y métodos definidos en la superclase. Esto es válido para todos los niveles, las clases herederas heredan todos los métodos y atributos de todos los niveles de clases previos a ella.

Polimorfismo.

Todo aquello que la clase necesite pero no debe mostrar, es decir, todo lo que sea privado se llamará implantación. Los métodos a través de los cuales se accede a lo privado se denominan interfaz. En la implantación existen la sobrecarga y la sobre escritura.

Un objeto que es creado a partir de una subclase será también un objeto del tipo de la superclase. A un objeto de tipo cubo se le puede asignar un objeto del tipo caja, ya que una caja es un tipo de cubo.

Cuando es necesario determinar de que subclase proviene un objeto es necesario utilizar la palabra **instanceof** que nos indica a partir de que clase se instanció o creo una instancia del objeto. Por ejemplo, si tenemos un arreglo de figuras geométricas y se desea almacenar en éste cualquier objeto que herede a partir de esta clase; podemos incluir en el arreglo al triángulo, al cuadrado, etc. ya que heredan las características de ser figuras geométricas.

Servlets.

La API (Interfaz de Programación de Aplicaciones) Java Servlet es una API que sirve para programar desde el lado del servidor. Se puede emplear para escribir programas de CGI (Interfaz Común de Puerta de Enlace) como servlets de Java, y lo admiten la mayoría de servidores web conocidos.

Los servlets son el equivalente a los applets, pero en el lado del servidor. Se escriben en la API Servlet y están instalados en el directorio servlets de un servidor web. Y pueden ser invocados en el URL siguiente:
http://localhost:8080/servlet/ServletName.class[?argumentos]

Las ventajas que tienen los servlets sobre los programas CGI son que el servidor, al invocarlos, utiliza muy pocos recursos, en que se les proporciona acceso directo a los recursos del servidor y que están escritos en Java.

Los servlets son pequeños programas personalizados de servidor que ejecuten tareas muy especializadas.

Los servlets son extensiones del servidor que están escritas en Java y que vienen asociadas a URL determinados. Cuando el URL de un servlet recibe una solicitud de un navegador web, el servidor invoca al servlet para procesar la solicitud. El servidor dota al servlet con toda la información necesaria para procesar la solicitud. También proporciona un mecanismo para que el servlet envíe información de respuesta al navegar web.

Servlets son precompilados por el programador, y puestos en el servidor de páginas (web server) que los carga, compila y ejecuta al momento que un usuario remoto lo solicita, generalmente a través de una página html.

Es de tener muy en cuenta que cuando el servidor de páginas carga y ejecuta un servlet, lo va mantener activo en memoria en su propio proceso por toda la vida del servidor de páginas (es decir mientras no se reinicie el servidor).

Es por esta situación que los servlets presentan ventajas y desventajas, la ventaja mas aparente es que si otros usuarios piden el mismo servlet, el servidor de páginas le responderá muy rápidamente, la desventaja aparente, es que si el servlet es muy especializado o de poco uso, estará vivo en memoria y consumiendo los recursos del computador, es por esta razón que se deberá seleccionar muy cuidadosamente cuales programas o aplicaciones se deberán construir con esta tecnología de servlets.

Otra elemento muy importante a tomar en cuenta en este tema, es que servlets y jsp (java server pages) interactúan con los usuarios a través de páginas html, es decir no son programas visuales como JFRAME y APPLETS que interactúan por medio de ventanas o formas visuales.

Nuestra interfase con los usuarios es a través de páginas html, es decir una página html proporcionará los datos al servlet y también lo activa, el servlet recibe los datos, los procesa y le responde al usuario con otra página html.

Los servlets ofrecen básicamente dos métodos: el método init() y el método service() (con las variantes doGet(), doPost(), etc.)

El método init() se ejecuta la primera vez que se llama al servlet. Es parecido al init() de los applets: Se ejecuta en la carga y el servlet queda persistente. Del mismo modo destroy() se ejecuta al descargarse.
Los servlets gozan de cierta fama de rápidos respecto a los CGI, incluso estando escritos en C. El truco consiste en obtener el objeto Connection en el init() y luego reutilizarlo para el resto de impactos. La obtención de un objeto Connection no es más que establecer un login en la base de datos. El proceso de login puede tardar unos segundos. Los servlets, al contrario que los CGI ofrecen la posibilidad de persistencia en memoria entre un impacto y otro, es decir mantener objetos ya instanciados permanentemente. Los CGI parten de cero cada vez y la persistencia se obtiene generalmente en base a ficheros.

El método init() se ejecuta la primera vez que llega un impacto. Es similar al init de un Applet. Una aplicación típica de init() es establecer sesión con una base de datos, así como establecer parámetros iniciales.
En los sucesivos impactos la conexión se mantiene abierta por lo que la ejecución global de un servlet puede ser más rápida que un CGI compilado en C.

JSP
Este sistema es una muestra de interrelacionar servlets con JSP.

Una variante de Java basada en servlets son las JSP (Java Server Pages)

Una página JSP es un documento text-based que describe como al procesar una solicitud se crea una respuesta. La descripción entremezcla datos plantilla con algunas acciones dinámicas y (fuerzas o influencias) en la plataforma de Java .

Java Server Page, es otra de las nuevas tecnologías para tratar de hacer más eficiente el modelo cliente-servidor y sobre todo la construcción de sistemas de comercio electrónico.

En este modelo, una página html también incluye código en java, es el servidor de páginas quien al estar mandando la página a la computadora remota, la compila y la convierte en un servlet.

Esta tecnología combina en una sola aplicación, tanto código html como código java.

El proceso de crear un jsp, es sencillo. Se crea un archivo normal con cualquier manejador de texto combinando código html y código java, se graba con extensión jsp, se manda al servidor y listo.

Cuando el usuario requiere un jsp, el servidor lo carga, lo compila, lo convierte a servlet y manda la página resultante al usuario remoto.

Las JavaServer Pages (JSP) están basadas en la tecnología de servlets. Cuando se combina con el uso de componentes JavaBeans, JSP proporcionar una capacidad que es al menos tan poderosa como los Servlets, posiblemente más que un servlet en crudo, y potencialmente mucho más fácil de usar.

La creación y compilación del Servlet es automática

Cada página JSP es compilada automáticamente en un servlet por el motor JSP. (Sólo podemos usar JSP en servidores que sean compatibles con JSP).

La creación y compilación automática del servlet ocurre la primera vez que se accede a la página. Dependiendo del comportamiento del servidor web, el servlet será grabado durante algún periodo de tiempo para utilizarlo una y otra vez sin necesidad de recrearlo y recompilarlo.

Por eso, la primera vez que se accede a la página, podría haber una pausa mientras que el servidor web crea y compila el servlet. Después de esto, los accesos a la página serán muchos más rápidos

¿Cómo se usan las página JSP?

Hay muchas formas diferentes de combinar JAP, Beans y Servlets. Como se mencionó antes, el uso juicioso de JSP hace posible combinar las mejores capacidades del HTML con los componentes de software reutilizables para crear aplicaciones del lado del servidor.

Esto hace muy práctico separar la lógica del negocio de la representación de los datos. Con esto, los programadores especializados en escribir JavaBeans que implementen la lógica del negocio, y los diseñadores de páginas especializados en

HTML pueden embeber llamadas a esos Beans desde el HTML sin necesidad de convertirse en expertos programadores Java.

JDBC

Sun ofrece un paquete, java.sql, que permite que los programas en este lenguaje accedan a los sistemas de administración de bases de datos relacionales (RDBMS). Dicho paquete de conectividad de bases de datos de Java (Java Database Connectivity Package o JDBC) permite la conexión e interacción con una base de datos relacional mediante el popular lenguaje SQL.

En Conclusión

Java ofrece un sin fin de posibilidades de uso y desarrollo de aplicaciones. Con base a lo más sencillo de la programación Orientada a Objetos

"El atractivo de Java radica en su simplicidad, familiaridad y cuidadosa selección de características que incluye y excluye. Java comparte su espíritu con C más que con ningún otro lenguaje de programación. Es un lenguaje de programación diseñado por programadores para programadores."[8]

Por su sencillez, facilidad de aprendizaje y para promover su uso como lenguaje de programación; Java es utilizado para realizar este sistema.

[8] Java 1.2 Al Descubierto, Jaime Jaworski, Prentice Hall, Madrid 1998.

CAPITULO II

"Análisis y Diseño del Sistema de Control para el ProSE-A."

Existen muchos métodos para realizar el análisis y diseño de sistemas Orientado a Objetos; uno de ellos, la notación UML es el más usado actualmente. El UML combina las mejores características de los métodos Orientados a Objetos y ha venido a simplificar, en mucho, éstos mismos métodos.

UML

El UML (Unified Modeling Language, Lenguaje de Modelado Unificado) se define como un "lenguaje que permite especificar, visualizar y construir los artefactos de los sistemas de software..."9. Es un sistema notacional destinado a los sistemas de modelado que utilizan conceptos orientados a objetos.

El UML es un estándar de la industria para construir modelos orientados a objetos. Nació en 1994 por iniciativa de Grady Booch y Jim Rumbaugh para combinar sus dos famosos métodos: el de Booch y el OMT (Object Modeling Technique, Técnica de Modelado de Objetos). Más tarde se les unió Ivar Jacobson, creador del método OOSE (Object-Oriented Software Engineering, Ingeniería de Software Orientada a Objetos) y de los casos de uso (Use Cases), una técnica muy eficaz para la determinación de las necesidades. En respuesta a una petición del OMG (Object Management Group, asociación para fijar los estándares de la industria) para definir un lenguaje y una notación estándar del lenguaje de construcción de modelos.

El Lenguaje de Modelado Unificado es el sucesor de los métodos de análisis y diseño orientado a objetos (OOA&D, Object Oriented Analisis and Design) que aparecieron al final de la década de los 80's y a principios de los 90's.

Sirve para: Modelar, Especificar, Construir, Documentar.

El UML provee una notación rica para la visualización de modelos. Está incluye los siguientes:

- Diagramas de casos de uso; para ilustrar las interacciones del usuario con el sistema.
- Diagramas de clases; para ilustrar la estructura lógica.
- Diagramas de secuencia y colaboración; para ilustrar el comportamiento.

[9] "*The UML specification documents*", Booch, G., I. y Rumbaugh, J. 1997. Rational Software Corp.

- Diagramas de estado; ilustran de igual forma el comportamiento.
- Diagramas de despliegue; para mostrar un mapa de la configuración del software y el hardware.
- Diagramas de componentes; para ilustrar la estructura física del software.

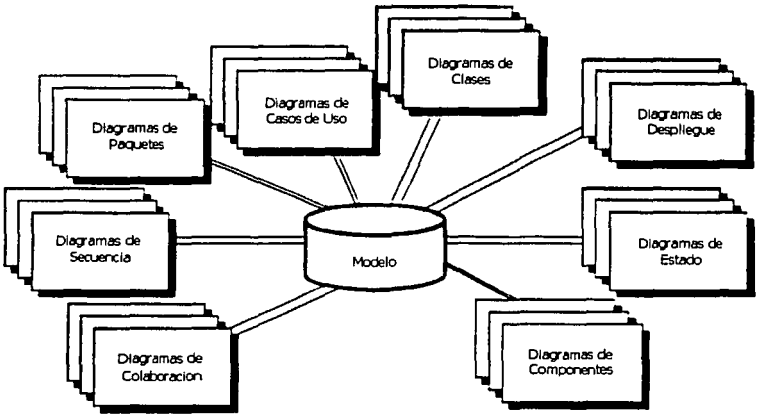
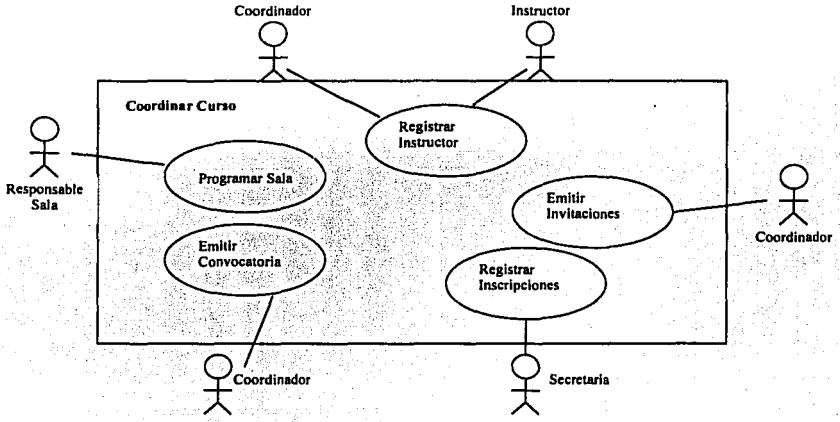


Fig.1 Modelando un sistema desde diferentes perspectivas.

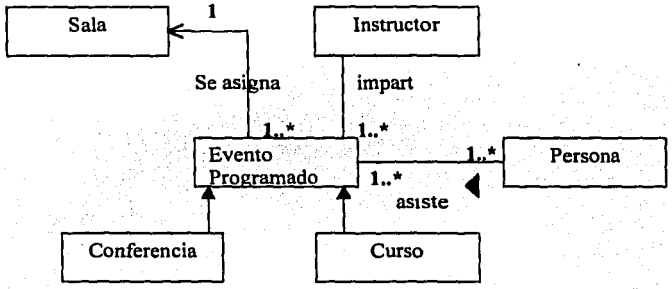
Diagrama de Casos de Uso



Objetivos:

Ilustrar los Roles tomados en una situación por los Usuarios (Actores).
Modelar los procesos que un Actor requiere de un sistema desde su propia perspectiva.

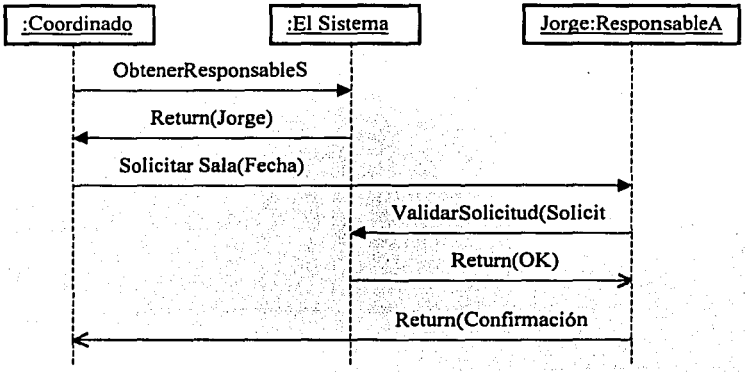
Diagrama de Clases



Objetivos:

Modelar las Clases participantes
Identificar sus Atributos
Identificar sus Operaciones
Mostrar Relaciones Estáticas entre Clases

Diagrama de Secuencia

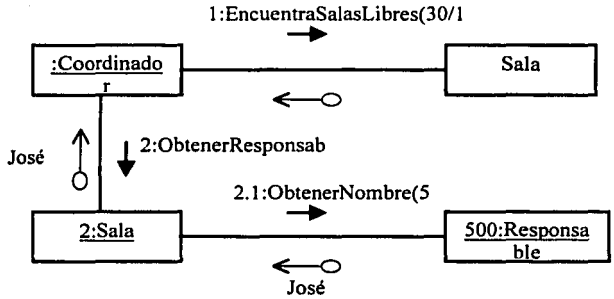


Objetivos:

Mostrar Eventos entre Objetos en secuencia
Mostrar Valores de retorno

Notas:
Es un Diagrama de Interacción "secuencial".

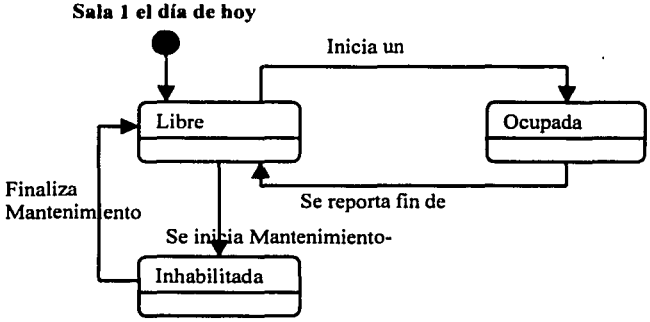
Diagrama de Colaboración



Objetivos:

Mostrar un grupo de Objetos y Ligas describiendo un Escenario
Mostrar Eventos (mensajes) pasando entre los Objetos. Puede incluir retorno de resultados.
Mostrar el orden relativo de los Eventos. Los llamados anidados son mostrados por numeración anidada.

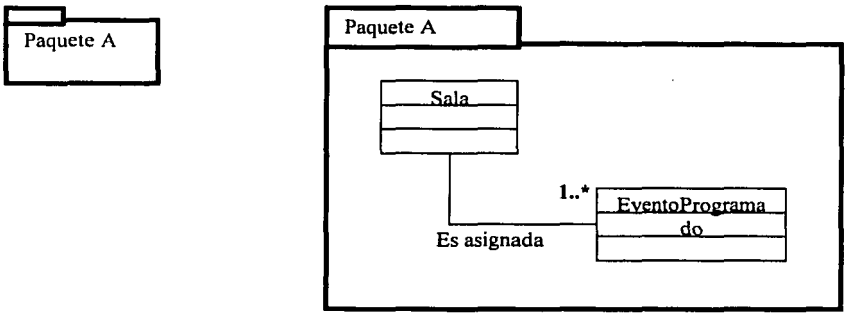
Diagrama de Transición de Estados



Objetivos:

Mostrar Estados, Eventos y Transiciones
Describir Ciclos de Vida de Objetos
Identificar Operaciones relacionadas con Estados y Eventos

Diagrama de Paquetes



Es un grupo de elementos modelo (Clases, otros Paquetes, etc.)

Objetivos:

Agrupar diversos tipos de cosas: Módulos, Modelos, Subsistemas, Grupos Físicos, etc.

En base a esta **Notación UML** fue que se llevo a cabo el Análisis y Diseño del sistema de Control para el Programa de seguimiento a Egresados de Aragón (ProSE-A); tomando en cuenta los siguientes requerimientos.

Definición de los Requerimientos del Sistema .

Panorama General.

La Universidad Nacional Autónoma de México considera aún como alumnos a todos aquellos que terminaron sus créditos académicos pero se encuentran realizando su servicio social o están en proceso de tesis, puesto que no se han titulado y no se pueden considerar egresados en toda la extensión de la palabra.

Llevar un control eficiente de ese tipo de alumnos y de los egresados, principalmente de la Escuela Nacional de Estudios Profesionales Campus Aragón, podrá mostrar las oportunidades que los alumnos de esta institución tienen en el campo laboral, y servirá de lazo de unión entre ellos y su Alma Mater.

Planteamiento del problema.

Con el fin de llevar a cabo éste control se realizó un sistema en el cual, vía una página web los alumnos y egresados pueden acceder a sus datos personales con el fin de mantenerlos siempre actualizados, así como tener la oportunidad de no perder el contacto con sus compañeros de generación y/o carrera y con la escuela misma para así mantenerse informados de eventos realizados en la universidad que pudieran ser de su interés.

Los nuevos lenguajes de programación permiten que los navegadores puedan cargar programas de ayuda capaces de manipular todo tipo de información; eliminando las barreras del tiempo y la distancia y permitiendo a la gente compartir información y trabajar en colaboración.

Como se había mencionado anteriormente Java permite hacer cosas excitantes con las páginas web. De manera que permite una gran interactividad, lo que marca la diferencia en las páginas web actuales. Un ejemplo de ello seria una aplicación en donde el usuario pueda hacer transacciones y estas se actualicen en tiempo real.

Además de la gran ventaja que tiene Java sobre otros lenguajes: la independencia de la plataforma. Lo que le permite abarcar la mayoría de la máquinas de hoy en día.

La máxima de Java es: "Escriba una vez, ejecútelo en cualquier parte". Esta declaración no esta hecha en vano: Java se ejecuta en la mayoría de las plataformas de los principales sistemas operativos.[10]

Considerando todas estas ventajas el lenguaje Java es utilizado para la realización del Sistema de Control para el Programa de Seguimiento de Egresados de la Escuela Nacional de Estudios Profesionales Campus Aragón (ProSE-A).

Uso.

El Sistema de Control para el Programa de Seguimiento de Egresados de la Escuela Nacional de Estudios Profesionales Campus Aragón (ProSE-A) es una herramienta de uso general, en la que se conjugan el uso de lenguajes de programación orientados a objetos, el diseño de bases de datos relacionales y la programación en web.

Metas.

El objetivo principal del presente trabajo es elaborar un sistema confiable y seguro para el Control del Programa de Seguimiento de Egresados de la Escuela Nacional de Estudios Profesionales Campus Aragón (ProSE-A). El análisis del Sistema fue realizado con ayuda de una de las herramientas Case existentes en el mercado actual llamada Rational Rose; la cual esta orientada a la notación **UML** utilizada en la Programación Orientada a Objetos.

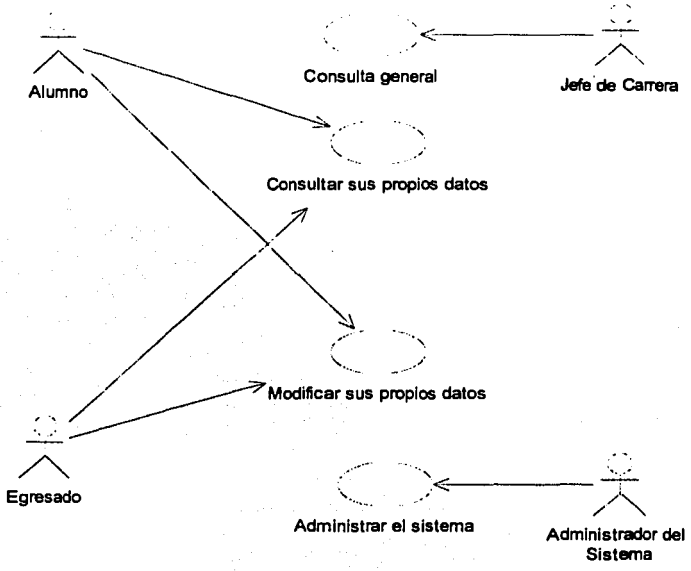
Funciones.

Las funciones principales del Sistema de Control para el Programa de Seguimiento de Egresados de la Escuela Nacional de Estudios Profesionales Campus Aragón (ProSE-A) son:

- El Alumno o Egresado puede Consultar sus propios datos,
- El Alumno o Egresado puede Actualizar sus propios datos,
- La Consulta General por parte de los Jefes de Carrera y
- La Administración el Sistema.

[10] Java 1.2 Al Descubierto, Jaime Jaworski, Prentice Hall, Madrid 1998.

Modelo de Casos de Uso.



Descripción de Actores.

Alumno

Es la persona que aun se encuentra inscrita en algunas materias

Egresado

Es la persona que ya se ha titulado.

Jefe de Carrera

Es la persona encargada de una de las carreras que se imparten en el plantel.

Administrador del Sistema

Es la persona que esta encargada de administrar y mantener la información de los Egresados de la UNAM Aragón.

Descripción de Paquetes de Casos de Uso.

En este sistema se consideran los siguientes paquetes:

- Package Alumno
- Package Egresado
- Package Administrador

Descripción de Casos de Uso.

- ***Consultar sus propios datos***
  Este caso de uso es iniciado por el Alumno o Egresado. Provee la capacidad de consultar su propia información una vez que el Administrador lo haya registrado.

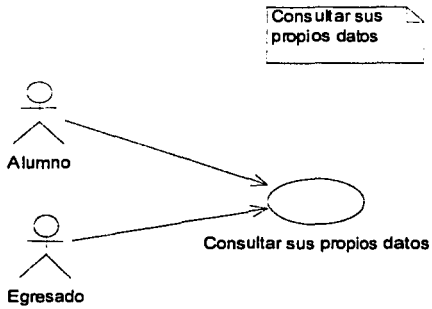


Fig 1. Diagrama de Caso de Uso

- ***Modificar sus propios datos***
  Este caso de uso es iniciado por el Alumno o Egresado. Provee la capacidad de modificar su propia información una vez que el Administrador lo haya registrado.

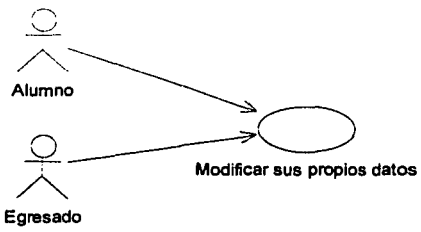


Fig 2. Diagrama de Caso de Uso

- ***Consulta General***
  Este caso de uso es iniciado por el Administrador o Jefe de Carrera. Provee la capacidad de consultar toda la información una vez que el Administrador la haya registrado.

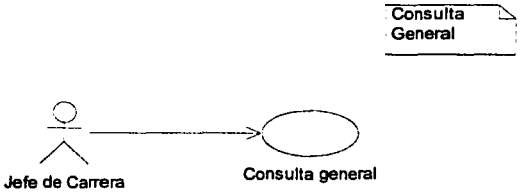


Fig 3. Diagrama de Caso de Uso

- ***Administrar el Sistema***
  Este caso de Uso es iniciado por el Administrador del Sistema. Provee las capacidades de consultar, dar de alta, dar de baja y modificar la información de los Alumnos y Egresados.

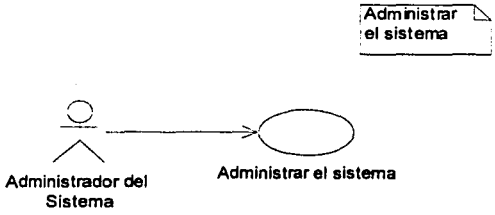


Fig 4. Diagrama de Caso de Uso

Casos de Uso.

Consultar sus propios datos.

Descripción:

Provee la capacidad de consultar la información propia del Alumno o Egresado.

Flujo de eventos:

- Precondiciones:
  - El Administrador del sistema deberá cargar la información.

Flujo Normal:

1. El alumno solicita la entrada al sistema
2. El sistema verifica si puede tener acceso y sus privilegios
3. El sistema da acceso y muestra los datos que se tienen del alumno.

Flujo de Excepciones:

E1: La solicitud de entrada al sistema no es valida. El alumno es notificado de esto. El alumno puede volver a introducir la información.

Diagrama de Colaboración

Descripción:

1. El alumno solicita la entrada al sistema
2. El sistema verifica si puede tener acceso y sus privilegios
3. El sistema da acceso y muestra los datos que se tienen del alumno.

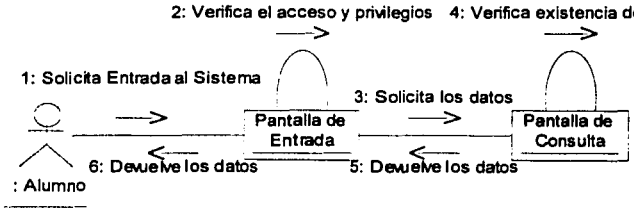


Fig 5. Diagrama de Colaboración

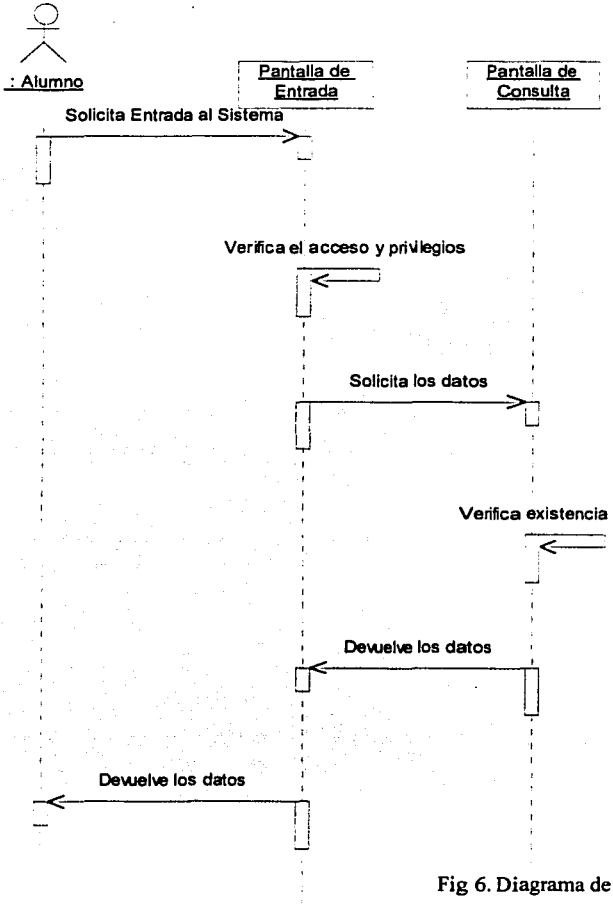


Fig 6. Diagrama de Secuencia

Modificar sus propios datos.

Descripción:

Provee la capacidad de modificar la información propia del Alumno o Egresado.

Flujo de eventos:

- Precondiciones:
  - El Administrador del sistema deberá cargar la información.

Flujo Normal:

1. El alumno solicita la entrada al sistema
2. El sistema verifica si puede tener acceso y sus privilegios
3. El sistema da acceso y muestra los datos que se tienen del alumno.
4. El alumno modifica los datos que le esta permitido modificar.
5. El sistema guarda la información.

Flujo de Excepciones:

E1: La solicitud de entrada al sistema no es valida. El alumno es notificado de esto. El alumno puede volver a introducir la información.

E2: Alguno de los datos no corresponde al formato del campo. El alumno es notificado de esto. El alumno puede volver a introducir la información.

Diagrama de Colaboración

Descripción:

1. El alumno solicita la entrada al sistema
2. El sistema verifica si puede tener acceso y sus privilegios
3. El sistema da acceso y muestra los datos que se tienen del alumno.
4. El alumno modifica los datos.
5. El sistema guarda la información.

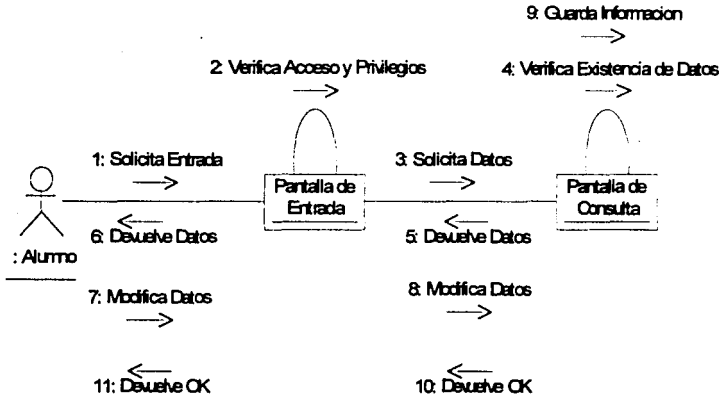


Fig 7. Diagrama de Colaboración

Karina Janette Olvera Alvarez
Ingeniería en Computación
Generación 95 - 99

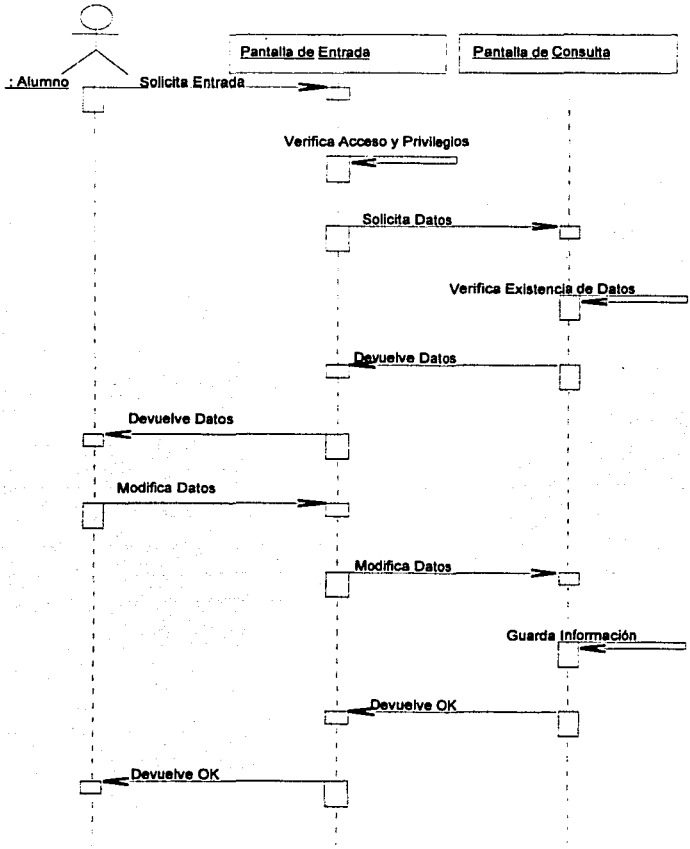


Fig 8. Diagrama de Secuencia

Karina Janette Olvera Alvarez
Ingeniería en Computación
Generación 95 - 99

Consulta General.

Descripción:

Provee la capacidad de consultar toda información del sistema.

Flujo de eventos:

- Precondiciones:
  - El Administrador del sistema deberá cargar la información.

Flujo Normal:

1. El administrador o jefe de carrera solicita la entrada al sistema
2. El sistema verifica si puede tener acceso y sus privilegios
3. El sistema da acceso y muestra los datos que se tienen en general de todos los alumnos y egresados; previamente filtrada.

Flujo de Excepciones:

E1: La solicitud de entrada al sistema no es válida. El administrador o jefe de carrera es notificado de esto. El administrador o jefe de carrera puede volver a introducir la información.

Diagrama de Colaboración

Descripción:

1. El administrador o jefe de carrera solicita la entrada al sistema
2. El sistema verifica si puede tener acceso y sus privilegios
3. El sistema da acceso y muestra los datos que se tienen en general de todos los alumnos y egresados; previamente filtrada.

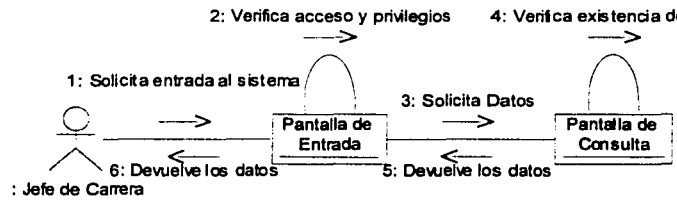


Fig 9. Diagrama de Colaboración

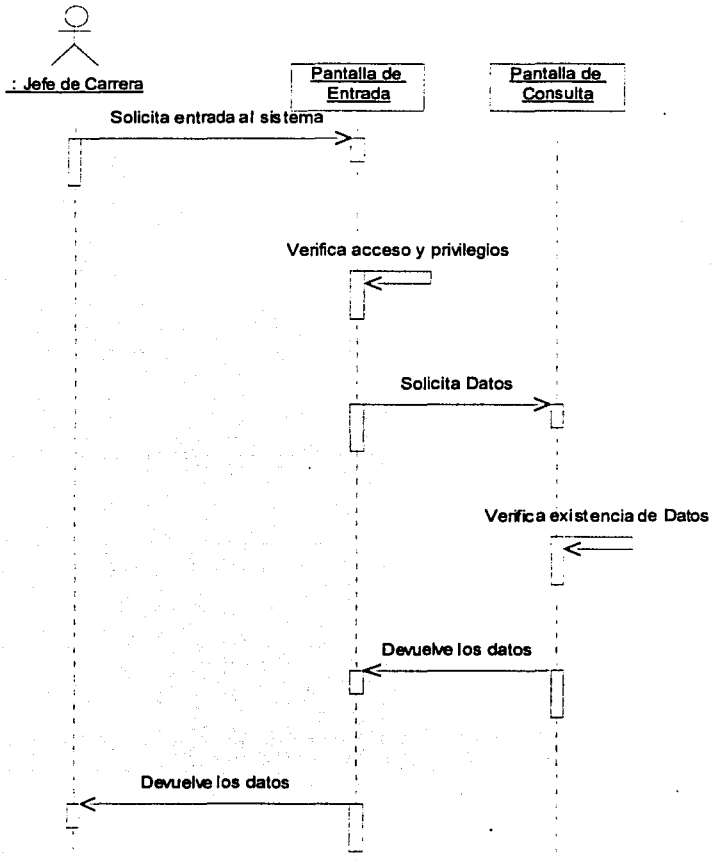


Fig 10. Diagrama de Secuencia

Administrar el Sistema.

Descripción:

Provee la capacidad de dar de alta, baja, modificar o consultar toda la información del sistema.

Flujo de eventos:

- Precondiciones:
  - Ninguna

Flujo Normal:

1. El administrador solicita la entrada al sistema
2. El sistema verifica si puede tener acceso y sus privilegios
3. El sistema da acceso y da la opción de realizar una consulta o una alta.
4. El administrador solicita una consulta.
5. El sistema muestra los datos solicitados y da la opción a modificar los datos.
6. El administrador modifica los datos.
7. El sistema guarda la información.
8. Si se elige una alta el sistema muestra los campos en blanco y permite introducir la información.
9. El sistema guarda la información.

Flujo de Excepciones:

E1: La solicitud de entrada al sistema no es válida. El alumno es notificado de esto. El alumno puede volver a introducir la información.

E2: Alguno de los datos no corresponde al formato del campo. El administrador es notificado de esto. El administrador puede volver a introducir la información.

Diagrama de Colaboración

Descripción:

1. El administrador solicita la entrada al sistema
2. El sistema verifica si puede tener acceso y sus privilegios
3. El sistema da acceso y da la opción de realizar una consulta o una alta.
4. El administrador solicita una consulta.
5. El sistema muestra los datos solicitados y da la opción a modificar los datos.
6. El administrador modifica los datos.
7. El sistema guarda la información.
8. Si se elige una alta el sistema muestra los campos en blanco y permite introducir la información.
9. El sistema guarda la información.

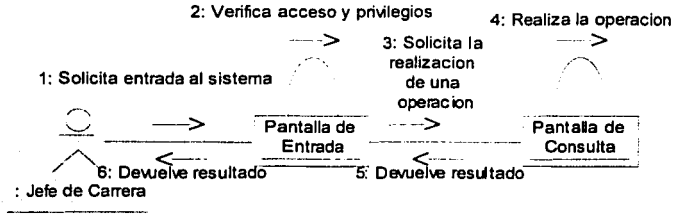


Fig 11. Diagrama de Colaboración

Kanna Janette Olvera Alvarez
Ingeniería en Computación
Generación 95 - 99

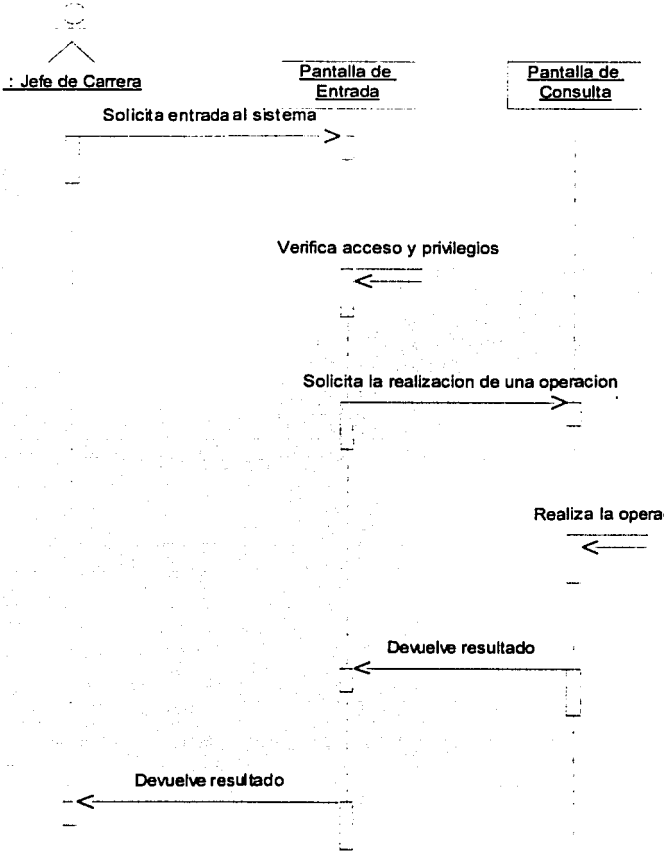


Fig 12. Diagrama de Secuencia

Así mismo; se cuenta con un diseño de base de datos, formado por las tablas que se encuentran en el siguiente diagrama. El cual paso por un proceso de Normalización en base a las Cinco Reglas o Niveles de Normalización que se conocen actualmente; las tres primeras que fueron perfiladas por el Dr. E.F.Codd en su escrito de 1972, "Further Normalization of the Data Base Relational Model" (Referente a la normalización de las Bases de Datos Relacionales). Y las otras reglas que han sido teorizadas por posteriores Matemáticos/Algebristas.

Primer nivel de Formalización/Normalización. (F/N)

- Eliminar los grupos repetitivos de la tablas individuales.
- Crear una tabla separada por cada grupo de datos relacionados.
- Identificar cada grupo de datos relacionados con una clave primaria.

Segundo nivel de F/N

- Crear tablas separadas para aquellos grupos de datos que se aplican a varios registros.
- Relacionar estas tablas mediante una clave externa.

Tercer nivel de F/N.

- Eliminar aquellos campos que no dependan de la clave.

Cuarto Nivel de F/N.

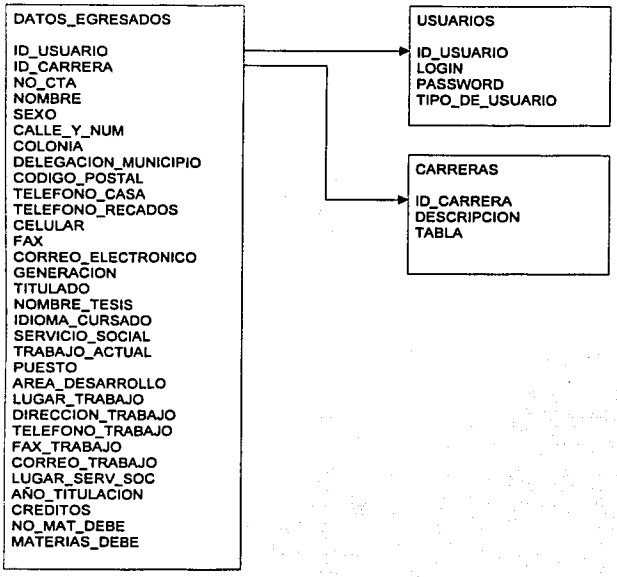
- Existen tres tipos de relaciones entre los datos: uno-a-uno, uno-con-varios y varios-con-varios
- En las relaciones varios-con-varios, entidades independientes no pueden ser almacenadas en la misma tabla.

Quinto Nivel de F/N.

- Existe otro nivel de normalización que se aplica a veces, pero es de hecho algo esotérico y en la mayoria de los casos no es necesario para obtener la mejor funcionalidad de nuestra estructura de datos o aplicación. Su principio sugiere:
  - La tabla original debe ser reconstruida desde las tablas resultantes en las cuales a sido troceada.

  Los beneficios de aplicar esta regla aseguran que no se ha creado ninguna columna extraña en las tablas y que la estructura de las tablas que has creado sea del tamaño justo que tiene que ser.

Diagrama Entidad - Relación.



CAPITULO III

"Desarrollo del Sistema de Control para el Programa de Seguimiento de Egresados de la Escuela Nacional de Estudios Profesionales Campus Aragón (ProSE-A)."

Karina Janette Olvera Alvarez
Ingeniería en Computación
Generación 95 - 99

Desarrollo

El desarrollo del sistema de control para el ProSE-A se realizó de la siguiente manera: Después de realizado el análisis y diseño en base a la notación UML utilizada en el paradigma de Orientación a Objetos; se inició la construcción del sistema en Lenguaje Java.

Dicha construcción comenzó retomando el hecho de que existían dos pantallas principales; la Pantalla de Entrada y la Pantalla de Consulta.

En donde interactúan con el Sistema todos los actores: Alumno, Egresado, Jefe de Carrera y Administrador del Sistema.

La construcción de la Pantallas se ideo de tal manera que pudieran recibir y mandar información al usuario, y entre ellas mismas para interactuar con la base de datos.

Tomando en cuenta las ventajas y facilidades para desarrollo que proporciona el lenguaje Java. El desarrollo del sistema tuvo las siguientes limitaciones.

Limitaciones

La idea original para el sistema era el uso de Applets de Java, aprovechando las ventajas que traen consigo, tales como botones, etiquetas, bordes, barras de herramientas y de menús, listas, conjuntos, estructuras, etc. Una gran variedad de elementos visuales para su construcción. Sobre todo utilizando el API de Swing.

La desventaja de éstos es que cuentan con un gran mecanismo de seguridad. Lo que no permite el acceso de información a través de ellos.

Los Navegadores actuales imponen las siguientes restricciones a los applets que se cargan a través de la Red:

- Un applet no puede cargar librerías ni definir métodos nativos.
- No puede leer ni escribir ficheros en el Host en el que se está ejecutando.
- No puede realizar conexiones en la Red, excepto con el Host del que fue cargado.
- No puede arrancar ningún programa en el Host donde se está ejecutando.
- No puede leer ciertas propiedades del sistema.
- Las ventanas que proporcionan los applets tienen un aspecto diferente a las de cualquier aplicación.

Fue entonces cuando se pensó en alternar su funcionamiento con Servlets, ya que estos hacen las peticiones de información a la base de datos y regresan los valores como parámetros al applet.

Un servlet generalmente es utilizado para procesar formas (requisiciones de usuarios),verificar ("authenticate") usuarios, generar contenido dinámico, etc.

Los Servlets son las respuesta de la tecnología Java a la programación CGI. Son programas que se ejecutan en un servidor Web y construyen páginas Web. Construir páginas Web al vuelo es útil (y comúnmente usado) por un número de razones:

Las páginas Web pueden estar basadas en datos enviados por el usuario. Por ejemplo, las páginas de resultados de los motores de búsqueda se generan de esta forma, y los programas que procesan pedidos desde sites de comercio electrónico también.

Los datos cambian frecuentemente. Por ejemplo, un informe sobre el tiempo o páginas de cabeceras de noticias podrían construir la página dinámicamente, quizás devolviendo una página previamente construida y luego actualizándola.

Las páginas Web pueden usar información desde bases de datos corporativas u otras fuentes. Por ejemplo, usaríamos servlets para hacer una página Web en una tienda on-line (en línea) que liste los precios actuales y el número de artículos en existencia.

Pero la misma seguridad del applet complicaba la independencia de plataformas, propia de Java; por lo que se decidió eliminar el uso de applets para sustituirlos por JSP's (Java Server Pages).

Java Server Pages (JSP) es una tecnología que nos permite mezclar HTML estático con HTML generado dinámicamente. Muchas páginas Web que están construidas con programas CGI son casi estáticas, con la parte dinámica limitada a muy pocas localizaciones. Pero muchas variaciones CGI, incluyendo los servlets, hacen que generemos la página completa mediante nuestro programa, incluso aunque la mayoría de ella sea siempre lo mismo. JSP nos permite crear dos partes de forma separada.

Los JSP's nos dan la ventaja de que trabajan conjuntamente con los servlets y son muy ligeros en las páginas web; lo que nos permite la interacción, a través de la red, con la base de datos.

Los JSP's consisten en codigo HTML o XML en el que se insertan bloques especiales de código. El código es ejecutado en el serivdor y los resultados son páginas dinámicas que son enviadas al navegador del usuario. A pesar de que los JSP son simples de construir, tienen a su disposición todo el poder de la orientación a objetos de Java y la Java Server API. Los JSP hacen un gran uso de JavaBeans, que son clases que siguen el patrón de los constructores sin argumentos (requeridos en JSP) y los métodos públicos Get y Set.

Un servlet es muy similar a un JSP, inclusive un JSP se convierte eventualmente en un servlet, la diferencia es que un servlet solo contiene lenguaje Java, mientras que un JSP contiene Java y HTML.

Finalmente el sistema fue desarrollado con JSP's y Servlets en interacción con una base de datos Access (con opción a ser cambiada por Oracle posteriormente) .

Presentación

Se realizo una presentación general del primer prototipo del sistema a todos los posibles usuarios; en este caso a todos los Jefes de Carrera del plantel . El prototipo estaba basado en la información proporcionada por el Jefe de Carrera de Ingeniería en Computación; e incluía todos los datos del cuestionario de seguimiento a egresados proporcionado en los "Desayunos de Egresados".[11]

[11] El cual se presenta como un Anexo al presente trabajo de Tesis.

Al terminar dicha presentación; en donde se expuso la funcionalidad completa del sistema; se requirió información propia de todos los Jefes de Carrera para complementar el prototipo. Con lo que se realizaron diversas entrevistas personalizadas con los Jefes de Carrera correspondientes.

Entrevistas

Todas las entrevistas fueron de gran ayuda para desarrollar un prototipo mas robusto del Sistema y así ampliar la funcionalidad del mismo para abarcar a todos los alumnos y egresados de las diferentes carreras con las que cuenta el plantel.

Algunas de las observaciones más relevantes fueron:

- En algunas carreras es obligatorio el que se curse un idioma, lo que se debe tomar muy en cuenta.
- Se debe indicar el lugar donde se realizó el Servicio Social.
- También debe indicarse la o las actividades realizadas en el Servicio Social.
- Si ya es titulado; indique en que fecha se tituló.
- Deberá incluir alguna liga a otra página, tal como la de la Escuela, la de la Universidad, la de Administración Escolar o la carrera en especifico.
- Podría incluirse el dato de Postgrados o Especialidades cursadas.
- Preguntar el porcentaje de Créditos acumulado.
- Si todavía debe materias, indicar cuantas y cuales.
- Si lo especifica su carrera, indicar cual es su Área de Especialidad en el trabajo actual.
- Si lo conociera; el año en que dejo de asistir a la Escuela.
- La forma de titulación elegida; si existiera mas de una posibilidad en su carrera.

Con esta información se complemento el prototipo del sistema que después se presento nuevamente a los Jefes de Carrera y autoridades del plantel.

CONCLUSIONES .-

Pruebas

- **Pruebas Internas al Sistema**
  Se realizaron pruebas internas a la sistema de tal manera que se pudiera comprobar la funcionalidad general del mismo. Abarcando las disciplinas de pruebas de funcionalidad, confiabilidad y desempeño.

- **Pruebas con Usuarios**
  - **Capacitación**
    Se contará con un proceso de capacitación a usuarios, de uso y manejo del sistema. Además de un manual de usuario.
  - **Uso**
    El sistema se pondrá en operación con toda su funcionalidad, para alumnos y personal de la escuela; para llevar a cabo una evaluación del mismo.
  - **Cuestionario de resultados y sugerencias**
    Se llevará a cabo una encuesta por medio de un cuestionario de resultados y sugerencias para los usuarios; a fin de determinar cualquier modificación o mantenimiento que el sistema requiera.

Análisis Costo – Beneficio del Sistema:

¿Cómo se debe determinar la factibilidad de un proceso?

La definición de factibilidad va mucho mas allá del uso común del término. Para los proyectos de Sistemas, la factibilidad es valorada en tres formas principales: operacional, técnica y económicamente.

Un proyecto debe ser factible en las tres formas para merecer un desarrollo posterior. El estudio de factibilidad no es un estudio de sistema completo, en vez de ello se usa el estudio de factibilidad para recopilar datos burdos para la administración, para que a su vez les permitan tomar una decisión sobre si deben continuar con el estudio del sistema

Los datos para el estudio de factibilidad pueden ser recolectados por medios de entrevistas, el tipo de entrevista requerido esta directamente relacionado con el problema u oportunidad que esta siendo sugerida. El Analista de Sistema entrevista a aquellos que frecuentemente necesitan ayuda y a los que están directamente involucrados con el proceso de toma de decisión.

Aunque es importante atacar el problema adecuado, el analista de sistemas no debe gastar mucho tiempo haciendo estudios de factibilidad, debido a que muchos proyectos serán solicitados y solamente unos cuantos serán ejecutados. El estudio de factibilidad debe estar ampliamente comprimido en el tiempo, comprendiendo varias actividades en un pequeño lapso.

Las fases que se deben tener en cuenta en el estudio de factibilidad:

1. Fundamentos del proyecto.
2. Definición de objetivos.
3. Determinación de recursos.
4. Evaluación de la factibilidad.
5. Estimación del tiempo requerido.

Cabe mencionar que para la calendarización de un proyecto existen en el mercado una serie de utilitarios que llevan a la optimización de un proyecto, estos

frecuentemente están basados en diagramas, como ser el de Gantt, Pert, etc. Con estos utilitarios se llegaría a la planeación de proyectos basados en computadoras, que seria un gran paso en la automatización de las actividades.

Para la evaluación de este proyecto fue necesario:
Evaluar la situación actual de la Escuela con respecto al Programa de Seguimiento a Egresados.
Analizar las necesidades de información a partir de los objetivos de programa.
Revisar los planes a mediano y largo plazo.
Establecer los alcances del proyecto.

La forma en que se evaluó este proyecto fue:

Teniendo como base la necesidad de un sistema para el control del Programa de Seguimiento a Egresados de la ENEP – Aragón (ProSE-A); partiendo de cero, ya que no se contaba con ningún tipo de sistema con el cual se pudiera tener contacto con los alumnos a través de Internet.

Se tomo en cuenta que se realizaba en un lenguaje cuyo software es gratuito a través de Internet; con apoyo de una herramienta case; de la cual se contaba con una versión de prueba por noventa días proporcionada por el distribuidor.

Además de contar con un servidor de aplicaciones compatible con el lenguaje utilizado, el cual igualmente es un software gratuito a través de Internet.

Y sabiendo de antemano que el sistema ayudará en gran medida a la extensión de dicho programa escolar.

La conclusión del análisis Costo - Beneficio fue:

La aplicación trae consigo el máximo de beneficios contra el mínimo de costos. Por lo que es muy factible su implementación.

Conclusiones

La programación a objetos, tiene un rumbo un poco desconocido. Muchas veces se cree que los lenguajes visuales son orientados a objetos, sin embargo la realidad es que estos lenguajes son basados en objetos y emulan la orientación a objetos. Mediante el uso de algún lenguaje de modelado como el UML se puede obtener una programación a objetos completa. UML utiliza elementos gráficos como son: casos de uso, diagramas de secuencia, diagramas de colaboración y diagramas de estado entre otros, facilitándonos el descubrimiento de clases nuevas. En la actualidad hay herramientas que nos facilitan la utilización del mismo.

El UML (Unified Modeling Language) es un lenguaje de modelado usado para especificar, visualizar y documentar los artefactos de un sistema orientado a objetos en desarrollo. Sustituye a los métodos de análisis y diseño orientado a objetos que aparecieron a finales de los años ochenta y principios de los noventa, unificando los métodos de Booch, Rumbaugh (OMT), Jacobson y abarcando también ideas de otros métodos. Al ser un lenguaje de modelado y no un método, consiste en la notación principalmente gráfica, que cualquier método puede usar para expresar su diseño.

El comportamiento del sistema es documentado en el modelo de casos de usos que nos muestran las funciones que realiza el sistema (caso de uso), los medios con los que interactúa el sistema (actor) y las relaciones que existen entre estos (diagramas de caso de uso).
El flujo de eventos de un caso de uso es expresado en escenarios mediante el uso de diagramas de interacción los cuales pueden ser diagrama de secuencia y diagrama de colaboración. Los de secuencia muestran las interacciones de los objetos, ordenadas por su secuencia en el tiempo, y las interacciones de los objetos organizadas alrededor de los objetos y las ligas entre ellos y el de colaboración.

UML nos ayuda a descubrir las clases del sistema. Si se utiliza de una manera adecuada, se podrá contar al final del análisis con los atributos y operaciones necesarios en la clase, y son los que darán significado y utilidad a la aplicación que se desarrolla.

El diagrama de transición de estados muestra todos los estados de un objeto, los eventos que recibe el objeto y las acciones resultantes a realizar.

Una de las principales ventajas del UML es la utilización del desarrollo interactivo e incremental. Este desarrollo nos permite ir construyendo el sistema poco a poco. El desarrollo incremental permite ir integrando en cada una de las interacciones los diversos componentes que integran el sistema, por ejemplo, captura de requerimientos, análisis, diseño, implementación y pruebas. Este tipo de desarrollo conduce a eliminar o reducir al máximo los riesgos que conlleva un sistema, siendo atacados los mayores riesgos primero. Existen muchos más conceptos de UML pero se necesita dedicarle mayor tiempo para poder explicar, o es conveniente, incluso asistir al curso completo para poder abarcarlos de una manera amplia.

Existe una herramienta para poder aplicar todos estos conceptos: Rational Rose que está diseñado para proveer al desarrollador con un conjunto de herramientas de modelado visual para el desarrollo de soluciones robustas y eficientes a necesidades de negocio reales. Mediante esta herramienta se abarcan los conceptos de UML necesarios para un desarrollo óptimo. Rational Rose es capaz de generar partes del código para utilizar en el desarrollo como son declaración de clases, de atributos y de operaciones, facilitando de este modo, el trabajo del desarrollador.

Este código puede ser generado en distintos lenguajes, como son Visual Basic, C++ o Java. Siendo este último el que se utilizó para la realización de este sistema; pero ¿porqué Lenguaje Java?

La plataforma Java es fundamentalmente una nueva forma de desarrollar tecnología informática, basada en el poder de las redes y en la idea de que el mismo software debería ejecutarse en diferentes tipos de computadoras, dispositivos personales y otros equipos.
Con Java, se puede usar la misma aplicación desde cualquier equipo, ya sea una PC, computadora Macintosh, una terminal o cualquier tipo de tecnología nueva como los video-teléfonos de Internet.

La razón por la que Java es tan importante es, ¡la red!.

Con Java, la Internet y las redes privadas se vuelven su ambiente de trabajo. Acoplada con el poder de las redes, la plataforma Java está ayudando a los usuarios

de computadoras a hacer cosas que eran antiguamente inimaginables. Por ejemplo, los usuarios pueden de forma segura accesar información personal y aplicaciones cuando están lejos de la oficina usando cualquier computadora que esté conectada a Internet, así que será capaz de accesar aplicaciones desde un teléfono móvil basado en la plataforma Java o incluso usar Smart Cards para obtener acceso desde un simple cajero automático hasta teleféricos.

¿Porqué tecnología Java? Las redes requieren de software que sea portable, modular y seguro, es decir, todas las áreas donde Java brilla, porque fue diseñado para su uso sobre redes desde un principio.

La tecnología Java elimina muchos de los problemas asociados con la instalación y la ejecución de las aplicaciones. Eso es porque generalmente el usuario de Java no tiene que configurar, cargar, o instalar nada. Las actualizaciones son automáticas, haciendo la instalación y configuración obsoleta.

Es toda una nueva forma de pensar en computadoras. Sólo basta con hacer click en una liga o presionar un botón y estará ahí. Más importante, desde el inicio, la plataforma Java fue diseñada para ejecutar programas de forma segura en las redes, lo cual significa que se integra de forma segura con los sistemas existentes en su red.

Para resumir este trabajo diremos que:

El Sistema de Control para el Programa de Seguimiento a Egresados de Aragón (ProSE-A) es una herramienta de fácil uso, construida en Lenguaje Java con la ayuda de la notación UML que llevará el control de la gran cantidad de egresados de la ENEP Aragón, los cuales, por diversas razones dejan truncos sus estudios para incorporarse al campo laboral lo antes posible. Así, este sistema crea un lazo entre los egresados y su escuela para mantener contacto directo entre ellos y, de ser posible, orientarlos para completar su carrera; o ir todavía mas allá, en estudios de postgrado.

GLOSARIO .-

Base de datos
Cualquier conjunto de datos organizados para su almacenamiento en la memoria de una computadora, diseñado para facilitar su mantenimiento y acceso de una forma estándar. Los datos suelen aparecer en forma de texto, números o gráficos. Desde su aparición en la década de 1950, se han hecho imprescindibles para las sociedades industriales.

Hay cuatro modelos principales de bases de datos: el modelo jerárquico, el modelo en red, el modelo relacional (el más extendido hoy en día; los datos se almacenan en tablas a los que se accede mediante consultas escritas en SQL) y el modelo de bases de datos deductivas. Otra línea de investigación en este campo son las bases de datos orientadas a objeto, o de objetos persistentes.

Base de datos relacional
Tipo de base de datos o sistema de administración de bases de datos, que almacena información en tablas (filas y columnas de datos) y realiza búsquedas utilizando los datos de columnas especificadas de una tabla para encontrar datos adicionales en otra tabla. En una base de datos relacional, las filas representan registros (conjuntos de datos acerca de elementos separados) y las columnas representan campos (atributos particulares de un registro). Al realizar las búsquedas, una base de datos relacional hace coincidir la información de un campo de una tabla con información en el campo correspondiente de otra tabla y con ello produce una tercera tabla que combina los datos solicitados de ambas tablas. Por ejemplo, si una tabla contiene los campos NÚM-EMPLEADO, APELLIDO, NOMBRE y ANTIGÜEDAD y otra tabla contiene los campos DEPARTAMENTO, NÚM-EMPLEADO y SALARIO, una base de datos relacional hace coincidir el campo NÚM-EMPLEADO de las dos tablas para encontrar información, como por ejemplo los nombres de los empleados que ganan un cierto salario o los departamentos de todos los empleados contratados a partir de un día determinado. En otras palabras, una base de datos relacional utiliza los valores coincidentes de dos tablas para relacionar información de ambas. Por lo general, los productos de bases de datos para microcomputadoras o microordenadores son bases de datos relacionales.

BASIC
Beginner's All-Purpose Symbolic Instruction Code: Código de Instrucción Simbólica Multipropósito para Principiantes. Lenguaje de programación, creado en 1963, sencillo y muy difundido.

Bibliotecas
Conjunto de programas con un formato determinado para ser utilizados por los desarrolladores de aplicaciones, con el objetivo principal de evitar la repetición de procesos. Llamadas también librerías.

C
Lenguaje de programación desarrollado en 1972 por el estadounidense Dennis Ritchie en los Laboratorios Bell. Debe su nombre a que su predecesor inmediato había sido llamado lenguaje de programación B. Aunque muchos consideran que C es un lenguaje ensamblador más independiente de la máquina que un lenguaje de alto nivel, su estrecha asociación con el sistema operativo UNIX, su enorme popularidad y su homologación por el American National Standards Institute (ANSI) lo han convertido quizá en lo más cercano a un lenguaje de programación estandarizado en el sector de microordenadores o microcomputadoras y estaciones de trabajo. C es un lenguaje compilado que contiene un pequeño conjunto de funciones incorporadas dependientes de la máquina. El resto de las funciones de C son independientes de la máquina y están contenidas en bibliotecas a las que se puede acceder desde programas escritos en C. Estos programas están compuestos por una o más funciones definidas por el programador, por lo que C es un lenguaje de programación estructurada.

C++
Una versión orientada a objetos del lenguaje de programación C, desarrollada por Bjarne Stroustrup a comienzos de la década de 1980 en los Laboratorios Bell.

En línea
Activado y listo para operación, con capacidad de comunicarse o de ser controlado por una computadora. Por ejemplo, una impresora está en línea cuando está lista para imprimir. Una base de datos está en línea cuando puede ser utilizada por un usuario conectado a la computadora que la contiene.

Gráficos de mapas de bits
BMP o Bit Map. Su nombre procede de mapas de bits. Son archivos de tipo gráfico en el que cada punto de la pantalla queda representando por uno o mas bits, dependiendo de si es blanco y negro, tonos de grises o color.
interfase: Elemento de transición o conexión que facilita el intercambio de datos. El teclado, por ejemplo, es una interfase entre el usuario y la computadora.

Gráficos orientados a objetos
También llamados gráficos estructurados. Son gráficos de ordenador o computadora basados en el uso de elementos de construcción, como líneas, curvas, círculos y rectángulos. Los gráficos orientados a objetos, utilizados por ejemplo en diseño asistido por computadora y en programas de dibujo e ilustración, describen un dibujo matemáticamente, como un conjunto de instrucciones que crean los elementos de la imagen. Este sistema se opone al de los gráficos de mapas de bits, otro método muy extendido para crear imágenes, que representa los gráficos como un conjunto de puntos en blanco y negro o en color que sigue un patrón determinado. Los gráficos orientados a objetos permiten al usuario manipular objetos como unidades completas, como por ejemplo al cambiar el largo de una línea o al aumentar el tamaño de un círculo, mientras que los gráficos de mapas de bits requieren pintar de nuevo los puntos individuales de una línea o un círculo. Debido a que los objetos están descritos matemáticamente, los gráficos orientados a objetos se pueden estratificar, girar y ampliar con relativa facilidad.

Interfaz
Punto en el que se establece una conexión entre dos elementos, que les permite trabajar juntos. En el campo de la informática se distinguen diversos tipos de interfaces que actúan a diversos niveles, desde las interfaces claramente visibles, que permiten a las personas comunicarse con los programas, hasta las imprescindibles interfaces *hardware*, a menudo invisibles, que conectan entre sí los dispositivos y componentes dentro de los ordenadores o computadoras.

Interfaz de línea de comandos
Esta es la interfaz más básica, pero también la que proporciona un mayor control. En esta interfaz puede escribir cualquier comando seguido de alguna tecla para proceder a la ejecución correspondiente. Esta interfaz puede contar con algunas funciones similares a las de shell avanzadas, como auto completar según el contexto y las

combinaciones de teclas con Ctrl al escribir comandos. Además, la teclas de flecha, Inicio, Fin y Supr pueden funcionar de forma similar al comando bash de shell.

Interfaz de programación de aplicaciones
Conjunto de rutinas que utiliza un programa de aplicación para solicitar y efectuar servicios de nivel inferior ejecutados por un sistema operativo informático. Un programa de aplicación efectúa dos tipos de tareas: las relacionadas con el trabajo que se está realizando, por ejemplo aceptar la entrada de texto o de números en un documento u hoja de cálculo, y las relacionadas con las tareas de mantenimiento, como la gestión de archivos y la presentación de la información en la pantalla. Estas tareas de mantenimiento son realizadas por el sistema operativo y la interfaz de programación de aplicaciones (API) proporciona al programa los medios para comunicarse con el sistema, indicándole qué tarea básica del sistema debe realizar y cuándo. En los equipos que funcionan con una interfaz gráfica de usuario (GUI), como el Apple Macintosh, una API también ayuda a los programas de aplicación a gestionar las ventanas, menús, iconos... En las redes de área local (LAN), una API como la NetBIOS de IBM proporciona a las aplicaciones métodos uniformes de solicitar servicios a los niveles inferiores de la red.

Interfaz gráfica de usuario
Tipo de visualización que permite al usuario elegir comandos, iniciar programas y ver listas de archivos y otras opciones utilizando las representaciones visuales (íconos) y las listas de elementos del menú. Las selecciones pueden activarse bien a través del teclado o con el ratón.

Para los autores de aplicaciones, las interfaces gráficas de usuario ofrecen un entorno que se encarga de la comunicación con la computadora. Esto hace que el programador pueda concentrarse en la funcionalidad, ya que no está sujeto a los detalles de la visualización ni a la entrada a través del ratón o del teclado. También permite a los programadores crear programas que realicen de la misma forma las tareas más frecuentes, como guardar un archivo, porque la interfaz proporciona mecanismos estándar de control como ventanas y cuadros de diálogo. Otra ventaja es que las aplicaciones escritas para una interfaz gráfica de usuario son independientes de los dispositivos: a medida que la interfaz cambia para permitir el uso de nuevos dispositivos de entrada y salida, como un monitor de pantalla grande o un dispositivo óptico de almacenamiento, las aplicaciones pueden utilizarlos sin necesidad de cambios.

Interfaces de usuario
Cuentan con el diseño gráfico, los comandos, mensajes y otros elementos que permiten a un usuario comunicarse con un programa. Las microcomputadoras disponen de tres tipos básicos de interfaces de usuario (que no necesariamente son excluyentes entre sí): la interfaz de línea de comandos, reconocible por los símbolos A o C del sistema MS-DOS, que responde a los comandos introducidos por el usuario; la interfaz controlada por menús utilizada en muchas aplicaciones (por ejemplo Lotus 1-2-3) ofrece al usuario una selección de comandos, permitiéndole elegir uno de ellos presionando la letra correspondiente, desplazando el cursor con las teclas de dirección o apuntando con el *mouse* (ratón); y la interfaz gráfica de usuario, una característica de los equipos Apple Macintosh y de los programas basados en ventanas, representa visualmente los conceptos, por ejemplo un escritorio, y permite al usuario no sólo controlar las opciones de los menús, sino también el tamaño, la posición y el contenido de una o más ventanas o áreas de trabajo que aparezcan en pantalla.

Internet
Interconexión de redes informáticas que permite a las computadoras conectadas comunicarse directamente. El término suele referirse a una interconexión en particular, de carácter planetario y abierto al público, que conecta redes informáticas de organismos oficiales, educativos y empresariales. También existen sistemas de redes más pequeños llamados *intranet*, generalmente para el uso de una única organización.

La tecnología de Internet es una precursora de la llamada 'superautopista de la información', un objetivo teórico de las comunicaciones informáticas que permitiría proporcionar a colegios, bibliotecas, empresas y hogares acceso universal a una información de calidad que eduque, informe y entretenga. A principios de 1996 estaban conectadas a Internet más de 25 millones de computadoras en más de 180 países, y la cifra sigue en aumento.

Intérpretes y compiladores
La traducción de una serie de instrucciones en lenguaje ensamblador (el código fuente) a un código máquina (o código objeto) no es un proceso muy complicado y se realiza normalmente por un programa especial llamado compilador. La traducción de un código fuente de alto nivel a un código máquina también se realiza con un

compilador, en este caso más complejo, o mediante un intérprete. Un compilador crea una lista de instrucciones de código máquina, el código objeto, basándose en un código fuente. El código objeto resultante es un programa rápido y listo para funcionar, pero que puede hacer que falle el ordenador si no está bien diseñado. Los intérpretes, por otro lado, son más lentos que los compiladores ya que no producen un código objeto, sino que recorren el código fuente una línea cada vez. Cada línea se traduce a código máquina y se ejecuta. Cuando la línea se lee por segunda vez, como en el caso de los programas en que se reutilizan partes del código, debe compilarse de nuevo. Aunque este proceso es más lento, es menos susceptible de provocar fallos en la computadora.

Lenguaje compilado
Un lenguaje cuyos programas se traducen a código máquina antes de ejecutarse, a diferencia de un lenguaje interpretado, cuyos programas se traducen y ejecutan instrucción por instrucción.

Lenguajes de alto nivel
Por lo general se piensa que los ordenadores son máquinas que realizan tareas de cálculos o procesamiento de textos. La descripción anterior es sólo una forma muy esquemática de ver una computadora. Hay un alto nivel de abstracción entre lo que se pide a la computadora y lo que realmente comprende. Existe también una relación compleja entre los lenguajes de alto nivel y el código máquina.

Los lenguajes de alto nivel son normalmente fáciles de aprender porque están formados por elementos de lenguajes naturales, como el inglés. En BASIC, el lenguaje de alto nivel más conocido, los comandos como "IF CONTADOR = 10 THEN STOP" pueden utilizarse para pedir a la computadora que pare si CONTADOR es igual a 10. Por desgracia para muchas personas esta forma de trabajar es un poco frustrante, dado que a pesar de que las computadoras parecen comprender un lenguaje natural, lo hacen en realidad de una forma rígida y sistemática.

Lenguajes de bajo nivel
Vistos a muy bajo nivel, los microprocesadores procesan exclusivamente señales electrónicas binarias. Dar una instrucción a un microprocesador supone en realidad enviar series de unos y ceros espaciadas en el tiempo de una forma determinada. Esta secuencia de señales se denomina código máquina. El código representa normalmente datos y números e instrucciones para manipularlos. Un modo más fácil

de comprender el código máquina es dando a cada instrucción un mnemónico, como por ejemplo STORE, ADD o JUMP. Esta abstracción da como resultado el ensamblador, un lenguaje de muy bajo nivel que es específico de cada microprocesador.

Los lenguajes de bajo nivel permiten crear programas muy rápidos, pero que son a menudo difíciles de aprender. Más importante es el hecho de que los programas escritos en un bajo nivel sean altamente específicos de cada procesador. Si se lleva el programa a otra máquina se debe rescribir el programa desde el principio.

Lenguaje de consulta estructurado
Un sublenguaje utilizado en bases de datos para consultar, actualizar y manejar bases de datos relacionales. Se deriva de un proyecto de investigación de IBM, que creó el "lenguaje estructurado de consulta en inglés" (SEQUEL) en la década de los setenta. El SQL es un estándar aceptado en productos de bases de datos. A pesar de que no se trata de un lenguaje de programación como puedan serlo C o Pascal, puede utilizarse en el diseño de consultas interactivas y puede incluirse en una aplicación como un conjunto de instrucciones de manejo de datos. El SQL estándar cuenta también con elementos destinados a la definición, modificación, control y protección de los datos. Tanto los usuarios técnicos como los que no lo son pueden utilizar este lenguaje.

Lenguaje de programación
Cualquier lenguaje artificial que puede utilizarse para definir una secuencia de instrucciones para su procesamiento por un ordenador o computadora. Es complicado definir qué es y qué no es un lenguaje de programación. Se asume generalmente que la traducción de las instrucciones a un código que comprende la computadora debe ser completamente sistemática. Normalmente es la computadora la que realiza la traducción.

Microprocesadores
Es el chip más importante de una computadora. Su velocidad se mide en MHz (Megahertz).

Modelo relacional
El modelo relacional fue propuesto originariamente por E.F. Codd en un ya famoso artículo de 1970. Gracias a su coherencia y facilidad de uso, el modelo se ha

convertido en los años 80 en el más usado para la producción de DBMS. La estructura fundamental del modelo relacional es precisamente esa, "relación", es decir una tabla bidimensional constituida por líneas (tuple) y columnas (atributos). Las relaciones representan las entidades que se consideran interesantes en la base de datos. Cada instancia de la entidad encontrará sitio en una tupla de la relación, mientras que los atributos de la relación representarán las propiedades de la entidad.

Programa
Instrucciones que varían según el lenguaje que se utiliza, pero cuyo fin es el de controlar las acciones que tiene que llevar a cabo el ordenador y sus periféricos.

Programación orientada a objetos
Un estilo de programación en el que un programa se contempla como un conjunto de objetos limitados que, a su vez, son colecciones independientes de estructuras de datos y rutinas que interactúan con otros objetos. Una clase define las estructuras de datos y rutinas de un objeto. Un objeto es una instancia de una clase, que se puede usar como una variable en un programa. En algunos lenguajes orientados a objetos, éste responde a mensajes, que son el principal medio de comunicación. En otros lenguajes orientados a objeto se conserva el mecanismo tradicional de llamadas a procedimientos.

Programación estructurada
Término general que se refiere a un tipo de programación que produce código con un flujo limpio, un diseño claro y un cierto grado de modularidad o de estructura jerárquica. Entre los beneficios de la programación estructurada se encuentran la facilidad de mantenimiento y la legibilidad por parte de otros programadores.

Shell
Un shell es un término utilizado para identificar una interface o la parte más externa de un programa. Algunas aplicaciones y sistemas operativos proveen diferentes shells o frentes para que al usuario le sea más sencillo interactuar con un programa.
En el caso de un shell de Unix, éste sirve de intermediario entre el usuario y el sistema operativo del servidor (Unix es un sistema operativo multiusuario y multitarea, es decir, que puede atender a varias personas, máquinas y tareas al mismo tiempo). El shell de Unix se encarga de aceptar los comandos por parte del usuario, verificando su correcta sintaxis y luego enviando las órdenes a otra parte del sistema para su ejecución.

convertido en los años 80 en el más usado para la producción de DBMS. La estructura fundamental del modelo relacional es precisamente esa, "relación", es decir una tabla bidimensional constituida por líneas (tuple) y columnas (atributos). Las relaciones representan las entidades que se consideran interesantes en la base de datos. Cada instancia de la entidad encontrará sitio en una tupla de la relación, mientras que los atributos de la relación representarán las propiedades de la entidad.

Programa
Instrucciones que varían según el lenguaje que se utiliza, pero cuyo fin es el de controlar las acciones que tiene que llevar a cabo el ordenador y sus periféricos.

Programación orientada a objetos
Un estilo de programación en el que un programa se contempla como un conjunto de objetos limitados que, a su vez, son colecciones independientes de estructuras de datos y rutinas que interactúan con otros objetos. Una clase define las estructuras de datos y rutinas de un objeto. Un objeto es una instancia de una clase, que se puede usar como una variable en un programa. En algunos lenguajes orientados a objetos, éste responde a mensajes, que son el principal medio de comunicación. En otros lenguajes orientados a objeto se conserva el mecanismo tradicional de llamadas a procedimientos.

Programación estructurada
Término general que se refiere a un tipo de programación que produce código con un flujo limpio, un diseño claro y un cierto grado de modularidad o de estructura jerárquica. Entre los beneficios de la programación estructurada se encuentran la facilidad de mantenimiento y la legibilidad por parte de otros programadores.

Shell
Un shell es un término utilizado para identificar una interface o la parte más externa de un programa. Algunas aplicaciones y sistemas operativos proveen diferentes shells o frentes para que al usuario le sea más sencillo interactuar con un programa.
En el caso de un shell de Unix, éste sirve de intermediario entre el usuario y el sistema operativo del servidor (Unix es un sistema operativo multiusuario y multitarea, es decir, que puede atender a varias personas, máquinas y tareas al mismo tiempo). El shell de Unix se encarga de aceptar los comandos por parte del usuario, verificando su correcta sintaxis y luego enviando las órdenes a otra parte del sistema para su ejecución.

Los sistemas Unix pueden tener diferentes shells, los más utilizados son el C shell (csh), el Bourne (bash) y el Korn (ksh). Cada uno de ellos ofrece un lenguaje diferente de comandos.
Así, por medio de un shell una persona puede tener acceso a los recursos de una máquina.

Sistema operativo
Programa que administra los demás programas en una computadora.

SQL
Structured Query Language. Lenguaje de programación que se utiliza para recuperar y actualizar la información contenida en una base de datos. Fue desarrollado en los años 70 por IBM. Se ha convertido en un estándar ISO y ANSI.

Sun Microsystems, Inc.
Compañía estadounidense de ordenadores o computadoras fundada en 1983, con sede central en Mountain View, California. La compañía ha sido líder en la fabricación de estaciones de trabajo de computadoras. Su rápido éxito se debe a su capacidad para cubrir la demanda surgida entre ingenieros y especialistas técnicos de estaciones de trabajo UNIX potentes pero relativamente asequibles. A finales de 1991, Sun tenía una participación del 30% en el mercado de estaciones de trabajo, situándose a gran distancia de su más próximo competidor. La estrategia de la compañía para la década de 1990 se basa en la venta de equipos y *software* de estaciones de trabajo a un grupo de clientes más amplio. Las compañías de Wall Street fueron algunos de los primeros clientes comerciales. Tras ellas vinieron compañías aéreas, bancos, compañías de seguros y autoridades municipales. Reconociendo la importancia de los nuevos sistemas de *software*, Sun colaboró con la competencia para facilitar el intercambio de datos y programas entre diferentes sistemas de computadoras. En 1993, con su filial FirstPerson Inc., Sun anunció su entrada en el mercado de la electrónica de consumo con computadoras portátiles que se pueden conectar a sistemas de redes.

UNIX
Sistema operativo multiusuario que incorpora multitarea. Fue desarrollado originalmente por Ken Thompson y Dennis Ritchie en los laboratorios AT&T Bell en 1969 para su uso en minicomputadoras. El sistema operativo UNIX tiene diversas variantes y se considera potente, más transportable e independiente de equipos

concretos que otros sistemas operativos porque está escrito en lenguaje C. El UNIX está disponible en varias formas, entre las que se cuenta AIX, una versión de UNIX adaptada por IBM (para su uso en estaciones de trabajo basadas en RISC), A/UX (versión gráfica para equipos Apple Macintosh) y Mach (un sistema operativo reescrito, pero esencialmente compatible con UNIX, para las computadoras NeXT).

URL

Uniform Resource Locator. Se conoce por este nombre a las direcciones dentro de Internet, normalmente, aunque no necesariamente, refiriéndonos a páginas Web. En este caso se distinguen por iniciarse con http:// No obstante es una simplificación para el usuario el referenciarlas de esta forma, en realidad son secuencias de números que se dirigen de forma inequívoca a una dirección. Esto se conoce como DNS.

World Wide Web

World Wide Web (también conocida como Web o WWW) es una colección de ficheros, denominados lugares de Web o páginas de Web, que incluyen información en forma de textos, gráficos, sonidos y vídeos, además de vínculos con otros ficheros. Los ficheros son identificados por un localizador universal de recursos (URL, siglas en inglés) que especifica el protocolo de transferencia, la dirección de Internet de la máquina y el nombre del fichero. Por ejemplo, un URL podría ser http://www.encarta.es/msn.com. Los programas informáticos denominados exploradores —como Navigator, de Netscape, o Internet Explorer, de Microsoft— utilizan el protocolo http para recuperar esos ficheros. Continuamente se desarrollan nuevos tipos de ficheros para la WWW, que contienen por ejemplo animación o realidad virtual (VRML). Hasta hace poco había que programar especialmente los lectores para manejar cada nuevo tipo de archivo. Los nuevos lenguajes de programación (como Java, de Sun Microsystems) permiten que los exploradores puedan cargar programas de ayuda capaces de manipular esos nuevos tipos de información.[12]

[12] *Enciclopedia Microsoft® Encarta® 98* © 1993-1997 Microsoft Corporation.

ANEXOS .-

Karina Janette Olvera Alvarez
Ingeniería en Computación
Generación 95 - 99

Universidad Nacional Autónoma de México
Campus Aragón
Jefatura de la Carrera de Ingeniería en Computación
Programa de Seguimiento a Egresados
ProSE-A

Nombre:_____
Apellido Paterno Apellido Materno Nombre(s)
Sexo: M( ) F( ) Dirección:_____
Calle No. Exterior No. Interior

_____
Colonia Delegación o Municipio Código Postal Entidad Federativa

Teléfono Particular:_____ Teléfono(recado):_____ Celular:_____

Fax:_____ Correo Electrónico:_____

Generación:_____ No. Cuenta:_____ Titulado: Si( ) No( )
Año Ingreso – Año Egreso

En Proceso( ) Tesis o Tesina( ) Nombre de Tesis:_____

Idiomas Cursados:_____ Servicio Social:_____

Trabajo Actual:_____ Puesto:_____ Lugar de Trabajo:_____

Dirección:_____

Teléfono:_____ Fax:_____ Correo Electrónico:_____

Si tienes los datos de algunos compañeros de tu generación, puedes colaborar con nosotros anotándolos con su Nombre Completo, Dirección, Teléfono en el siguiente espacio:

Nombre	Dirección	Teléfono	Correo Electrónico

Servicios.

El sistema de control para el ProSE-A cuenta con los siguientes servicios:

- Consulta de datos del Egresado
- Actualización de los datos del Egresado
- Consulta General por Carrera

En lo que respecta a los Egresados se cuenta con un módulo de consulta (para verificar información) de sus propios datos con la opción de actualización de los mismos.

Así mismo en lo que respecta a los Funcionarios de la Escuela Nacional de Estudios Profesionales Aragón (ENEP-A), se cuenta con un módulo de consultas generales por carrera con opción de generar reportes en base a esa información.

Al entrar al sistema aparece la Pantalla Principal que presenta elementos de seguridad; por lo que, para ingresar al sistema se requerirá introducir una Clave de Usuario y una Contraseña.

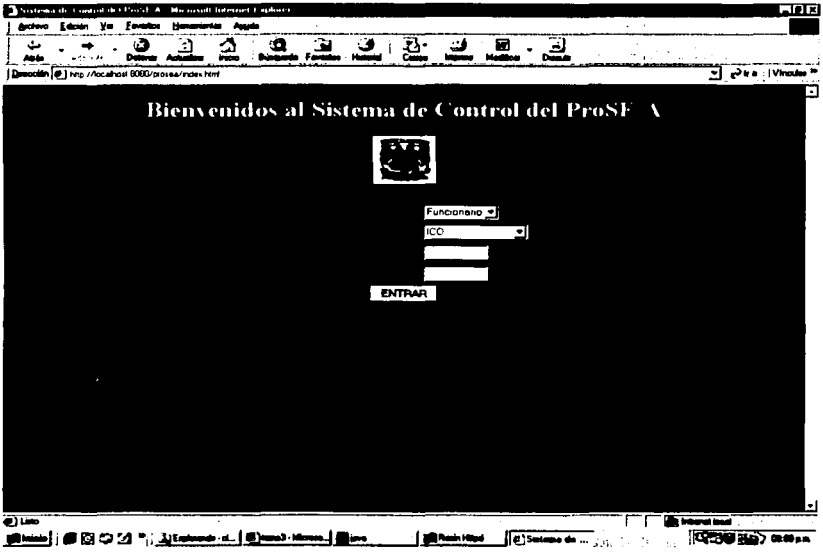
La Clave de Usuario se compone del número de cuenta correspondiente a su registro escolar en la UNAM, con el siguiente formato: 999999999 (nueve dígitos), omitiendo el guión que separa el digito verificador; agregando un cero en la parte más significativa si su generación es menor al 2000; en tal caso si su número de cuenta es 9561507-7 su clave de usuario será 095615077.

La Contraseña se compone de la fecha de nacimiento y tiene el siguiente formato: ddmmaaaa; en donde dd es el día, mm el mes y aaaa el año; por ejemplo si la fecha de nacimiento es el 27 de Marzo de 1976, la contraseña será 27031976.

Además debe elegirse, de una lista de opciones el Tipo de Usuario y la Carrera correspondiente. Cabe señalar que en base a toda esta información proporcionada por el usuario se le dará acceso a su correspondiente módulo en el sistema.

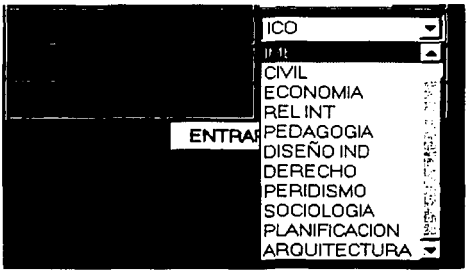
Conociendo el Sistema de Control para el Prose-A...

Pantalla Principal

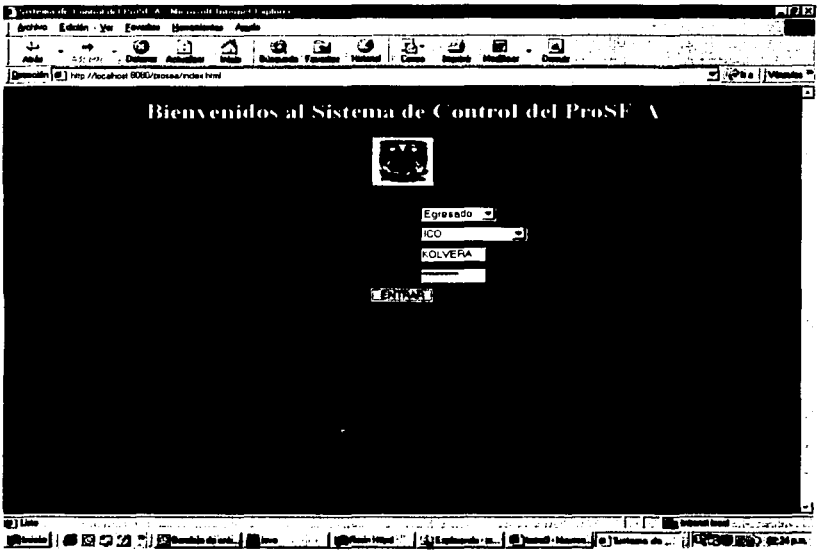


Lista de opciones sobre el Tipo de Usuario:

Lista de opciones sobre la Carrera:

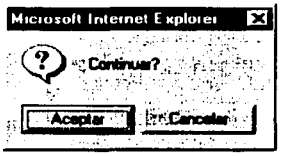


Yo soy un Egresado, ¿que debo hacer para usar el sistema ?

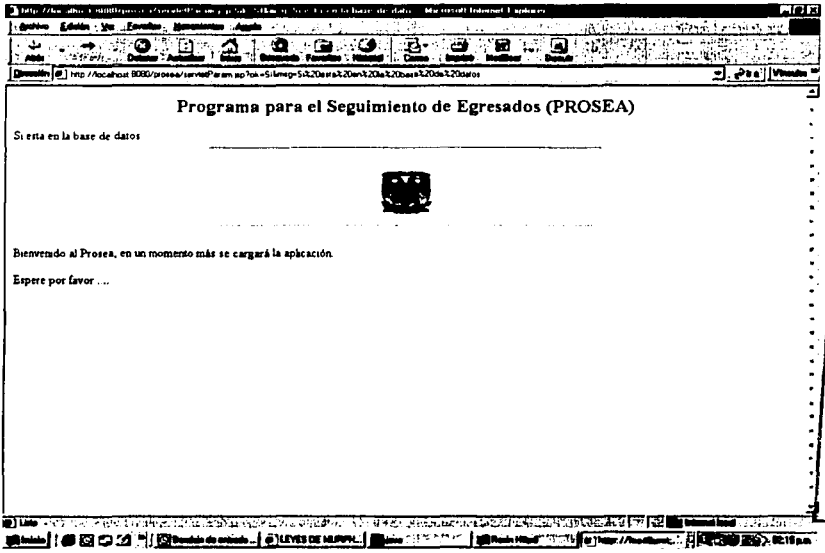


Después de haber digitado su Clave de Usuario y Contraseña, y de haber elegido el Tipo de Usuario: EGRESADO y su Carrera correspondiente de las listas de opciones, usted debe dar clic en el botón ENTRAR.

Inmediatamente después el sistema le proporciona un cuadro de diálogo en donde se le pide confirmar si desea continuar con la operación, o desea permanecer en esa misma pantalla.

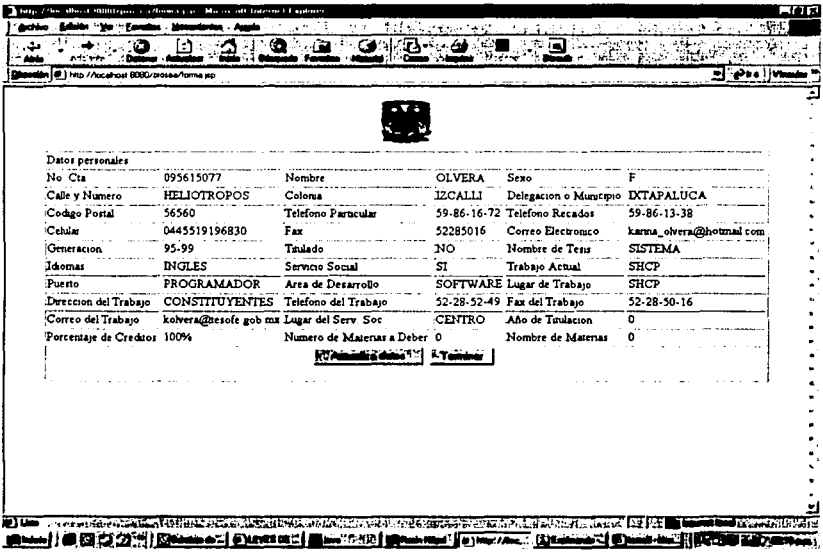


Al darle clic en el botón ACEPTAR del cuadro de diálogo; comienza el proceso de autorización del usuario en el sistema. Si la Clave de Usuario y la Contraseña son correctas para el tipo de Usuario: EGRESADO y para la carrera correspondiente entonces aparecerá la pantalla de Bienvenida.



En este momento se esta llevando a cabo el proceso de recopilación de los datos registrados en el sistema.

Y unos segundos después aparecerá la pantalla correspondiente a sus datos personales y escolares.

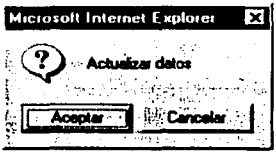


Datos personales

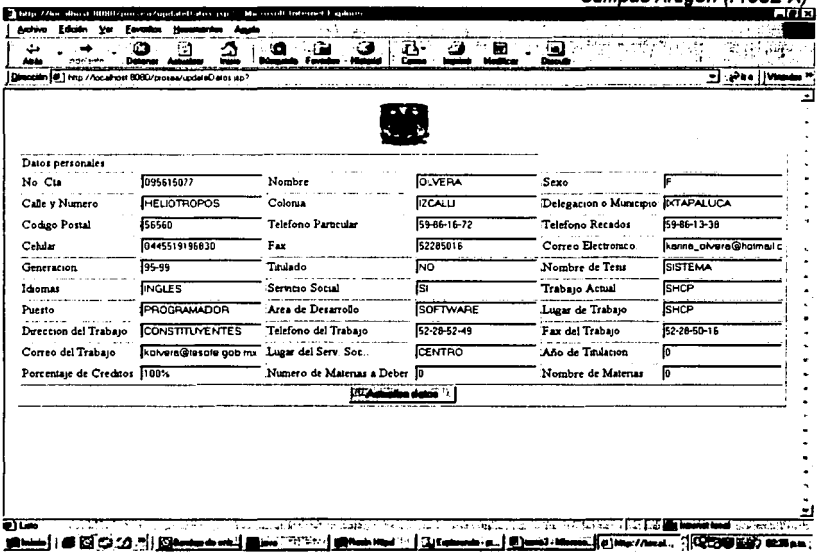
No Cta	095615077	Nombre	OLVERA	Sexo	F
Calle y Numero	HELIOTROPOS	Colonia	IZCALLI	Delegacion o Municipio	IXTAPALUCA
Codigo Postal	56560	Telefono Particular	59-86-16-72	Telefono Recados	59-86-13-38
Celular	0445519196830	Fax	52285016	Correo Electronico	karina_olvera@hotmail.com
Generacion	95-99	Titulado	NO	Nombre de Tesis	SISTEMA
Idiomas	INGLES	Servicio Social	SI	Trabajo Actual	SHCP
Puesto	PROGRAMADOR	Area de Desarrollo	SOFTWARE	Lugar de Trabajo	SHCP
Direccion del Trabajo	CONSTITUYENTES	Telefono del Trabajo	52-28-52-49	Fax del Trabajo	52-28-50-16
Correo del Trabajo	kolvera@tesofe.gob.mx	Lugar del Serv. Soc	CENTRO	Año de Titulacion	0
Porcentaje de Creditos	100%	Numero de Materias a Deber	0	Nombre de Materias	0

Ya estoy dentro del sistema, pero mis datos están incompletos o no son los más actuales, ¿cómo los modifico?

Para actualizar los datos debe dar clic en el botón ACTUALIZA DATOS; enseguida el sistema proporciona un cuadro de diálogo en donde se le pide confirmar si desea actualizar los datos, o desea cancelar la petición.



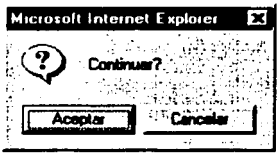
Al darle clic en el botón ACEPTAR ; el sistema proporciona la pantalla donde los campos están abiertos para modificarse, o ser rellenados correctamente.



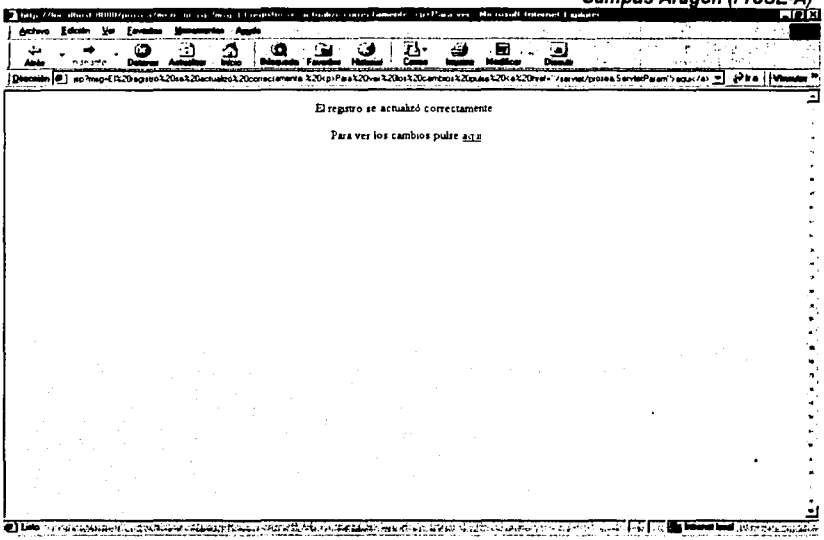
Datos personales					
No Cta	095615077	Nombre	OLVERA	Sexo	F
Calle y Numero	HELIOTROPOS	Colonia	IZCALLI	Delegacion o Municipio	CTAPALUCA
Codigo Postal	56560	Telefono Particular	59-86-16-72	Telefono Recados	59-86-13-38
Celular	0445519186830	Fax	52285016	Correo Electronico	karina_olvera@hotmail.c
Generacion	95-99	Titulado	NO	Nombre de Tesis	SISTEMA
Idiomas	INGLES	Servicio Social	SI	Trabajo Actual	SHCP
Puesto	PROGRAMADOR	Area de Desarrollo	SOFTWARE	Lugar de Trabajo	SHCP
Direccion del Trabajo	CONSTITUYENTES	Telefono del Trabajo	52-28-52-49	Fax del Trabajo	52-28-50-15
Correo del Trabajo	kolvera@issste.gob.mx	Lugar del Serv. Soc.	CENTRO	Año de Titulacion	0
Porcentaje de Creditos	100%	Numero de Materias a Deber	0	Nombre de Materias	0

Actualiza datos

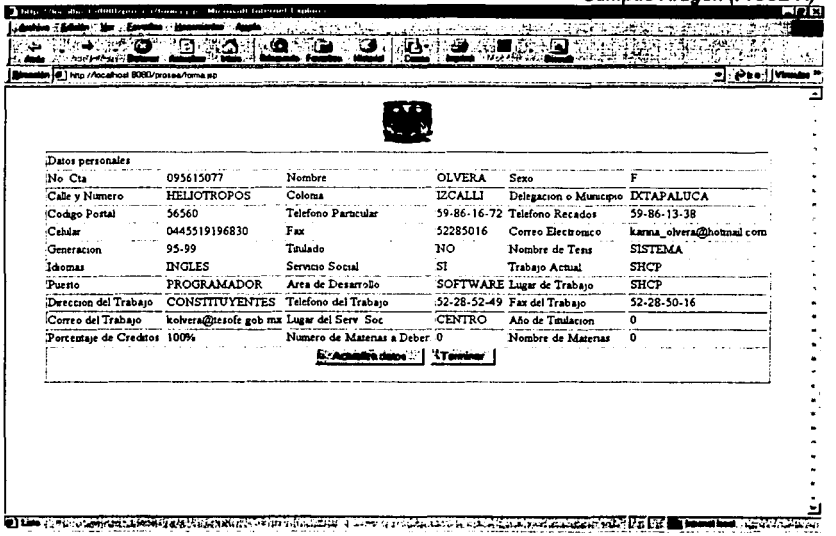
Al terminar de actualizar la información debe dar clic en el botón ACTUALIZA DATOS, la cual pedirá confirmación de la operación mediante otro cuadro de diálogo; si desea continuar debe dar clic en el botón ACEPTAR.



Enseguida el sistema le confirmará que EL REGISTRO SE ACTUALIZÓ CORRECTAMENTE. Para poder corroborar que los cambios fueron realizados el sistema le proporciona una "liga" hacia la página de consulta de sus datos.



Al dar clic en la palabra AQUÍ, el sistema le mostrará sus datos ya actualizados.



Datos personales

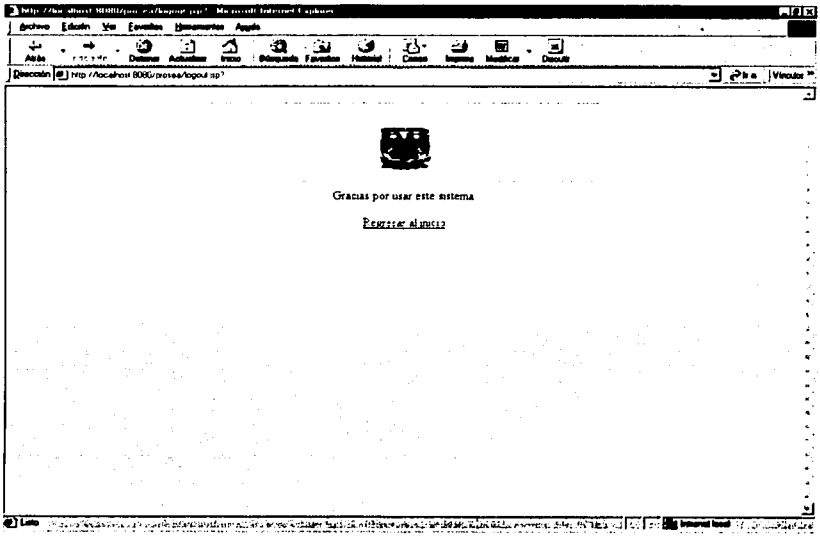
No Cta	095615077	Nombre	OLVERA	Sexo	F
Calle y Numero	HELIOTROPOS	Coloma	IZCALLI	Delegacion o Municipio	IXTAPALUCA
Codigo Postal	56560	Telefono Particular	59-86-16-72	Telefono Recados	59-86-13-38
Celular	0445519196830	Fax	52285016	Correo Electronico	kanna_olvera@hotmail.com
Generacion	95-99	Titulado	NO	Nombre de Tesis	SISTEMA
Idiomas	INGLES	Servicio Social	SI	Trabajo Actual	SHCP
Puesto	PROGRAMADOR	Area de Desarrollo	SOFTWARE	Lugar de Trabajo	SHCP
Direccion del Trabajo	CONSTITUYENTES	Telefono del Trabajo	52-28-52-49	Fax del Trabajo	52-28-50-16
Correo del Trabajo	kolvera@tesofe.gob.mx	Lugar del Serv. Soc.	CENTRO	Año de Titulacion	0
Porcentaje de Creditos	100%	Numero de Materias a Deber	0	Nombre de Materias	0

Actualizar datos | Terminar

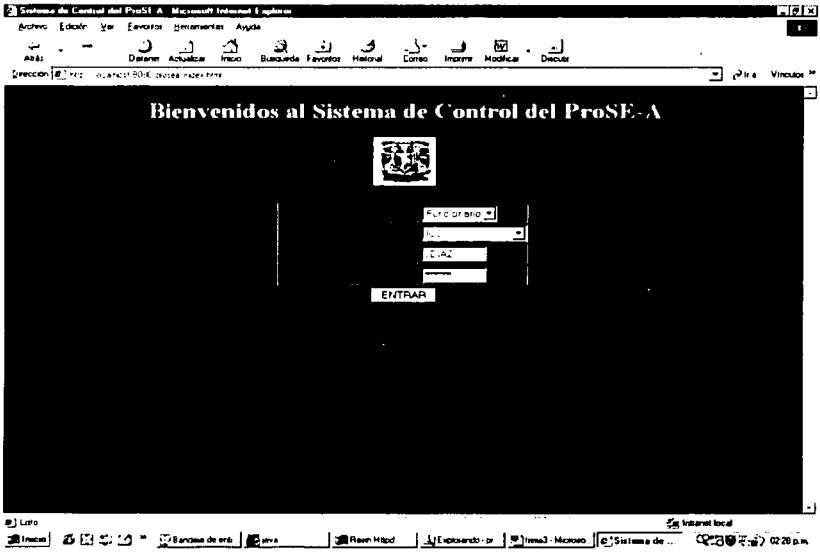
Ya actualicé mis datos; ¿cómo salgo del sistema?

Para salir por completo del sistema debe dar clic en el botón TERMINAR de la página en donde se encuentra la consulta de datos. Inmediatamente después aparece un cuadro de diálogo para confirmar la operación, o cancelarla.

Al dar clic en el botón ACEPTAR el sistema cierra su sesión y proporciona una página en donde le agradece su participación, y le ofrece una "liga" a la página de inicio del sistema.

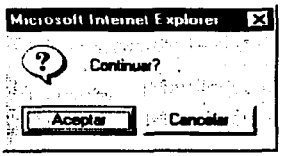


Pero, yo soy un Funcionario de la ENEP-A, ¿funciona el sistema igual para mí?

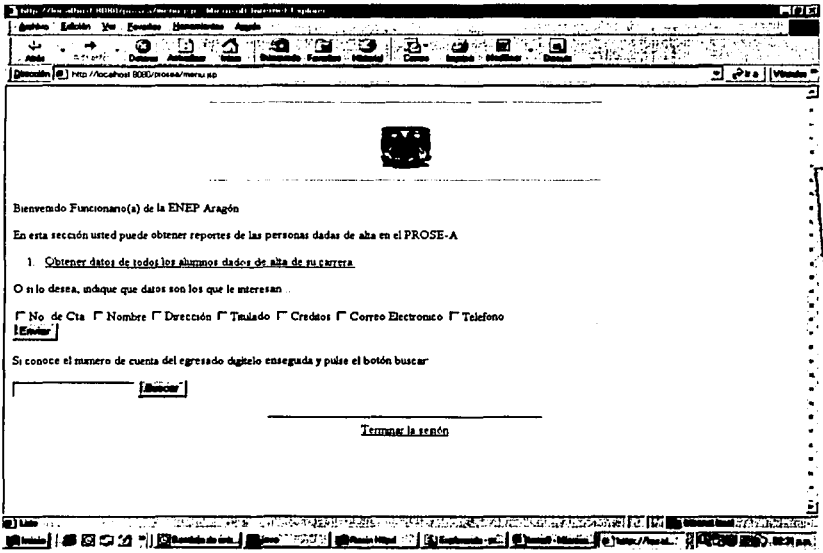


Casi igual... después de haber digitado su Clave de Usuario y Contraseña, y de haber elegido el Tipo de Usuario: FUNCIONARIO y su Carrera correspondiente de las listas de opciones, usted debe dar clic en el botón ENTRAR.

Inmediatamente después el sistema le proporciona un cuadro de diálogo en donde se le pide confirmar si desea continuar con la operación, o desea permanecer en esa misma pantalla.



Al darle clic en el botón ACEPTAR del cuadro de diálogo; comienza el proceso de autorización del usuario en el sistema. Si la Clave de Usuario y la Contraseña son correctas para el tipo de Usuario: FUNCIONARIO y para la carrera correspondiente entonces aparecerá la pantalla de Bienvenida.

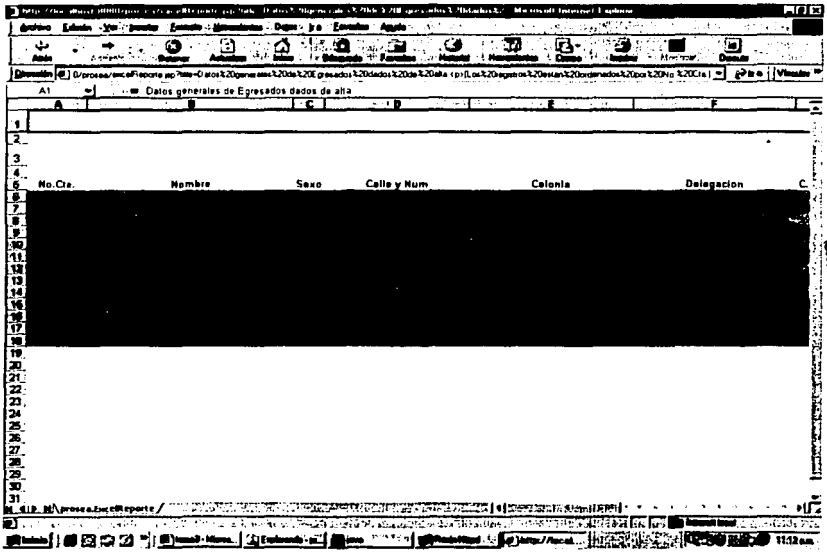


Esta pantalla, además de darle la bienvenida le ofrece varias opciones para la generación de reportes de la información de los egresados de la carrera correspondiente.

La primera opción es obtener una lista completa de los datos de todos los Egresados dados de alta en el sistema que corresponden a su carrera en específico.

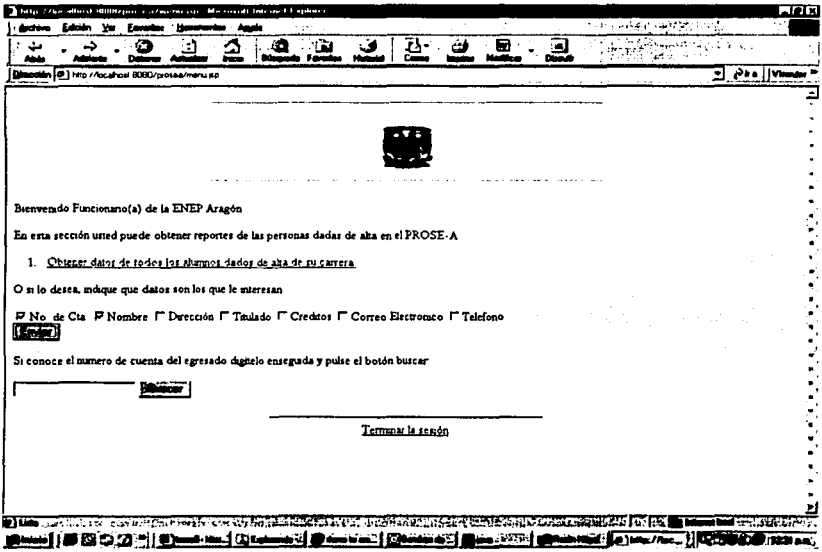
¿Cómo obtengo ese reporte?

Al dar clic sobre la "liga" correspondiente en la página de Bienvenida; o sea sobre el texto: 1. Obtener datos de todos los alumnos dados de alta de su carrera. El sistema proporcionara un reporte en formato de Hoja de Excel, como el que aparece a continuación.



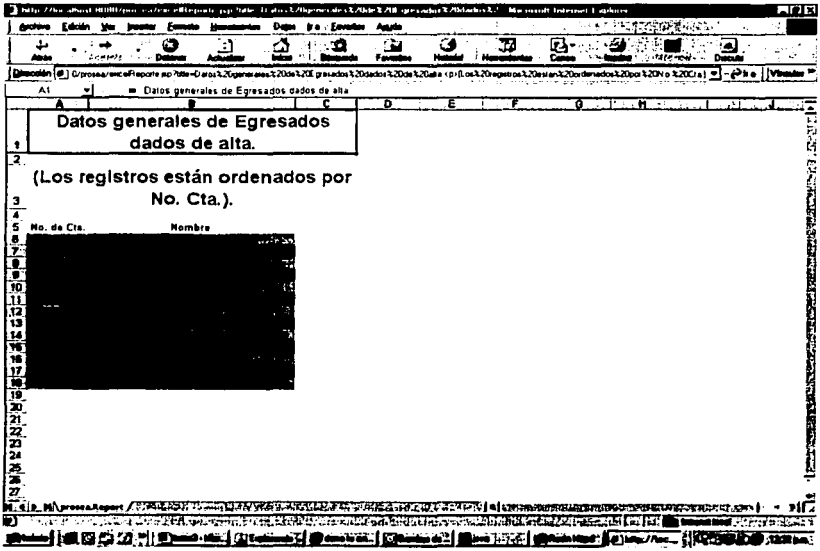
Si deseo solo algunos datos del reporte anterior, ¿cómo los puedo conseguir?

La segunda opción de reporte puede ser la solución; ya que a través de ella se obtiene una lista compuesta por los datos que usted mismo elige, que pueden ser: Número de Cuenta, Nombre, Dirección, Si está Titulado, Porcentaje de Créditos, Correo Electrónico o Teléfono; de todos los Egresados correspondientes a su carrera



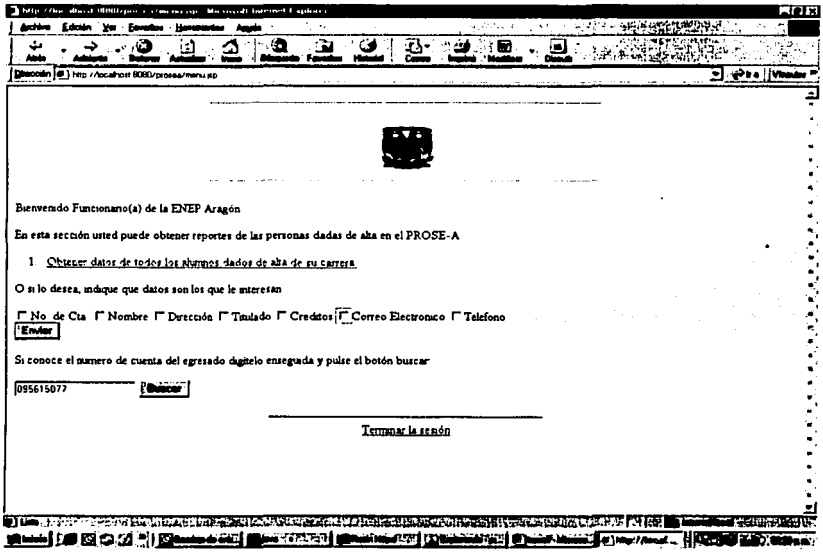
¿Y cómo obtengo ese reporte?

Para elegir los datos con los que se formará el reporte basta con marcarlos con un clic del mouse en su cuadrito correspondiente de la página anterior, y para generar el reporte basta con dar clic en el botón ENVIAR. Al instante se genera el reporte en formato Excel.



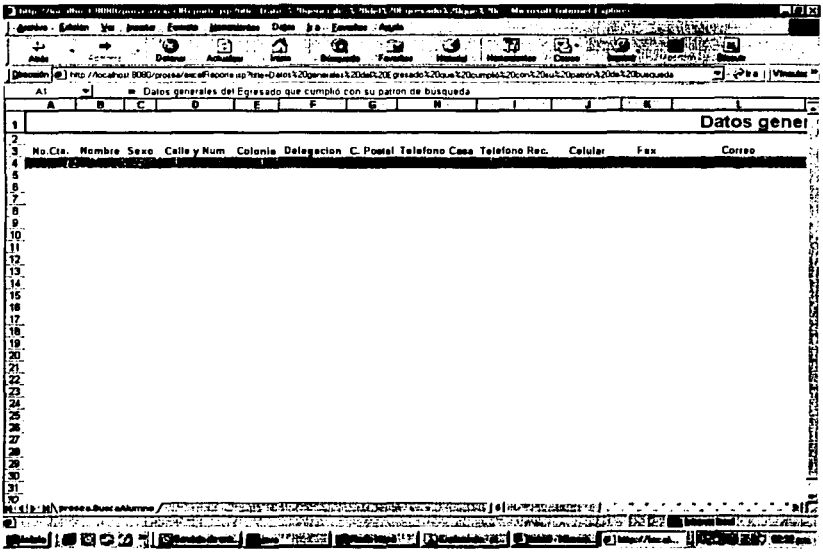
Conozco el número de cuenta del egresado, ¿puedo sacar un reporte solo de sus datos?

Si; el sistema puede ejecutar un reporte con los datos específicos de un solo registro basándose en el Número de Cuenta del Egresado. Ésta es la tercera opción de reporte con la que cuenta el sistema.

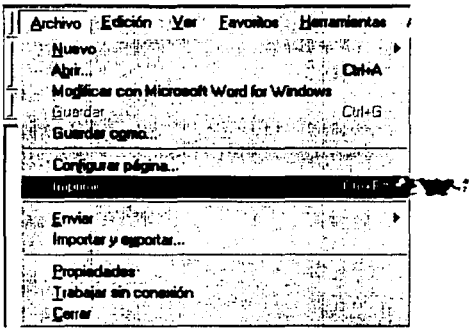


¿Y cómo obtengo este reporte?

Con solo teclear el número de cuenta correspondiente, en el formato de nueve dígitos sin guión, en el campo adjunto al botón BUSCAR y después dar clic en el botón antes mencionado, el sistema generará el reporte del registro individual, en formato Excel.



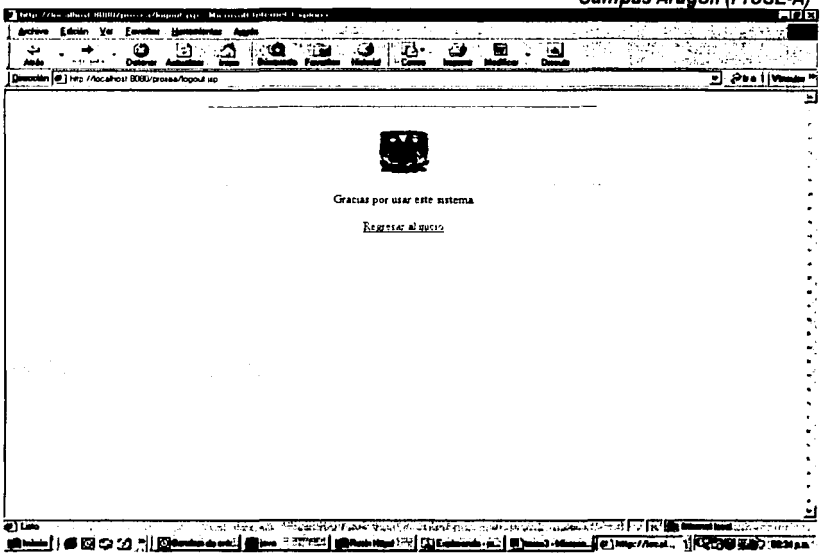
Para imprimir o guardar los reportes basta con seguir los pasos comunes para realizar esas tareas desde el navegador.



Ya ejecute mis reportes; ¿cómo salgo del sistema?

Para salir por completo del sistema debe regresar a la página de bienvenida, después de imprimir o guardar su reporte; para ello solo se utilizan las "flechas" de la barra de herramientas del navegador, con las que se va hacia "atrás" o hacia "adelante".

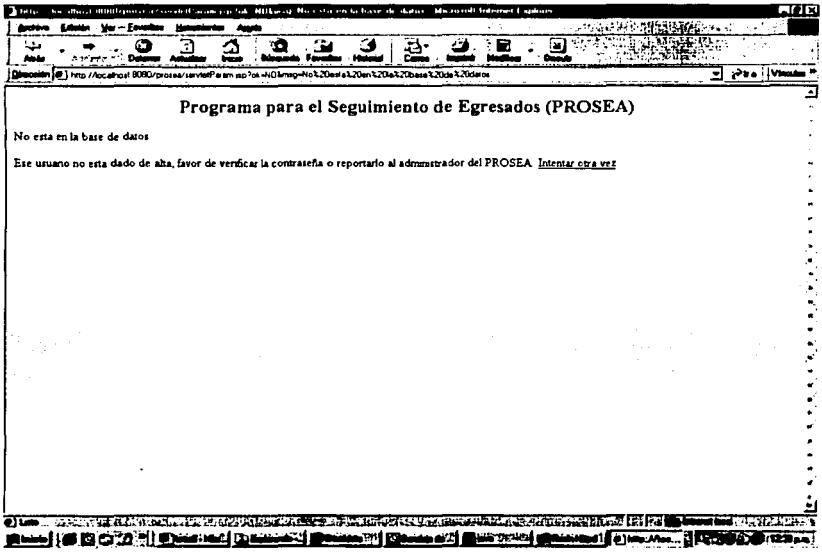
Ya estando en la página principal de donde podemos generar todos los reportes, solo resta dar clic en la "liga" TERMINAR LA SESIÓN. Inmediatamente después el sistema cierra su sesión y muestra una página en donde le agradece su participación, y le ofrece una "liga" a la página de inicio del sistema.



Al dar clic en la "liga" REGRESAR AL INICIO el sistema proporciona de nuevo la pantalla principal de inicio.

¿Cómo puedo saber si mi Clave de Usuario y Contraseña no son los correctos?

Si su Clave de usuario y/o su Contraseña tienen algún error, el sistema le mostrará una página en donde le informa que no encuentra su clave y le ofrece una "liga" a la página de inicio para volver a intentar teclearla.



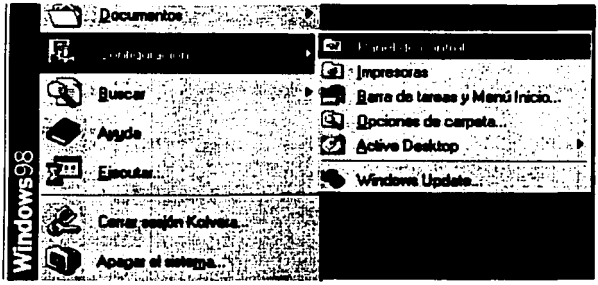
Manual Técnico

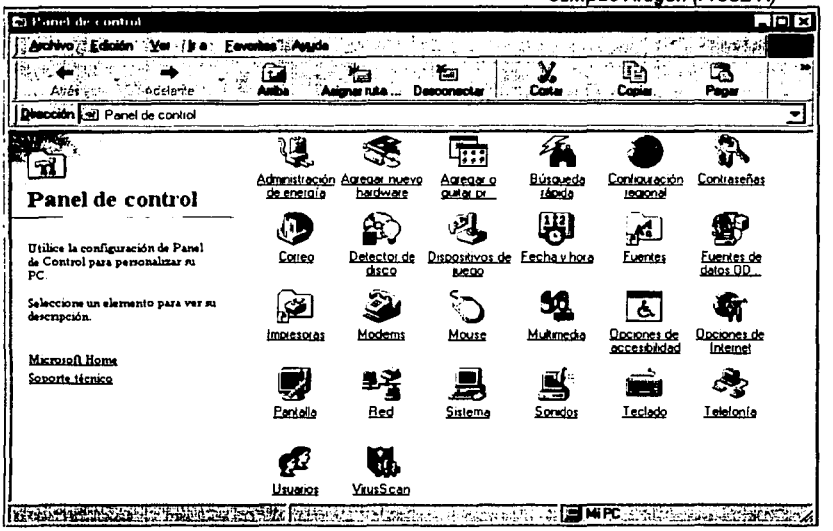
Instalando el Sistema de Control para el Prose-A...

Para el correcto funcionamiento del Sistema de Control para el Programa de Seguimiento a Egresados de la Escuela Nacional de Estudios Profesionales Aragón (ProSE-A) se debe configurar la localización de la base de datos e instalar el Servidor Web que pondrá en la red dicho sistema.

La base de datos que se utilizó para realizar las pruebas durante el desarrollo del sistema fue Access; la cual, en ambiente Windows 98 se debe configurar de la siguiente manera:

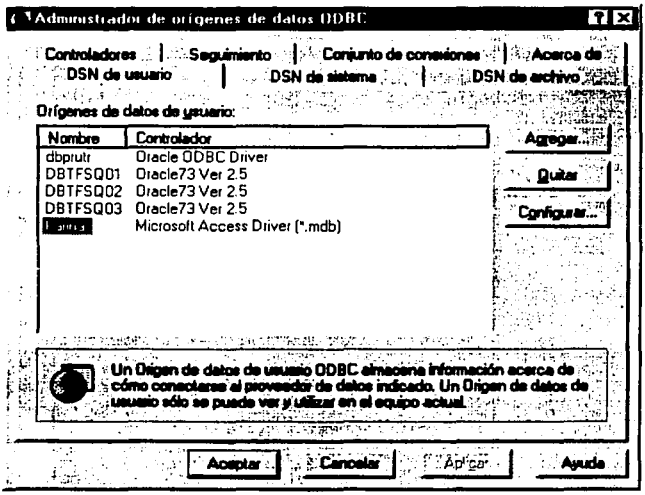
En la pantalla del Panel de Control, que se encuentra dentro de la opción de configuración:



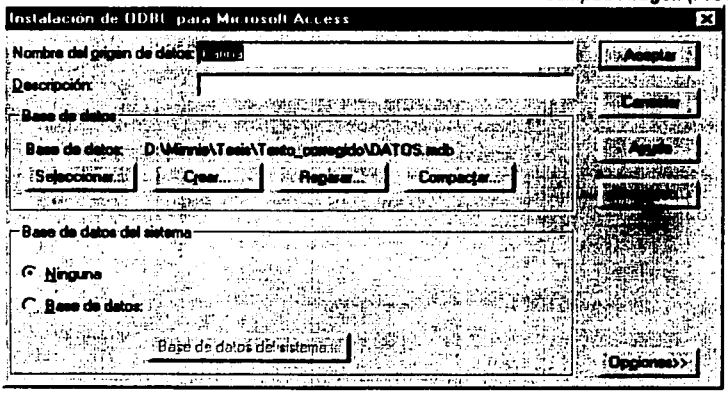


Dar clic en el icono de Fuentes de Datos ODBC:

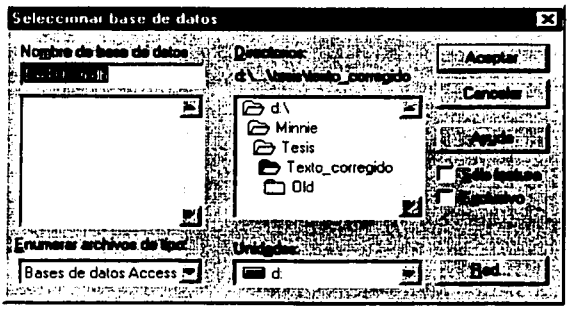
Con lo que aparecerá la siguiente pantalla en donde se configura la localización de la base de datos:



Se da clic en el botón "Agregar..." y en la siguiente pantalla, se coloca el nombre de la base de datos:



Para localizar la base de datos; se da clic en el botón "Seleccionar", de la pantalla anterior, para que se indique, en la pantalla siguiente, en donde se encuentra.



Así, al dar clic en el botón "Aceptar" de cada una de las pantallas, se habrá configurado la base datos para ser usada.

El servidor web que se utilizó para las pruebas en el desarrollo del sistema fue RESIN 2.1.0; de la empresa Caucho Technology, Inc.

El servidor web Resin es standalone (independiente), y comienza a escuchar peticiones de HTTP sobre el puerto 8080. El Resin puede ser usado para el desarrollo o la evaluación del software mismo.

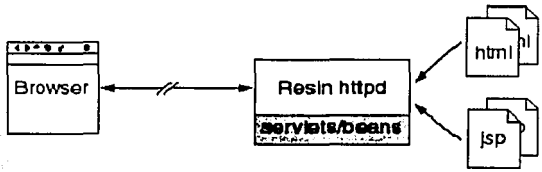
Los pasos para su uso son:

1. Instale JDK 1.2 o una versión posterior. Sobre Unix, ponga las variables del JAVA_HOME :
   SET CLASSPATH=.;C:\ Temporal;
   SET PATH=%PATH%;C:\jdk1.2.2\bin
2. Unzip resin-2.1.0.zip
3. Ponga los archivos JSP (y HTML) en el directorio c:/resin-2.1.0/doc/ como en el ejemplo: resin-2.1.0/doc/hello.jsp
4. Ponga los servlets (.java y .class) en el directorio c:/resin-2.1.0/doc/ WEB-INF/classes/ como en el ejemplo: resin-2.1.0/doc/WEB-INF/classes/test/HelloServlet.class
5. Ponga los archivos WAR en el directorio c:/resin-2.1.0/ webapps/ como en el ejemplo: resin-2.1.0/webapps/hello.war
6. Ejecute resin-2.1.0/bin/httpd.sh sobre Unix o resin-2.1.0/bin/httpd.exe sobre Win32 (dando doble clic sobre este archivo).
7. Coloque en su navegador la dirección: http: // localhost:8080
8. Cree las aplicaciones web directamente sobre Resin en resin-2.1.0/webapps/hello/hello.jsp, tal como en el ejemplo : resin-2.1.0/webapps/hello/hello.jsp
9. Si lo necesita, modifique la configuración en el archivo conf/resin.conf

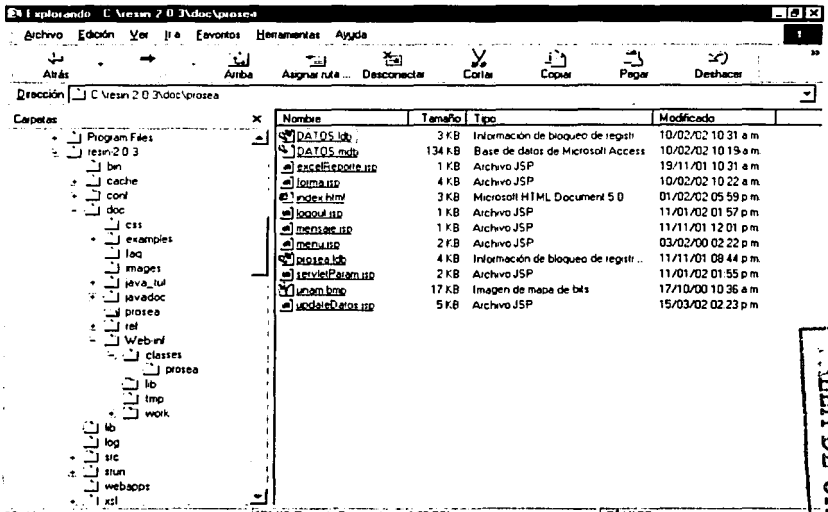
Antes de que usted esté listo de desplegar el servidor, estos son todos los pasos que se tuvieron que haber realizado para comenzar con Resin.

Sobre Unix, el Resin también puede ser configurado para correr con Netscape. La idea es la misma en cuanto al servidor Apache.

En la siguiente imagen se muestra la Configuración estándar de Resin.



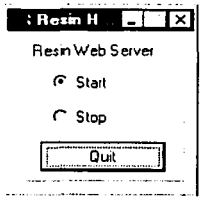
En la imagen siguiente se muestra la ubicación de los archivos del sistema.

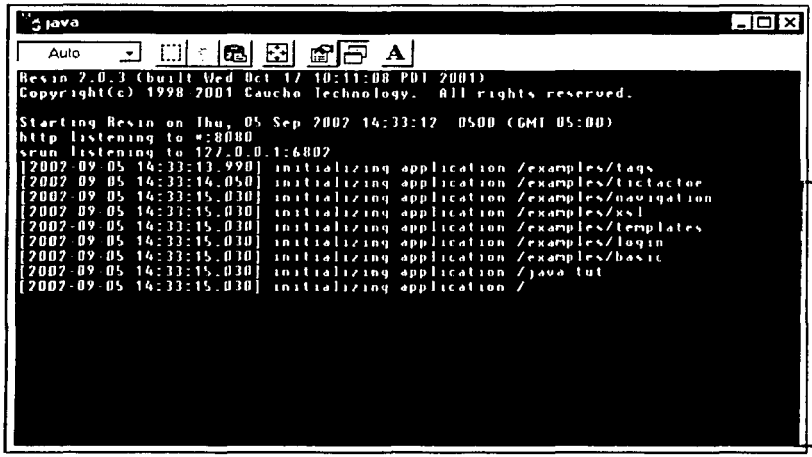


Pantalla Principal de Resin

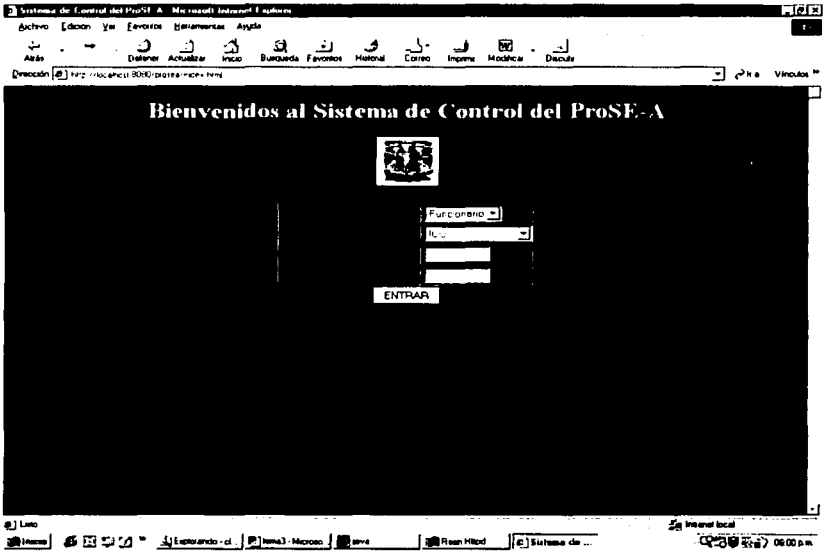
Acceso al archivo resin-2.1.0/bin/httpd.exe

Al ejecutar el archivo anterior Resin muestra las siguientes pantallas:





Finalmente se puede tener acceso al sistema, en el navegador, con la dirección web: http: //localhost:8080/prosea/index.html



BIBLIOGRAFÍA .-

1. **"Java, El lenguaje de Programación de Internet"**; Marcombo, Barcelona, España, 1996.

2. Au, Makower, WebWak; **"Java Programming Basics"**; Mis:Press; U.S.A; 1996.

3. Bartlett, Leslie, Simkin; **"Java Programming Explorer"**, Coriolis Group Books, U.S.A, 1998.

4. Booch Grady; **"Object-Oriented Analusis and Design With Applications"**; Addison-Wesley Object Technology Series.

5. Cary A. Jordan; **"Java 1.1 Certification Training Guide"**, New Riders, U.S.A., 1998.

6. Eriksson Hans-Erik, Penker Magnus; **"UML Toolkit"**.

7. Espeset, Tonny; **"Kick Ass Java Programming"**; Coriolis Group Books; U.S.A; 1996.

8. Fowler Martin, Scott Kendall (colaborador), Booch Grady; **"UML Distilled: Applying the Standard Object Modeling Language"**; Addison-Wesley Object Technology Series.

9. Hobbs, Ashton; **"Aprendiendo programación para bases de datos con JDBC en 21 días"**; Prentice Hall; Madrid España; 1998.

10. Jacobson, Ivar; **"Object-Oriented Software Engineering: A Use Case Driven Approach"**; Addison-Wesley Object Technology Series.

11. Jaworski Jamie; **"Java 1.2 Al Descubierto"**; Prentice Hall; Madrid España; 1999.

12. Larman, Craig; **"Applying UML and Patterns: An Introduction to Objetct-Oriented Analysis and Design"**.

13. Liang, Daniel; **"An Introduction to Java Programming"**, QUE Education&Trainning; U.S.A; 1998.

14. Morgan, Mike;**"Descubre Java 1.2"**; Prentice Hall Iberia; Madrid, 1999.

15. Reese, George; **"Database Programing with JDBC and Java"**, O'REILLY, U.S.A, 1997.

16. Rinehart, Martín; **"Desarrollo de Bases de Datos en Java"**; Mc Graw Hill, Madrid, 1998.

17. Rumbaugh James, Blaha Michael (colaborador), Premerlani William, Eddy Frederick, Lorensen Bill, Lorenson William (colaborador); **"Object-Oriented Modeling and Design"**.

18. Siddalingaiah, Lockwood; **"Java How-To, The Definitive Java Problem-Solver"**; Wait Group Press, U.S.A, 1996.

19. Walsh, Fronckowiak; **"Java Bible"**; IDG Books, U.S.A.