00623
22

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CONTADURÍA Y ADMINISTRACIÓN

SISTEMA DE BOLSA DE TRABAJO PARA UNA AGENCIA DE EMPLEO

DISEÑO DE UN SISTEMA PARA UNA ORGANIZACIÓN

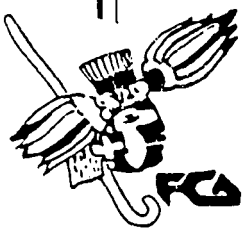
QUE PARA OBTENER EL TÍTULO DE:

LICENCIADO EN INFORMÁTICA

PRESENTAN:

OMAR ROJAS GÓMEZ
RUBEN ZULBARÁN JUÁREZ

ASESORA:
L.I. LUZ MARÍA RAMÍREZ ROMERO



MÉXICO, D.F. 2003

A

Agradecimientos:

A mis padres
Por el apoyo económico, moral y sentimental en todo mi desarrollo escolar.

A mis amigos
Por haber estado conmigo, por las burlas, por los chistes, por el apoyo.

A Rubén
Por aguantarme en el desarrollo de este trabajo, por demostrar que no nos diéramos por vencidos.

A Luz
Por su tiempo, ayuda, apoyo, por sus correcciones siempre atinadas, por recibirnos para las revisiones hasta en las noches, por soportarnos, hacernos reir.

Omar

B

Agradecimientos:

A mis padres:
Por su amor y apoyo que me brindaron para llevar a cabo con éxito esta tarea.

A mi familia:
Por haberme ofrendado cariño y paciencia.

A Omar:
Por haberme brindado su amistad y haberme permitido concluir con él una etapa importante en nuestras vidas.

A Luz:
Por haber sido una guía y un gran apoyo durante todo este proceso.

Rubén

ÍNDICE

INTRODUCCIÓN

El presente trabajo muestra todas las etapas del desarrollo de un sistema. Se presentan las actividades preliminares al análisis, como es la definición del objetivo general, alcances del sistema, los requerimientos del usuario, un estudio costo – beneficio y un estudio de factibilidad.

A continuación se realiza el análisis basado en la metodología Yourdon, en el cual se muestra el diagrama de contexto del sistema, los diagramas de flujo de datos a primer y segundo nivel, un diagrama entidad relación, un diccionario de datos y las miniespecificaciones correspondientes a los diagramas de flujo de datos.

Finalmente se detalla el diseño del sistema, donde se define la arquitectura a implementar, el diagrama de estructura, los diagramas de clases que se presentan como auxiliares en el diseño de la aplicación, el diseño de la base de datos, el diccionario de datos de la implementación del sistema, las interfaces de usuario y los reportes que generará el sistema.

Para mayor comprensión del presente trabajo se agregaron una serie de anexos que detallan la simbología utilizada en el análisis y diseño de la base de datos, los estereotipos utilizados en los diagramas de clases y un ejemplo de la implementación del proyecto.

PROPÓSITO DEL SISTEMA

Introducción

En este capítulo se especifica el Objetivo General del sistema de Bolsa de Trabajo, que es el motivo por el cual se desarrolla el sistema. También se definen los objetivos particulares que son las funciones específicas que el sistema debe cumplir, además se delimitará los alcances del sistema, qué *debe* y qué *no debe* hacer el sistema; a través de una lista de acontecimientos que indica las actividades que el sistema tiene que cumplir y una lista de exclusiones que delimitan el ámbito del sistema.

1.1. Objetivo General

Proponer un sistema de Bolsa de Trabajo capaz de clasificar y brindar información automáticamente a compañías oferentes de empleos, solicitantes de los mismos y al público en general.

1.2. Objetivos Particulares

- Permitir a las compañías oferentes de trabajo, colocar sus ofertas con sus especificaciones y revisar los curricula de candidatos de empleo según sus especificaciones.
- Permitir a los solicitantes colocar sus curricula, revisar las ofertas de trabajo de su rama de especialización, conocer los empleos mejor pagados, empresas que han requerido más candidatos e índices de demanda de carreras.
- Permitir al público en general revisar reportes estadísticos sobre las carreras mejor pagadas, más solicitadas, de mayor demanda de requerimientos, menos solicitadas y de mayor rapidez de colocación.

1.3. Alcances del Sistema

1.3.1. Lista de acontecimientos

1) Registro de las compañías oferentes de trabajo
   a) Colocar las ofertas de trabajo con los requisitos, carreras y sueldo propuesto.
   b) Permitir actualización de datos de las compañías y de las ofertas de trabajo.
   c) Acceder a los curricula de los solicitantes
   d) Dar de baja la oferta de trabajo ya satisfechas o expiradas.

2) Registro de los solicitantes.
   a) Colocar los curricula, habilidades disponibles y sueldo deseado.
   b) Actualizar sus datos, curricula y habilidades.
   c) Acceder a las ofertas de trabajo de su ramo.
   d) Permitir acceso a reportes sobre empleos mejor pagados, empresas que han requerido más candidatos e índices de demanda de carrera.

3) Acceso al público en general a reportes de carreras mas demandadas, carreras mejor pagadas, de mayor rapidez de colocación, menos solicitadas y de mayor demanda de requerimientos.

1.3.2. Exclusiones del sistema

1) No permitirá al público en general ver ofertas y demandas de trabajo sin estar previamente registrado.
2) No validará la veracidad de la información colocada por las compañías y los solicitantes.

2. ESTUDIO PRELIMINAR

Introducción

En este capítulo se presentan los requerimientos del cliente, que son las funciones que debe cumplir el sistema. También presenta un estudio de costo – beneficio que evalúa los insumos, su costo y las ventajas que aportará el sistema. Además presenta un estudio de factibilidad que nos muestra la situación actual de negocio, las necesidades a cubrir, las implicaciones de desarrollar el sistema, el escenario donde operará, los posibles riesgos que pueda tener y las características de los usuarios finales.

Por último determina las actividades a realizarse y se calcula el tiempo estimado para llevar a cabo cada una de ellas.

2.1. Requerimientos del Cliente

Registro de la Empresa

- El registro es gratuito
- Debe llenar el formulario con los datos que se le piden
- Debe proporcionar obligatoriamente un nombre de usuario y una contraseña
- Después del registro tendrá acceso a los servicios del sistema

Servicios a las empresas registradas

- Publicación de ofertas de empleo durante un tiempo determinado por la empresa.
- Búsqueda de curricula de los usuarios registrados
- Modificar los datos de su empresa
- Modificar los datos de las ofertas de empleo publicadas
- Dar de baja su cuenta en el sistema

Registro de curriculum vitae

- El registro es gratuito
- Debe llenar el formulario con los datos que se le piden
- Debe proporcionar obligatoriamente un nombre de usuario y una contraseña
- Después del registro tendrá acceso a los servicios del sistema

Servicios a las personas que registraron su curriculum vitae

- Acceso a las ofertas de empleo publicadas por las empresas de su ramo profesional
- Modificar los datos de su curriculum vitae
- Dar de baja su curriculo vitae del sistema

Servicios al público en general

- Acceso al público en general a reportes de carreras más demandadas, carreras mejor pagadas, de mayor rapidez de colocación, menos solicitadas y de mayor demanda de requerimientos.

2.2. Estudio Costo-Beneficio

Operaciones y costos.

Operación	Tiempo	Personal	Costo
Análisis	6 semanas	2 analistas	$160,000.00
Diseño	1 semana	2 diseñadores DB	$40,000.00
	2 semanas	2 Diseñadores de pantallas y salidas	$40,000.00
	2 semanas	2 Diseñadores de Arquitectura	$60,000.00
Programación		2 Programadores	
Modulo de conexión a BD	1 semana	1 Programador	$15,000.00
Módulo de captura de Empresas	1 semana	1 Programador	$7,000.00
Módulo de captura de Solicitantes	1 semana	1 Programador	$7,000.00
Modulo de subir archivos	1 semana	1 Programador	$7,000.00
Modulo de modificación de Empresas	1 semana	1 Programador	$5,000.00
Módulo de modificación de Solicitantes	1 semana	1 Programador	$5,000.00
Modulo de generación de ofertas	1 semana	1 Programador	$6,000.00
Módulo de modificación de ofertas	1 semana	1 Programador	$5,000.00
Módulo de generación de reportes	1 semana	1 Programador	$13,000.00
Pruebas	2 semanas	3 Expertos	$30,000.00
		Total	$400,000.00

La base de datos, el sistema operativo y el lenguaje de programación son libres y no tienen costo.
La empresa cuenta con una computadora Pentium IV, 1.5 Ghz, 256 MB en RAM y 40 GB de disco duro y administrador de sistemas.

Este sistema ofrece competitivas ventajas, que superan por un amplio margen su costo inicial.

- Ahorro de gastos de publicación de ofertas de trabajo[1].
- Ahorro de gastos de búsqueda de candidatos.
- Ahorro de tiempo en publicación de ofertas de trabajo.
- Ahorro de tiempo en búsqueda de candidatos idóneos.
- Ahorro de tiempo en contacto con candidatos elegidos.
- Ahorro de tiempo en la búsqueda de ofertas.
- Ahorro de tiempo en la presentación de los curricula y habilidades disponibles.

[1] El costo de una publicación en un periódico es de $481 por centímetro de columna más IVA y es por sólo 15 días. Fuente: Instituto Federal Electoral; Catálogo 2003 de Tarifas de Medios.

El sistema tiene grandes posibilidades de llevarse a cabo, ya que le permitirá a la empresa un ahorro considerable en gastos de operación (tanto materiales como humanos) y un inestimable ahorro de tiempo en la selección de personal.

2.3. Estudio de Factibilidad

2.3.1. Motivación del Negocio

1) Situación Actual
   a) Dependencia de la publicación en periódicos
   b) Información disponible limitada
   c) Costo elevado
   d) Comunicación de los interesados unidireccional
   e) Lentitud

2) Necesidades
   a) Independencia respecto a los periódicos
   b) Mayor disponibilidad de información
   c) Intercomunicación entre los interesados
   d) Bajo costo
   e) Rapidez

3) Implicaciones
   a) Trabajo de desarrollo y mantenimiento del sistema.
   b) Necesidad de inversión al desarrollo y mantenimiento del sistema.
   c) Recursos dedicados a la operación del sistema.

4) Situación óptima
   a) Facilidad de uso
   b) Disponibilidad de información
   c) Rapidez

2.3.2. Perfil del Cliente

Tipo de Usuario	Expectativas	Metas y Prioridades	Requerimientos	Restricciones	Disponibilidad al cambio
Empresas	Conocer a los candidatos Rapidez Automatización Información para tomar la mejor opción Amplio mercado	Ahorro de tiempo Disminución del costo Amplitud de opciones	Automatización en la obtención de la información. Disponibilidad Facilidad de uso	Desconocimiento del sistema	Si
Solicitantes	Rapidez Ampliar el horizonte de búsqueda Conocer ofertas	Ahorro de tiempo Disminución del gasto	Disponibilidad Facilidad de uso	Desconocimiento del sistema	Si
Público en general	Evaluar y considerar las opciones laborales	Obtener información	Facilidad de uso Disponibilidad	Desconocimiento del sistema	Si
Administrador del sistema	Fácil manejo y mantenimiento	Mantener la integridad de la información	Disponibilidad de acceso Mantenimiento transparente	Desconocimiento del sistema	Si

2.3.3. Factores críticos de Éxito

1) Rapidez para el acceso de información
   Gracias a que el Sistema Bolsa de Trabajo será capaz de clasificar y brindar información automáticamente a compañías oferentes de empleos, solicitantes de los mismos y al público en general, los resultados del sistema serán de rápido acceso

2) Disponibilidad del sistema
   El Sistema de Bolsa de Trabajo podrá ser accedido desde cualquier computadora con acceso a Internet, a cualquier hora, sin distinción de personas

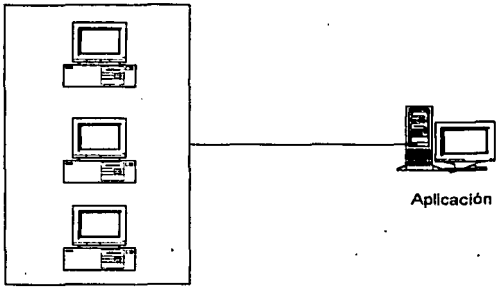
3) Facilidad de uso
   El Sistema de Bolsa de Trabajo puede ser utilizado por cualquier persona con conocimientos básicos de computación y de Internet

4) Amplitud del mercado
   Como será implantado en Internet el mercado potencial es amplio, además de que no tiene costo por su uso.

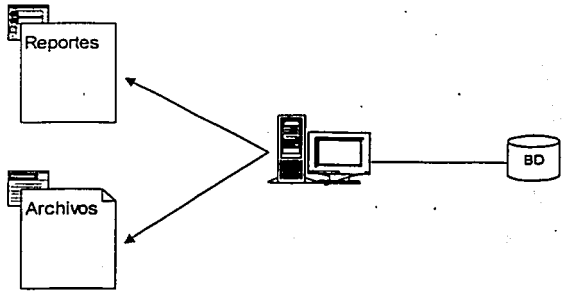
5) Competencia
   Dado que ya existen otras bolsas de trabajo en Internet, este sistema pretende brindar servicios agregados a los usuarios más jóvenes, que les puedan ayudar a elegir un área de desarrollo de acuerdo a sus habilidades y gustos personales.

2.3.4. Escenario Operacional

1) Instalación del sistema
   a) Aplicación web
   b) Público en general



2) Datos almacenados
   a) Base de datos
   b) Archivos
   c) Generación automatizada de reportes



3) Manejo por parte de los usuarios
   a) Tipo de usuario
   b) Acceso vía Internet
   c) Captura mediante formularios

2.3.5. Matriz de Riesgos

Riesgo	Solución	Responsable
Pérdida de datos	Respaldo de la información contenida en la Base de Datos	Administrador del sistema
Errores de captura	Módulo de validación de entradas	Desarrolladores
Hacking	Implantación de un Firewall	Administrador del sistema

Dado lo anterior, el sistema no presenta dificultades o inconvenientes para su desarrollo, debido a que hay una necesidad latente del sistema, usuarios capaces de usar el mismo, un escenario operativo favorable y una matriz de riesgo que se puede controlar.

2.3.6. Actividades

Las actividades a realizar para llevar a cabo el desarrollo del sistema requerido son:

Análisis

- Estudio Preliminar
- Análisis

Diseño

- Diseño de arquitectura
- Diagramas de clases
- Diseño de la BD
- Diseño de pantallas y salidas

Implementación

- Modulo de conexión a BD
- Módulo de captura de Empresas
- Módulo de captura de Solicitantes
- Modulo de subir archivos
- Modulo de modificación de Empresas
- Módulo de modificación de Solicitantes
- Modulo de generación de ofertas
- Módulo de modificación de ofertas
- Módulo de generación de reportes

Pruebas y liberación

- Pruebas
- Instalación y liberación

Estas actividades quedan planteadas en el diagrama de Gantt siguiente.

2.3.7. Diagrama de Gantt

Actividades	Tiempo (Semanas)																					
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
Estudio Preliminar	■	■																				
Análisis			■	■	■	■																
Diseño de arquitectura							■															
Diagramas de clases								■														
Diseño de la BD									■													
Diseño de pantallas y salidas										■	■											
Modulo de conexión a BD												■										
Módulo de captura de Empresas													■									
Módulo de captura de Solicitantes														■								
Modulo de subir archivos												■										
Modulo de modificación de Empresas													■									
Módulo de modificación de Solicitantes														■								
Modulo de generación de ofertas														■								
Módulo de modificación de ofertas															■							
Módulo de generación de reportes															■	■	■					
Pruebas																		■	■			
Instalación y liberación																		■	■			
	Feb.		Mar				Abril				May				Jun				Jul			

Fecha de Inicio: 18 febrero 2003
Fecha de termino: 5 Julio 2003

3. ANÁLISIS

Introducción

En este capítulo se detalla cual es el problema y lo que se necesita para resolverlo, para llevarlo a cabo se presenta el diagrama de contexto que muestra la relación del sistema con las entidades con las que interactúa, indicando los datos que recibe y los que genera al exterior.

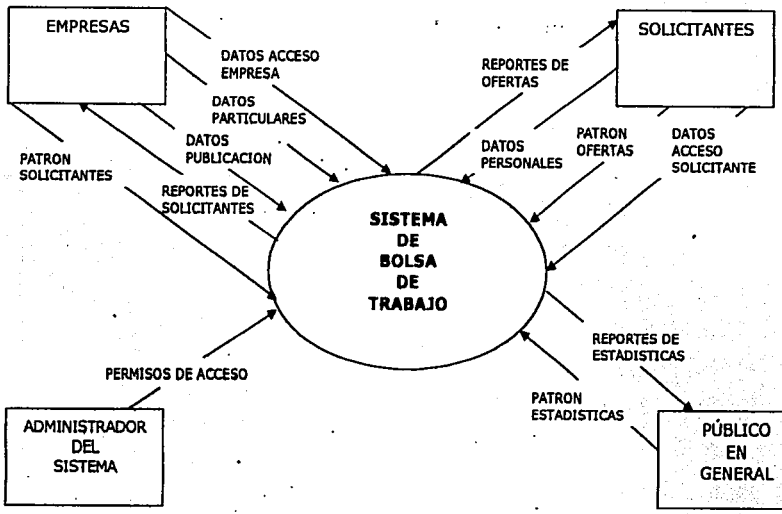
A continuación se presenta una serie de diagramas de flujo de datos, que definen el movimiento interno de los datos que el sistema recibe y además define la modularidad del mismo.

Además presenta un diagrama Entidad – Relación que modela las entidades principales que componen el sistema y su interrelación.

Después se muestra el diccionario de datos, que contiene la definición de los datos manejados en los diagramas de flujo.

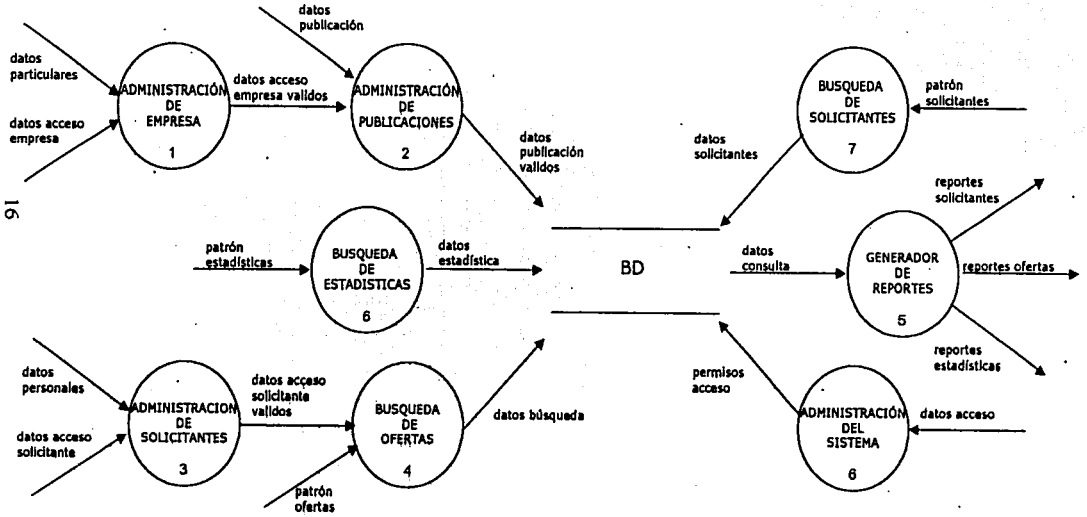
Finalmente están las miniespecificaciones, que son los algoritmos de cada módulo del diagrama de flujo de datos.

3.1. Relación del sistema con su entorno (Diagrama de Contexto)

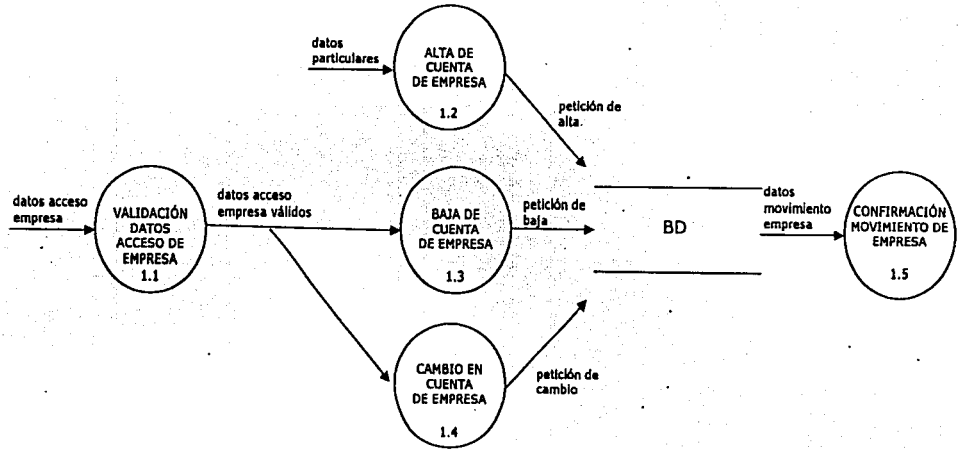


3.2. Diagramas de Flujo de Datos

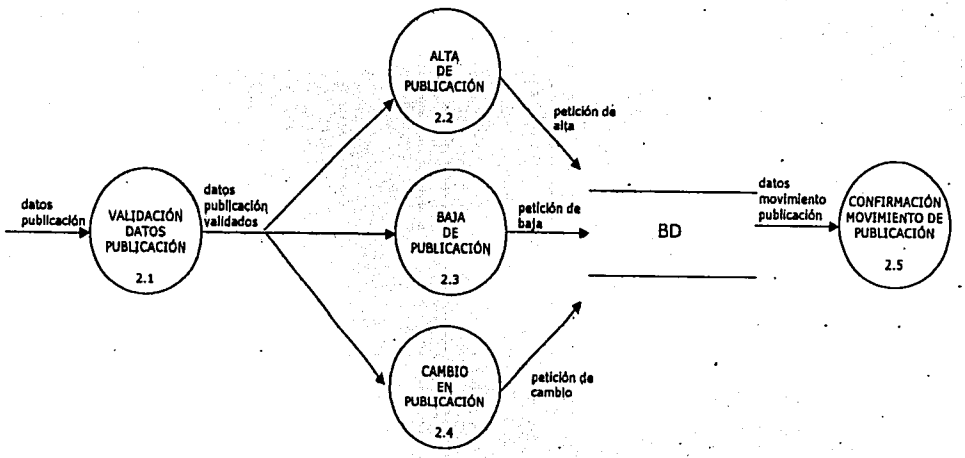
DFD 0 SISTEMA DE BOLSA DE TRABAJO



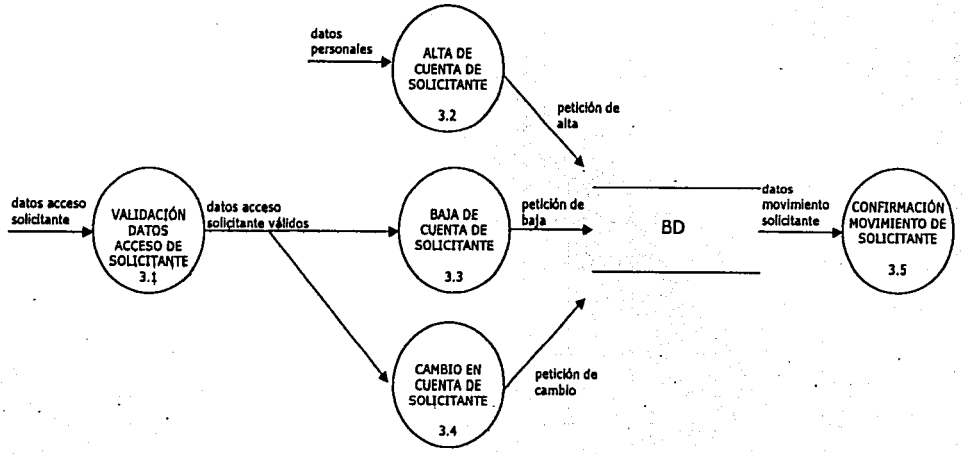
DFD 1 ADMINISTRACIÓN DE EMPRESA



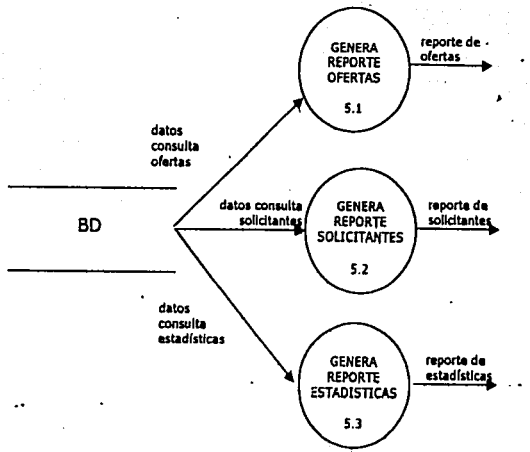
DFD 2 ADMINISTRACIÓN DE PUBLICACIONES



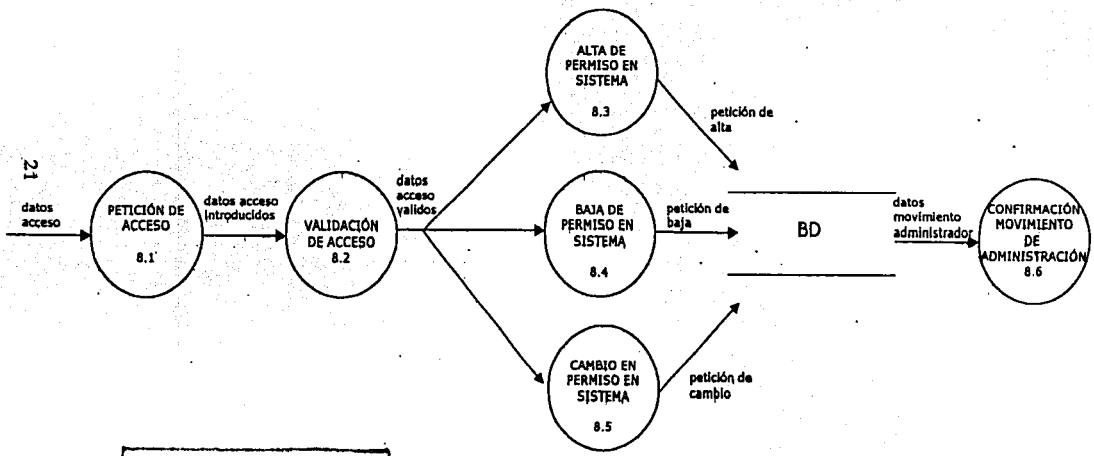
DFD 3 ADMINISTRACIÓN DE SOLICITANTE



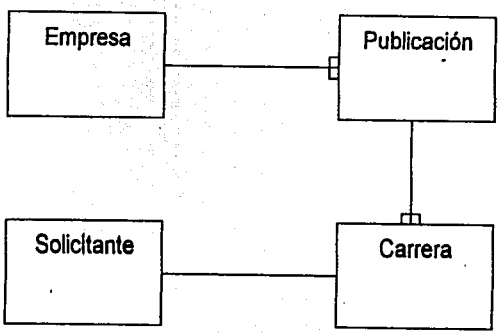
DFD 5 GENERADOR DE REPORTES



DFD 8 MODULO DE ADMINISTRACIÓN



3.3. Diagrama Entidad – Relación



3.4. Diccionario de Datos

Datos de acceso empresa		{login | passwd}	
login	#*	Clave única del usuario	varchar(8)
passwd	*	Clave de acceso del usuario	Varchar(10)

Datos particulares		{denominacion social | @RFC | e_mail | telefono | direccion}	
enominacion social	*	Nombre o denominación social del inscrito en el sistema	Varchar(50)
RFC	#*	Registro Federal de Contribuyentes	Varchar(13)
e_mail	*	Correo electrónico	Varchar(30)
telefono	*	Teléfono del inscrito en el sistema	Varchar(15)
direccion	*	Domicilio del inscrito en el sistema	Varchar(60)

Datos publicación		{@RFC | clave | carrera solicitada | puesto | conocimientos | sueldo | fecha_publicacion}	
RFC	#* FK	Registro Federal de Contribuyentes	Varchar(13)
clave	#*	Identificador en el sistema	Number
carrera_solicitada	*	Nombre de la carrera solicitada	Varchar(25)
puesto	*	Nombre del puesto solicitado	Varchar(25)
conocimientos	*	Habilidades del solicitante	Varchar(40)
sueldo	*	Monto que se ofrece al solicitante	Varchar(10)
fecha_publicacion	*	Fecha en que se introduce en el sistema la publicación	date

Datos personales		{nombre | @RFC | e_mail | telefono | direccion | sueldo | curriculum | carrera}	
nombre	*	Nombre del inscrito en el sistema	Varchar(50)
RFC	#*	Registro Federal de Contribuyentes	Varchar(13)
e_mail	*	Correo electrónico del inscrito en el sistema	Varchar(30)
telefono	*	Teléfono del inscrito en el sistema	Varchar(15)
direccion	*	Domicilio del inscrito en el sistema	Varchar(60)
sueldo	*	Monto que solicita el solicitante	Varchar(10)
curriculum	*	Documento con la información de las habilidades del solicitante	Varchar(3000)
conocimientos	*	Habilidades del solicitante	Varchar(40)
carrera	*	Nombre de la carrera solicitada	Varchar(25)

Patrón de solicitantes		{carrera | conocimientos}	
carrera	O	Nombre de la carrera solicitada	Varchar(25)
conocimientos	O	Habilidades del solicitante	Varchar(40)

Patrón de ofertas		{carrera_solicitada | denominacion social }	
carrera_solicitada	O	Nombre de la carrera solicitada	Varchar(25)
Conocimientos	O	Habilidades del solicitante	Varchar(40)
Nombre	O	Nombre o denominación social del inscrito en el sistema	Varchar(50)

Patrón de estadísticas		{carrera_solicitada}	
carrera_solicitada	O	Nombre de la carrera solicitada	Varchar(25)

Datos consulta solicitantes		{nombre | e_mail | carrera_solicitada | telefono}	
nombre	*	Nombre del inscrito en el sistema	Varchar(50)
e_mail	*	Correo electrónico del inscrito en el sistema	Varchar(30)
carrera_solicitada	*	Nombre de la carrera solicitada	Varchar(25)
telefono	*	Teléfono del inscrito en el sistema	Varchar(15)

Datos consulta ofertas		{denominacion social | e_mail | telefono | puesto | conocimientos | sueldo}	
denominación social	*	Nombre o denominación social del inscrito en el sistema	Varchar(50)
e_mail	*	Correo electrónico	Varchar(30)
telefono	*	Teléfono del inscrito en el sistema	Varchar(15)
puesto	*	Nombre del puesto solicitado	Varchar(25)
conocimientos	*	Habilidades del solicitante	Varchar(40)
sueldo	*	Monto que se ofrece al solicitante	Varchar(10)

Datos consulta estadísticas		{carrera_solicitada | sueldo | demanda}	
carrera_solicitada	*	Nombre de la carrera solicitada	Varchar(25)
sueldo	*	Monto que se ofrece al solicitante	Varchar(10)
demanda	*	Porcentaje determinado según la búsqueda, la carrera, etc.	Number

3.5. Miniespeficaciones del Sistema de Bolsa de Trabajo

Proceso 1 Administración de Empresa

1.1 Validación Datos Acceso de Empresa

```
Inicio
   Leer (login,password);
   Conectar BD;
   Verificar (login, password);
   Si ( validos (login,password) )
        Redireccionar(Movimiento de Empresa)
   De lo contrario
        Salir
   Fin Si;
   Cerrar BD;
Fin
```

1.2 Alta de Cuenta de Empresa

```
Inicio
   Leer (Datos Empresa);
   Validar(Datos Empresa);
    Si Datos Empresa == validos
        Conectar BD;
        Insertar(Datos Empresa);
        Desconectar BD;
    De lo contrario
        Imprimir ("Sus datos son incorrectos");
    Fin si
Fin
```

1.3 Baja de Cuenta de Empresa

```
Inicio
   Conectar BD;
   Buscar (Datos Empresa);
   Presentar(Datos Empresa);
    Si borrar es verdadero
        Borrar(Datos Empresa);
    De lo contrario
        Imprimir ("No se eliminaron sus datos");
    Fin si
    Desconectar BD;
Fin
```

1.4 Cambio de Cuenta de Empresa

```
Inicio
   Conectar BD;
   Buscar (Datos Empresa);
   Presentar(Datos Empresa);
   Leer (Datos Empresa);
   Validar(Datos Empresa);
   Si Datos Empresa == validos
        Actualiza(Datos Empresa);
   De lo contrario
        Imprimir ("Sus datos son incorrectos");
   Fin si
   Desconectar BD;
Fin
```

1.5 Confirmación de Movimiento de Empresa

```
Inicio
   Presentar(Datos Empresa);
Fin
```

Proceso 2 Administración de Publicaciones

2.1 Validación Datos de Publicación

```
Inicio
   Leer (Datos Publicación);
   Conectar BD;
   Verificar (Datos Publicación);
   Si ( validos (Datos Publicacion) )
        Redireccionar(Movimiento de Publicacion)
   De lo contrario
        Salir
   Fin Si;
   Cerrar BD;
Fin
```

2.2 Alta de Publicación

```
Inicio
   Leer (Datos Publicación);
   Validar(Datos Publicación);
   Si Datos Publicación == validos
        Conectar BD;
        Insertar(Datos Publicación);
        Desconectar BD;
   De lo contrario
        Imprimir ("Sus datos son incorrectos");
   Fin si
Fin
```

2.3 Baja de Publicación

```
Inicio
   Conectar BD;
   Buscar (Datos Publicación);
   Presentar(Datos Publicación);
    Si borrar es verdadero
         Borrar(Datos Publicacion);
    De lo contrario
        Imprimir ("No se eliminaron sus datos");
    Fin si
    Desconectar BD;
Fin
```

2.4 Cambio de Publicación

```
Inicio
   Conectar BD;
   Buscar (Datos Publicación);
   Presentar(Datos Publicación);
    Leer (Datos Publicación);
   Validar(Datos Publicación);
    Si Datos Publicación == validos
       Actualiza(Datos Publicación);
    De lo contrario
        Imprimir ("Sus datos son incorrectos");
    Fin si
    Desconectar BD;

Fin
```

2.5 Confirmación de Movimiento de Publicación

```
Inicio
   Presenta(Datos Publicación);
Fin
```

Proceso 3 Administración de Solicitante

3.1 Validación de Datos de Acceso de Solicitante

```
Inicio
   Leer (login,password);
   Conectar BD;
   Verificar (login, password);
   Si ( validos (login,password) )
        Redireccionar(Movimiento de Solicitante)
   De lo contrario
         Salir
   Fin Si;
   Cerrar BD;
Fin
```

3.2 Alta de Cuenta de Solicitante

```
Inicio
  Leer (Datos Solicitante);
  Validar(Datos Solicitante);
   Si Datos Solicitante == validos
       Conectar BD;
       Insertar(Datos Solicitante);
       Desconectar BD;
   De lo contrario
       Imprimir ("Sus datos son incorrectos");
   Fin si
Fin
```

3.3 Baja de Cuenta de Solicitante

```
Inicio
   Conectar BD;
   Buscar (Datos Solicitante);
   Presentar(Datos Solicitante);
   Si borrar es verdadero
        Borrar(Datos Solicitante);
   De lo contrario
       Imprimir ("No se eliminaron sus datos");
   Fin si
   Desconectar BD;
Fin
```

3.4 Cambio de Cuenta de Solicitante

```
Inicio
   Conectar BD;
   Buscar (Datos Solicitante);
   Presentar(Datos Solicitante);
   Leer (Datos Solicitante);
   Validar(Datos Solicitante);
   Si Datos Solicitante == validos
       Actualizar(Datos Solicitante);
   De lo contrario
       Imprimir ("Sus datos son incorrectos");
   Fin si
   Desconectar BD;
Fin
```

3.5 Confirmación de Movimiento de Solicitante

```
Inicio
  Presentar(Datos Solicitante);
Fin
```

Proceso 4 Búsqueda de Ofertas

Inicio
Lee(Patrón búsqueda Ofertas);
Conectar BD;
Busca (Patrón búsqueda Ofertas);
Desconectar BD;
Llama(Generador de Reportes);
Fin

Proceso 5 Generador de Reportes

5.1 Genera Reporte de Ofertas

Inicio
Recibe (Patrón de búsqueda de ofertas);
Genera(Reporte de ofertas);
Imprime (Reporte de ofertas);
Fin

5.2 Genera Reporte de Solicitantes

Inicio
Recibe (Patrón de búsqueda de solicitantes);
Genera(Reporte de solicitantes);
Imprime (Reporte de solicitantes);
Fin

5.3 Genera Reporte de Estadística

Inicio
Recibe (Patrón de búsqueda de estadísticas);
Genera(Reporte de estadísticas);
Imprime (Reporte de estadísticas);
Fin

Proceso 6 Búsqueda de Solicitantes

Inicio
Lee(Patrón búsqueda Solicitantes);
Conectar BD;
Busca (Patrón búsqueda Solicitantes);
Desconectar BD;
Llama(Generador de Reportes);
Fin

Proceso 7 Búsqueda de Estadísticas

```
Inicio
   Lee(Patrón búsqueda Estadísticas);
   Conectar BD;
   Busca (Patrón búsqueda Estadísticas);
   Desconectar BD;
   Llama(Generador de Reportes);
Fin
```

Proceso 8 Administración del Sistema

8.1 Petición de Acceso

```
Inicio
   Leer (login,password);
   Llama(Validación de Acceso);
Fin
```

8.2 Validación de Acceso

```
Inicio
   Conectar BD;
   Verificar (login, password);
   Si ( validos (login,password) )
        Redireccionar(Movimiento de Administración)
   De lo contrario
        Salir
   Fin Si;
   Cerrar BD;
Fin
```

8.3 Alta de Permiso de Sistema

```
Inicio
   Leer (Datos [Empresa][Oferta][Solicitante] );
   Validar(Datos [Empresa][Oferta][Solicitante]);
   Si Datos == validos
        Conectar BD;
        Insertar(Datos [Empresa][Oferta][Solicitante]);
        Desconectar BD;
   De lo contrario
        Imprimir ("Sus datos son incorrectos");
   Fin si
Fin
```

8.4 Baja de Permiso de Sistema

```
Inicio
   Conectar BD;
   Buscar (Datos [Empresa][Oferta][Solicitante]);
   Presentar(Datos [Empresa][Oferta][Solicitante]);
    Si borrar es verdadero
         Borrar(Datos [Empresa][Oferta][Solicitante]);
    De lo contrario
        Imprimir ("No se eliminaron sus datos");
    Fin si
    Desconectar BD;
Fin
```

8.5 Cambio de Permiso de Sistema

```
Inicio
   Conectar BD;
   Buscar (Datos [Empresa][Oferta][Solicitante]);
   Presentar(Datos [Empresa][Oferta][Solicitante]);
    Leer (Datos [Empresa][Oferta][Solicitante]);
   Validar(Datos [Empresa][Oferta][Solicitante]);
    Si Datos [Empresa][Oferta][Solicitante] == validos
        Actualiza(Datos [Empresa][Oferta][Solicitante]);
    De lo contrario
        Imprimir ("Sus datos son incorrectos");
    Fin si
    Desconectar BD;
Fin
```

8.6 Confirmación de Movimiento de Administración

```
Inicio
  Presentar(Datos[Empresa][Oferta][Solicitante]);
Fin
```

4. DISEÑO

Introducción

En este capítulo se define la arquitectura a través de un diagrama que presenta los usuarios, su descripción y la forma en que se relacionan con el sistema.

A continuación se presenta un diagrama de clases con el que se diseña todo el funcionamiento del sitio, éste está basado en los estereotipos estándares de la extensión de UML para diseño Web. (ver Anexo II).

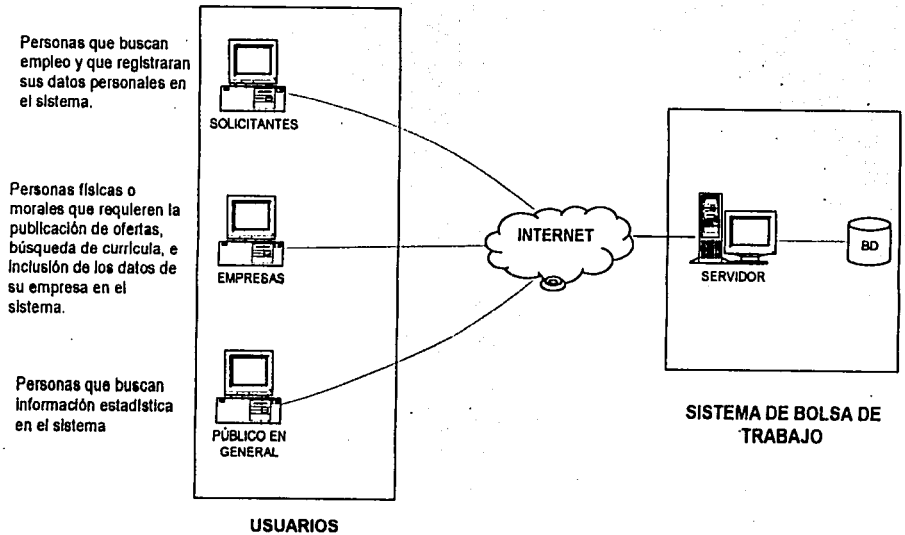
Después muestra un diagrama de estructura que define todos los módulos jerarquizados en los cuales se divide el sistema, así como los datos que devuelven a su módulo superior.

También se diseña la base de datos a través de un diagrama entidad – relación específico donde los atributos de cada entidad y sus relaciones. Para especificar este diagrama hay un diccionario de datos de la implementación del sistema, el cual indica todos los atributos de las entidades y sus características.

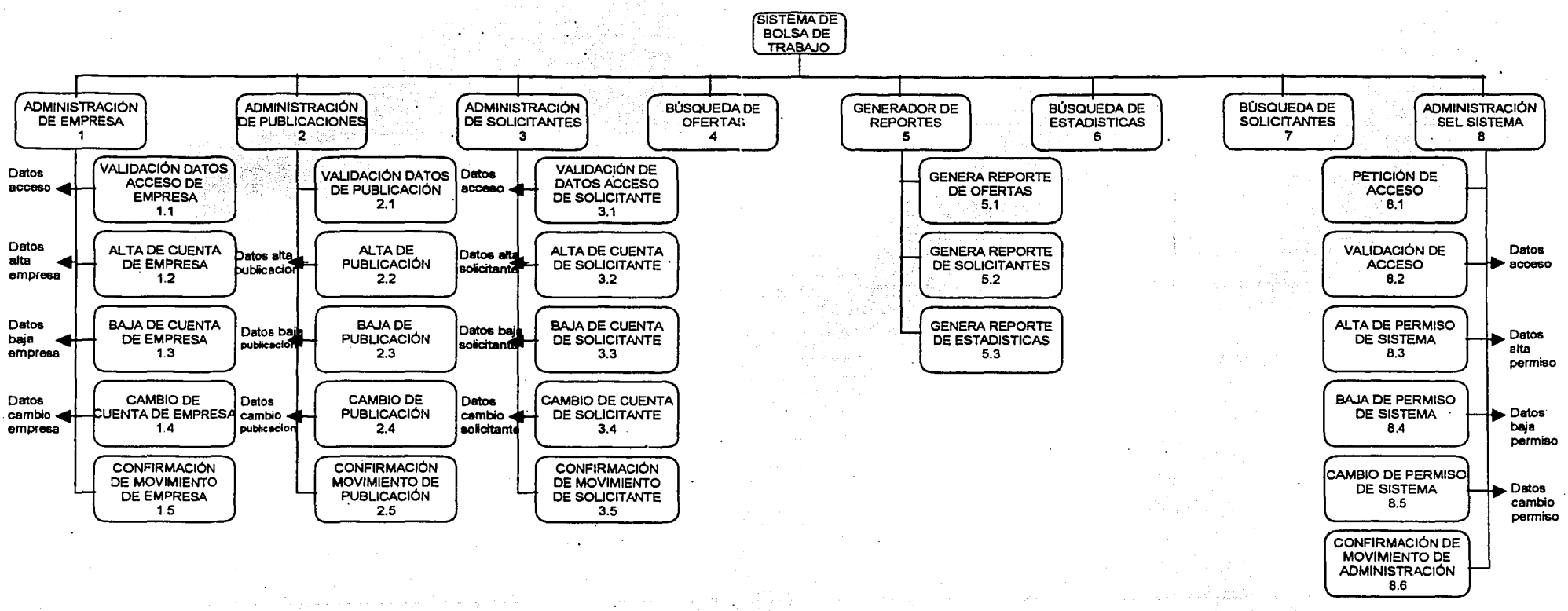
Finalmente se muestran las pantallas y reportes con los cuales el usuario interactuará con el sistema.

(En el anexo 3 se muestra un ejemplo del resultado de la implementación del sistema).

4.1. Definición de la Arquitectura



4.2. Diagrama de Estructura



4.3. Diagrama de Clases, como extensión del diagrama de Estructura para la utilización del Lenguaje Java.

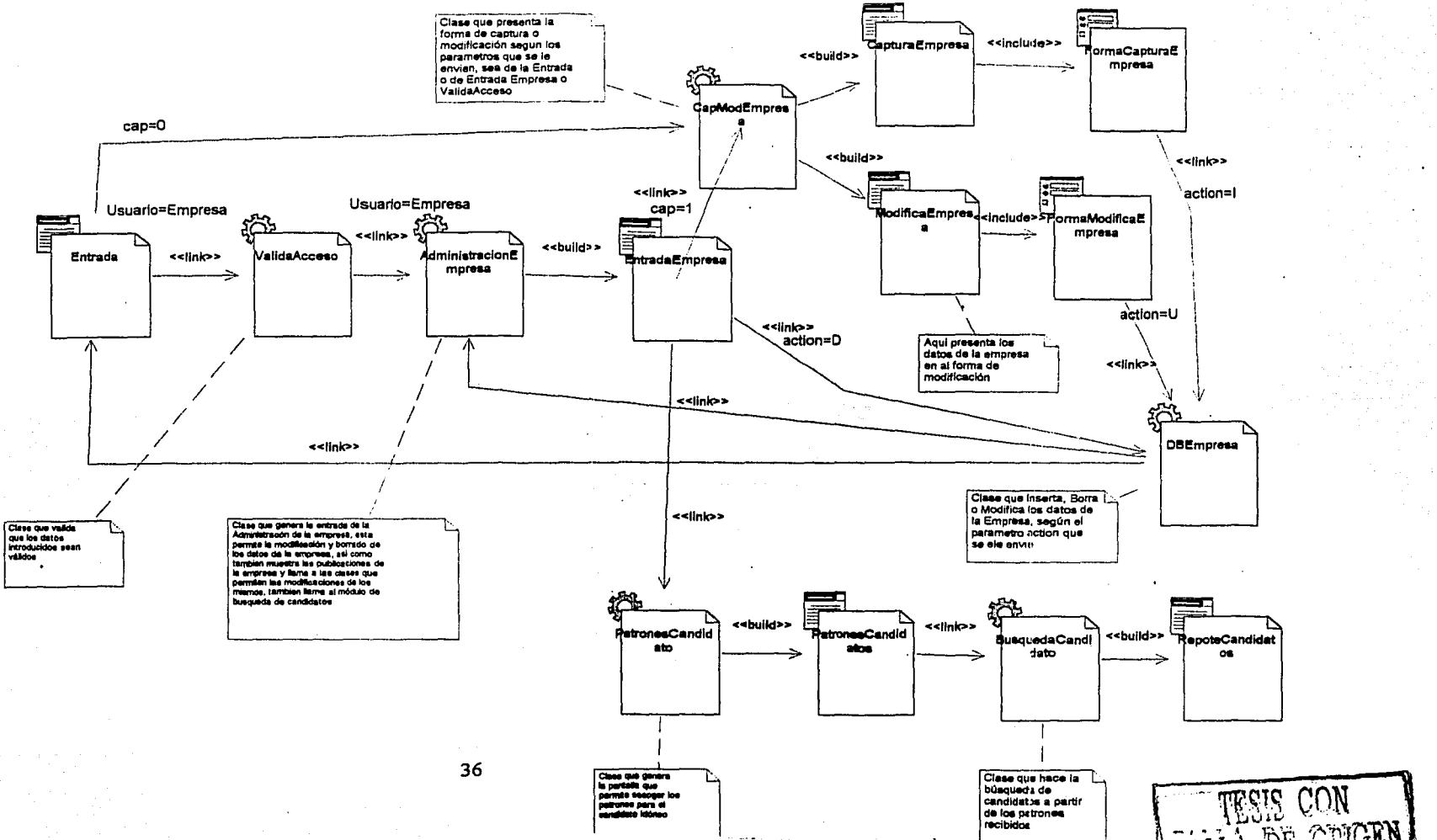


Diagrama de Clases (Publicacion)

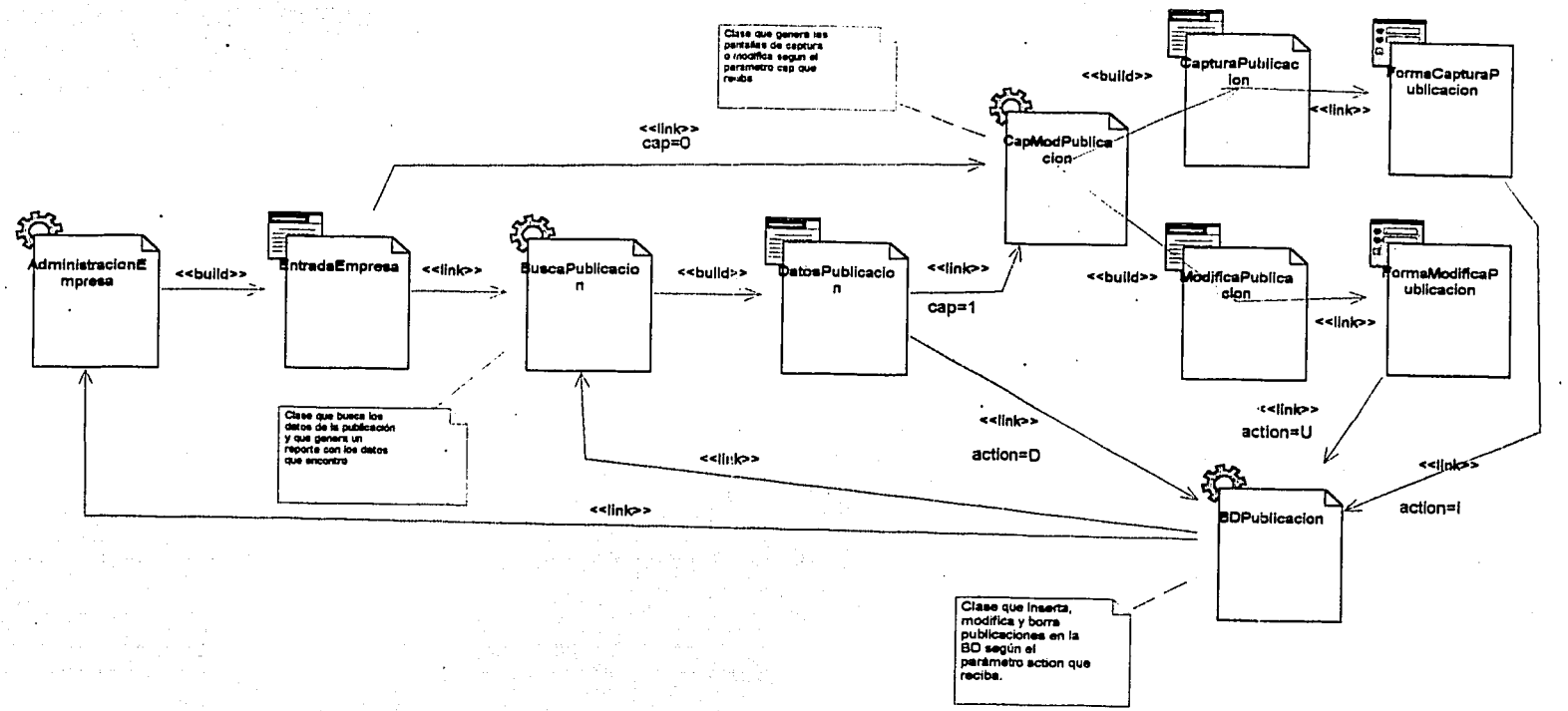


Diagrama de Clases (Publico en General)

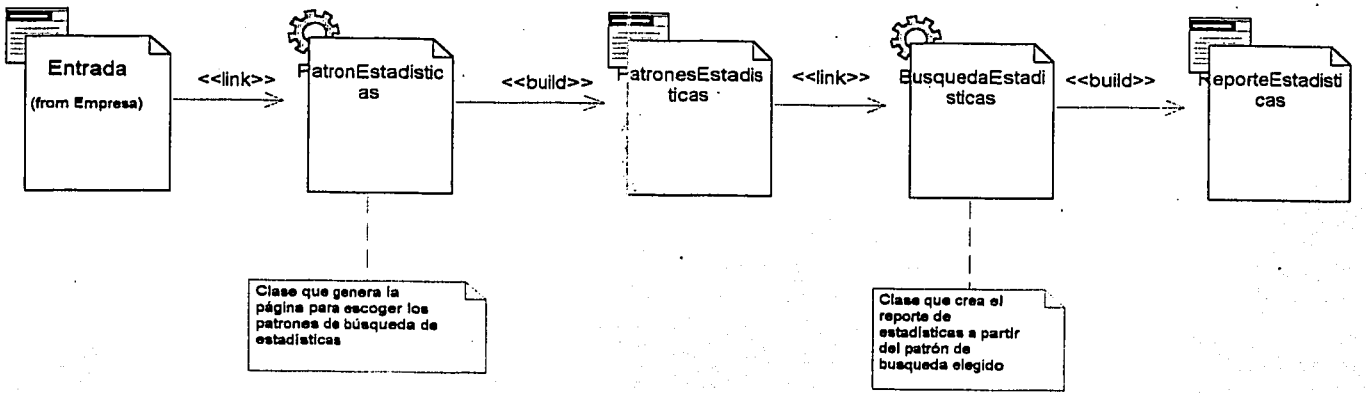


Diagrama de Clases (Solicitante)

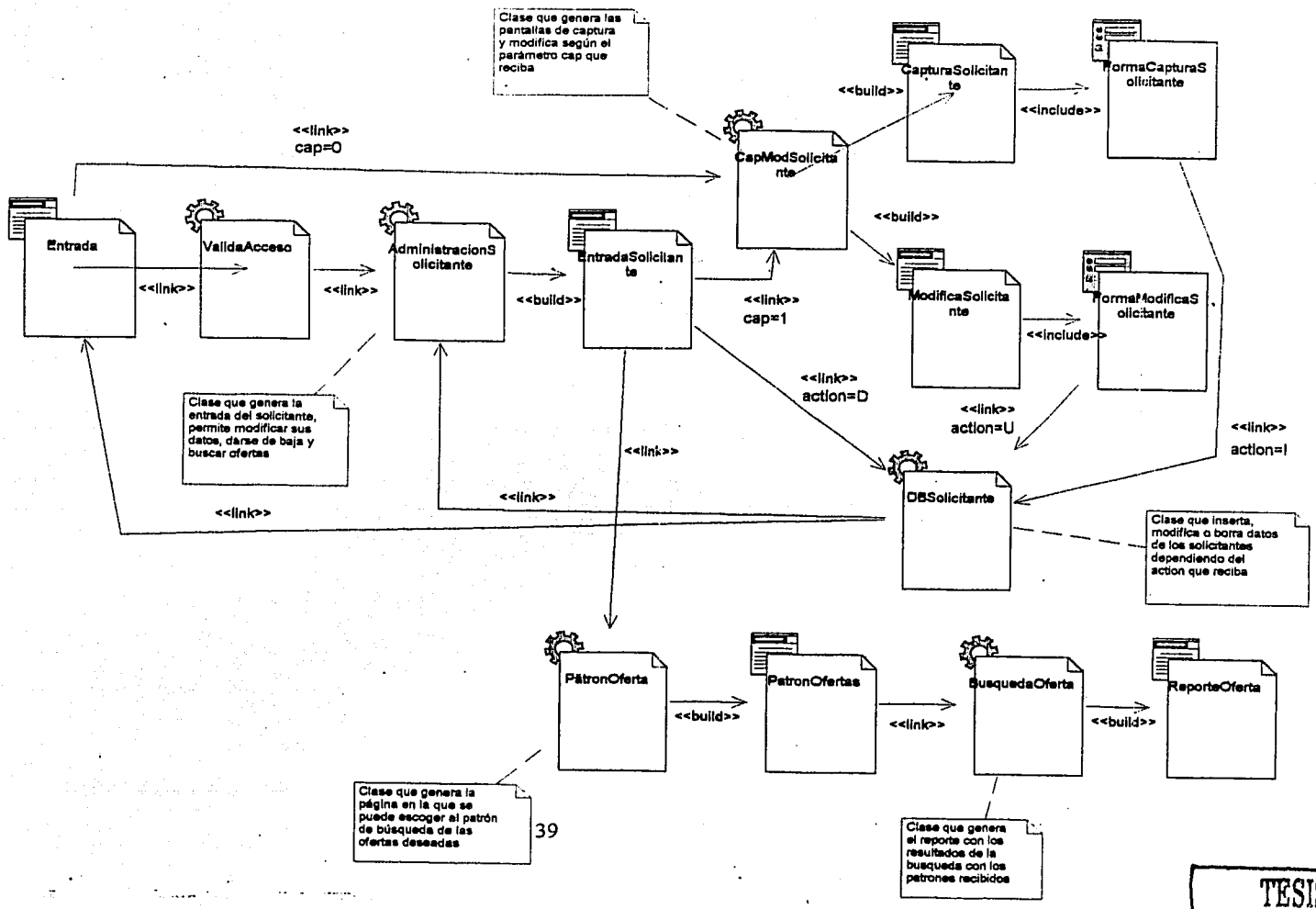
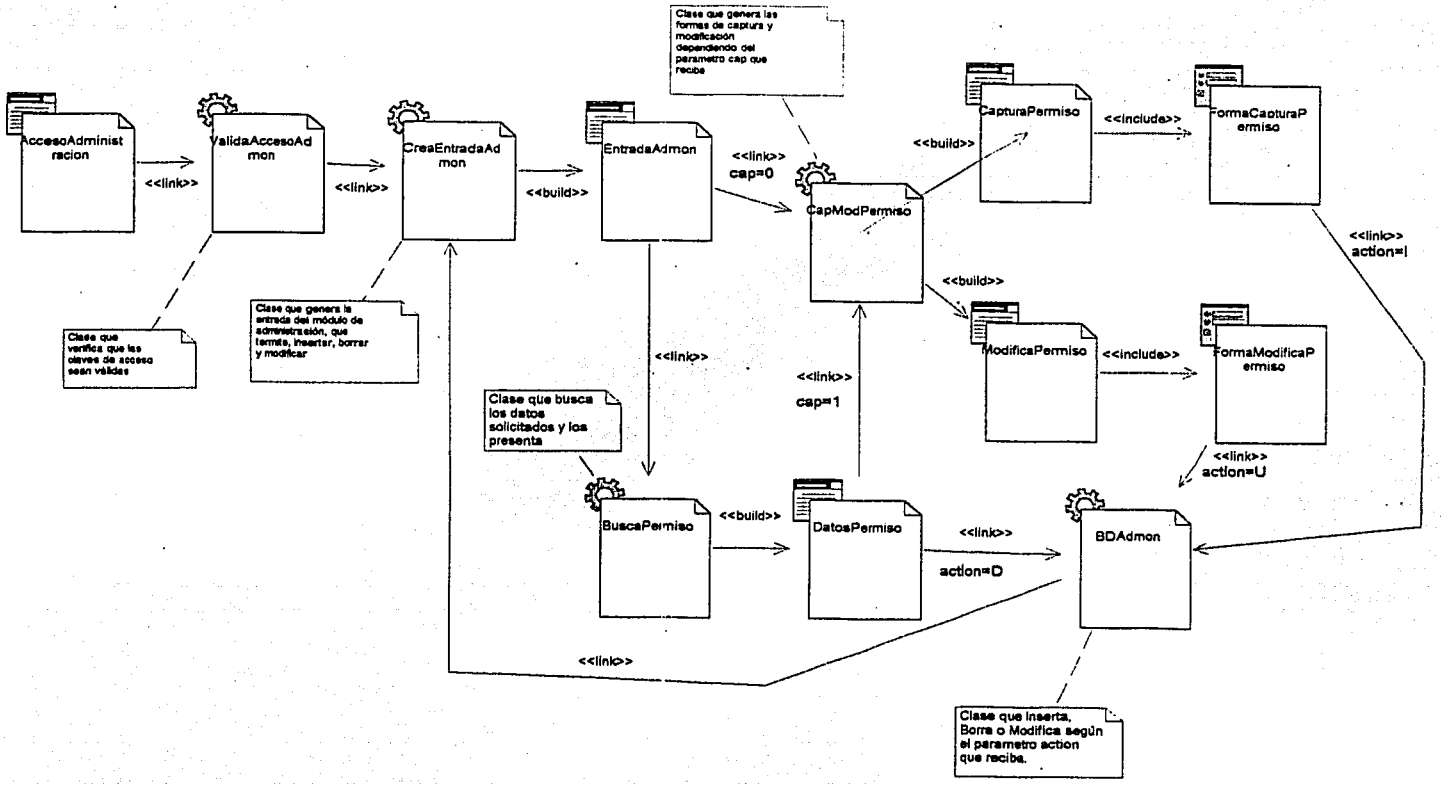
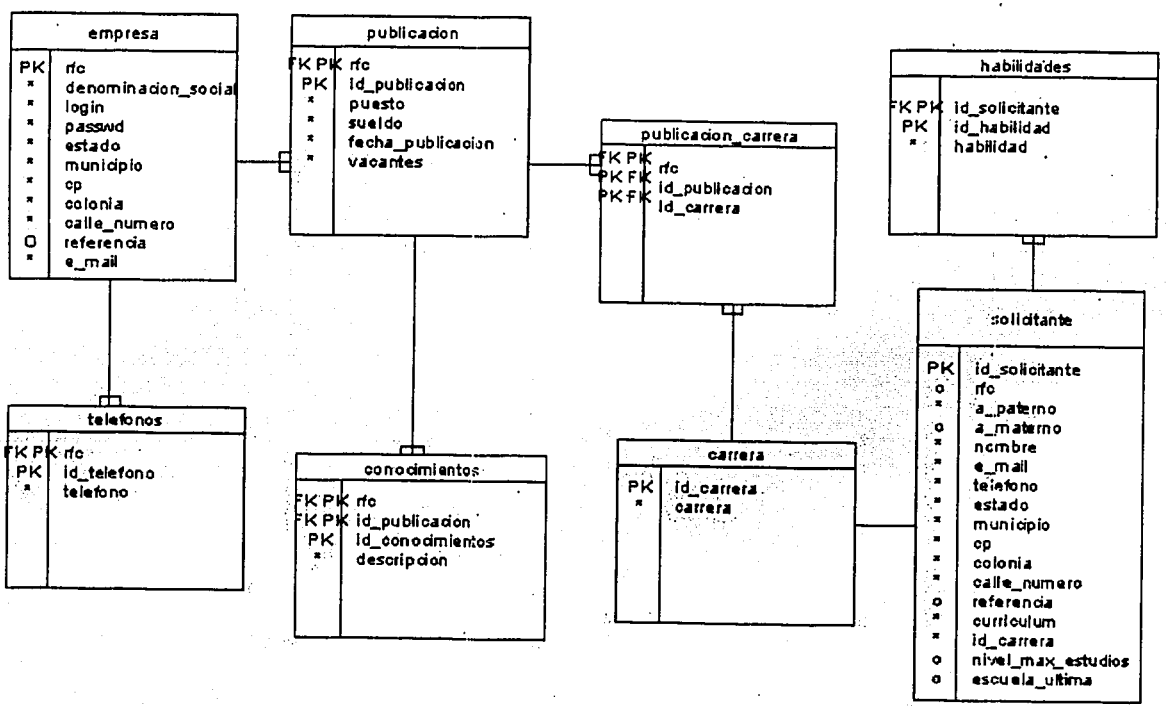


Diagrama de Clases (Administración del sistema)



4.4. Diseño de la Base de Datos

DIAGRAMA ENTIDAD-RELACIÓN



4.5. Diccionario de Datos de la Implementación del Sistema

1	empresa					
ID	NOMBRE	DESCRIPCION	ORIGEN	TIPO	PRIORIDAD	LONGITUD
1.1	rfc	Registro federal de contribuyentes de la empresa		varchar	PK	13
1.2	denominacion_social	Nombre o razón de la empresa		varchar	*	50
1.3	login	login de acceso de empresa		varchar	*	10
1.4	passwd	contraseña de acceso de la empresa		varchar	*	10
1.5	estado	Estado del domicilio de la empresa		varchar	*	30
1.6	municipio	Municipio del domicilio de la empresa		varchar	*	40
1.7	cp	Código postal del domicilio de la empresa		varchar	*	5
1.8	colonia	Colonia del domicilio de la empresa		varchar	*	100
1.9	calle_numero	Calle y Número del domicilio de la empresa		varchar	*	100
1.10	referencia	Referencia de localización del domicilio de la empresa		varchar	O	100
1.11	e_mail	Dirección de correo electrónico de la empresa		varchar	*	30

2	telefonos					
ID	NOMBRE	DESCRIPCION	ORIGEN	TIPO	PRIORIDAD	LONGITUD
2.1	rfc	Registro federal de contribuyentes de la empresa	empresa	varchar	FK PK	13
2.2	id_telefono	identificador del tipo de teléfono		Int	PK	2
2.3	telefono	Número telefónico		varchar	*	20

3	publicacion					
ID	NOMBRE	DESCRIPCION	ORIGEN	TIPO	PRIORIDAD	LONGITUD
3.1	rfc	Registro federal de contribuyentes de la empresa	empresa	varchar	FK PK	13
3.2	id_publicacion	identificador de la publicación		int	PK	3
3.3	puesto	Nombre del puesto		varchar	*	20
3.4	sueldo	Sueldo o salario		number	*	10
3.5	fecha_publicacion	Fecha de la publicación		date	*	
3.6	vacantes	Número de vacantes		int	*	3

4	conocimientos					
ID	NOMBRE	DESCRIPCION	ORIGEN	TIPO	PRIORIDAD	LONGITUD
4.1	rfc	Registro federal de contribuyentes de la empresa	empresa	varchar	FK PK	13
4.2	id_publicacion	identificador de la publicación	publicacion	int	FK PK	3
4.3	id_conocimientos	identificador de conocimientos		int	PK	2
4.4	descripcion	Descripción de conocimientos		varchar	*	150

5	Publicación_carrera					
ID	NOMBRE	DESCRIPCION	ORIGEN	TIPO	PRIORIDAD	LONGITUD
5.1	rfc	Registro federal de contribuyentes de la empresa	empresa	varchar	FK PK	13
5.2	id_publicacion	Identificador de la publicación	publicacion	int	FK PK	3
5.3	id_carrera	Identificador de carrera	carrera	int	FK PK	3

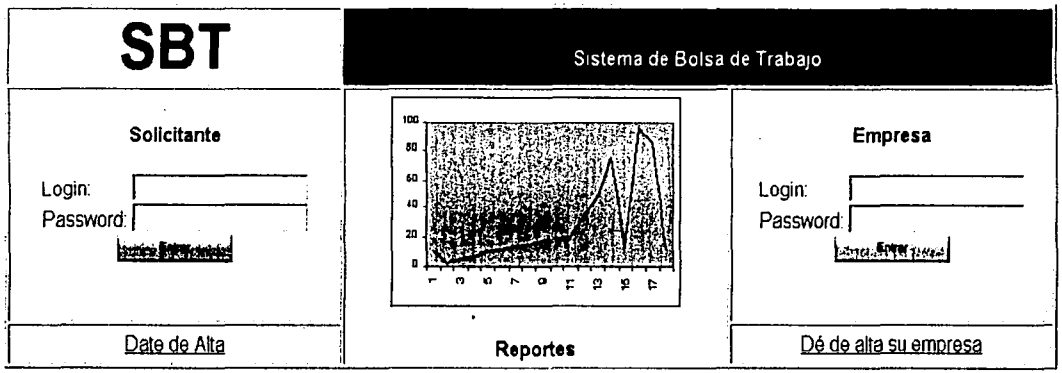
6	carrera					
ID	NOMBRE	DESCRIPCION	ORIGEN	TIPO	PRIORIDAD	LONGITUD
6.1	id_carrera	identificador de carrera		int	FK PK	3
6.2	carrera	Descripción de la carrera		varchar	*	50

7	habilidades					
ID	NOMBRE	DESCRIPCION	ORIGEN	TIPO	PRIORIDAD	LONGITUD
7.1	id_solicitante	identificador de solicitante	solicitante	int	FK PK	5
7.2	id_habilidad	identificador de la habilidad		int	PK	3
7.3	habilidad	Descripción de la habilidad		varchar	*	70

8	**solicitante**					
ID	**NOMBRE**	**DESCRIPCION**	**ORIGEN**	**TIPO**	**PRIORIDAD**	**LONGITUD**
8.1	id_solicitante	identificador de solicitante		int	PK	5
8.2	rfc	Registro Federal de Contribuyentes del Solicitante		varchar	O	13
8.3	a_paterno	Apellido paterno del solicitante		varchar	*	30
8.4	a_materno	Apellido materno del solicitante		varchar	O	30
8.5	nombre	Nombre del solicitante		varchar	*	50
8.6	e_mail	Dirección de correo electrónico del solicitante		varchar	*	30
8.7	telefono	Teléfono del solicitante		varchar	*	20
8.8	estado	Estado del domicilio del solicitante		varchar	*	30
8.9	municipio	Municipio del domicilio del solicitante		varchar	*	40
8.10	cp	Código postal del domicilio del solicitante		varchar	*	5
8.11	colonia	Colonia del domicilio del solicitante		varchar	*	100
8.12	calle_numero	Calle y Número del domicilio del solicitante		varchar	*	100
8.13	referencia	Referencia de localización del domicilio del solicitante		varchar	O	100
8.14	curriculum	Dirección del Curriculum del Solicitante		varchar	*	70
8.15	id_carrera	identificador de carrera	carrera	int	*	3
8.16	nivel_max_estudios	Nivel máximo de estudios del solicitante		varchar	O	50
8.17	escuela_ultima	Última escuela a la que asistió el solicitante		varchar	O	100

4.6. Diseño de la Interfaz Gráfica del Usuario

Pagina de inicio



Alta de Cuenta Empresa

SBT Sistema de Bolsa de Trabajo

Alta de Empresa

Instrucciones: Llene los datos que se le solicitan, las etiquetas con un * son obligatorios.

Datos de la Empresa

RFC*:
Denominación Social*:

Domicilio

Estado*: DISTRITO FEDERAL Delegación o Municipio: GUSTAVO A. MADERO

Colonia*:
Calle y Numero*:
Código Postal*:

Medio de Comunicación

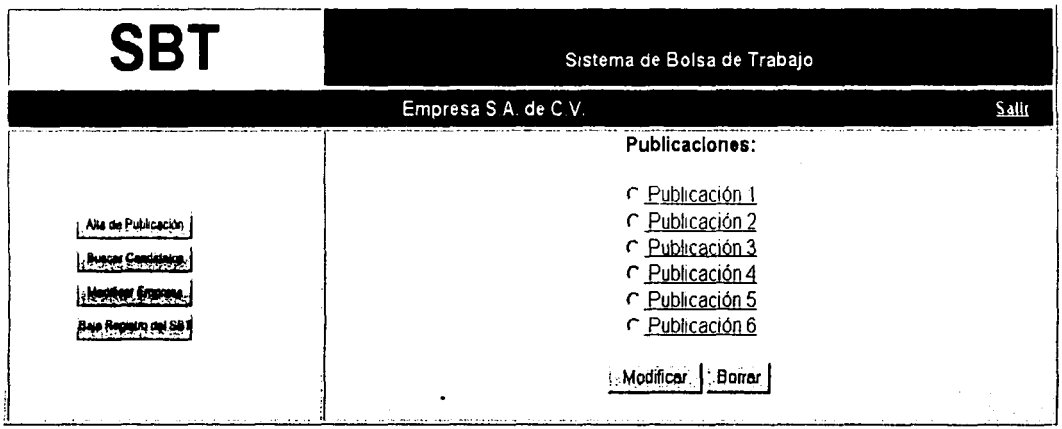
E_mail*:
Teléfono(s)*: Agregar

Medio de Identificación

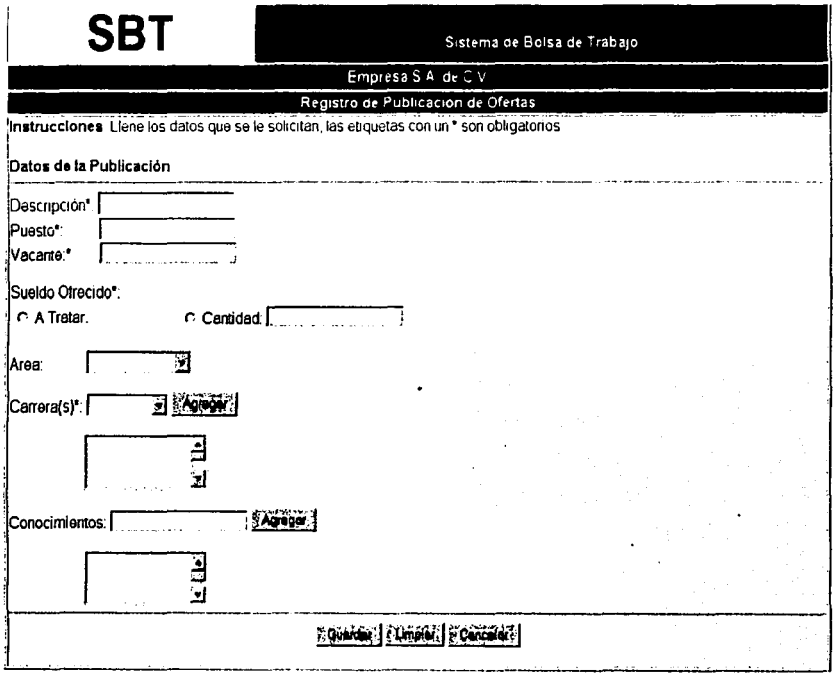
Login*:
Password*:
Repita su Password*:

Guardar Limpiar Salir

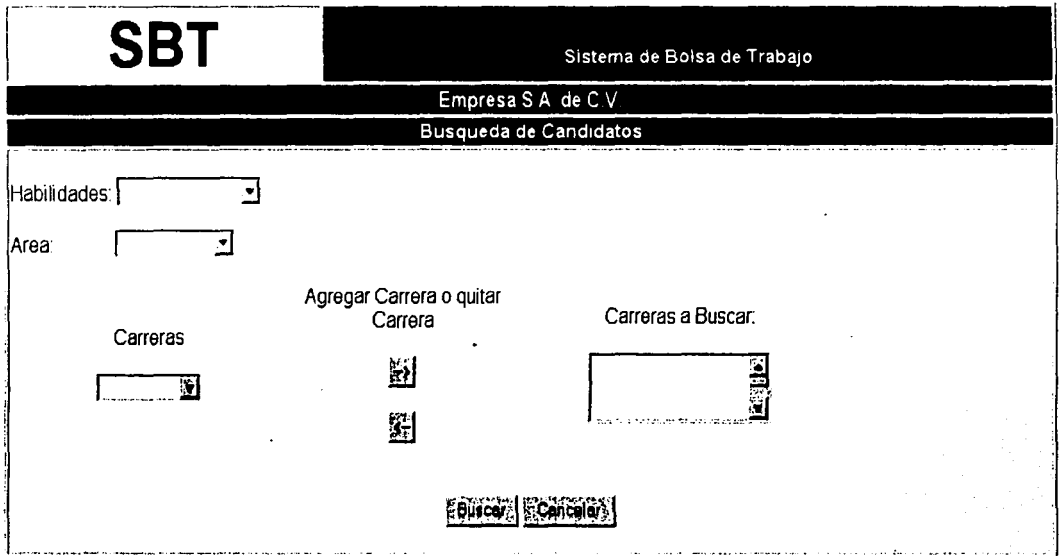
Administración de Publicación



Alta de Publicación



Búsqueda de Solicitantes



Alta de Cuenta de Solicitante

SBT Sistema de Bolsa de Trabajo

Alta de Solicitante

Instrucciones: Llene los datos que se le solicitan, las etiquetas con un * son obligatorios.

Datos Personales

RFC:
Nombre*:
Apellido Paterno*:
Apellido Materno:

Domicilio

Estado*: DISTRITO FEDERAL Delegación o Municipio: GUSTAVO A. MADERO

Colonia*:
Calle y Numero*:
Código Postal*:

Medio de Comunicación

E_mail*:
Teléfono*:

Datos Académicos

Carrera*:
Habilidades*:
Curriculum*:

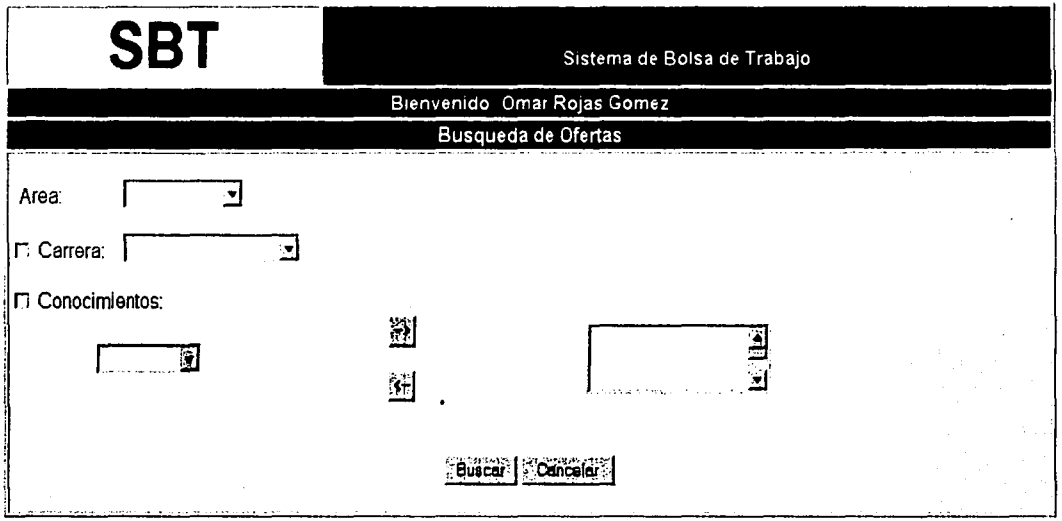
Medio de Identificación

Login*:
Password*:
Repita su Password*:

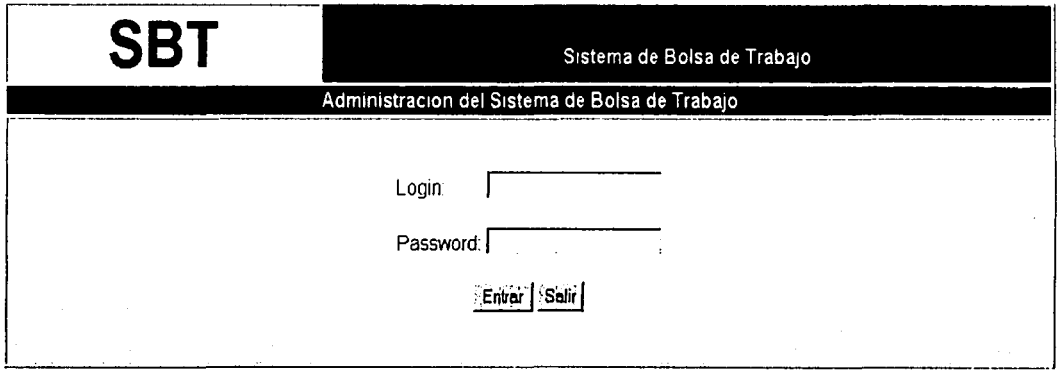
Administración de Solicitantes

SBT	Sistema de Bolsa de Trabajo
Bienvenido Omar Rojas Gomez	Salir
Buscar Oferta [redacted] [redacted]	**Ofertas que a usted le interesarían:** Oferta 1 Oferta 2 Oferta 3 Oferta 4 Oferta 5

Búsqueda de Ofertas



Petición de Acceso



Administración del Sistema de Bolsa de Trabajo

SBT

Sistema de Bolsa de Trabajo

Administracion del Sistema de Bolsa de Trabajo

Administración Empresa	**Administración Solicitantes**
Empresas que no han hecho publicaciones en más de 3 meses.	Solicitantes que no hay hecho cambios en sus datos por más de 3 meses
Empresa S.A. 1	Solicitante 1
Empresa S.A. 2	Solicitante 2
Empresa S.A. 3	Solicitante 3
Empresa S.A. 4	Solicitante 4
Empresa S.A. 5	Solicitante 5

Generador de Reportes

SBT

Sistema de Bolsa de Trabajo

Reportes y Estadisticas

Patrón de Búsqueda

- Conocimientos Requeridos
- Indice de Solicitación de Carreras
- Empresas con Mayor Demanda de Solicitantes
- Carreras Mejor Pagadas

Número de Registros Solicitados 10

Buscar | Cancelar

4.7. Diseño de Reportes

Reporte de Ofertas

SBT	Sistema de Bolsa de Trabajo
Candidatos	

Ofertas Disponibles:

oferta 1
oferta 2
oferta 3
oferta 4
oferta n

Descripción de Oferta

SBT

Sistema de Bolsa de Trabajo

Publicaciones

	Nombre de empresa
Descripción :	Desarrollo de aplicaciones web, en leguajes Java, PHP, Perl, y con BD PostgreSQL, Oracle y Sybase. Necesario Sistema Operativo Linux.
Puesto :	Analista Programador WEB
No. de Vacantes :	3
Sueldo :(mensual)	$ 12,500
Carreras :	Informática Ing. en Computación
Conocimientos :	PHP, JAVA, ORACLE, PERL
Correo Electrónico :	gcarrillo@yahoo.com.mx
Teléfonos :	1623-4879

Descripción Solicitante

SBT Sistema de Bolsa de Trabajo

Candidatos

Nombre	Rubén Zulbarán Juárez
Carrera	Lic en Informática
Conocimientos	PHP, JAVA, ORACLE, PERL
Curriculum :	Curriculum Vitae.doc
Correo Electrónico :	rubenzj@wololo.com
Teléfonos :	2345-8920

Nombre :	Omar Rojas Gómez
Carrera :	Lic en Informática
Conocimientos :	PHP, PYTHON, ORACLE, POSTGRESQL
Curriculum :	Curriculum Vitae.doc
Correo Electrónico :	jsbryor@wololo.com
Teléfonos :	8920-4475

Índice solicitado de carreras

SBT Sistema de Bolsa de Trabajo

Indice Solicitado de Carreras

Carrera :	Numero de empresas que lo solicitan :	Porcentaje :
Carrera 1	13	N %
Carrera 2	13	N %
Carrera 3	13	N %
Carrera 4	13	N %
Carrera n	13	N %

Carreras mejor pagadas

SBT — Sistema de Bolsa de Trabajo

Carreras mejor pagadas

Carrera :	Limites Inferior - Superior :	Promedio :
Carrera 1	$ 5,000 - $ 12,500	N
Carrera 2	$ 5,000 - $ 12,500	N
Carrera 3	$ 5,000 - $ 12,500	N
Carrera 4	$ 5,000 - $ 12,500	N
Carrera n	$ 5,000 - $ 12,500	N

Conocimientos Requeridos

SBT

Sistema de Bolsa de Trabajo

Conocimientos Requeridos

Habilidades :	Numero de empresas que lo solicitan :
Habilidad 1	13
Habilidad 2	133
Habilidad 3	15
Habilidad 4	13
Habilidad n	13

Empresas que más solicitantes requieren

SBT Sistema de Bolsa de Trabajo

Empresas que mas solicitantes requieren

Empresas :	Numero de vacantes :
Empresa 1	7
Empresa 2	7
Empresa 3	7
Empresa 4	3
Empresa n	5

4.8. Herramientas de Desarrollo

Para el desarrollo de este sistema, se utilizará el siguiente Software:

Sistema Operativo: Linux Red Hat 7.3.

Servidor Web: Apache 1.4,
Tomcat 3.5

Lenguaje de Programación: Java SDK 1.3.1, SDKEE 1.2.1
HTML 4.0
JavaScript 1.2

Base de Datos: PostgreSQL 7.3

Justificación

Se eligió el Sistema Operativo Linux Red Hat 7.3 por su fiabilidad, facilidad de uso, sus aplicaciones y la facilidad de licenciamiento que nos brinda.

El servidor Web Apache 1.4 se eligió por su confiabilidad, robustez, sencillez y su compatibilidad con otros proyectos, como Tomcat; el proyecto Tomcat se acopla bien al servidor Web Apache lo cual nos permite utilizar el potencial de Java SDKEE 1.2.1.

El lenguaje de programación Java, en sus dos variantes SDK 1.3.1 y SDKEE 1.2.1, se eligió por ser robusto, confiable, sencillo y seguro. Este lenguaje permite desarrollar sistemas con rapidez, seguridad y con una gran capacidad de modularidad.

El Manejador de Base de Datos PostgreSQL 7.3 se eligió porque cumple con los estándares de SQL 98, por lo tanto para su mantenimiento cualquier administrador de Base de Datos puede utilizarlo.

CONCLUSIONES

El Sistema de Bolsa de Trabajo cumplió satisfactoriamente el objetivo general, ya que se cubrieron los requerimientos solicitados por el usuario:

- Registro de la empresa. Se analizó, diseñó e implementó un módulo con las características necesarias para registrar y actualizar la información general de empresas que se registran en el sistema.
- Servicios a las empresas registradas: Se analizó, diseñó e implementó un módulo con las características necesarias para registrar, actualizar y eliminar publicaciones de ofertas de trabajo, para buscar a los candidatos idóneos para sus puestos disponibles.
- Registro de currículum vitae: Se analizó, diseñó e implementó un módulo con las características necesarias para registrar y actualizar los currícula de las personas que se registran en el sistema.
- Servicios a las personas que registraron su currículum vitae: Se analizó, diseñó e implementó un módulo con las características necesarias para buscar publicaciones de ofertas de trabajo de las empresas.
- Servicios al público en general: Se analizó, diseñó e implementó un módulo con las características necesarias para obtener reportes estadísticos sobre ofertas de empleo publicadas.

Todo lo anterior se pudo llevar a cabo gracias a lo concreto y específico de la metodología Yourdon, apoyada por la extensión de UML para el diseño de aplicaciones Web, que nos dio la facilidad de identificar los componentes (HTML, scripts de navegador y scripts de servidor) que se manejarían para la programación e implementación del Sistema de Bolsa de Trabajo. Gracias a ésto se obtuvo orden, claridad, limpieza y agilidad en este trabajo, ya que las necesidades quedaron bien definidas.

ANEXO I

Simbología utilizada en el Diccionario de Datos, Diagrama de Base de Datos y Diccionario de Datos de la Implementación del Sistema

Símbolo	Significado
PK	Llave primaria de una tabla de la Base de datos, es obligatorio, único y no nulo.
FK	Llave foránea, indica la relación entre dos tablas, esta debe ser una llave primaria en de las tablas referenciadas.
*	Campo obligatorio de una tabla.
O	Campo que puede ser nulo en una tabla.

ANEXO II

Modelando Diseño de Aplicaciones Web con UML

Las aplicaciones Web se han hecho muy populares debido en parte a las herramientas y tecnologías que permiten su desarrollo con rapidez, y porque los diseñadores de sistemas han reconocido situaciones donde una aplicación Web tiene ventajas significativas sobre las aplicaciones tradicionales.

Normalmente el avance de las aplicaciones Web han sido las *herramientas de desarrollo* y se pone poca atención en el *proceso de desarrollo*, debido a que los *ambientes de desarrollo* actuales permiten crear aplicaciones sin un análisis y diseño serios. Cualquier sistema con una complejidad media necesita ser diseñado y modelado, desafortunadamente el modelado de estas aplicaciones no es tan obvio.

El modelado es importante debido a que ayuda a manejar la complejidad. Las aplicaciones Web pueden hacerse complejas con rapidez. Un sistema puede ser representado de muchas formas con modelos consistentes. Cada modelo tiene un propósito y una audiencia, la cual es el arquitecto Web y el diseñador. Es importante que se modelen los artefactos con un apropiado nivel de abstracción.

Como el artefacto principal de las aplicaciones Web son las páginas, se creería que su modelado detallado sería esencial, pero en un diseño de un modelo, la interfaz del usuario es irrelevante y no tiene efecto en la lógica del negocio. Los *Scripts*, especialmente los del servidor, tienen un efecto sobre el comportamiento del negocio, o pueden ser completamente la implementación de las reglas de negocio.

Extensión de UML

Los diseñadores de UML reconocen que el lenguaje no siempre es perfecto para cada situación. Hay momentos cuando en el proceso de desarrollo sería mejor agregar información adicional o aplicar una semántica distinta cuando se modelan diferentes elementos.

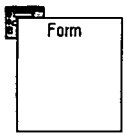
UML ha definido un mecanismo para permitir extender la semántica de modelos específicos. El mecanismo de extensión permite la inclusión de nuevos atributos, semántica y restricciones distintas. Cuando se colocan todos estos elementos nuevos forman una extensión de UML.

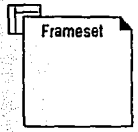
La extensión da la posibilidad de asignar diferentes iconos a clases estereotipadas. El problema de que una pagina Web tenga diferentes scripts y variables ejecutadas tanto en el servidor como en el cliente, se resuelve definiendo estereotipos, un método en el servidor y un método en el cliente, esto permite distinguir atributos y métodos de un objeto página.

La extensión de UML define estereotipos concretos para clases, estos son: Server Page, Client Page, Form, Frameset, Target y Scriplet. Cada uno de ellos representa un componente Web (HTML, Script del navegador o Script del servidor) y la relación existente entre ellos.

Los estereotipos que se manejaron en el presente trabajo se definen en la siguiente tabla:

Estereotipos

Estereotipo de clase	Icono	Descripción
Server Page (Página del servidor)		El Estereotipo Server Page representa todas las rutinas, subrutinas y funciones que se ejecutan en el servidor. Los Scripts del cliente o el formato de la interfaz no son parte de este estereotipo. Se puede relacionar con otros componentes que existan en el servidor lo cual incluye objetos de negocio o componentes de acceso a base de datos
Client Page (Página del Cliente)		El Client Page representa la pagina que se ejecuta en el navegador del cliente, esta está asociada con componentes que se ejecutan en el cliente, incluyendo Applets, Controles Active X y otros estereotipos del Modelo.
Form (Forma)		El estereotipo Form tiene los atributos de los elementos de la pagina, los métodos del estereotipo Client Page tienen acceso a todos los atributos de estereotipo Form, y la relación entre estos es que un Client Page contiene Forms.

Frameset (Juego de Cuadros)		El estereotipo Frameset representa la forma en que se divide una pagina en cuadros cada uno de ellos con un nombre, y que estos pueden contener distintos estereotipos Client Page. Por lo regular contiene un estereotipo Target que es donde se mostrarían las Client Page llamadas.
Target (Cuadro Destino)		El estereotipo Target representa el cuadro destino que manda llamar un estereotipo Client Page, que esta contenida en un estereotipo Frameset. Target solo esta contenido dentro del estereotipo Frameset.
Scriptlet (Scripts de Navegador)		Un estereotipo Scriptlet es una página del cliente en memoria del navegador, contiene referencias a componentes y controles que son reutilizados en las páginas del cliente.

Referencia Electrónica

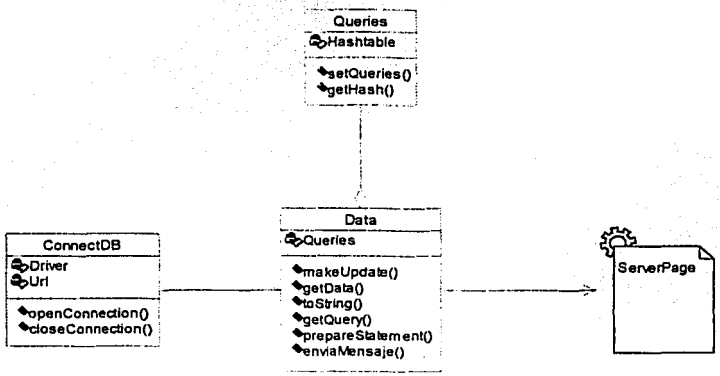
http://www.rational.com/products/whitepapers/100462.jsp

ANEXO III

Implementación del sistema

La implementación del sistema se desarrolló en base a los diagramas de clases expuestos en el capítulo 4, es importante hacer notar que cada estereotipo ServerPage es el encargado de realizar las operaciones con la Base de Datos.

Para simplificar el desarrollo de este sistema, se construyó un módulo de conexión a la Base de Datos, que se diagrama a continuación:



La clase ConnectDB es la responsable de crear conexiones a la Base de Datos y enviárselas a la clase Data, también se hace cargo de cerrar dichas conexiones.

La Clase Queries es un contenedor de todas las sentencias de SQL que se utilizan en el sistema.

La Clase Data utiliza las conexiones que pide a ConnectDB, las sentencias de SQL de la clase Queries, para realizar todas las inserciones, actualizaciones, eliminaciones y búsquedas que el estereotipo ServerPage necesita para cumplir la lógica del negocio.

A continuación se presentan la implementación del módulo Alta de Solicitante, incluyendo las clases del diagrama anterior.

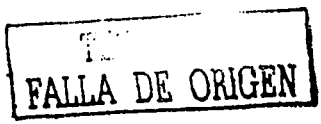
Clase ConnectDB

```
package DBAccess;

import java.sql.*;


/**
Clase que se encarga de abrir y cerrar coneexiones segun sean las
circunstancias.
*/
public class ConnectDB{

    static final String DRIVER="org.postgresql.Driver";
    static final String CONN = "jdbc:postgesql://localhost/DBTrabajo";

/**
Método que abre una conexión, con el driver declarado y el url de la
conexión, regresa un objeto de tipo Connection.
*/

      public static Connection openConnection(){
            Connection conn = null;
            try{
                  Class.forName(DRIVER).newInstance();
                  conn = DriverManager.getConnection(CONN);
            }catch(Exception e){
                  System.out.println(e.toString() );
                  conn = null;
            }
            return conn;
      }

/**
Método que cierra una conexión, recibe la conexión como parametro.
*/


      public static void closeConnection(Connection conn){
            try{
                  conn.close();
            }catch(Exception e){
                  System.out.println(e.toString() );
            }
      }

}
```

Clase Queries

```
package DBAccess;

import java.util.Hashtable;
/**
      Clase contenedora de Queries.
*/
public class Queries{

      /**
            Variable que contiene los queries.
      */
      Hashtable ht = new Hashtable();

      /**
            Constructor de la clase que manda llamar al metodo
            setQueries.
      */

      public Queries(){
            setQueries();
      }

      /**
            Metodo que pone los queries en la variable Hashtable.
      */
      public void setQueries(){

            //Queries para el modulo de solicitante.
            //Alta de solicitante
            ht.put("count_solicitante","select
            ifnull(max(id_solicitante),0) + 1  from solicitante");
            ht.put("valida_login","select  count(*)   from  solicitante
      where       login=?");
            ht.put("insert_solicitante","insert into solicitante
            values(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)");
            ht.put("insert_habilidades","insert into habilidades
            values(?,?,?)");
            ht.put("modify_file","update solicitante set curriculum=?
            where login=?");

            //Modificacion de solicitante
            ht.put("select_solicitantes","select
            rfc,nombre,a_paterno,a_materno,estado,municipio,colonia,
            calle_numero, "+

      "referencia,cp,e_mail,telefono,nivel_max_estudios,id_carrera,
            escuela_ultima,login,passwd " +
            " from solicitante where id_solicitante=?");
            ht.put("select_habilidades","select      habilidad      from
            habilidades where id_solicitante=?");
            ht.put("modify_solicitante","update solicitante set
            rfc=?,nombre=?,a_paterno=?,a_materno=?,estado=?,municipio=?
      ,
            colonia=?, " +
            "calle_numero=?, referencia=?,cp=?,e_mail=?,telefono=?,
            nivel_max_estudios=?,id_carrera=?, " +
```

```
            " escuela_ultima=?,passwd=? where id_solicitante=?");
            //Baja de Solicitante
            ht.put("delete_solicitante","delete from solicitante where
            id_solicitante=?");
            ht.put("delete_habilidades","delete from habilidades where
            id_solicitante=?");
            //Admon solicitante
            ht.put("select_nombre_sol","select id_solicitante,
            concat(nombre,'    '    ,a_paterno,'    ',a_materno)    from
      solicitante       where login=?");
            ht.put("select_ofertas_interes"," select p.id_publicacion,
            p.rfc  from  publicacion_carrera  p,solicitante  s  where
            p.id_carrera = s.id_carrera and id_solicitante=?");
      }

      /**
            Metodo que regresa la variable Hashtable.
      */

      public Hashtable getHash(){
            return ht;
      }
}
```



Clase Data

```
package DBAccess;

import DBAccess.*;
import java.sql.*;
import java.io.*;
import java.util.Hashtable;
import java.util.Vector;
import javax.servlet.*;
import javax.servlet.http.*;

/**
      Clase que ejecuta los Queries regresando un arreglo
bidimensional con los datos obtenidos
      y tembien contiene metodos para ejecutar Inserts y Updates.
*/

public class Data{

      static Queries q = new Queries();

/**
Metodo que realiza una actualizacion o inserción a la BD, recibe como
parametro el nombre
del Query contenido en la clase Queries, ejecuta la instrucción y
regresa un booleano, verdadero si no hubo fallas y falso si las hubo.
*/

      public static boolean makeUpdate ( String query ){
            try{
                  Connection conn = ConnectDB.openConnection();
                  conn.setAutoCommit(false);
                  String stamt = getQuery(query);
                  PreparedStatement ps = prepareStatement(conn,stamt);
                  int prueba = ps.executeUpdate();
                  if(prueba == 0){
                        conn.rollback();
                        return false;
                  }
                  conn.commit();
                  ps.close();
                  ConnectDB.closeConnection(conn);
                  return true;
            }catch(Exception e){
                  e.printStackTrace();
            }
            return false;
      }

/**
Metodo que realiza una actualizacion o inserción a la BD,
      recibe como parametro el nombre del Query contenido en la clase
Queries, y un único parametro,      ejecuta la instrucción y regresa un
booleano, verdadero si no hubo fallas y falso si las hubo.
*/
```

```
	public static boolean makeUpdate ( String query,String param ){
		try{
			Connection conn = ConnectDB.openConnection();
			conn.setAutoCommit(false);
			String stamt = getQuery(query);
			PreparedStatement ps = prepareStatement(conn,stamt);
			ps.setString(1,param);
			int prueba = ps.executeUpdate();
			if(prueba == 0){
				conn.rollback();
				return false;
			}
			conn.commit();
			ps.close();
			ConnectDB.closeConnection(conn);
			return true;
		}catch(Exception e){
			e.printStackTrace();
		}
		return false;
	}

/**
Metodo que realiza una actualizacion o inserción a la BD, recibe como
parametro el nombre del Query contenido en la clase Queries, y arreglo
que contiene los parametros, ejecuta la instrucción y regresa un
booleano, verdadero si no hubo fallas y falso si las hubo.
*/

	public static boolean makeUpdate ( String query,String[] params
){
		try{
			Connection conn = ConnectDB.openConnection();
			conn.setAutoCommit(false);
			String stamt = getQuery(query);
			PreparedStatement ps = prepareStatement(conn,stamt);
			for(int i = 0; i< params.length; i++){
				ps.setString(i+1,params[i]);
			}
			int prueba = ps.executeUpdate();
			if(prueba == 0){
				conn.rollback();
				return false;
			}
			conn.commit();
			ps.close();
			ConnectDB.closeConnection(conn);
			return true;
		}catch(Exception e){
			e.printStackTrace();
		}
		return false;
	}

/**
```

```
Metodo que realiza busqueda a la BD, recibe como parametro el nombre
del Query contenido en la clase Queries, ejecuta la busqueda y regresa
un
arreglo bidimensional con los datos obtenidos.
*/

      public static String[][] getData ( String query ){
            String [][] array = null;
            try{
                  Connection conn = ConnectDB.openConnection();
                  String stamt = getQuery(query);
                  PreparedStatement ps = prepareStatement(conn,stamt);
                  ResultSet rs = ps.executeQuery();
                  array = toString(rs);
                  ps.close();
                  ConnectDB.closeConnection(conn);

            }catch(Exception e){
                  e.printStackTrace();
            }
            return array;

      }

/**
Metodo que realiza busqueda a la BD, recibe como parametro el nombre
del Query contenido en la clase Queries y un único parametro, ejecuta
la busqueda y regresa arreglo bidimensional con los datos obtenidos.
*/

      public static String[][] getData ( String query, String param ){
            String [][] array = null;
            try{
                  Connection conn = ConnectDB.openConnection();
                  String stamt = getQuery(query);
                  PreparedStatement ps = prepareStatement(conn,stamt);
                  ps.setString(1,param);
                  ResultSet rs = ps.executeQuery();
                  array = toString(rs);
                  ps.close();
                  ConnectDB.closeConnection(conn);

            }catch(Exception e){
                  e.printStackTrace();
            }
            return array;

      }


/**
Metodo que realiza busqueda a la BD, recibe como parametro el nombre
del Query contenido en la clase Queries y un arreglo con todos los
parametros, ejecuta la busqueda y regresa arreglo bidimensional con
los datos obtenidos.
*/
```

```
	public static String[][] getData ( String query, String[] params
){
		String [][] array = null;
		try{
			Connection conn = ConnectDB.openConnection();
			String stamt = getQuery(query);
			PreparedStatement ps = prepareStatement(conn,stamt);
			for(int i = 0; i< params.length; i++){
				ps.setString(i+1,params[i]);
			}
			ResultSet rs = ps.executeQuery();
			array = toString(rs);
			ps.close();
			ConnectDB.closeConnection(conn);

		}catch(Exception e){
			e.printStackTrace();
		}
		return array;

	}

/**
Metodo que convierte el Resulset obtenido en la busqueda del
cualquiera de los metodos getData, y lo covierte en un arreglo de
cadenas, que devuelve despues.
*/
	public static String[][] toString(ResultSet rs){
		String[][] array = null;
		try{
			ResultSetMetaData rsmd = rs.getMetaData();
			//preparando el arreglo por linea.
			int val = rsmd.getColumnCount();
			String linea[] = new String[ val ];
			Vector v = new Vector(); //agregaremos cada renglon
al vector
			//los encabezados
			for(int i = 0; i < linea.length; i++){
				linea[i] = rsmd.getColumnName(i+1) ;
			}
			v.add(linea);
			while( rs.next() ){
				linea = new String[ val ];
				for(int i = 0; i < linea.length; i++){
					linea[i] = rs.getString(i+1);
				}
				v.add(linea);
			}
			//creando el arreglo bidimensional con los elementos
del Vector.
			array = new String[v.size()][linea.length];
			for(int j = 0; j<array.length;j++){
				array[j]=(String[])(v.elementAt(j));
			}
		}catch(Exception e){
			e.printStackTrace();
		}
		return array;
```

```
	}


/**
Metodo que obtiene el Query contenido en el HashTable de queries,
recibe como parámetro el nombre del query y resgresa una cadena que es
el query colocado en la clase Queries.
*/

	public static String getQuery(String key){
		Hashtable ht = q.getHash();
		Object statement = ht.get(key);
		return statement.toString();
	}

/**
Metodo que crea el PreparedStatement, recibe como parametros, la
conexión y el query	obtenido del HashTable de la clase Queries,
crea el PreparedStatement y lo devuelve, al	metodo que lo llamo.
*/
	public static PreparedStatement prepareStatement(Connection
conn, String query){
		PreparedStatement ps = null;
		try{
			ps = conn.prepareStatement(query);
		}catch(Exception e){
			e.printStackTrace();
		}
		return ps;
	}

/**
Metodo que envia un mensaje de error, o de aviso al usuario, una vez
que se esta realizando alguna operacion en el servidor.
*/
	public static void enviaMensaje(String msg, HttpServletResponse
response){
		try{
			PrintWriter out = response.getWriter();
			response.setContentType("text/html");
			out.println("<html>"+
				"<head></head>"+
				"<body
onLoad=\"javascript:alert('"+msg+"'),history.go(-1)\">"+
				"</body>"+
				"</html>");
			out.flush();
			out.close();
		}catch(Exception e){
			e.printStackTrace();
		}
	}
}
```

Clase DBSolicitante

```
package trabajo.solicitante;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import DBAccess.*;

public class DBSolicitante extends HttpServlet{
/**
Metodo que recibe las peticiones del usuario por el método Get, recibe
un objeto HttPServletRequest y un objeto HttpServletResponse
*/
      public void doGet(HttpServletRequest request,
HttpServletResponse response) throws ServletException{
            doPost(request,response);
      }

/**
Metodo que recibe las peticiones del usuario por el método Post,
recibe un objeto HttPServletRequest y un objeto HttpServletResponse

*/

      public void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException{
            try{
                  String tipo = request.getParameter("tipo");
                  if(tipo.equals("C")){
                        capturaSolicitante(request,response);
                  }else if(tipo.equals("M")){
                        modificaSolicitante(request,response);
                  }else{
                        eliminaSolicitante(request,response);
                  }
            }catch(Exception e){
                  e.printStackTrace();
                  Data.enviaMensaje("Hubo un error en la
transacción",response);
            }
      }

/**
Metodo capturaSoliciante, guarda en la Base de Datos  los datos que se
introdujeron en la forma de captura, recibe un objeto
HttPServletRequest y un objeto HttpServletResponse
*/
      public void capturaSolicitante(HttpServletRequest
request,HttpServletResponse response)  throws Exception{
            String tipo = request.getParameter("tipo");
            String rfc = request.getParameter("rfc");
            String nombre = request.getParameter("nombre");
            String a_paterno = request.getParameter("a_paterno");
            String a_materno = request.getParameter("a_materno");
            String estado = request.getParameter("estado");
            String distrito = request.getParameter("distrito");
            String colonia = request.getParameter("colonia");
```

```
            String calle = request.getParameter("calle");
            String referencia = request.getParameter("referencia");
            String cp = request.getParameter("cp");
            String e_mail = request.getParameter("e_mail");
            String telefono = request.getParameter("telefono");
            String nivel = request.getParameter("nivel");
            String carrera = request.getParameter("carrera");
            String escuela = request.getParameter("escuela");
            String[] habilidades =
request.getParameterValues("habilidades");
            String login = request.getParameter("login");
            String passwd = request.getParameter("passwd");
            String file = "file";
            rfc = rfc.toUpperCase();
            nombre = nombre.toUpperCase();
            a_paterno = a_paterno.toUpperCase();
            a_materno = a_materno.toUpperCase();
            colonia = colonia.toUpperCase();
            calle = calle.toUpperCase();
            referencia = referencia.toUpperCase();
            escuela = escuela.toUpperCase();
            String[][] login_old = Data.getData("valida_login",login);
            String[][] id_solicitante =
Data.getData("count_solicitante");
            if( login_old[1][0].equals("0") ){
                  String[] param =
{id_solicitante[1][0],rfc,a_paterno,a_materno,nombre,e_mail,telefono,e
stado,distrito,cp,colonia,calle,referencia,file,carrera,nivel,escuela,
login,passwd};
                  boolean transaccion =
Data.makeUpdate("insert_solicitante",param);
                  for(int i=0; i<habilidades.length; i++){
                        habilidades[i] = habilidades[i].toUpperCase();
                        String[] param_habilidades =
{id_solicitante[1][0],""+(i+1),habilidades[i]};
                        transaccion =
Data.makeUpdate("insert_habilidades",param_habilidades);
                  }
                  outHTML(tipo,login,response);
            }else{
                  Data.enviaMensaje("Ya existe el login que escogio,
favor de escribir otro",response);
            }
      }

/**
Metodo modificaSoliciante, modifica en la Base de Datos  los datos que
se introdujeron en la forma de captura, recibe un objeto
HttPServletRequest y un objeto HttpServletResponse
*/


      public void modificaSolicitante(HttpServletRequest
request,HttpServletResponse response)throws Exception{
            String tipo = request.getParameter("tipo");
            String id = request.getParameter("id_solicitante");
            String login = request.getParameter("login");
            String rfc = request.getParameter("rfc");
```

```
            String nombre = request.getParameter("nombre");
            String a_paterno = request.getParameter("a_paterno");
            String a_materno = request.getParameter("a_materno");
            String estado = request.getParameter("estado");
            String distrito = request.getParameter("distrito");
            String colonia = request.getParameter("colonia");
            String calle = request.getParameter("calle");
            String referencia = request.getParameter("referencia");
            String cp = request.getParameter("cp");
            String e_mail = request.getParameter("e_mail");
            String telefono = request.getParameter("telefono");
            String nivel = request.getParameter("nivel");
            String carrera = request.getParameter("carrera");
            String escuela = request.getParameter("escuela");
            String[] habilidades =
request.getParameterValues("habilidades");
            String passwd = request.getParameter("passwd");
            rfc = rfc.toUpperCase();
            nombre = nombre.toUpperCase();
            a_paterno = a_paterno.toUpperCase();
            a_materno = a_materno.toUpperCase();
            colonia = colonia.toUpperCase();
            calle = calle.toUpperCase();
            referencia = referencia.toUpperCase();
            escuela = escuela.toUpperCase();
            String[] param =
{rfc,nombre,a_paterno,a_materno,estado,distrito,colonia,calle,referenc
ia,cp,e_mail,telefono,nivel,carrera,escuela,passwd,id};
            boolean transaccion =
Data.makeUpdate("modify_solicitante",param);
            //eliminando las habilidades anteriores
            transaccion = Data.makeUpdate("delete_habilidades",id);
            for(int i=0; i<habilidades.length; i++){
                  habilidades[i] = habilidades[i].toUpperCase();
                  String[] param_habilidades =
{id,""+(i+1),habilidades[i]};
                  transaccion =
Data.makeUpdate("insert_habilidades",param_habilidades);
            }
            outHTML(tipo,login,response);
      }
/**
Metodo eliminaSoliciante, elimina de la Base de Datos  los datos del
solicitante, recibe un objeto HttPServletRequest y un objeto
HttpServletResponse
*/

      public void eliminaSolicitante(HttpServletRequest
request,HttpServletResponse response)throws Exception{
            String tipo = request.getParameter("tipo");
            String id = request.getParameter("id_solicitante");
            boolean delete = Data.makeUpdate("delete_habilidades",id);
            delete = Data.makeUpdate("delete_solicitante",id);
            outHTML(tipo,"none",response);
      }

/**
```

```
Metodo outHTML, ejecuta la repuesta del programa hacia el cliente,
recibe un objeto String tipo, un objeto Stirng login y un objeto
HttpServletResponse
*/

     public void outHTML(String tipo,String login,
HttpServletResponse response)throws Exception{
            PrintWriter pw = response.getWriter();
            if(tipo.equals("D") ){
                  pw.println("<html><head></head><body
onLoad=\"parent.location='/jsp/Entrada.jsp'\"></body></html>");
            }else{
                  pw.println("<html><head></head><body
onLoad=\"parent.location='/jsp/UploadFile.jsp?login="+ login
+"'\"></body></html>");
            }
            pw.flush();
            pw.close();
      }



}
```

REFERENCIAS

Bibliografía

Análisis de Diseño de Sistemas; Kendall Kenneth; Prentice Hall; Mexico 1990.

Análisis y Diseño de Sistemas de Información; Zen James; Mc Graw Hill; México 1991.

Estructura de Datos, Algoritmos, Abstracción y Objetos; Joyanes Aguilar Luis, I. Zahonero Martines Ignacio; Mc Graw Hill, España 1998.

Java en Pocas Palabras; Flanagan David; Mc Graw Hill – O'Reilly; México D.F. 1997.

Java Servlet Programming; Hunter Jason, O'Reilly, USA, 2001.

Javascript: The Definitive Guide; Flanagan David;

Modern Structured Analysis; Yourdon Edward; Prentice Hall; USA 1989

Sistemas de Información Administrativa; Murdick Robert; Prentice Hall Hispanoamericana; México 1986.

The Official Red Hat X86 Installation Guide; Red Hat, INC; USA 2002.

Referencias Electrónicas

http://jakarta.apache.org/tomcat/

http://java.sun.com/

http://wp.netscape.com/eng/mozilla/3.0/handbook/javascript/

http://www.apache.org/

http://www.postgresql.org/

http://www.rational.com/

http://www.redhat.com/