Universidad Autónoma Metropolitana
División de Ciencias Básicas e Ingeniería
Ingeniería en Computación

Sistema informático para manejo de curricula vitarum (CV)

Alumnos:

Cadena Méndez Carlos Enrique (204305763)

Víctor Hugo Peña Ramírez (204305323)

Asesorado por:

M. en C. Rafaela Blanca Silva López

No. Económico: 17114

Índice de contenidos

1. Sistema informático para manejo de curricula vitarum (CV) 7
1.1. Antecedentes . . . 7
1.2. Justificación . . . 7
1.3. Objetivos Generales . . . 9
1.4. Objetivos Particulares . . . 9

2. Curriculum Vitae 10
2.1. Definición de curriculum vitae . . . 10
2.1.1. Objetivos . . . 10
2.1.2. Tipos de Curriculum Vitae . . . 10
2.1.3. Que busca una empresa en un currículum y como los manipula? . . . 11
2.2. Estructura de un Curriculum Vitae . . . 11

3. XML 13
3.1. Introducción . . . 13
3.2. Definición de XML . . . 14
3.3. Antecedentes de XML . . . 14
3.4. Enfoque de XML . . . 15
3.5. Estructura de un documento XML . . . 15
3.6. Documentos XML bien formados . . . 16
3.7. Partes de un documento XML . . . 17
3.7.1. Prólogo . . . 17
3.7.2. Cuerpo . . . 17
3.7.3. Elementos . . . 17
3.7.4. Atributos . . . 17
3.7.5. Entidades predefinidas . . . 18
3.7.6. Secciones CDATA . . . 18
3.7.7. Comentarios . . . 18
3.8. Validez . . . 18
3.9. Document type definition (DTD) . . . 18
3.9.1. Ejemplo DTD . . . 18
3.10. XML Schemas (XSD) . . . 20
3.10.1. Componentes . . . 20
3.10.2. Estructura . . . 20
3.10.3. XML namespaces . . . 22
3.10.4. Declaración de un namespace . . . 22
3.11. Ventajas de los Schemas frente a los DTDs . . . 23

4. Diseño 24
4.1. Casos de Uso . . . 24
4.1.1. Caso de Uso: Registrar . . . 24
4.1.2. Caso de Uso: Validar Cuenta . . . 24
4.1.3. Caso de Uso: Editar Currículum . . . 25
4.1.4. Caso de uso: Personalizar Currículum . . . 27
4.1.5. Caso de Uso: Generar Currículum . . . 28
4.1.6. Caso de uso: Seleccionar Plantilla . . . 29
4.1.7. Caso de Uso: Generar Reportes . . . 29
4.1.8. Caso de Uso: Administrar Usuarios . . . 30

4.2. Diagramas de Clases . . . 32
4.2.1. Usuarios, Menú y Contacto . . . 32
4.2.2. Currículum . . . 32
4.2.3. Personal . . . 32
4.2.4. Dirección . . . 32
4.2.5. Teléfono . . . 32
4.2.6. Correo electrónico . . . 32
4.2.7. Escolaridad . . . 32
4.2.8. Experiencia Laboral . . . 32
4.2.9. Afición . . . 32
4.2.10. Curso . . . 32
4.2.11. Documentación . . . 32
4.2.12. Habilidad . . . 32
4.2.13. Idioma . . . 32
4.2.14. Publicación . . . 32

5. Navegación y Base de Datos **44**
5.0.15. Diagrama de Navegación . . . 44
5.0.16. Diagrama Entidad Relación de Base de Datos . . . 45
5.0.17. Esquema Físico de Base de Datos . . . 45

6. Manual de Usuario **52**
6.1. Información General . . . 52
6.1.1. Módulos . . . 52
6.1.2. Usuarios . . . 52
6.2. Estructura del Sitio . . . 53
6.3. Inicio . . . 53
6.3.1. Registro . . . 54
6.3.2. Acerca . . . 54
6.3.3. Contacto . . . 54
6.4. Inicio de Administrador . . . 54
6.5. Usuarios . . . 55
6.5.1. Administrar Usuarios . . . 55
6.5.2. Alta de Usuarios . . . 55
6.6. Inicio de Supervisor . . . 55
6.6.1. Búsquedas . . . 55
6.7. Inicio de Currículum . . . 55
6.8. Formulario . . . 56
6.9. Personalizar . . . 56
6.10. Plantillas . . . 56
6.11. Generar . . . 56

7. Manual Técnico **61**
7.1. Instalación y Configuración de Requerimientos . . . 61
7.1.1. Instalación JDK 6 . . . 61
7.1.2. Instalación Apache Tomcat 6.0.18 . . . 61

8. Conclusiones **62**

9. Bibliografía **64**

Índice de códigos

3.1. Ejemplo de código HTML 13
3.2. Comparación entre una página HTML y un documento XML 14
3.3. Ejemplo de la estructura de un documento XML 15
3.4. Ejemplo de un documento XML bien formado. 17
3.5. Ejemplo DTD para validar documentos XML basados en una currícula. 19
3.6. Ejemplo de documento XML que hace uso del DTD currícula de la figura 3.5 19
3.7. Ejemplo de estructura de un documento esquema vacío 20
3.8. Ejemplo de esquema XML 21
3.9. Ejemplo de problemas que tiene un documento XML sin la utilización de namespace 22
3.10. Ejemplo de la declamación de un namespace en un documento XML. 23
3.11. Ejemplo de utilización de un namespace en XML 23

5.1. Esquema Físico de Base de Datos 51

Índice de figuras

2.1. Tipos de Curriculum Vitae . . . 10

3.1. Estructura de árbol que se genera a partir del documento XML . . . 16

4.1. Caso de Uso Registrar . . . 25
4.2. Caso de Uso Validar Cuenta . . . 25
4.3. Caso de Uso Editar Currículum . . . 27
4.4. Caso de Uso Personalizar Currículum . . . 28
4.5. Caso de Uso Generar Currículum . . . 28
4.6. Caso de Uso Seleccionar Plantilla . . . 29
4.7. Caso de Uso Generar Reportes . . . 30
4.8. Caso de Uso Administrar Usuarios . . . 32
4.9. Diagrama de Clases Usuarios y Menú . . . 33
4.10. Diagrama de Clases Usuarios y Menú . . . 34
4.11. Diagrama de Clases Currículum . . . 35
4.12. Diagrama de Clases Personal . . . 35
4.13. Diagrama de Clases Dirección . . . 36
4.14. Diagrama de Clases Teléfono . . . 37
4.15. Diagrama de Clases Correo electrónico . . . 37
4.16. Diagrama de Clases Escolaridad . . . 38
4.17. Diagrama de Clases Experiencia Laboral . . . 39
4.18. Diagrama de Clases Afición . . . 40
4.19. Diagrama de Clases Curso . . . 41
4.20. Diagrama de Clases Documentación . . . 42
4.21. Diagrama de Clases Habilidad . . . 42
4.22. Diagrama de Clases Idioma . . . 43
4.23. Diagrama de Clases Publicación . . . 43

5.1. Mapa del Sitio . . . 44
5.2. Diagrama Entidad Relación . . . 45

6.1. Estructura del Sitio . . . 52
6.2. Sección Superior . . . 53
6.3. Sección Inferior . . . 53
6.4. Inicio Superior . . . 53
6.5. Inicio Inferior . . . 54
6.6. Registro . . . 54
6.7. Acerca . . . 54
6.8. Contacto . . . 55
6.9. Inicio de Administrador Superior . . . 55
6.10. Inicio de Administrador Inferior . . . 56
6.11. Usuarios Superior . . . 56
6.12. Usuarios Inferior . . . 57
6.13. Administrar Usuarios . . . 57
6.14. Alta de Usuarios . . . 57
6.15. Búsquedas Superior . . . 57
6.16. Búsquedas Inferior . . . 58
6.17. Inicio de Currículum Superior . . . 58
6.18. Inicio de Currículum Inferior . . . 58

6.19. Formulario 58
6.20. Formulario 59
6.21. Formulario 59
6.22. Personalizar 59
6.23. Personalizar 59
6.24. Plantillas 60
6.25. Generar 60

Capítulo 1

Sistema informático para manejo de curricula vitarum (CV)

1.1. Antecedentes

Curriculum vitae es un concepto latino que significa "carrera de la vida". Surgió en contraposición y por analogía a cursus honorum, que se utilizaba para denominar la carrera profesional de los magistrados romanos. Como una forma de simplificar el concepto, suele utilizarse sólo el término currículum, incluso puede usarse la abreviatura C.V.

En la actualidad, la palabra currículum permite referirse al conjunto de experiencias laborales y educacionales de un sujeto. El currículum resulta un requisito casi ineludible a la hora de presentarse para solicitar un empleo.

Es común enfrentar la tarea de redactar un currículum cuando se solicita un empleo y es claro que sería de gran ayuda tener un sistema que sirva como guía en el proceso de su elaboración, ya que muchas veces no se tiene idea de la información que la empresa requiere para poder hacer un buen análisis del perfil.

La mayoría de las organizaciones que reúnen currícula para buscar el mejor perfil, asignan a una persona para que revise la información y determine quién tiene las aptitudes que requiere la empresa.

Para minimizar el tiempo de búsqueda de una persona con ciertas aptitudes, las empresas han optado por organizar los currícula tomando en cuenta la información general que contienen, por ejemplo: se pueden clasificar por el área de trabajo, la experiencia laboral, las fechas, etc. Esta clasificación ayuda a facilitar la búsqueda, sin embargo para realizarla se debe de revisar manualmente cada uno de los currícula; esto es lo que se desea evitar para agilizar todo el proceso.

Existen algunos sistemas que intentan resolver este problema homogenizando la estructura del currículum, para mantener el orden de la información y aplicar un estilo personalizado a su presentación. Uno de estos sistemas es llamado CVONLINE , un recurso en línea que permite crear un currículum, además de enviarlo a un determinado correo o generar un archivo.

Este servicio facilita la tarea de elaboración, pero no permite la administración de la información que se ha introducido. Esta solución informática fue creada pensando en usuarios que desean crear su currículum, sin embargo no tiene funcionalidad en las empresas u organizaciones que están buscando administrar toda esta información.

En este proyecto se tomarán en cuenta las dos partes, por un lado se podrá editar y diseñar un currículum, y por el otro se manipulará y administrará el mismo.

1.2. Justificación

En la actualidad, las empresas u organizaciones que publican vacantes para ocupar un puesto de trabajo, piden como requisito fundamental presentar un currículum en donde se encuentre la información resumida que

describa las aptitudes y el perfil laboral de los aspirantes a la vacante.

Este problema se complica aún más cuando los currícula tienen características totalmente diferentes, como por ejemplo: cuando existen currícula en diferentes idiomas o cuando tienen información que se presenta en diferente orden.

Estos problemas se pueden resolver por medio del sistema informático que estamos proponiendo, el cual será un proyecto de software libre que controlará, administrará y manipulará toda la información de una forma que facilite la elaboración y el diseño, así como también la búsqueda de información en un conjunto de currícula.

La implementación de un sistema de administración de currícula optimizará el tiempo de búsqueda de información, que ayudará a elegir rápidamente el currículum que más se adapte al perfil solicitado por la empresa; este proceso será mas eficiente y exacto.

El proyecto necesita un modelo de datos que sea rápido y eficaz a la hora de realizar consultas y mostrar resultados, así como un diseño de módulos e interfaz que sea eficiente y claro, tanto para el usuario, como para los desarrolladores. Estas actividades sólo pueden realizarse correctamente por un Ingeniero en computación.

1.3. Objetivos Generales

Diseñar e implementar un sistema de información que controle currícula vitarum (CV) y minimice el tiempo de búsqueda de información en una organización.

1.4. Objetivos Particulares

- Determinar las necesidades para el manejo de currícula vitarum en una empresa de consultoría, donde continuamente se requiere generar diferentes formatos de estos con diversos enfoques y en diferentes idiomas.
- Diseñar un sistema que cubra con las necesidades planteadas en el análisis anterior, identificando la interacción entre los módulos y el almacenamiento en los repositorios de datos.
- Investigar las especificaciones del lenguaje de programación XML (Extensible Markup Language) con el fin de generar los repositorios de datos y se enlazarán con los módulos del sistema.
- Diseñar una base de datos que agilice y optimice las búsquedas con un formato óptimo para el almacenamiento de datos semi-estructurados basados en XML.
- Diseñar las plantillas que se utilizarán para generar un currículum vitae.
- Diseñar las plantillas que se utilizarán para generar reportes de las búsquedas de información.
- Implementar el diseño del sistema y las plantillas que se utilizarán para generar reportes de las búsquedas de información, utilizando tecnologías abiertas y/o estándares.
- Implementar el diseño de la base de datos y las plantillas que se utilizaran para generar un currículum vital utilizando tecnologías abiertas y/o estándares.
- Documentar las diferentes etapas del desarrollo.

Capítulo 2

Curriculum Vitae

2.1. Definición de curriculum vitae

Curriculum vitæ es un concepto latino que significa "carrera de la vida". Surgió en contraposición y por analogía a cursus honorum, que se utilizaba para denominar la carrera profesional de los magistrados romanos. Como una forma de simplificar el concepto, suele utilizarse sólo el término currículum, incluso puede usarse la abreviatura C.V.

2.1.1. Objetivos

- Dar a conocer a la empresa y al futuro empleador los datos básicos de la persona que lo envía en el ámbito profesional.
- Preseleccionar de candidatos para las pruebas profesionales.
- Servir de apoyo en las entrevistas de selección.
- Conformar un módulo para la gestión de conocimiento de la empresa.

2.1.2. Tipos de Curriculum Vitae

Fundamentalmente existen dos tipos de currículum:

Tipo		Cronológico	Funcional
Información		Ordenado por fechas: ascendente o descendente	Ordenado por funciones desarrolladas
Ventajas		Información detallada por años	Resalta trabajos acordes con el puesto
Utilizar	Cuando	Poca experiencia	Mucha experiencia, cambios frecuentes en el trabajo
	Quien	Recién titulados	Personas con amplio historial formativo y profesional

Figura 2.1: Tipos de Curriculum Vitae

2.1.3. Que busca una empresa en un currículum y como los manipula?

Las empresas tomaran en cuenta diversos aspectos que darán como consecuencia la elección de solo uno de los aspirantes al puesto.

- **Buena presentación:** Este documento es el primer contacto con la empresa, se debe dar una buena impresión. Las empresas buscan seriedad por lo tanto se debe elaborar un documento impecable, sin tachaduras ni borrones.
- **Brevedad:** Las empresas buscan que se les proporcione información interesante y útil ya que lo más seguro es que la persona responsable de selección haya recibido muchos CV.
- **Concisión y claridad:** También por cuestiones de tiempo y de cantidad de información las empresas requieren que estos documentos utilicen un estilo directo y que se anoten con precisión nombres de empresas, cargos, fechas, títulos, etc.

Para minimizar el tiempo de búsqueda de una persona con ciertas aptitudes, las empresas han optado por organizar los currícula tomando en cuenta la información general que contienen, se pueden clasificar por:

- Área de trabajo
- Experiencia laboral
- Fechas
- Educación
- Profesión

Esta clasificación ayuda a facilitar la búsqueda, sin embargo para realizarla se debe de revisar manualmente cada uno de la currícula; esto es lo que se desea evitar para agilizar todo el proceso. En este proyecto se podrá editar y diseñar un currículum, y por el otro se manipulará y administrará el mismo.

2.2. Estructura de un Curriculum Vitae

Un currículum vitae es la herramienta más importante que una persona tiene para promocionarse y desarrollar una carrera profesional, esto desde el punto de vista de la persona que busca empleo. También es el modo que tienen los empleadores de conocer y escoger de entre sus competidores al futuro personal. Es el modo en que las personas pueden abrirse las puertas y por lo tanto su elaboración requiere de tiempo y reflexión.

A través del currículum, lo que la persona desea es darse a conocer y conseguir una entrevista. Este documento muestra el historial académico y profesional, es una tarjeta de presentación en la que la persona encargada de contratar al personal conocerá a los candidatos al puesto.

Como los aspirantes al puesto no suelen ser pocos, entonces las empresas buscan que un currículum sea breve, conciso, bien hecho, entendible y organizado, con el fin de hacer una elección correcta y en el menor tiempo posible.

Por lo tanto la idea de un buen currículum podría ser que debe estar bien estructurado con los datos perfectamente localizados y situados en el apartado correspondiente para su fácil localización. De esta forma se consigue una interpretación rápida y sencilla, sin dejar lugar a dudas sobre la adecuación del candidato.

Los apartados más comunes del Currículum Vitae son:

- Datos Personales
  - Nombre y Apellidos.
  - Dirección: Calle, número, piso, código postal, población y provincia.
  - Teléfono: Preferiblemente dos, el particular y otro de contacto.
  - E-mail.
- Formación Académica

- Título de Escuela Superior (Doctorados, Maestría, etc.): Título, Centro, Ciudad, Fechas de Inicio-Fin.
- Licenciatura Universitaria: Título, Centro, Ciudad, Fechas de Inicio-Fin.
- Estudios Oficiales.
- Se pueden añadir las Becas de Estudio recibidas en el apartado correspondiente.

- Formación Extra Académica
  - Cursos de Capacitación para la Profesión: Reflejar aquéllos cursos que tengan una especial relevancia para el puesto de trabajo que se solicita. Los cursos que se mencionen o bien deben tener un prestigio reconocido o que sean de larga duración. Indicar las horas de duración.
  - Cursos de Formación: Se puede añadir la participación en seminarios o jornadas, siempre y cuando éstos tengan relación directa con el puesto solicitado.
  - Cursos de Idiomas.
- Experiencia Profesional
  - Ocupación Actual: Nombre de la Empresa, Sector, Fechas de Permanencia, Denominación del Puesto y Funciones.
  - Prácticas en Empresas: Nombre de la Empresa, Sector, Fechas de Permanencia, Denominación del Puesto y Funciones.
  - Experiencias Anteriores o Trabajos Realizados en períodos de estudio. Intenta relacionar la experiencia con el trabajo al que aspiras.
  - Puedes utilizar un orden cronológico directo, desde la primera empresa a la última, o un orden cronológico inverso, desde la última a la primera; dependerá de la experiencia que te interese resaltar.
- Idiomas
  - Nivel de Conocimiento y Capacidades. De cada idioma indica el grado de dominio: Bajo (sólo sabes traducir), Medio (hablas y escribes con dificultad) y Alto (hablas y escribes correctamente).
  - Debes ser realista, pues en muchas empresas realizan una prueba de idiomas antes de la contratación
- Publicaciones y Otras Actividades Profesionales
  - Se deben seguir las mismas reglas que para la experiencia profesional.
- Otros Datos de Interés
  - Este apartado es opcional. Puedes incluir datos interesantes a destacar, pero difícilmente encuadrables en los apartados anteriores, tales como: Movilidad geográfica, carnet de conducir, vehículo propio.

Capítulo 3

XML

3.1. Introducción

Por lo general la mayoría de las personas en promedio han tenido algún contacto con las computadoras e Internet, esto quiere decir que han tenido contacto con la manipulación del Web que cuenta con uno de los principales recursos disponibles en Internet que son las páginas HTML (Hypertext Markup Language). Estas páginas están construidas principalmente por instrucciones llamadas etiquetas (tags) que son las encargadas de describir la estructura, diseño y presentación del contenido que se muestra en una página Web.

El lenguaje HTML nos sirve para comprender mejor el significado de XML, podemos decir que una página desarrollada con el lenguaje HTML es un conjunto de etiquetas que encierran y separan cada uno de los elementos que conforman una página Web. Por ejemplo: Para crear una página debemos tener algunas etiquetas principales que nos separan cada uno de las partes que conforman a una página HTML.

```
1  <html>
2    <head>
3          <title>
4               Ejemplo Página HTML
5          </title>
6    </head>
7    <body>
8          <p>
9               Ejemplo HTML
10         </p>
11   </body>
12 </html>
```

Código 3.1: Ejemplo de código HTML

En el código 5.0.17 podemos ver como cada uno de los elementos principales que conforman a una página HTML están encerrado por las etiquetas características de HTML, esto nos permite separar y saber que tipo de etiquetas deben de ir dentro de otras, para formar una estructura anidada de etiquetas.

Una vez que hemos definido la estructura de una página HTML, podemos decir que HTML es un documento XML pero con restricciones, un documento XML al igual que HTML esta conformado por etiquetas pero estas etiquetas no deben de tener un léxico restringido como lo tiene HTML. Es decir, mientras la página HTML debe de tener estrictamente las etiquetas de <html>, </html>, <head>, </head>, <body>, </body>, etc. en XML se pueden definir las etiquetas que mejor especifiquen el contenido <documentoWEB>, </documentoWEB>, <principal>, </principal>, <contenido>, </contenido>, etc.

Este ejemplo solo es una analogía para poder entender mejor el concepto principal de XML, ya que un navegador Web no es capaz de interpretar nuestro documento XML para mostrarlo como una página HTML. Por lo tanto XML no es un lenguaje que nos ayude a mejorar las páginas Web, sino que es un lenguaje que

```
<html>
   <head>
         <title>
              Ejemplo Página HTML
         </title>
   </head>
   <body>
         <p>
              Ejemplo HTML
         </p>
   </body>
</html>
```

```
<documentoWEB>
     <principal>
          <titulo>
               Ejemplo Documento XML
          </titulo>
     </principal>
     <contenido>
          <parrafo>
               Ejemplo XML
          </parrafo>
     </contenido>
</documentoWEB>
```

Código 3.2: Comparación entre una página HTML y un documento XML

describe el tipo de información que se maneja en un documento.

3.2. Definición de XML

El nombre de XML se conforma de las sigla en inglés de Extensible Markup Language (lenguaje de marcas extensible), al igual que HTML es un lenguaje basado en marcas pero a diferencia de HTML es considerado un metalenguaje ya que es una forma de especificar lenguajes. Fue desarrollado por el World Wide Web Consortium (W3C) adaptando y simplificando el lenguaje SGML. Por lo tanto XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades.

3.3. Antecedentes de XML

En los años 70 la empresa IBM invento un lenguaje llamado GML (Generalized Markup Language), que surgió por la necesidad de almacenar grandes cantidades de información ordenadamente y con capacidad de interoperabilidad entre las aplicaciones. Este lenguaje gustó a la ISO, por lo que en 1986 trabajaron para normalizarlo, creando SGML (Standard Generalized Markup Language), capaz de adaptarse a un gran numero de aplicaciones y entornos de desarrollo. A partir de él se han creado otros sistemas para almacenar información de los cuales el más destacado ha sido la creación del lenguaje XML.

En el año 1989 Tim Berners Lee creó la Web, y junto con ella el lenguaje HTML. La especificación de este lenguaje se definió con ayuda del lenguaje SGML para que fuera un formato apegado a un estándar. Debido a esto fue adoptado rápidamente por la comunidad y diversas organizaciones creando sus propios visores de HTML (navegadores Web). Esto provoco una riña entre ellos por crear el visor HTML mas avanzado creando sus propias etiquetas, definiciones e instrucciones provocando un crecimiento descontrolado del lenguaje HTML.

El W3C en 1998 empezó el desarrollo de XML con la mentalidad de solucionar las carencias y defectos que tiene el lenguaje HTML como por ejemplo:

- En HTML se mezcla el contenido con los estilos que se aplican al texto y a la propia página.
- No permite la interoperabilidad entre las aplicaciones para el intercambio de información.
- La presentación de la información de una página Web depende del navegador utilizado.

Para hacer esto XML deja de lado muchas características de SGML que estaban pensadas para facilitar la escritura manual de documentos. XML en cambio está orientado a hacer las cosas más sencillas para los programas automáticos que necesiten interpretar el documento.

3.4. Enfoque de XML

Hoy en día existen diferentes paradigmas de programación como lo puede ser la programación estructurada o la programación orientada a objetos. XML es un lenguaje que propone la programación basada en documentos, es decir que un programa que recibe entradas y produce salidas, se puede cambiar a una arquitectura en donde se recibe un documento y como salida se produce otro documento que es estándar para que otras aplicaciones puedan utilizarlo y recibirlo como entrada. Como se ha mencionado una de los principales objetivos que se tiene en mente al utilizar XML es lograr la estandarización del intercambio de información, hoy en día la mayoría de las aplicaciones tienen el problema de no poder intercambiar información, esto es debido a que cada una de ellas tiene una manera propia de recibir datos y producir resultados. Este problema se puede solucionar utilizando documentos XML para intercambiar información.

3.5. Estructura de un documento XML

La tecnología XML busca dar solución al problema de expresar información estructurada de la manera más abstracta y reutilizable posible. Que la información sea estructurada quiere decir que se compone de partes bien definidas, y que esas partes se componen a su vez de otras partes. Con una estructura correctamente planteada se puede obtener un árbol bien definido con la estructura de las partes que conforman al documento. Estas partes se llaman elementos, y en XML se definen mediante etiquetas.

```
1 <curriculum>
2     <datosgenerales>
3         <nombre>
4             <apellidop>Perez </apellidop>
5             <apellidom>    Hernandez </apellidom>
6             <nombres>Juan Carlos </nombres>
7         </nombre>
8         <fechanac>
9             <dia>31</dia>
10            <mes>01</mes>
11            <anio>1985</anio>
12        </fechanac>
13    </datosgenerales>
14 </curriculum>
```

Código 3.3: Ejemplo de la estructura de un documento XML

Si observamos el diagrama de la figura 2.2 cada uno de los elementos que se presentan es una parte del documento, cada uno de estos elementos esta representado por un par de etiquetas en la figura 2.1. Una etiqueta consiste en una marca hecha para separar los elementos que se encuentran en el documento editado. Podemos definir cada elemento como un pedazo de información con un sentido claro y definido.

Las etiquetas tienen la forma <nombre> correspondencia < /nombre>, donde nombre es el nombre del elemento que se está señalando y correspondencia es la información que estará delimitada por las etiquetas. Las etiquetas pueden estar anidadas, pero toda etiqueta que se abra (<nombre>) debe de tener correspondencia con una etiqueta de cierre (< /nombre>) respetando el orden jerárquico.

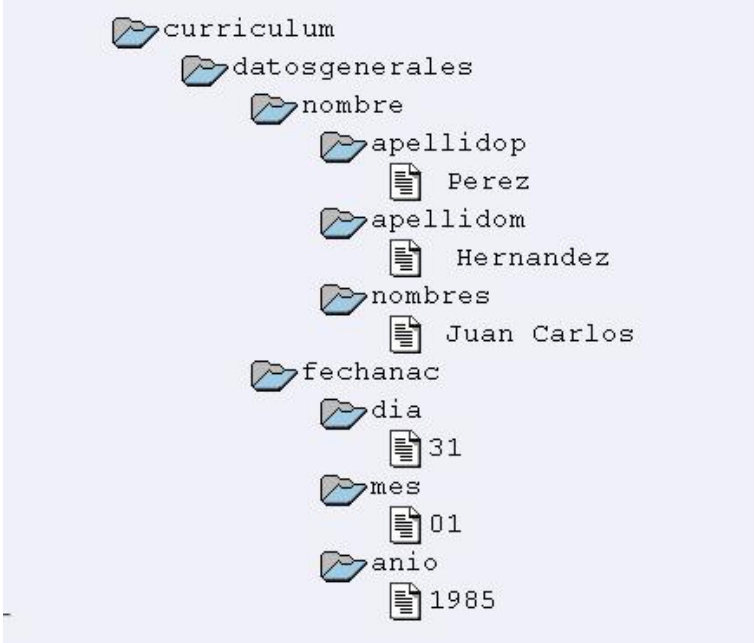

Figura 3.1: Estructura de árbol que se genera a partir del documento XML

3.6. Documentos XML bien formados

Cuando un documento XML cumple con todas las definiciones básicas de formato se dice que es un documento bien formado. Un documento bien formado es el que no tienen errores léxicos en lenguaje definido de XML y por lo tanto puede analizarse sintacticamente con un parser que cumpla con la sintaxis del estándar XML. Un documento XML puede cumplir con el requisito de ser bien formado, sin embargo existe otro requisito que debe de cumplir llamado validez del documento que se explica posteriormente.

Todos los documentos XML deben de cumplir como requisito mínimo que estén bien formados esto quiere decir que deben de cumplir lo siguiente:

- En algunas ocasiones no se utiliza DTD (Document Type Definition), en este caso el documento debe comenzar con una Declaración de Documento Standalone.
  - <?xml version="1.0" encoding='iso-8859-1' ? >
- Los documentos XML solo pueden tener un elemento raíz del que todos los demás sean parte.
- Todas las etiquetas deben estar equilibradas: Esto significa que todos los elementos que contengan datos de tipo carácter deben tener etiqueta de apertura y de cierre <nombre>< /nombre>.
  - <apellidop> Perez < /apellidop>
- Las etiquetas deben de estar correctamente anidadas dentro de otras respetando el orden de las etiquetas.
- En las etiquetas se pueden definir atributos, todos los valores de los atributos deben ir entrecomillados ya sea entre comillas simples o dobles.
  - <nombre='nombres'>
- Pueden existir elementos vacíos, esto es, que pueden tener etiqueta de apertura pero no de cierre. Esto es posible pero la etiqueta se de be de terminar con: '/ >'.
- Si, dentro de una etiqueta existe la necesidad de poner un símbolo como (¡, ¿, &) debe de colocarse como &lt, &amp, &gt respectivamente.

Los nombres de las etiquetas pueden ser alfanuméricos, comenzando con una letra.

```
1 <?xml version=''1.0''encoding='iso-8859-1' ?>
2 <curriculum>
3      <datosgenerales>
4           <nombre>
5                <apellidop>Perez </apellidop>
6                <apellidom>    Hernandez </apellidom>
7                <nombres>Juan Carlos </nombres>
8           </nombre>
9           <fechanac>
10               <dia>31</dia>
11               <mes>01</mes>
12               <anio>1985</anio>
13          </fechanac>
14     </datosgenerales>
15     <documentacion>
16          <rfc>peh850131</rfc>
17          <afilIMSS>4645648</afilIMSS>
18          <pasaporte>98789</pasaporte>
19          <licenciaM>
20               <licencia>si</licencia>
21               <clase>B</clase>
22               <numero>567575</numero>
23          </licenciaM>
24     </documentacion>
25 </curriculum>
```

Código 3.4: Ejemplo de un documento XML bien formado.

3.7. Partes de un documento XML

Un documento XML está formado por el prólogo y por el cuerpo del documento.

3.7.1. Prólogo

Aunque no es obligatorio, los documentos XML pueden empezar con unas líneas que describen la versión XML, el tipo de documento y otras cosas.

El prólogo contiene:

- Una declaración XML. Es la sentencia que declara al documento como un documento XML.
- Una declaración de tipo de documento. Enlaza el documento con su Schema para el soporte de las etiquetas.
- Uno o más comentarios e instrucciones de procesamiento.

3.7.2. Cuerpo

A diferencia del prólogo, el cuerpo no es opcional en un documento XML, el cuerpo debe contener un y solo un elemento raíz, característica indispensable también para que el documento esté bien formado.

3.7.3. Elementos

Los elementos XML pueden tener contenido (más elementos, caracteres o ambos), o bien ser elementos vacíos.

3.7.4. Atributos

Los elementos pueden tener atributos, que son una manera de incorporar características o propiedades a los elementos de un documento. Deben ir entre comillas.

3.7.5. Entidades predefinidas

Entidades para representar caracteres especiales para que, de esta forma, no sean interpretados como marcado en el procesador XML.

3.7.6. Secciones CDATA

Es una construcción en XML para especificar datos utilizando cualquier carácter sin que se interprete como marcado XML. Solo se utiliza en los atributos. Permite que caracteres especiales no rompan la estructura. Ejemplo:

<![CDATA["contenido especial: áéíóúñ& "]]>

3.7.7. Comentarios

Comentarios a modo informativo para el programador que han de ser ignorados por el procesador.
Los comentarios en XML tienen el siguiente formato:

<! − −− Comentario − − − >

3.8. Validez

Como se ha expuesto, un documento "bien formado" solamente se refiere a su estructura sintáctica básica, es decir, que se componga de elementos, atributos y comentarios que se especifican en la sintaxis de XML. La necesidad que existe por validar un documento XML es por que no solo se debe de verificar la sintaxis del documento, si no que también la estructura de los elementos presentes, ya que cada aplicación que utilice estos documentos XML debe de tener definida una estructura determinada.

Por ejemplo una aplicación puede pedir como entrada un documento XML en el cual sea obligatorio llevar el campo Nombre si a esta aplicación se le proporciona un documento XML sin que especifique este campo, se validara y no tomara este documento aunque sea bien formado por que debe de cumplir con las validaciones que solicita la aplicación.

Esta relación entre elementos se especifica en un documento externo o definición, expresada como DTD (Document Type Definition) o como XSchema. Crear una definición equivale a crear un nuevo lenguaje de marcado, para una aplicación específica.

3.9. Document type definition (DTD)

La DTD define los tipos de elementos, atributos y entidades permitidas, y puede expresar algunas limitaciones para combinarlos. Los documentos XML que se ajustan a su DTD son denominados válidos.

Siglas en inglés de Document Type Definition. La definición de tipo de documento (DTD) es una descripción de estructura y sintaxis de un documento XML o SGML que especifica las restricciones que se deben de cumplir para validar cada uno de los documentos XML. Su función básica es la descripción del formato de datos, para usar un formato común y mantener la consistencia entre todos los documentos que utilicen la misma DTD. De esta forma, dichos documentos, pueden ser validados, conocen la estructura de los elementos y la descripción de los datos que trae consigo cada documento, y pueden además compartir la misma descripción y forma de validación dentro de un grupo de trabajo que usa el mismo tipo de información.

Una DTD describe:

- Elementos: indican qué etiquetas son permitidas y el contenido de dichas etiquetas.
- Estructura: indica el orden en que van las etiquetas en el documento.
- Anidamiento: indica qué etiquetas van dentro de otras.

3.9.1. Ejemplo DTD

Un ejemplo de una DTD XML muy simple, para describir una lista de personas:

```
<ELEMENT Curricula (curriculum*)>
<ELEMENT curriculum (datosgenerales,datosparticulares?)>
<ELEMENT datosgenerales (nombre,fechanac)>
<ELEMENT nombre (apellidop,apellidom,nombres)>
<ELEMENT fechanac (dia,mes,anio)>
<ELEMENT datosparticulares (deporte)>
<ELEMENT apellidop (#PCDATA) >
<ELEMENT apellidom (#PCDATA) >
<ELEMENT nombres (#PCDATA) >
<ELEMENT dia (#PCDATA) >
<ELEMENT mes (#PCDATA) >
<ELEMENT anio (#PCDATA) >
<ELEMENT deporte (#PCDATA) >
```

Código 3.5: Ejemplo DTD para validar documentos XML basados en una currícula.

```
<?xml version=''1.0''encoding=''ISO-8859-1''?>
<DOCTYPE curricula SYSTEM ''curricula.dtd''>
<Curricula>
    <curriculum>
        <datosgenerales>
            <nombre>
                <apellidop>Perez </apellidop>
                <apellidom>    Hernandez </apellidom>
                <nombres>Juan Carlos </nombres>
            </nombre>
            <fechanac>
                <dia>31</dia>
                <mes>01</mes>
                <anio>1985</anio>
            </fechanac>
        </datosgenerales>
        <datosparticulares>
            <deporte>futbol</deporte>
        </datosparticulares>
    </curriculum>
</Curricula>
```

Código 3.6: Ejemplo de documento XML que hace uso del DTD currícula de la figura 3.5

3.10. XML Schemas (XSD)

Un Schema es algo similar a un DTD. Define qué elementos puede contener un documento XML, cómo están organizados y qué atributos y de qué tipo pueden tener sus elementos.

El XML Schema es un lenguaje que nos describe y restringe formalmente la estructura abstracta que debe cumplir un documento de XML. Es una evolución de un DTD para evitar las deficiencias del DTD y realizar un esquema más abstracto.

Algunas de las ventajas de Xschema con respecto a DTD son:

- Usan sintaxis de XML, logrando abstraer más la información.
- Permiten especificar los tipos de datos que el documento XML podrá utilizar, es decir, en el documento esquema se puede definir el tipo del contenido de un elemento y especificar si debe ser un número, cadena, texto, fecha, etc.
- Permiten utilizar diferentes espacios de nombres (namespace) para tener diversas validaciones para no degenerar la estructura del XML a pesar de la manipulación de diferentes personas.
- Debido a que su sintaxis esta basada en XML puede ser extensible.

3.10.1. Componentes

Los documentos esquema se construyen a partir de diferentes tipos de componentes:

- Elemento. (element) Es cada uno de los datos abstractos que entraran en las restricciones del documento XML. Los elementos pueden tener tipos de datos simples; pueden ocurrir más de una vez y/o pueden tener tipos complejos.
- Atributo. (attribute) Es una manera de incorporara características o propiedades a los elementos definidos en un documento esquema.
- Tipo Simple. (simpleType) Los tipos simples contienen datos, pero no pueden contener atributos o sub-elementos.
- Tipo Complejo. (complexType) Los tipos complejo permiten incluir sub-elementos y/o atributos y se definen enumerando los elementos incluidos. El contenido de los tipo complejo puede ser:
  - Contenido Simple (simpleContent) Al igual que el tipo simple este tipo de contenido contiene datos sin embargo no pueden contener atributos o sub-elementos.
  - Contenido Complejo (complexContent) Dentro de los componentes tipo complejo también puede haber tipos complejos que almacenen sub-elementos y/o atributos.
  - Secuencia (sequence) Se utiliza dentro de los tipos complejo para almacenar un grupo o secuencia de elementos.

3.10.2. Estructura

Ejemplo de la estructura de un documento esquema vacío:

```
1 <?xml version=''1.0''encoding=''ISO-8859-1''?>
2 <xsd:schema xmlns:xsd=''http://www.w3.org/2001/XMLSchema''version=''0.1''xml:lang=''es''>
3 </xsd:schema>
```

Código 3.7: Ejemplo de estructura de un documento esquema vacío

Ejemplo de la estructura de un documento esquema:

```
<?xml version=''1.0''encoding=''UTF-8''?>
<xsd:schema xmlns:xsd=''http://www.w3.org/2001/XMLSchema''>
    <xsd:element name=''Curricula''>
         <xsd:complexType>
            <xsd:sequence>
                 <xsd:element ref=''Curriculum''maxOccurs= ''1''/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

     <xsd:element name=''Curriculum''>
         <xsd:complexType>
            <xsd:sequence>
                 <xsd:element ref=''DatosGenerales''maxOccurs= ''1''/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

   <xsd:element name = ''DatosGenerales''>
         <xsd:complexType>
           <xsd:sequence>
                   <xsd:element ref=''Nombre''maxOccurs= ''1''/>
              <xsd:sequence>
         <xsd:complexType>
     </xsd:element>

     <xsd:element name = ''Nombre''>
         <xsd:complexType>
                  <xsd:sequence>
                       <xsd:element name = ''apellidop''type = ''xsd:
                       string''maxOccurs = ''1''/>
                       <xsd:element name = ''apellidom''type = ''xsd:
                       string''maxOccurs = ''1''/>
                       <xsd:element name = ''nombres''type = ''xsd:string''
                       maxOccurs = ''1''/>
                  <xsd:sequence>
         <xsd:complexType>
     </xsd:element>
</xsd:schema>
```

Código 3.8: Ejemplo de esquema XML

3.10.3. XML namespaces

Por lo general un documento XML esta creado con el fin de compartir e intercambiar información, el problema que puede surgir por el intercambio del mismo documento es que al manipular el documento se pueden ir definiendo etiquetas, si no se controla la definición de elementos y etiquetas se puede ocasionar un caos al tratar de parsear etiquetas con el mismo nombre pero que tienen diferente significado.

Podemos entender mejor el problema con un ejemplo: En la figura 4.1 se muestra un ejemplo de los problemas que puede ocasionar la falta de un namespace. Se puede observar en la figura que existen etiquetas repetidas que tienen diferente significado. Es decir la etiqueta nombre se encuentra definida dos veces, solo que en la primera definición se refiere al nombre de una persona y en la segunda definición se refiere al nombre de una empresa. Lo mismo ocurre con las etiquetas de dirección, calle, numero, colonia, cp. Ya que mientras unas hacen referencia a la dirección de una persona otras se refieren a la dirección de una empresa.

Es por esto que la W3C recomienda la utilización de espacio de nombres para proporcionar elementos y atributos con nombre único para resolver la ambigüedad existente entre elementos y atributos que tengan el mismo nombre.

```
<?xml version=''1.0''encoding=''ISO-8859-1''?>
<DOCTYPE curricula SYSTEM ''curricula.dtd''>
<Curricula>
    <curriculum>
        <datosgenerales>
            <nombre>
                <apellidop>Perez </apellidop>
                <apellidom>    Hernandez </apellidom>
                <nombres>Juan Carlos </nombres>
            </nombre>
            <direccion>
                <calle>Fco. Marquez</calle>
                <numero>44</numero>
                <colonia>Niños Heroes</colonia>
                <cp>54870</cp>
            </direccion>
        </datosgenerales>
        <experienciaLaboral>
            <empresa>
                <nombre>
                    Universidad Autonoma Metropolitana
                </nombre>
                <direccion>
                    <calle>Av. San Pablo</calle>
                    <numero>180</numero>
                    <colonia>Reynosa Tamaulipas</colonia>
                    <cp>02200</cp>
                </direccion>
            </empresa>
        </experienciaLaboral>
    </curriculum>
</Curricula>
```

Código 3.9: Ejemplo de problemas que tiene un documento XML sin la utilización de namespace

3.10.4. Declaración de un namespace

Un espacio de nombres se declara utilizando el atributo reservado de XML xmlns, cuyo valor debe ser un URI (Uniform Resourse Identifier) como por ejemplo:

```
<usr:usuario xmlns:usr='http://mipagina.com/proyecto/Espacio_de_nombres_XML/usuario'
             xmlns:emp='http://mipagina.com/proyecto/Espacio_de_nombres_XML/pedido'>

```

Código 3.10: Ejemplo de la declamación de un namespace en un documento XML.

Una vez que se hay declarado el espacio de nombres de XML cada una de las etiquetas definidas en el documento debe de empezar con el prefijo declarado. Cabe destacar que el URI utilizado no es funcional como hipervínculo, ya que solo sirve para el parser de XML, se recomienda utilizar como identificador una URI debido a que es difícil que diferentes personas puedan coincidir en la definición del espacio de nombres.

Una vez definido el espacio de nombres el documento XML puede tener una estructura como la siguiente:

```
<?xml version=''1.0''encoding=''ISO-8859-1''?>
 <usr:usuario xmlns:usr='http://mipagina.com/proyecto/Espacio_de_nombres_XML/usuario'
              xmlns:emp='http://mipagina.com/proyecto/Espacio_de_nombres_XML/pedido'>
<Curricula>
    <curriculum>
        <usr:datosgenerales>
            <usr:nombre>
                <usr:apellidop>Perez </apellidop>
                <usr:apellidom>     Hernandez </apellidom>
                <usr:nombres>Juan Carlos </nombres>
            </usr:nombre>
            <usr:direccion>
                <usr:calle>Fco. Marquez</calle>
                <usr:numero>44</numero>
                <usr:colonia>Niños Heroes</colonia>
                <usr:cp>54870</cp>
            </usr:direccion>
        </usr:datosgenerales>
        <emp:experienciaLaboral>
            <emp:empresa>
                <emp:nombre>
                    Universidad Autonoma Metropolitana
                </emp:nombre>
                <emp:direccion>
                    <emp:calle>Av. San Pablo</emp:calle>
                    <emp:numero>180</emp:numero>
                    <emp:colonia>Reynosa Tamaulipas</emp:colonia>
                    <emp:cp>02200</emp:cp>
                </emp:direccion>
            </emp:empresa>
        </emp:experienciaLaboral>
    </curriculum>
</Curricula>
```

Código 3.11: Ejemplo de utilización de un namespace en XML

3.11. Ventajas de los Schemas frente a los DTDs

Usan sintaxis de XML, al contrario que los DTDs, permiten especificar los tipos de datos y son extensibles.

Capítulo 4

Diseño

4.1. Casos de Uso

4.1.1. Caso de Uso: Registrar

Actores: Usuario.
Descripción: Cuando el usuario entra por primera vez al sistema es necesario que registre sus datos para obtener una cuenta.
Pre-condiciones: El usuario debe haber accesado al sistema por medio de un explorador Web.
Post-condiciones: El usuario estará registrado en el sistema y podrá utilizarlo.

Flujo Principal:
1. El usuario inicia el sistema.
2. El usuario selecciona la opción de registro.
3. El sistema muestra un formulario para ingresar nombre de usuario, contraseña y confirmación de contraseña.
4. El usuario llena el formulario con sus datos.
5. El sistema valida la información de los campos que proporcionó el usuario.
6. El sistema informa que el usuario se ha registrado con éxito.

Flujo Alternativo:
a. En cualquier momento - Falla el sistema.
    a.1 Se cancela la operación en el sistema.
    a.2 Se reinicia el sistema.
5.1. El sistema no reconoce los datos del usuario.
    5.1.1. El usuario no da al sistema todos los datos requeridos para realizar la operación de alta.
        5.1.1.1 El sistema informa al usuario que faltaron datos.
        5.1.1.2 El sistema solicita al usuario que le entregue los datos necesarios.
        5.1.1.3 El sistema regresa al estado 4.
    5.1.3. El usuario ya está registrado.
        5.1.3.1 El sistema informa al usuario que esa cuenta de usuario ya existe.
        5.1.3.2 El sistema solicita al usuario que ingrese otro usuario.
        5.1.3.3 El sistema regresa al estado 4.

4.1.2. Caso de Uso: Validar Cuenta

Actores: Administrador, Usuario y Supervisor.
Descripción: Se valida la cuenta de usuario para verificar que está registrado en el sistema.
Pre-condiciones: El usuario debe estar registrado en el sistema.
Post-condiciones: El usuario ingreso con éxito al sistema y se muestran las opciones de acuerdo a su tipo de usuario.

Flujo Principal:
1.El usuario inicia el sistema.

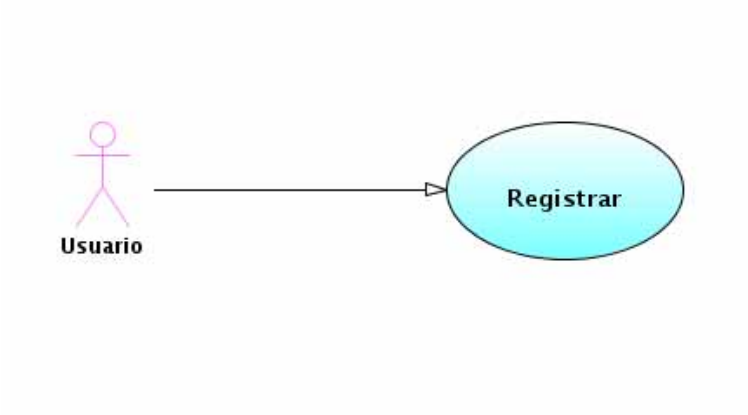

Figura 4.1: Caso de Uso Registrar

2.El sistema pide el usuario que ingrese cuenta de usuario y contraseña.
3.El usuario escribe su usuario y contraseña.
4.El sistema valida el usuario y contraseña.

Flujo Alternativo:
a. En cualquier momento - Falla el sistema.
    a.1 Se cancela la operación en el sistema.
    a.2 Se reinicia el sistema.
4.1. El sistema no reconoce la identidad del usuario.
    4.1.1. El sistema pregunta al usuario si desea crear una cuenta.
    4.1.2. El sistema muestra la forma de registro.
    4.1.3. El usuario crea una cuenta de usuario.
    4.1.4 El sistema regresa al estado 4.

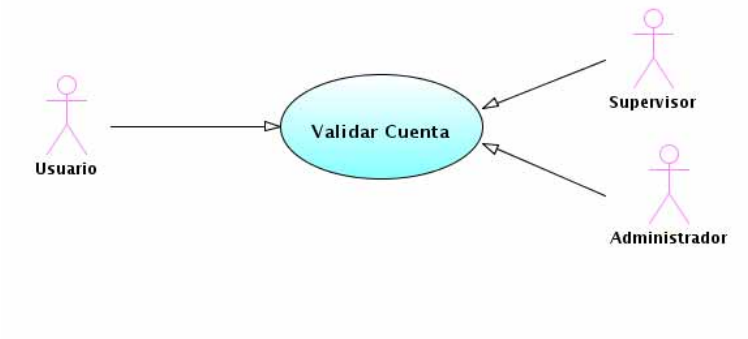

Figura 4.2: Caso de Uso Validar Cuenta

4.1.3. Caso de Uso: Editar Currículum

Actores: Usuario.
Descripción: El usuario ingresa toda la información que forma parte de su currículum.
Pre-condiciones: El usuario ingresó su cuenta de usuario y contraseña válidos.
Post-condiciones: Se crean y almacenan los registro necesarios para formar parte de un currículum con la información que proporcionó el usuario.

Flujo Principal:

1.El usuario selecciona la opción de crear currículum.

2.El sistema muestra un formulario para que el usuario ingrese sus datos.

3. El administrador despliega una de las secciones para editar (información personal, aficion, curso, dirección, documentaci

- 3.1. El usuario elige editar su información personal.
  - 3.1.1. El usuario realiza cambios en el formulario con la información que desea cambiar.
  - 3.1.2. El sistema valida la información de los campos que proporcionó el usuario.
  - 3.1.3. El sistema actualiza la información personal.
  - 3.1.4. El sistema confirma al usuario que se ha actualizado la información personal.
- 3.2. El usuario elige agregar una dirección.
  - 3.2.1. El usuario llena el formulario con la información de direccion.
  - 3.2.2. El sistema valida la información de los campos que proporcionó el usuario.
  - 3.2.3. El sistema da de alta la dirección.
  - 3.2.4. El sistema confirma al usuario que se ha dado de alta la dirección.
- 3.3. El usuario elige agregar un teléfono.
  - 3.3.1. El usuario llena el formulario con la información de teléfono.
  - 3.3.2. El sistema valida la información de los campos que proporcionó el usuario.
  - 3.3.3. El sistema da de alta el teléfono.
  - 3.3.4. El sistema confirma al usuario que se ha dado de alta el teléfono.
- 3.4. El usuario elige agregar un correo electrónico.
  - 3.4.1. El usuario llena el formulario con la información de correo electrónico.
  - 3.4.2. El sistema valida la información de los campos que proporcionó el usuario.
  - 3.4.3. El sistema da de alta el correo electrónico.
  - 3.4.4. El sistema confirma al usuario que se ha dado de alta el correo electrónico.
- 3.5. El usuario elige agregar una afición.
  - 3.5.1. El usuario llena el formulario con la información de aficiones.
  - 3.5.2. El sistema valida la información de los campos que proporcionó el usuario.
  - 3.5.3. El sistema da de alta la afición.
  - 3.5.4. El sistema confirma al usuario que se ha dado de alta la afición.
- 3.6. El usuario elige agregar un curso.
  - 3.6.1. El usuario llena el formulario con la información de curso.
  - 3.6.2. El sistema valida la información de los campos que proporcionó el usuario.
  - 3.6.3. El sistema da de alta el curso.
  - 3.6.4. El sistema confirma al usuario que se ha dado de alta el curso.
- 3.7. El usuario elige agregar documentación.
  - 3.7.1. El usuario llena el formulario con la información de documentación.
  - 3.7.2. El sistema valida la información de los campos que proporcionó el usuario.
  - 3.7.3. El sistema da de alta la documentación.
  - 3.7.4. El sistema confirma al usuario que se ha dado de alta la documentación.
- 3.8. El usuario elige agregar escolaridad.
  - 3.8.1. El usuario llena el formulario con la información de escolaridad.
  - 3.8.2. El sistema valida la información de los campos que proporcionó el usuario.
  - 3.8.3. El sistema da de alta la institución.
  - 3.8.4. El sistema confirma al usuario que se ha dado de alta la institución.
- 3.9. El usuario elige agregar una habilidad.
  - 3.9.1. El usuario llena el formulario con la información de habilidad.
  - 3.9.2. El sistema valida la información de los campos que proporcionó el usuario.
  - 3.9.3. El sistema da de alta la habilidad.
  - 3.9.4. El sistema confirma al usuario que se ha dado de alta la habilidad.
- 3.10. El usuario elige agregar un idioma.
  - 3.10.1. El usuario llena el formulario con la información de idioma.
  - 3.10.2. El sistema valida la información de los campos que proporcionó el usuario.
  - 3.10.3. El sistema da de alta el idioma.
  - 3.10.4. El sistema confirma al usuario que se ha dado de alta el idioma.
- 3.11. El usuario elige agregar una publicación.
  - 3.11.1. El usuario llena el formulario con la información de publicación.
  - 3.11.2. El sistema valida la información de los campos que proporcionó el usuario.
  - 3.11.3. El sistema da de alta la publicación.

3.11.4. El sistema confirma al usuario que se ha dado de alta la publicación.
3.12. El usuario elige agregar un trabajo.
3.12.1. El usuario llena el formulario con la información de trabajo.
3.12.2. El sistema valida la información de los campos que proporcionó el usuario.
3.12.3. El sistema da de alta el trabajo.
3.12.4. El sistema confirma al usuario que se ha dado de alta el trabajo.

Flujo Alternativo:
a. En todo momento - El sistema falla
a.1. Se cancela la operación que se estaba realizando.
2.a. El usuario no desea editar su currículum.
2.a.1. El usuario informa al sistema que ya no desea editar su currículum.
2.a.2. El sistema cancela la operación y regresa a su estado inicial.
3.1.a El administrador no desea editar su currículum.
3.1.a.1 El administrador informa al sistema que ya no desea editar su currículum.
3.1.a.2. El sistema vuelve al estado 2.

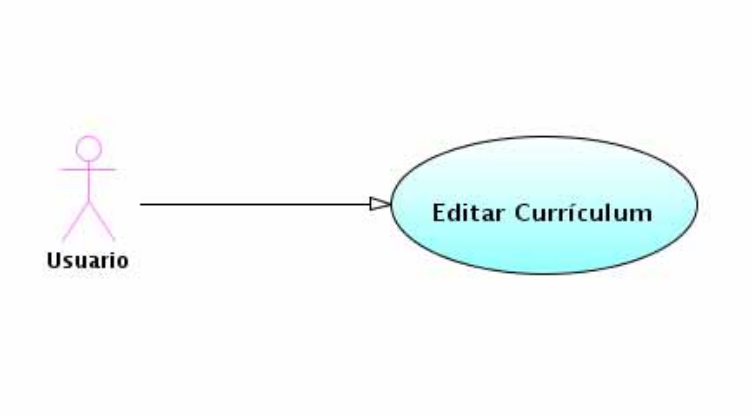

Figura 4.3: Caso de Uso Editar Currículum

4.1.4. Caso de uso: Personalizar Currículum

Actores: Usuario.
Descripción: El usuario edita el órden y los registros que aparecerán al generar un currículum.
Pre-condiciones: El usuario ingresó su cuenta de usuario y contraseña válidos.
Post-condiciones: El sistema guarda el órden y los registros que aparecerán al momento de generar el currículum del usu

Flujo Principal:
1.El usuario selecciona la opción de personalizar currículum.
2.El sistema muestra una pantalla con las diferentes secciones de las que esta compuesto el currículum.
3.El usuario arrastra y suelta los registros en la posición deseada.
4. El sistema confirma al usuario que se han guardado los cambios.
5.El usuario activa la bandera de los registros que desea que aparezcan en su currículum.
6. El sistema confirma al usuario que se han guardado los cambios.

Flujo Alternativo:
a. En cualquier momento - Falla el sistema.
a.1. Se cancela la operación en el sistema.
a.2. Se reinicia el sistema.

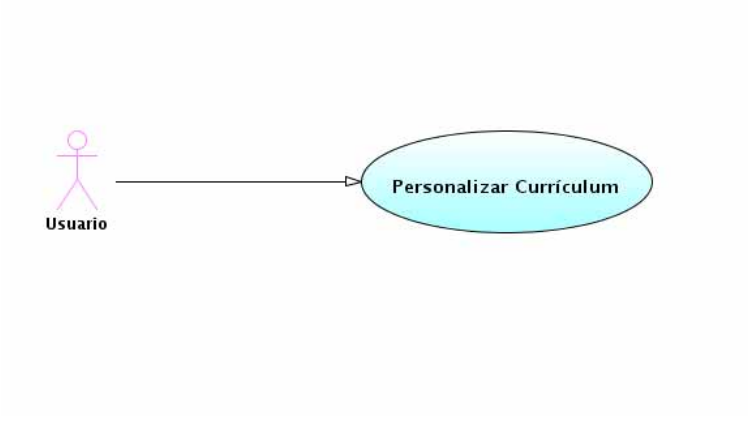

Figura 4.4: Caso de Uso Personalizar Currículum

4.1.5. Caso de Uso: Generar Currículum

Actores: Administrador, Supervisor y Usuario.
Descripción: El usuario puede generar su currículum en formato PDF.
Pre-condiciones: El usuario ingresó su cuenta de usuario y contraseña válidos. El usuario debe de tener por lo menos un registro en alguna sección del currículum.
Post-condiciones: El sistema genera y muestra el currículum en formato PDF.

Flujo Principal:
1.El usuario selecciona la opción de generar currículum.
2.El sistema muestra la currícula del usuario creados hasta el momento.
3.El sistema pregunta al usuario el nombre y destino del archivo a generar.
4.El usuario escribe un nombre y selecciona el destino del archivo que contendrá la información del currículum. 5.El sistema informa al usuario que se ha generado el archivo y lo muestra.

Flujo Alternativo: a. En cualquier momento - Falla el sistema.
a.1. Se cancela la operación en el sistema.
a.2. Se reinicia el sistema.
b. En cualquier momento el usuario desea seleccionar otro currículum.
b.1. El usuario cancela la operación que realiza en el momento.
b.2. El sistema regresa al estado 2.

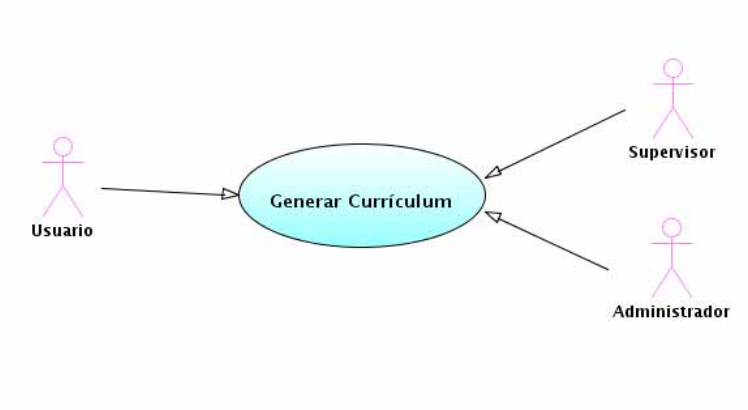

Figura 4.5: Caso de Uso Generar Currículum

4.1.6. Caso de uso: Seleccionar Plantilla

Actores: Usuario.
Descripción: El usuario selecciona una plantilla disponible que pueda ser aplicada al generar un currículum.
Pre-condiciones: El usuario ingresó su cuenta de usuario y contraseña válidos.
Post-condiciones: El sistema guarda la plantilla que aplicará al momento de generar el currículum del usuario.

Flujo Principal:
1.El usuario selecciona la opción de seleccionar plantilla.
2.El sistema muestra una lista de plantillas.
3.El usuario selecciona cada una de las plantillas para observar la vista previa.
4.El sistema muestra la vista previa de cada una de las plantillas.
5.El usuario pide al sistema guardar la última plantilla seleccionada.
6.El sistema guarda el tipo de plantilla que se aplicará.

Flujo Alternativo:
a. En cualquier momento - Falla el sistema.
    a.1. Se cancela la operación en el sistema.
    a.2. Se reinicia el sistema.
3.1 El usuario no guarda al seleccionar la plantilla.
    3.1.1 El sistema aplicará la última plantilla guardada en el sistema.
        7.1.1.1 El usuario decide sobrescribir la plantilla existente.
        7.1.1.2 El sistema regresa al estado 8.
    7.1.2. El sistema pide al usuario que ingrese otro nombre para la plantilla a crear.
    7.1.3. El sistema regresa al estado 3.

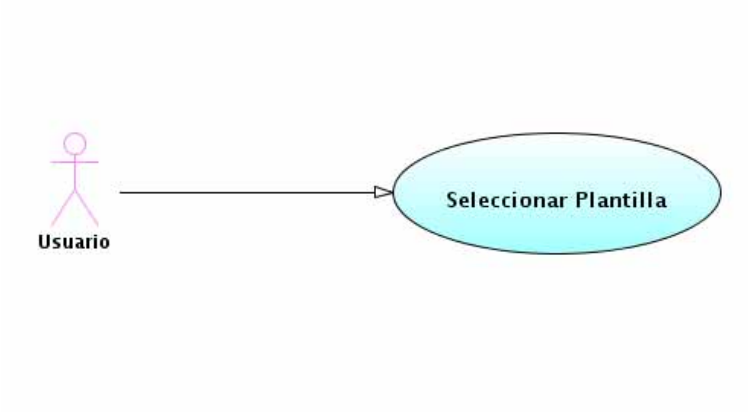

Figura 4.6: Caso de Uso Seleccionar Plantilla

4.1.7. Caso de Uso: Generar Reportes

Actores: Administrador y Supevisor.
Descripción: El usuario podrá consultar cualquier currículum seleccionando la información que debe ser mostrada.
Pre-condiciones: El usuario ingresó su cuenta de usuario y contraseña válidos.
Post-condiciones: El sistema mostrará un reporte con la currícula y la información que coincida con los datos que busca el administrador.

Flujo Principal: 1.El usuario selecciona la opción de generar reportes.
2.El sistema presenta un formulario al usuario para que ingrese los datos a consultar.
3.El usuario escribe los criterios de búsqueda para cada campo.
4.El sistema valida la información de los campos que proporcionó el usuario.

5.El sistema busca la currícula que coincide con los criterios.
6.El sistema muestra un reporte con la currícula que encontró.

Flujo Alternativo:
a. En cualquier momento - Falla el sistema.
a.1. Se cancela la operación en el sistema.
a.2. Se reinicia el sistema.
4.1 El usuario no ingreso ningún dato en el formulario.
4.1.1 El sistema indica al usuario que se generará un currículum en blanco.
4.1.1.1 El usuario decide agregar información al formulario.
4.1.1.1.2 El sistema regresa al estado 4.
4.1.2 El sistema regresa al estado 5.
4.2. El usuario ingresa una cadena inválida.
4.2.1 El sistema indica el error de datos inválidos.
4.2.2 El sistema regresa al estado 4.
5.1. El sistema no encuentra información que coincida con el criterio de búsqueda.
5.1.1. El sistema indica que no hay información que coincida con el criterio de búsqueda.
5.1.2. El sistema regresa al estado 1.

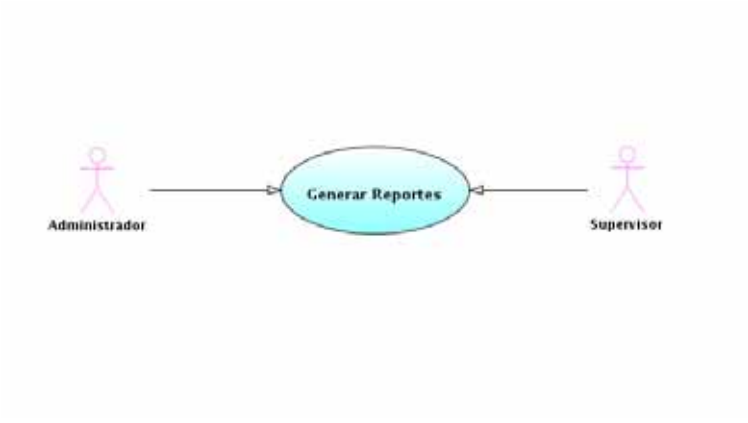

Figura 4.7: Caso de Uso Generar Reportes

4.1.8. Caso de Uso: Administrar Usuarios

Actores: Administrador.
Descripción: El administrador puede crear, modificar, consultar o eliminar cualquier usuario que esté registrado en el sistema.
Pre-condiciones: El administrador ingresó su cuenta de usuario y contraseña válidos.
Post-condiciones: El sistema crea, modifica, consulta o elimina una cuenta de usuario.

Flujo Principal:
1. El administrador selecciona la opción de administrar usuarios.
2. El sistema pregunta al administrador qué actividad desea realizar (altas, bajas, cambios o consultas).
3. El administrador elige una de las opciones (altas, bajas, cambios o consultas).
3.1. El administrador elige realizar altas.
3.1.1. El administrador indica al sistema el nombre de usuario, contraseña y tipo de usuario.
3.1.2. El sistema valida la información de los campos que proporcionó el usuario.
3.1.3. El sistema da de alta al usuario.
3.1.4. El sistema confirma al administrador que se ha dado de alta al usuario.
3.2. El administrador elige realizar bajas.
3.2.1 El administrador indica al sistema el usuario que desea dar de baja.
3.2.2 El administrador confirma al sistema que desea dar de baja al usuario escogido.

3.2.3 El sistema da de baja al usuario.
3.2.4 El sistema confirma que se ha dado de baja el usuario correctamente.
3.3. El administrador elige realizar cambios.
3.3.1 El administrador indica al sistema el usuario que desea modificar.
3.3.2 El sistema muestra el usuario que se modificarÃ¡.
3.3.3 El administrador indica al sistema las modificaciones que se deben hacer al usuario escogido.
3.3.4 El sistema realiza las modificaciones al usuario.
3.3.5 El sistema confirma al administrador que se han realizado las modificaciones correctamente.
3.4. El administrador elige realizar consultas.
3.4.1 El administrador indica al sistema la información que desea buscar.
3.4.2 El sistema muestra los datos del usuario o usuarios que se eligió consultar.

Flujo Alternativo:
a. En todo momento - El sistema falla
a.1. Se cancela la operación que se estaba realizando.
2.a. El administrador no desea administrar usuarios.
2.a.1. El administrador informa al sistema que ya no desea gestionar usuarios.
2.a.2. El sistema cancela la operación y regresa a su estado inicial.
3.1.a El administrador no desea realizar altas.
3.1.a.1 El administrador informa al sistema que ya no desea realizar altas.
3.1.a.2. El sistema vuelve al estado 2.
3.1.1.a. La cuenta de usuario ingresada al sistema ya existe.
3.1.1.a.1. El sistema informa al administrador que esa cuenta de usuario ya existe.
3.1.1.a.2. El sistema espera a que el administrador ingrese otro usuario.
3.1.1.a.3. El sistema regresa al estado 3.1.2.
3.1.2.a. El administrador no da al sistema todos los datos requeridos para realizar la operación de alta.
3.1.2.a.1. El sistema informa al administrador que faltaron datos.
3.1.2.a.2. El sistema solicita al administrador que le entregue los datos necesarios.
3.1.2.a.3. El sistema regresa al estado 3.1.2.
3.1.4.a - La operación de alta no se realiza correctamente
3.1.4.a.1. El sistema informa al administrador que ha ocurrido un error.
3.1.4.a.2. El sistema pregunta al administrador si desea que se reintente la operación o si desea que se cancele.
3.1.4.a.3. El administrador elige una opción (reintentar o cancelar la operación de alta).
3.1.4.a.3.1. El administrador elige reintentar la operación de alta.
3.1.4.a.3.1.1. El sistema vuelve al estado 3.1.4.
3.1.4.a.3.2. El administrador elige cancelar la operación de alta.
3.1.4.a.3.2.1. El sistema cancela la operación de alta.
3.1.4.a.3.2.2. El sistema vuelve al estado 3.1.
3.2.4.a - La operación de baja no se realiza correctamente
3.2.4.a.1. El sistema informa al administrador que ha ocurrido un error.
3.2.4.a.2. El sistema pregunta al administrador si desea que se reintente la operación o si desea que se cancele.
3.2.4.a.3. El administrador elige una opción (reintentar o cancelar la operación de baja).
3.2.4.a.3.1. El administrador elige reintentar la operación de baja.
3.2.4.a.3.1.1. El sistema vuelve al estado 3.2.4.
3.2.4.a.3.2 - El administrador elige cancelar la operación de baja.
3.2.4.a.3.2.1. El sistema cancela la operación de baja.
3.2.4.a.3.2.2. El sistema vuelve al estado 3.2
3.3.4.a - La operación de modificación no se realiza correctamente
3.3.4.a.1 El sistema informa al administrador que ha ocurrido un error.
3.3.4.a.2 El sistema pregunta al administrador si desea que se reintente la operación o si desea que se cancele.
3.3.4.a.3 El administrador elige una opción (reintentar o cancelar la operación de modificación)
3.3.4.a.3.1 El administrador elige reintentar la operación de modificación.
3.3.4.a.3.1.1 El sistema vuelve al estado 3.3.4.
3.3.4.a.3.2 - El administrador elige cancelar la operación de modificación.
3.3.4.a.3.2.1 El sistema cancela la operación de modificación.
3.3.4.a.3.2.2 El sistema vuelve al estado 3.3.
3.4.1.a No se encuentra el dato buscado en la ubicación (opción) elegida.
3.4.1.a.1 El sistema informa al administrador que no hubo resultados.
3.4.1.a.2 El sistema espera a que el administrador ingrese otros dato y opción.

3.4.1.a.3 El sistema regresa al estado 3.4.1.

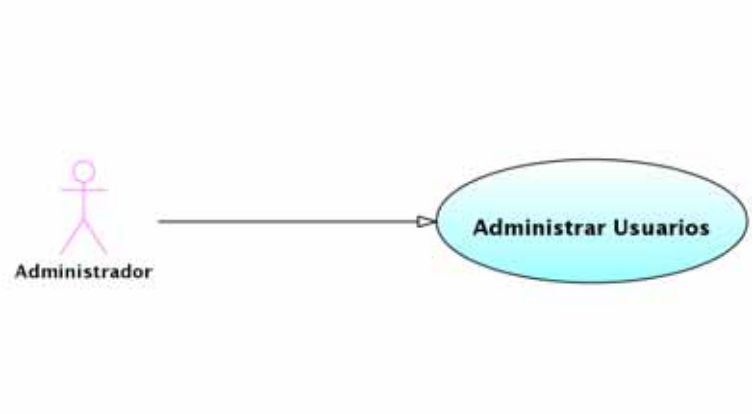

Figura 4.8: Caso de Uso Administrar Usuarios

4.2. Diagramas de Clases

4.2.1. Usuarios, Menú y Contacto

4.2.2. Currículum

4.2.3. Personal

4.2.4. Dirección

4.2.5. Teléfono

4.2.6. Correo electrónico

4.2.7. Escolaridad

4.2.8. Experiencia Laboral

4.2.9. Afición

4.2.10. Curso

4.2.11. Documentación

4.2.12. Habilidad

4.2.13. Idioma

4.2.14. Publicación

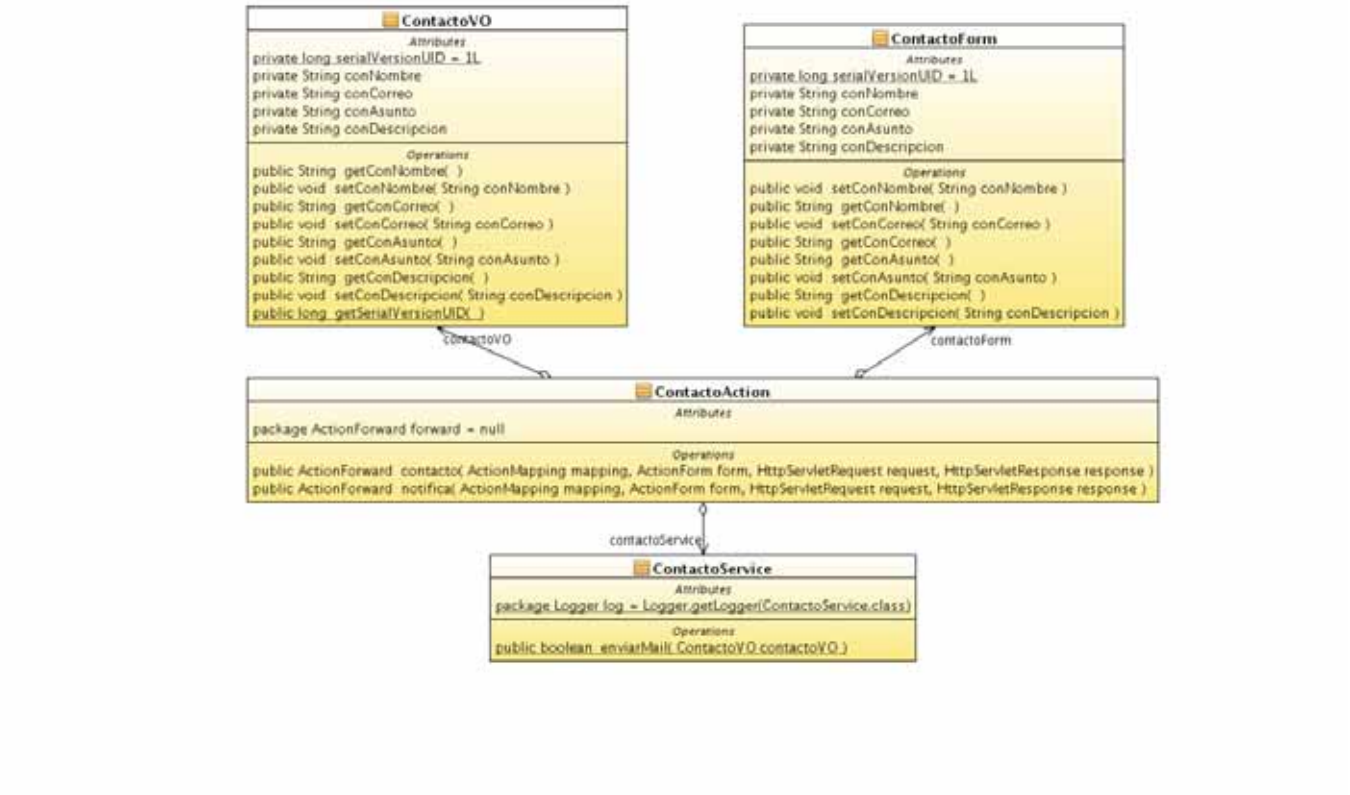

Figura 4.9: Diagrama de Clases Usuarios y Menú

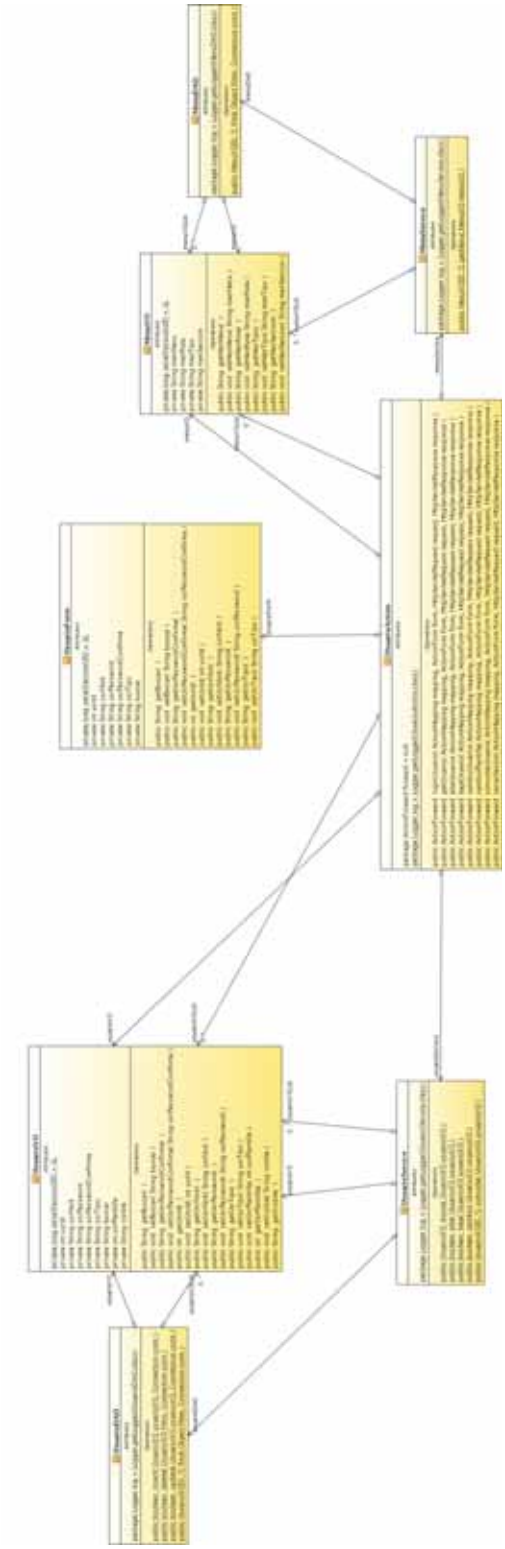

Figura 4.10: Diagrama de Clases Usuarios y Menú

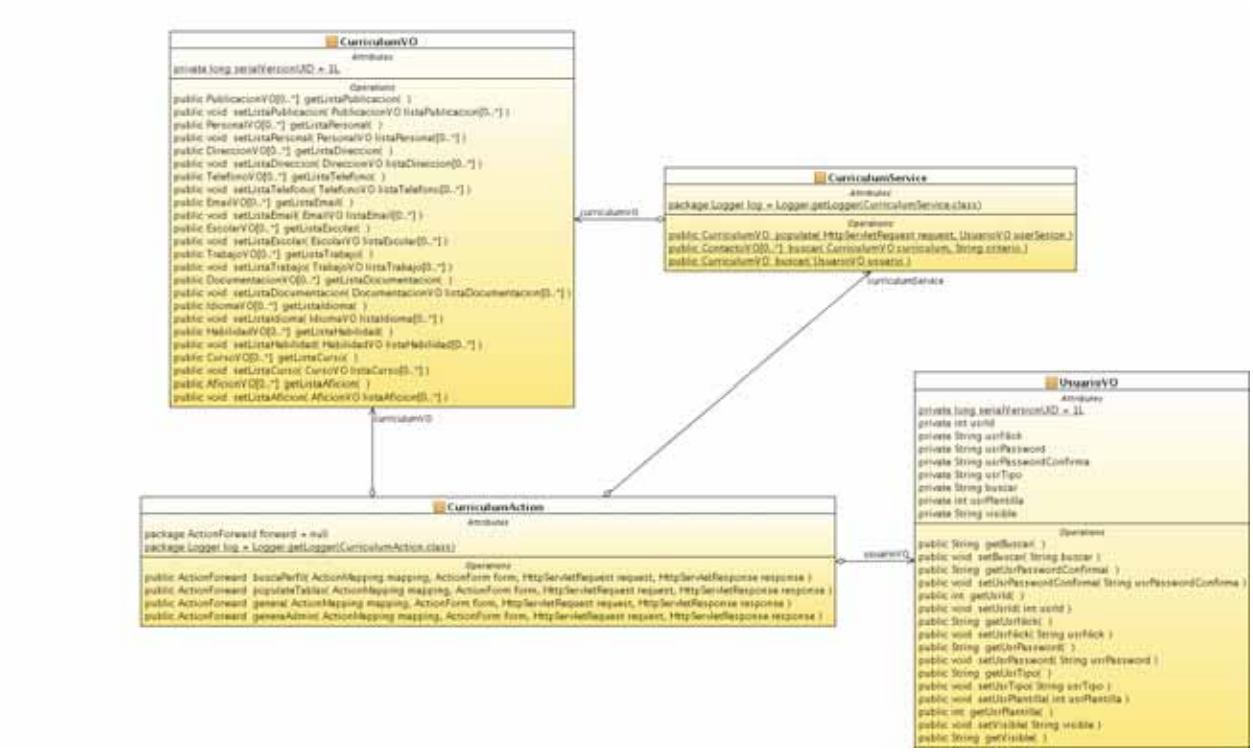

Figura 4.11: Diagrama de Clases Currículum

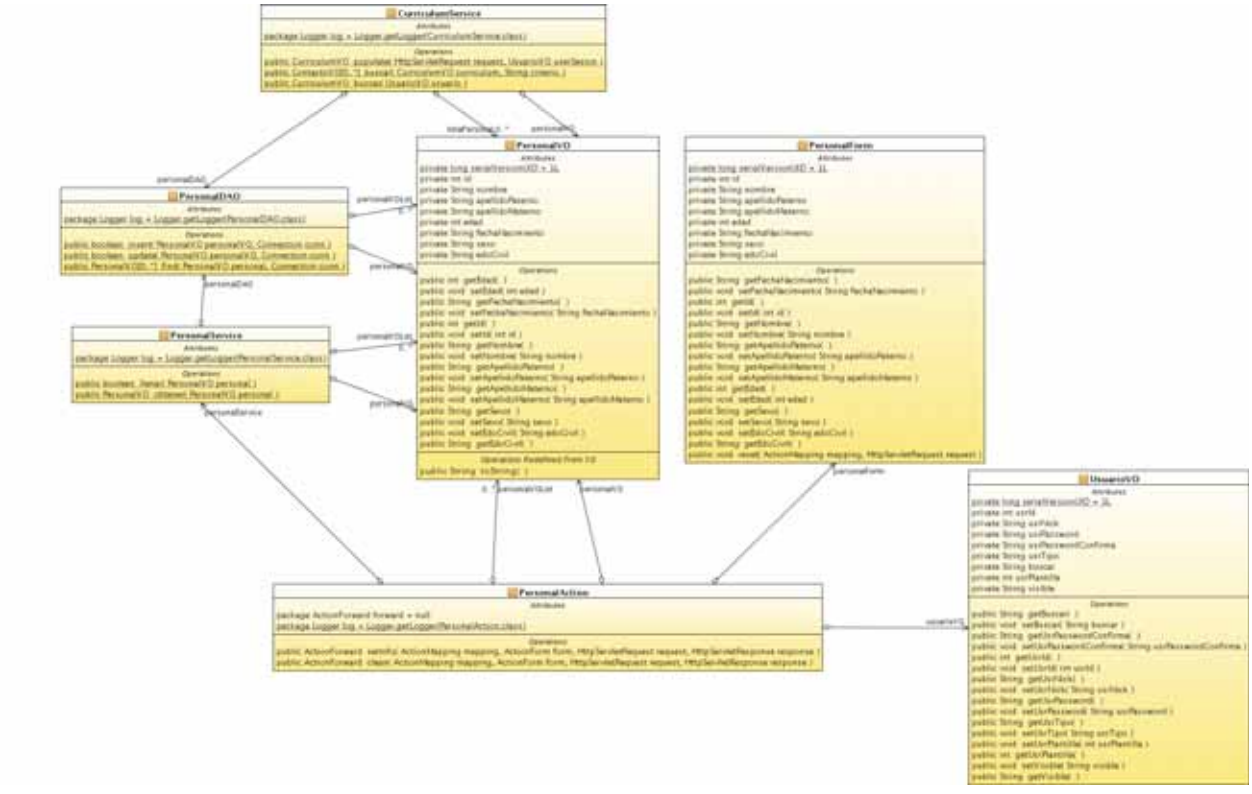

Figura 4.12: Diagrama de Clases Personal

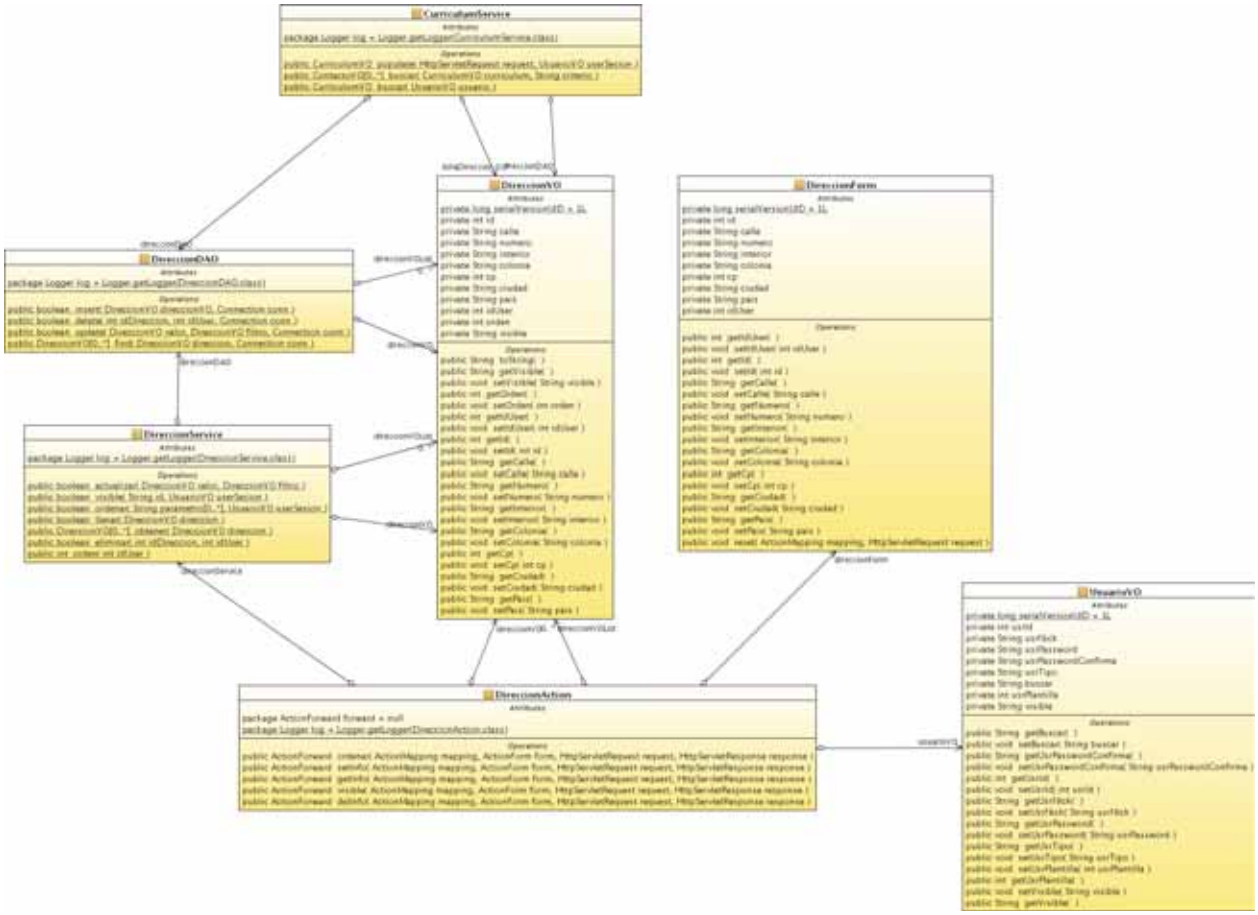

Figura 4.13: Diagrama de Clases Dirección

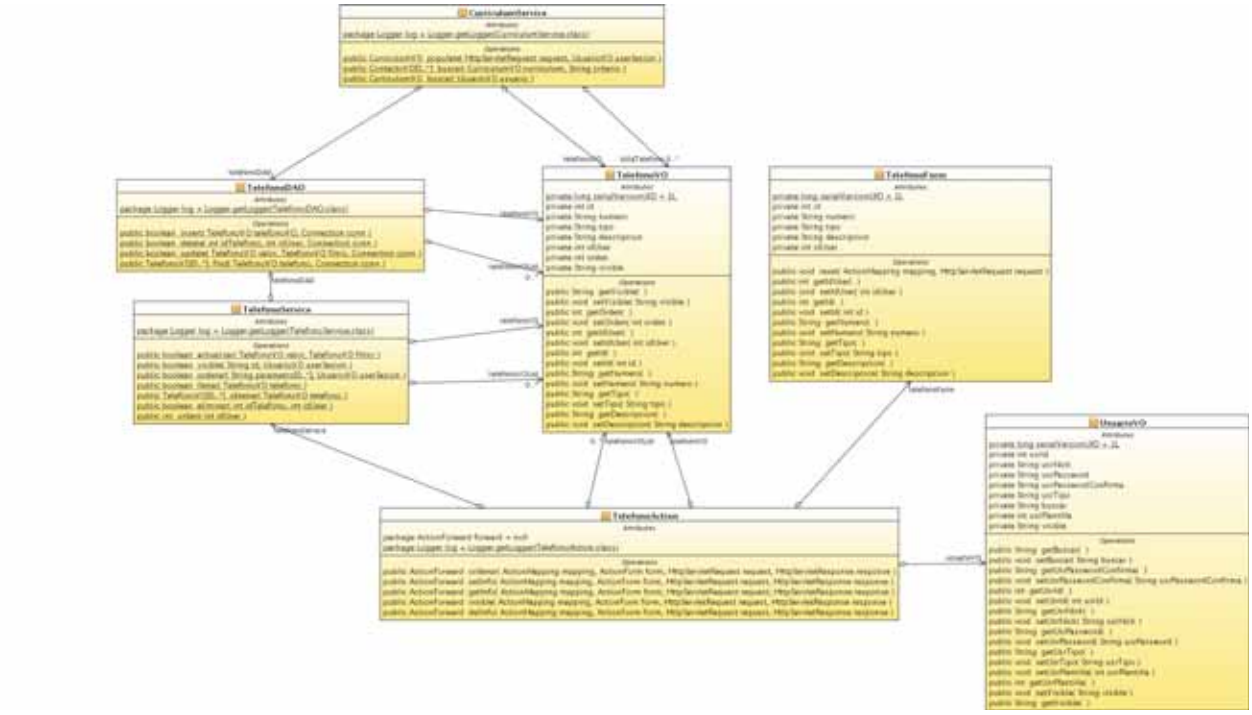

Figura 4.14: Diagrama de Clases Teléfono

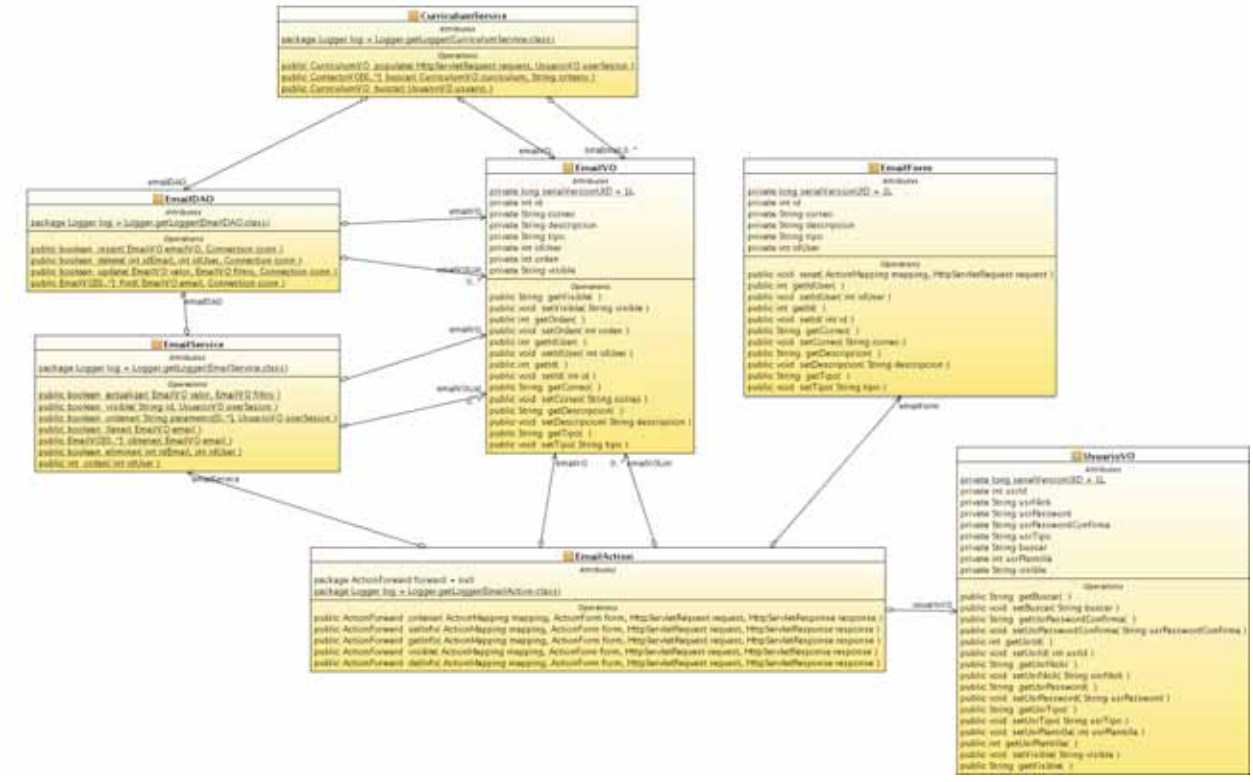

Figura 4.15: Diagrama de Clases Correo electrónico

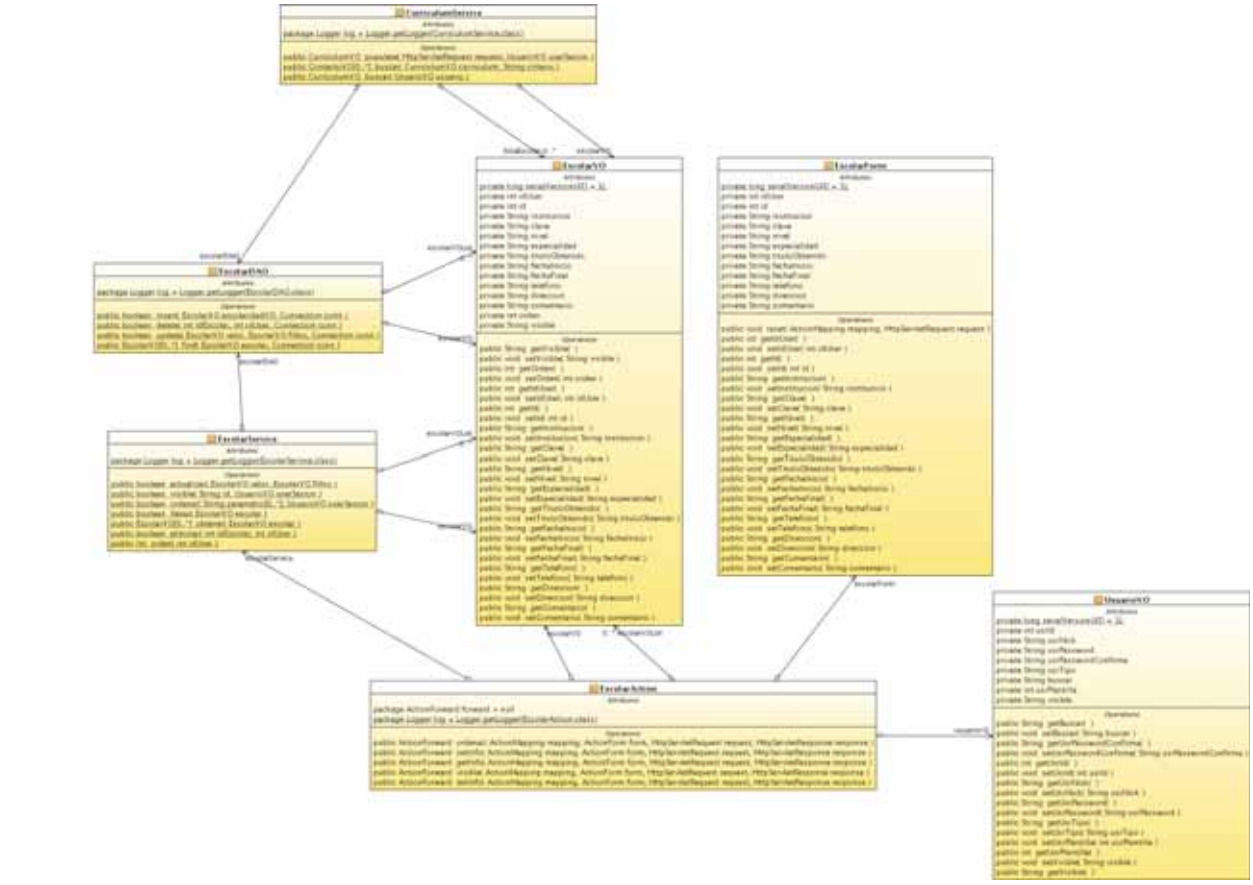

Figura 4.16: Diagrama de Clases Escolaridad

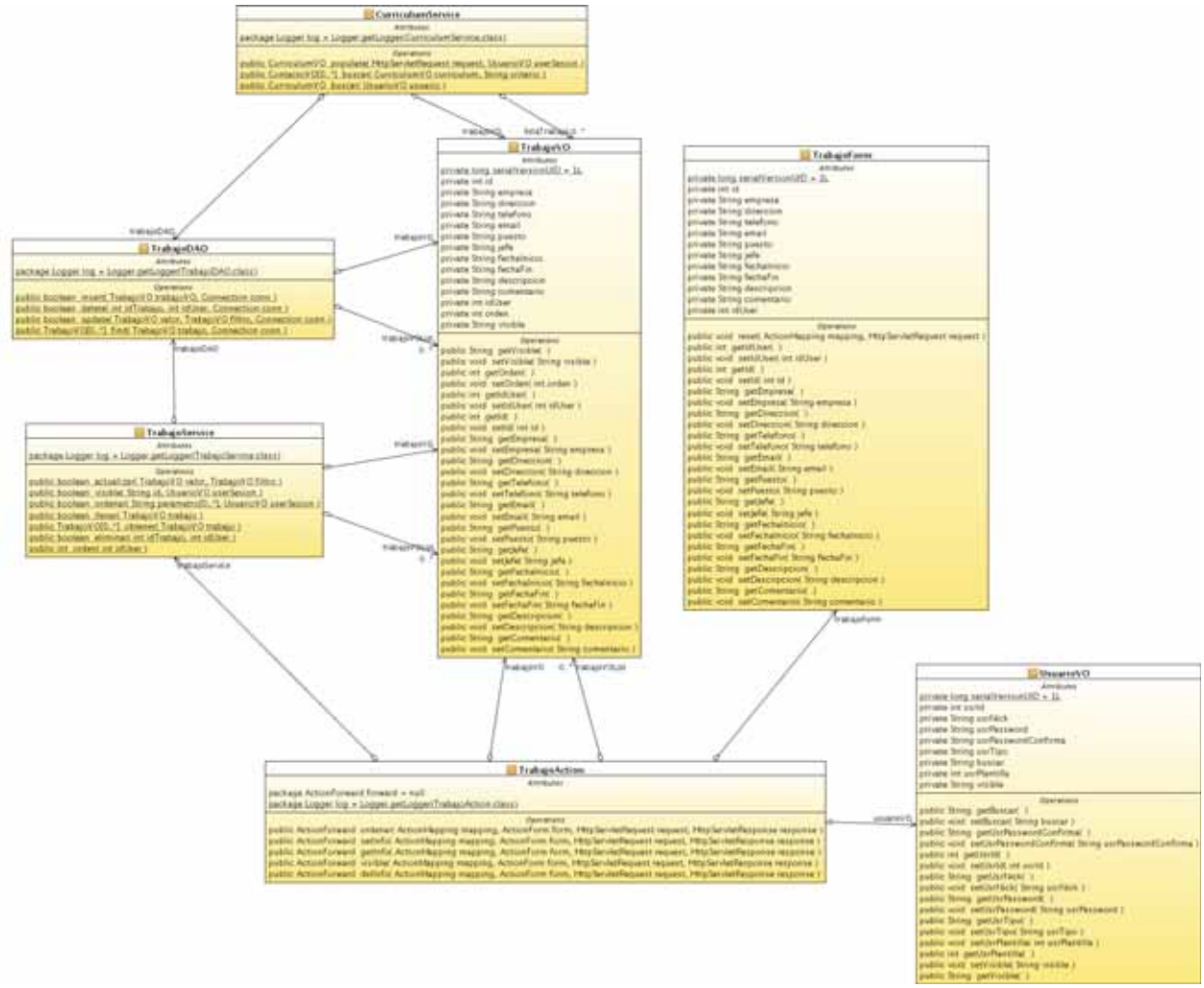

Figura 4.17: Diagrama de Clases Experiencia Laboral

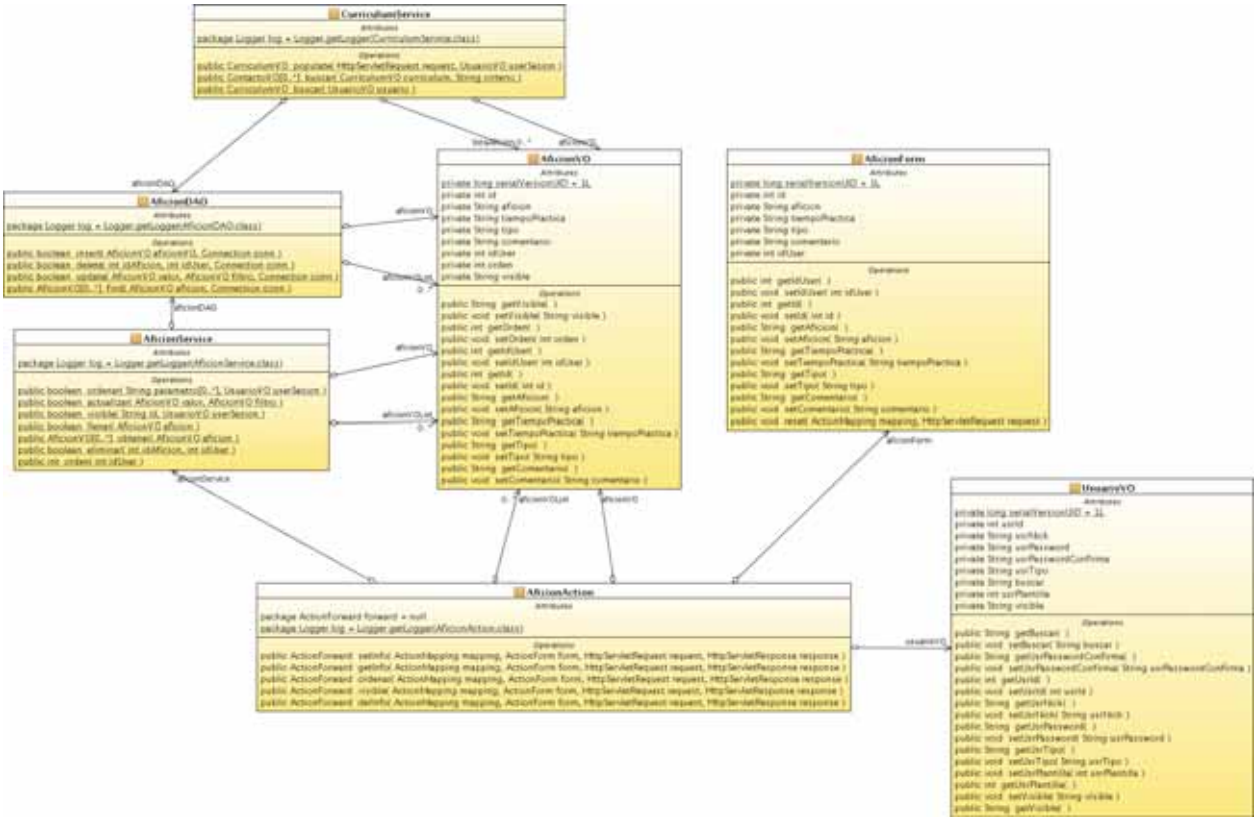

Figura 4.18: Diagrama de Clases Afición

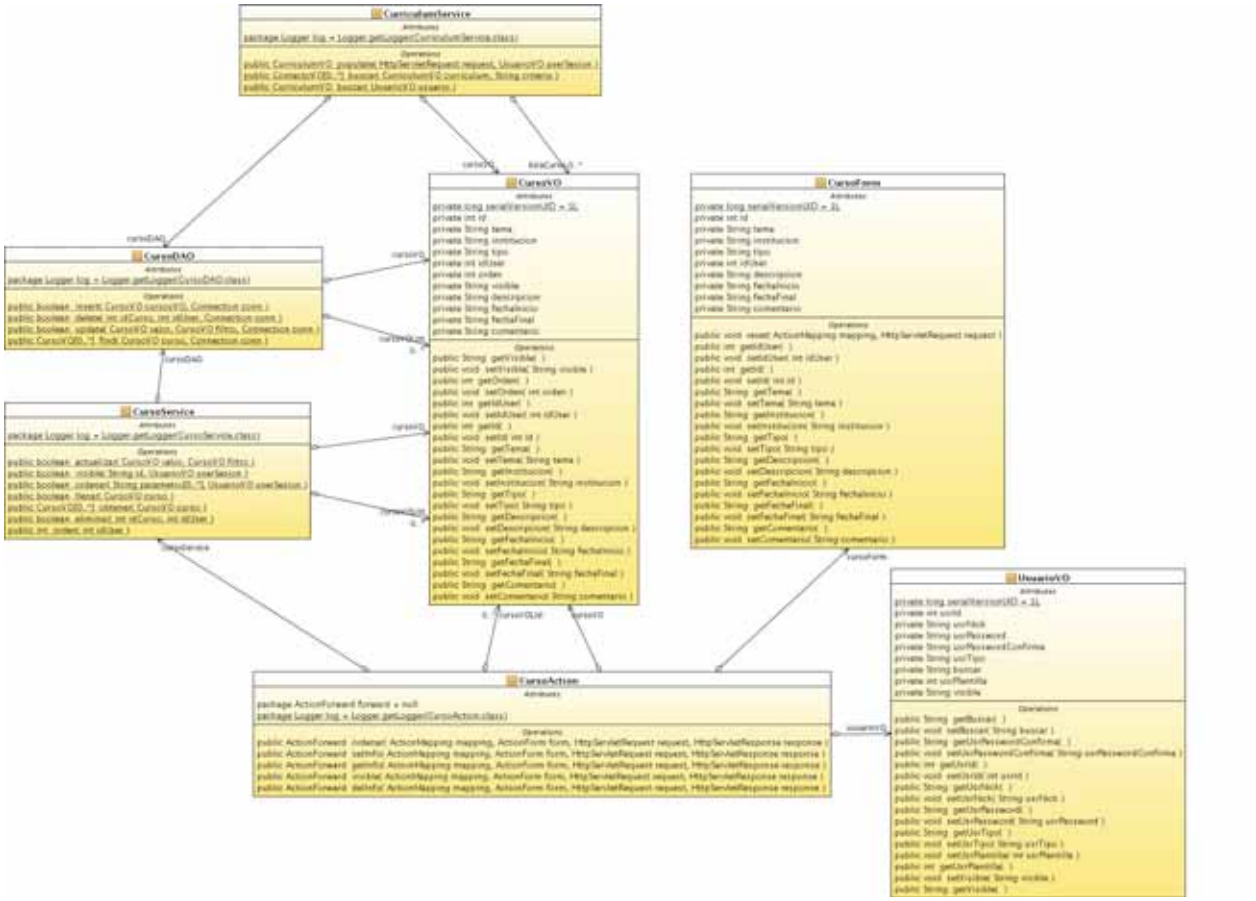

Figura 4.19: Diagrama de Clases Curso

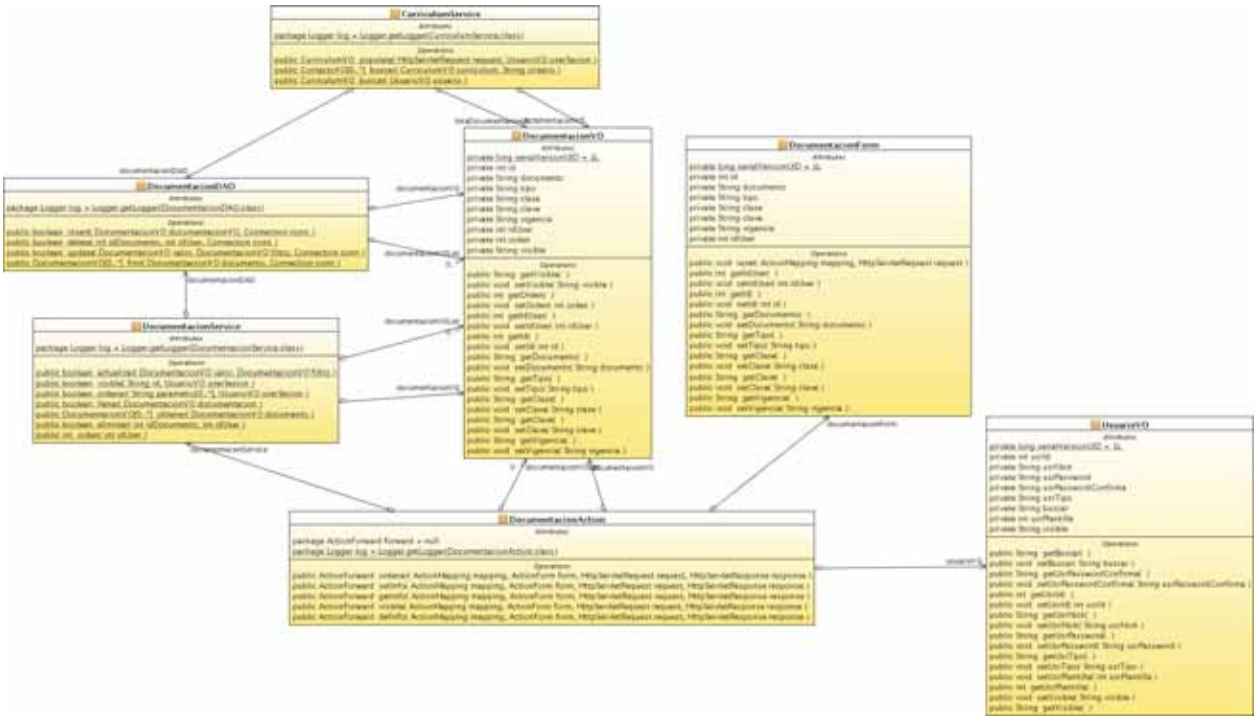

Figura 4.20: Diagrama de Clases Documentación

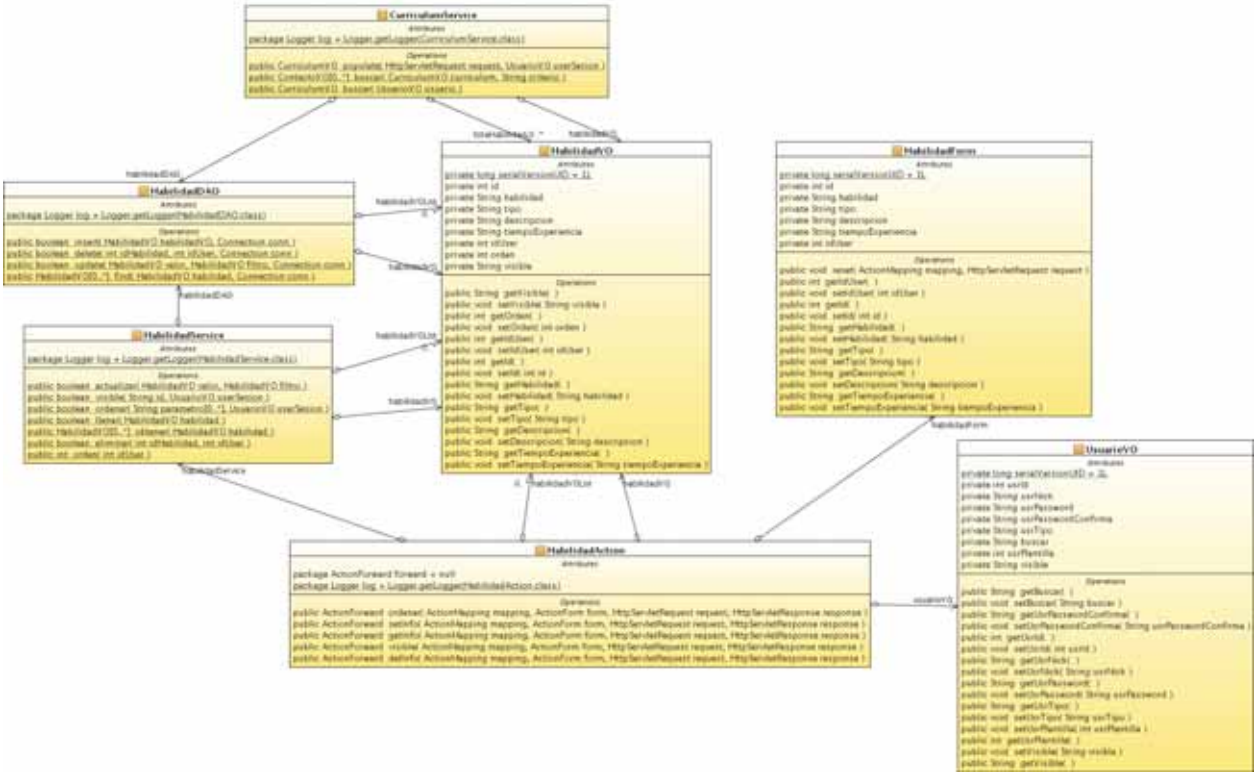

Figura 4.21: Diagrama de Clases Habilidad

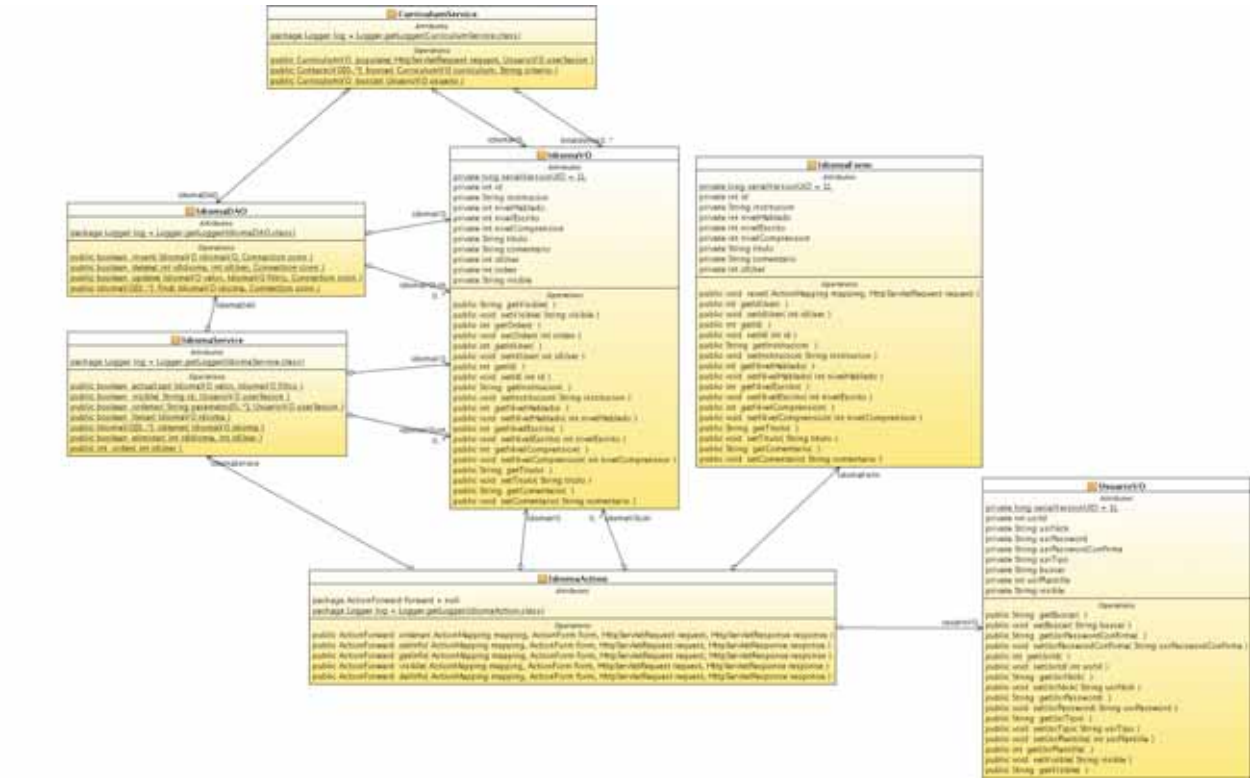

Figura 4.22: Diagrama de Clases Idioma

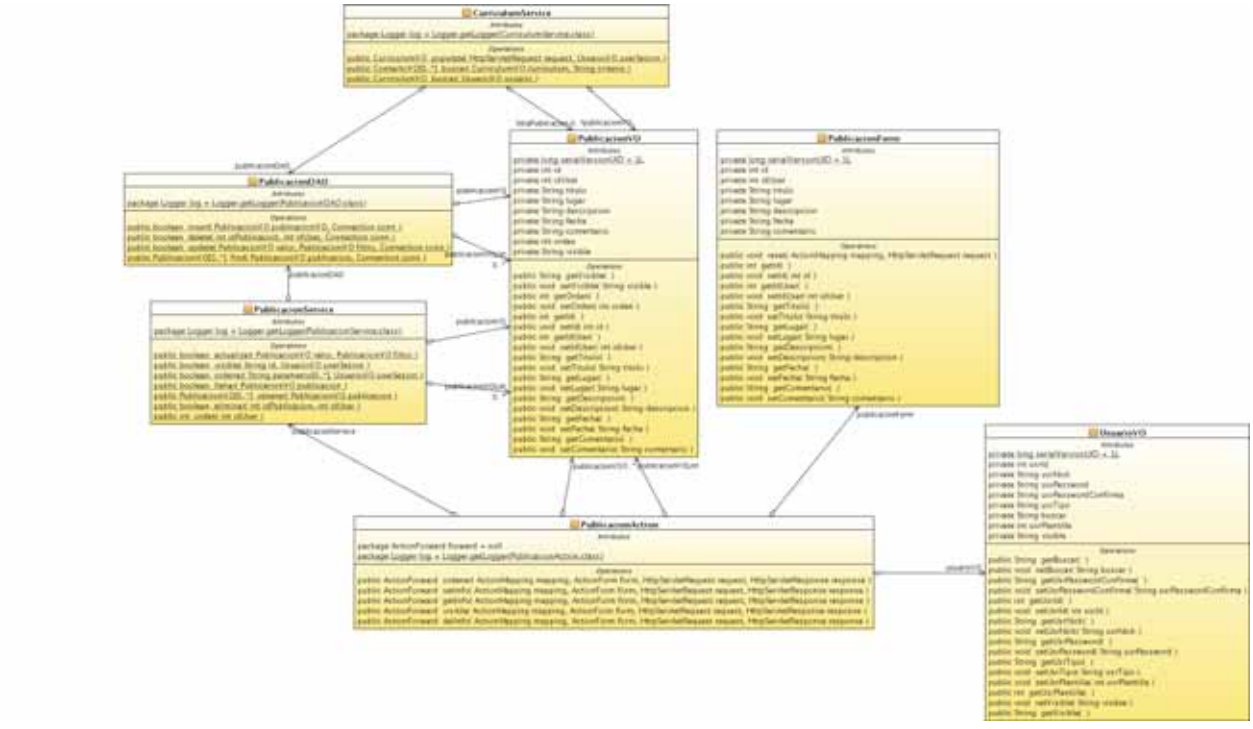

Figura 4.23: Diagrama de Clases Publicación

Capítulo 5

Navegación y Base de Datos

5.0.15. Diagrama de Navegación

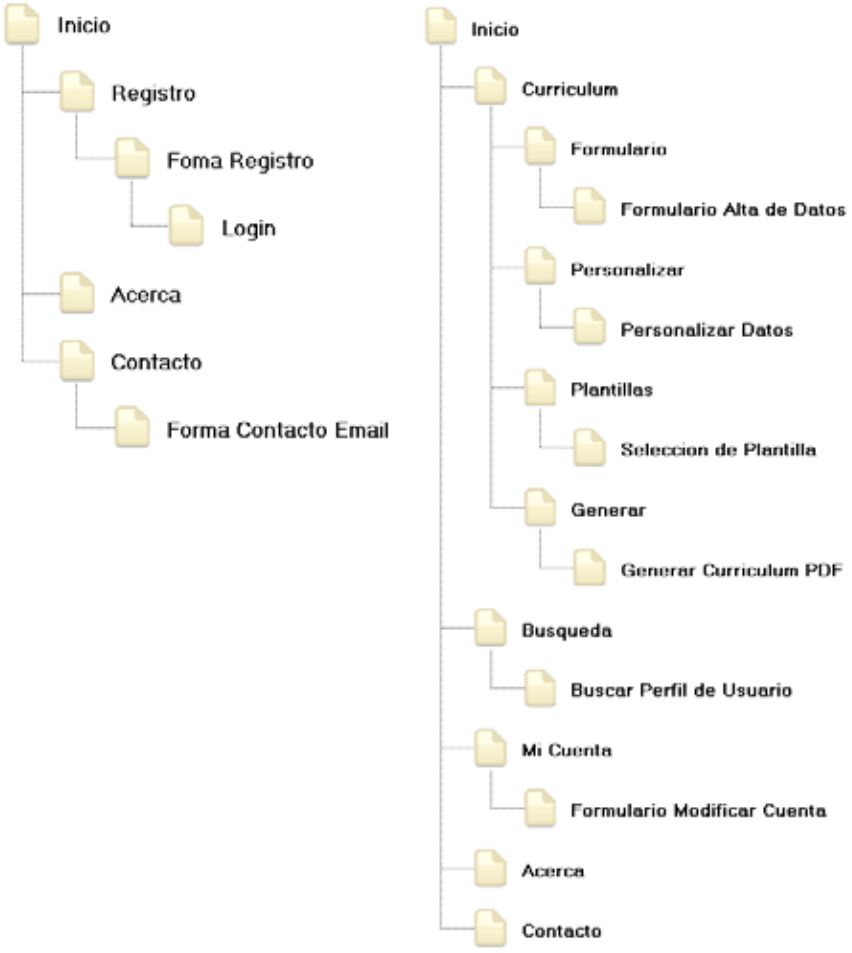

Figura 5.1: Mapa del Sitio

,

5.0.16. Diagrama Entidad Relación de Base de Datos

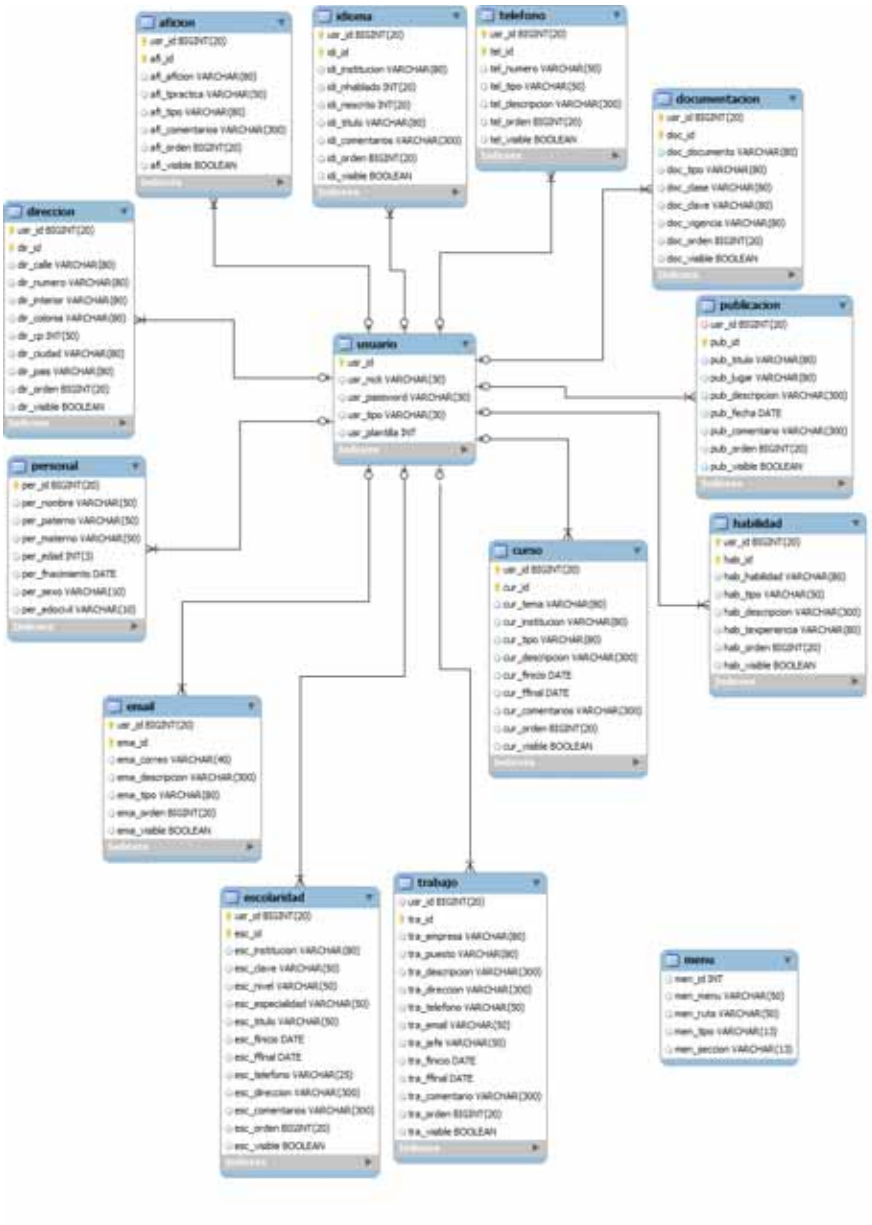

Figura 5.2: Diagrama Entidad Relación

5.0.17. Esquema Físico de Base de Datos

```
DROP DATABASE curricula;


CREATE DATABASE curricula;


USE curricula;


CREATE TABLE usuario
(
usr_id serial,
usr_nick varchar(30),
usr_password varchar(30),
usr_tipo varchar(30),
usr_plantilla int,
PRIMARY KEY (usr_id)
);


CREATE TABLE personal
(
per_id bigint(20) unsigned,
per_nombre varchar(50),
per_paterno varchar(50),
per_materno varchar(50),
per_edad int(3),
per_fnacimiento date,
per_sexo varchar(10),
per_edocivil varchar(10),
FOREIGN KEY (per_id) REFERENCES usuario (usr_id),
PRIMARY KEY (per_id)
);


CREATE TABLE aficion
(
usr_id bigint(20) unsigned,
afi_id serial,
afi_aficion varchar(80),
afi_tpractica varchar(50),
afi_tipo varchar(80),
afi_comentarios varchar(300),
afi_orden bigint(20) unsigned,
afi_visible bool,
PRIMARY KEY (usr_id, afi_id),
FOREIGN KEY (usr_id) REFERENCES usuario (usr_id)
);
```

```
CREATE TABLE curso
(
usr_id bigint(20) unsigned,
cur_id serial,
cur_tema varchar(80),
cur_institucion varchar(80),
cur_tipo varchar(80),
cur_descripcion varchar(300),
cur_finicio date ,
cur_ffinal date,
cur_comentarios varchar(300),
cur_orden bigint(20) unsigned,
cur_visible bool,
PRIMARY KEY (usr_id, cur_id),
FOREIGN KEY (usr_id) REFERENCES usuario (usr_id)
);


CREATE TABLE direccion
(
usr_id bigint(20) unsigned ,
dir_id serial,
dir_calle varchar(80) ,
dir_numero varchar(80),
dir_interior varchar(80) default NULL,
dir_colonia varchar(80),
dir_cp int(50) ,
dir_ciudad varchar(80) ,
dir_pais varchar(80),
dir_orden bigint(20) unsigned,
dir_visible bool,
PRIMARY KEY (usr_id, dir_id),
FOREIGN KEY (usr_id) REFERENCES usuario (usr_id)
);


CREATE TABLE documentacion
(
usr_id bigint(20) unsigned ,
doc_id serial,
doc_documento varchar(80),
doc_tipo varchar(80),
doc_clase varchar(80),
doc_clave varchar(80),
doc_vigencia varchar(80),
doc_orden bigint(20) unsigned,
doc_visible bool,
PRIMARY KEY (usr_id, doc_id),
FOREIGN KEY (usr_id) REFERENCES usuario (usr_id)
);
```

```
CREATE TABLE email
(
usr_id bigint(20) unsigned,
ema_id serial,
ema_correo varchar(40),
ema_descripcion varchar(300),
ema_tipo varchar(80),
ema_orden bigint(20) unsigned,
ema_visible bool,
PRIMARY KEY (usr_id, ema_id),
FOREIGN KEY (usr_id) REFERENCES usuario (usr_id)
);


CREATE TABLE escolaridad
(
usr_id bigint(20) unsigned,
esc_id serial,
esc_institucion varchar(80),
esc_clave varchar(50),
esc_nivel varchar(50) ,
esc_especialidad varchar(50),
esc_titulo varchar(50),
esc_finicio date,
esc_ffinal date,
esc_telefono varchar(25),
esc_direccion varchar(300),
esc_comentarios varchar(300),
esc_orden bigint(20) unsigned,
esc_visible bool,
PRIMARY KEY (usr_id,esc_id),
FOREIGN KEY (usr_id) REFERENCES usuario (usr_id)
);


CREATE TABLE habilidad
(
usr_id bigint(20) unsigned ,
hab_id serial,
hab_habilidad varchar(80) ,
hab_tipo varchar(50) ,
hab_descripcion varchar(300) ,
hab_texperiencia varchar(80) ,
hab_orden bigint(20) unsigned,
hab_visible bool,
PRIMARY KEY (usr_id,hab_id),
FOREIGN KEY (usr_id) REFERENCES usuario (usr_id)
);
```

```
CREATE TABLE idioma
(
usr_id bigint(20) unsigned ,
idi_id serial,
idi_institucion varchar(80),
idi_nhablado int(20) ,
idi_nescrito int(20),
idi_titulo varchar(80) ,
idi_comentarios varchar(300),
idi_orden bigint(20) unsigned,
idi_visible bool,
PRIMARY KEY (usr_id,idi_id),
FOREIGN KEY (usr_id) REFERENCES usuario (usr_id)
);


CREATE TABLE telefono
(
usr_id bigint(20) unsigned ,
tel_id serial,
tel_numero varchar(50) ,
tel_tipo varchar(50) ,
tel_descripcion varchar(300) default NULL,
tel_orden bigint(20) unsigned,
tel_visible bool,
PRIMARY KEY (usr_id,tel_id),
FOREIGN KEY (usr_id) REFERENCES usuario (usr_id)
);


CREATE TABLE trabajo
(
usr_id bigint(20) unsigned ,
tra_id serial,
tra_empresa varchar(80) ,
tra_puesto varchar(80) ,
tra_descripcion varchar(300) ,
tra_direccion varchar(300) ,
tra_telefono varchar(50) ,
tra_email varchar(50) ,
tra_jefe varchar(50) ,
tra_finicio date,
tra_ffinal date ,
tra_comentario varchar(300) ,
tra_orden bigint(20) unsigned,
tra_visible bool,
PRIMARY KEY (tra_id),
FOREIGN KEY (usr_id) REFERENCES usuario (usr_id)
);
```

```
CREATE TABLE publicacion
(
usr_id bigint(20) unsigned ,
pub_id serial,
pub_titulo varchar(80) ,
pub_lugar varchar(80) ,
pub_descripcion varchar(300) ,
pub_fecha date,
pub_comentario varchar(300) ,
pub_orden bigint(20) unsigned,
pub_visible bool,
PRIMARY KEY (pub_id),
FOREIGN KEY (usr_id) REFERENCES usuario (usr_id)
);


CREATE TABLE menu
(
men_id int unsigned,
men_menu varchar(50),
men_ruta varchar(50),
men_tipo varchar(13),
men_seccion varchar(13)
);


INSERT INTO usuario VALUES (1,'admin','pass','administrador','1');
INSERT INTO usuario VALUES (2,'super','pass','supervisor','1');
INSERT INTO usuario VALUES (3,'user','pass','usuario','1');


INSERT INTO menu VALUES ('1','Inicio','/inicio.do','administrador','menu');
INSERT INTO menu VALUES ('2','Curriculum','/curriculum.do','administrador','menu');
INSERT INTO menu VALUES ('3','Usuarios','/usuarios.do','administrador','menu');
INSERT INTO menu VALUES ('4','Busquedas','/busquedas.do','administrador','menu');
INSERT INTO menu VALUES ('5','Mi Cuenta','/autoModificarU.do','administrador','menu');
INSERT INTO menu VALUES ('6','Acerca','/acerca.do','administrador','menu');
INSERT INTO menu VALUES ('7','Contacto','/contacto.do','administrador','menu');


INSERT INTO menu VALUES ('1','Inicio','/inicio.do','administrador','menuC');
INSERT INTO menu VALUES ('2','Formulario','/formulario.do','administrador','menuC');
INSERT INTO menu VALUES ('3','Personalizar','/curriculumA.do?accion=populateTablas','administrador','menuC');
INSERT INTO menu VALUES ('4','Plantillas','/plantillas.do','administrador','menuC');
INSERT INTO menu VALUES ('5','Generar','/curriculumA.do?accion=genera','administrador','menuC');


INSERT INTO menu VALUES ('1','Inicio','/inicio.do','administrador','menuU');
INSERT INTO menu VALUES ('2','Administrar Usuarios','/buscarU.do?accion=consultaUsuario','administrador','menuU');
INSERT INTO menu VALUES ('3','Alta de Usuarios','/crearU.do','administrador','menuU');



INSERT INTO menu VALUES ('1','Inicio','/inicio.do','supervisor','menu');
INSERT INTO menu VALUES ('2','Busquedas','/busquedas.do','supervisor','menu');
INSERT INTO menu VALUES ('3','Mi Cuenta','/autoModificarU.do','supervisor','menu');
INSERT INTO menu VALUES ('4','Acerca','/acerca.do','supervisor','menu');
INSERT INTO menu VALUES ('5','Contacto','/contacto.do','supervisor','menu');
```

```
INSERT INTO menu VALUES ('1','Inicio','/inicio.do','usuario','menu');
INSERT INTO menu VALUES ('2','Curriculum','/curriculum.do','usuario','menu');
INSERT INTO menu VALUES ('3','Mi Cuenta','/autoModificarU.do','usuario','menu');
INSERT INTO menu VALUES ('4','Acerca','/acerca.do','usuario','menu');
INSERT INTO menu VALUES ('5','Contacto','/contacto.do','usuario','menu');


INSERT INTO menu VALUES ('1','Inicio','/inicio.do','usuario','menuC');
INSERT INTO menu VALUES ('2','Formulario','/formulario.do','usuario','menuC');
INSERT INTO menu VALUES ('3','Personalizar','/curriculumA.do?accion=populateTablas','usuario','menuC');
INSERT INTO menu VALUES ('4','Plantillas','/plantillas.do','usuario','menuC');
INSERT INTO menu VALUES ('5','Generar','/curriculumA.do?accion=genera','usuario','menuC');
```

Código 5.1: Esquema Físico de Base de Datos

Capítulo 6

Manual de Usuario

6.1. Información General

El siguiente manual tiene como objetivo orientar a los usuarios en el manejo, navegaciń y funcionalidad del sistema al administrar una currícula. El sistema consta de tres módulos principales: Administración de usuarios, Currículum y Búsquedas.

6.1.1. Módulos

- Administración de usuarios
  Registrar, buscar, editar y eliminar todos los usuarios que se encuentran registrados en el sistema.
- Currículum
  Editar y personalizar la informacón para generar un documento con el currículum en la plantilla elegida.
- Búsquedas
  Generar reportes de la currícula que contenga los parámetros de bśqueda ingresados.

Éstos y los demás módulos pueden ser ingresados por los diferentes usuarios dependiendo del rol que tengan asignado.

6.1.2. Usuarios

- Administrador
  Encargado de gestionar usuarios y tiene acceso a todos los módulos del sistema.
- Supervisor
  Se encarga de generar reportes y búsquedas de la currícula disponible.
- Usuario
  Ingresa información al sistema para constriur su curículum.

Figura 6.1: Estructura del Sitio

6.2. Estructura del Sitio

Cada pantalla del sistema esta dividida en 3 secciones:

- Menu
  Muestra las funcionalidades del sistema para cada pantalla.
- Barra Lateral
  Esta barra muestra si el usuario ha iniciado sesión y se incluyen ligas con información referente al desarrollo del sistema.
- Cuerpo
  El cuerpo depende de la funcionalidad que se ha elegido utilizar, en el caso de las páginas principales a su vez se divide en 2 secciones, en la parte superior se da una explicación referente al módulo en el que se encuentra el usuario, en la parte inferior se explica cada uno de los menus a los que se puede acceder.

Figura 6.2: Sección Superior

Figura 6.3: Sección Inferior

6.3. Inicio

Esta es la pantalla cuando se ingresa al sistema y no se ha iniciado una sesión, se tienen las opciones de Registro, Acerca y Contacto.

Figura 6.4: Inicio Superior

Figura 6.5: Inicio Inferior

6.3.1. Registro

En el registro se crea una cuenta de usuario para ingresar la información de un currículum, manipularla y seleccionar una plantilla para generar un documento. El usuario debe llenar el formulario con el nombre de usuario,la contrasena y la confirmación y guardar para que el sistema valide la información.

Figura 6.6: Registro

6.3.2. Acerca

En esta parte se muestra la información referente a los creadores y el proyecto.

Figura 6.7: Acerca

6.3.3. Contacto

En esta sección un usuario o persona ajena al sistema puede enviar un correo electrónico para contactar al administrador.

6.4. Inicio de Administrador

Esta es la pantalla cuando un administrador ingresa al sistema, se tienen las opciones de Currículum, Usuarios, Búsquedas, Mi cuenta Acerca y Contacto.

Figura 6.8: Contacto

Figura 6.9: Inicio de Administrador Superior

6.5. Usuarios

En este módulo el administrador puede agregar, editar y eliminar usuarios con el perfil alguno de los diferentes perfiles de administrador, supervisor o usuario. Se tienen las opciones de Administrar Usuarios y Alta de Usuarios.

6.5.1. Administrar Usuarios

Buscar, editar y eliminar todos los usuarios que se encuentran registrados en el sistema.

6.5.2. Alta de Usuarios

Agregar usuarios al sistema con el rol de administrador, supervisor o usuario común.

6.6. Inicio de Supervisor

Esta es la pantalla cuando un supervisor ingresa al sistema, se tienen las opciones de Búsquedas, Mi cuenta Acerca y Contacto.

6.6.1. Búsquedas

El usuario debe seleccionar el nombre de cada una de las secciones que componen el currÃculum para mostrar y llenar los campos con los patrones de bÃºsqueda de la currÃcula que deseas obtener.

Puede determinar si la consulta debe coincidir exactamente con los parÃ¡metros de bÃºsqueda (Y/AND) o si solamente debe contener alguno de estos (O/OR) seleccionando el radio button correcto.

6.7. Inicio de Currículum

Esta es la pantalla cuando un supervisor ingresa al sistema, se tienen las opciones de Búsquedas, Mi cuenta Acerca y Contacto.

Figura 6.10: Inicio de Administrador Inferior

Figura 6.11: Usuarios Superior

6.8. Formulario

Se ingresa la información necesaria para conformar un currículum, datos personales, localización, escolaridad, experiencia laboral, documentos importantes, idiomas, habilidades, cursos, publicaciones y aptitudes. El formulario esta separado por estas secciones y se pueden agregar y eliminar registros para cada una de estas.

6.9. Personalizar

Modificar el orden de los campos por prioridad o simplemente por gusto así como tambien indicar cuales deben aparecer al momento de generar el documento.

6.10. Plantillas

Analizar los tipos de plantillas y elegir una de las disponibles para aplicar al momento de exportar el currículum.

6.11. Generar

Exportar el currícumul a un documento en formato PDF.

Figura 6.12: Usuarios Inferior

Figura 6.13: Administrar Usuarios

Figura 6.14: Alta de Usuarios

Figura 6.15: Búsquedas Superior

Figura 6.16: Búsquedas Inferior

Figura 6.17: Inicio de Currículum Superior

Figura 6.18: Inicio de Currículum Inferior

Figura 6.19: Formulario

Figura 6.20: Formulario

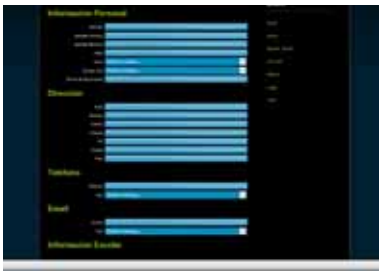

Figura 6.21: Formulario

Figura 6.22: Personalizar

Figura 6.23: Personalizar

Figura 6.24: Plantillas

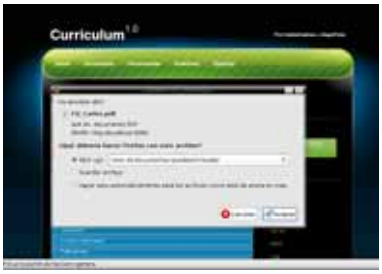

Figura 6.25: Generar

Capítulo 7

Manual Técnico

7.1. Instalación y Configuración de Requerimientos

Para el desarrollo de este sistema se utilizo JDK 6 y Apache Tomcat 6.0.18.

7.1.1. Instalación JDK 6

- Colocar el archivo jdk-6u10-linux-i586.bin en /usr/local y ejecutarlo

  ./jdk-6u10-linux-i586.bin

- Colocar el archivo jdk-6u10-linux-i586.bin en /usr/local y ejecutarlo

  ./jdk-6u10-linux-i586.bin

- Editar el archivo /etc/profile para establecer las variables de ambiente

  sudo nano /etc/profile

- Escribir al final del archivo

  JAVA_HOME=/usr/local/jdk1.6.0_10"
  PATH="$PATH:/usr/local/jdk1.6.0_10/bin"
  export JAVA_HOME
  export PATH

7.1.2. Instalación Apache Tomcat 6.0.18

- Colocar el archivo apache-tomcat-6.0.18.tar.gz en/usr/local y extraerlo

  sudo tar xvzf apache-tomcat-6.0.18.tar.gz

- Editar el archivo /usr/local/apache-tomcat-5.5.27/bin/catalina.sh para establecer la variable de ambiente JAVA_HOME

  sudo nano /usr/local/apache-tomcat-5.5.27/bin/catalina.sh

- Escribir al principio del archivo

  JAVA_HOME=/usr/local/jdk1.6.0_10"

- Iniciar Tomcat
  sudo /usr/local/apache-tomcat-6.0.18/bin/startup.sh
- Verificar en un explorador
  http://localhost:8080/
- Detener Tomcat
  sudo /usr/local/apache-tomcat-6.0.18/bin/shutdown.sh

Capítulo 8

Conclusiones

En la propuesta inicial del proyecto se detalla y se propone la utilización de un repositorio XML como medio de almacenamiento persistente de la información de cada uno de los registro que genera la información del usuario. Durante la investigación de las tecnologías que podríamos utilizar para el repositorio nos encontramos con varios gestores y bases de datos nativas de XML entre las cuales destacan los siguientes:

- dbXML
- Ipedo
- Infonyte
- Tamino
- Virtuoso
- XIndice
- XDBM
- eXist

El problema que encontramos con cada uno de estos gestores es que ninguno de ellos permite la actualización de los documentos XML, todos son considerados únicamente como repositorio de base de datos.

Nosotros basamos nuestra investigación en el repositorio XML eXist debido a que es el más popular, la integración con JAVA es sencilla y es compatible con la mayoría de lenguajes de consultas de XML como lo es XQuey, Xpath, tiene compatibilidad con los namespaces de los XML y uno de las características más significativas es que no está limitado al uso de esquema y puede funcionar sin la dependencia de este.

Al momento de la integración con XML pudimos realizar varios ejemplos enlazando JAVA con el repositorio de eXist realizamos algunas consultas para la obtención de datos pero al investigar las actualizaciones de documentos XML no dimos cuenta que si existe un módulo para realizar actualización, remplazo e inserción, pero es un módulo en desarrollo que todavía necesita optimizaciones y sirve correctamente cuando se está trabajando con una cantidad considerable de datos; pero en nuestro caso se trata de un repositorio que cuenta con muchos nodos anidados lo que ocasionaría mucho tiempo perdido en una actualización de un nodo.

Las pruebas para corroborar el tiempo de actualización de un repositorio lo realizamos con un archivo XML de ejemplo que contiene la descripción de una biblioteca, este contaba únicamente con 4 nodos: Título, autor, editorial, comentario; en este repositorio se intentó actualizar la editorial de los libros que tenían determinado autor y para 100 libros se tardó 3 segundos en actualizar, después se quiso insertar un nuevo tag o nodo entre editorial y comentario a todos los libros (año) esta actualización para los 100 libros tardo 5 segundos.

Por tales resultados decidimos que no era óptimo utilizar ese medio de almacenamiento ya que no significa ningún cambio significativo en la funcionalidad y por lo pronto con esta versión no tiene motivo para ser interoperable con otros sistemas. También teníamos la posibilidad de utilizar un repositorio XML comercial como lo es dbXML de Oracle que ya cuenta con módulos de actualización probados y funcionales, pero esto convertiría al proyecto en una aplicación dependiente de recursos comerciales.

Se espera que los módulos de actualización de eXist queden totalmente liberados para cambiar el acceso a datos (DAO) para que consulte la BD XML utilizando como lenguaje principal Xquery. Así muchos sistemas podrían consultar la información de los currículum en base a otras tecnologías como Web Service y protocolos SOAP.

Capítulo 9

Bibliografía

- Euro-CV, Jean-Pierre Thiollet, Top Editions, 1997. ISBN 2-87731-131-7.
- http://definicion.de/curriculum/, 5 de mayo de 2008.
- www.gestiopolis.com/canales8/rrhh/losrecursoshumanos/curriculum-vitae-tipos-y-generalidades.htm, 12 de Julio de 2008.
- http://websou.usal.es/empleo/curricul.pdf, 2 de Agosto de 2008.
- Learning XML: [creating self-describing data], Erik T. Ray, O'Reilly, 2003. ISBN 0596004206, 9780596004200.
- http://struts.apache.org/, enero de 2009.
- http://jquery.com/, enero de 2009.
- http://tomcat.apache.org/, mayo de 2008.
- http://java.sun.com/, mayo de 2008.
- http://www.mysql.com/, mayo de 2008.
- http://logging.apache.org/log4j/1.2/index.html, julio de 2009.
- http://www.exist-db.org/, julio de 2009.