MEJORAMIENTO Y EVOLUCIÓN DE LOS SISTEMAS SIMAT, SINEB

Ministerio de Educación Nacional

Contrato MEN-1006-2009

MANUAL TECNICO

SIMAT

V 2.0

Diciembre 2010

Historia de revisiones

Date	Versión	Descripción	Autor
31/07/2006	1.0	Creación.	Javier Sánchez, Pedro Buitrago EDESA
10/12/2010	2.0	Ajustes	Angelica Sánchez

Tabla de Contenido

Lista de Figuras

Documento de Arquitectura de Software

1. INTRODUCCION

1.1 Objetivo

Este documento sirve de guía para el personal técnico encargado del mantenimiento del Sistema de Matriculas SIMAT, en él se presentan aspectos relacionados con los requerimientos del sistema, el modelo de persistencia implementado y aspectos relacionados con instalación y DEPLOYMENT.

Este documento constituye el primer recurso de documentación que debe consultar el personal técnico encargado del soporte y mantenimiento del sistema. Se entiende que el lector tiene conocimientos previos relacionados con tecnología J2EE y FrameWorks como Tapestry y Hibernate.

1.2 Glosario

J2EE	Java 2 Enterprise Edition. Es la plataforma multinivel de Sun Microsystems para implementar aplicaciones empresariales de gran escala.
Hibernate	Un conjunto de herramientas y APIs para implementar persistencia en una aplicación J2EE. http://www.hibernate.org
Tapestry	Un framework de desarrollo para aplicaciones J2EE. Tapestry implementa un patrón MVC para la construcción y administración de interfaces usuario con tecnologías J2EE.
Patrones	Soluciones comprobadas a problemas típicos. Aplicar patrones de diseño a software aumenta la reusabilidad, la mantenibilidad, y la probabilidad de éxito.
Quartz	Es un sistema de programación de tareas que puede ser integrado con aplicaciones J2EE. Quartz permite que la definición de tareas sea implementada en java. De esta forma es posible ejecutar tareas en un momento del tiempo y/o con una frecuencia determinada.
Máquina Virtual de Java (JVM siglas en Inglés)	
DEPLOYMENT	
SIMAT	Siglas para Sistema de Matriculas, con este nombre se identifica la aplicación.

1.3 Referencias

[ALUR 2003]	Core J2EE Patterns, 2nd Edition. D. Alur, J. Crupi, D. Malks.

	Sun Microsystems Press. 2003
[JOHNSON 2003]	Expert one-on-one, J2EE Design and Development. R. Jonson. Wiley Publishing, 2003.
[BAUER 2004]	Hibernate in Action. C. Bauer, G. King. Manning Publications, 2004.
[LEWIS SHIP 2004]	Tapestry in Action. H. Lewis Ship. Manning Publications, 2004.
[GAMMA 1995]	Design Patterns. E. Gamma, R. Helm, R. Johnson, J. Vlissides. Addison-Wesley, 1995.
[TER 2004]	Documento de Términos de Referencia para el Diseño, Programación, Apoyo A Implantación, Capacitación Y Soporte Del Sistema De Apoyo Al Proceso De Matrícula Estudiantil. Concurso de Méritos 03/2004. Ministerio de Educación Nacional, 2004.
[EDESA 2004a]	Documento de Casos de Uso, v1.0. Unión Temporal EDESA-IIT, 2004.
[EDESA 2004b]	Documento de Procesos, v1.0. Unión Temporal EDESA-IIT, 2004.

1.4 Sobre el resto del documento

El documento pretende cubrir cada uno de los aspectos que requiere un ingeniero de software o analista para soportar SIMAT, de esta forma en el Capitulo 2. se presenta una descripción general de la arquitectura , el escenario de operación, los requerimientos de hardware y software y una descripción de los módulos del sistema. El Capítulo 3. describe la interfaz de usuario. En el Capítulo 4. se realiza una descripción de la persistencia y su implementación en SIMAT. El Capítulo 5. y 6. presentan el diagrama de Deployment y descripción de los archivos de configuración del mismo. El Capítulo 7. ilustra la creación de instaladores, su configuración y requisitos.

2. DESCRIPCION DEL SISTEMA

SIMAT es una aplicación WEB desarrollada en java y estructurada en tres capas, una capa de presentación, una capa de servicios y una capa de persistencia. La capa de presentación es soportada por Tapestry [1] y la capa de persistencia es manejada por Hibernate [2]. Este capítulo hace una introducción de los frameworks anteriormente nombrados y su relación entorno a la arquitectura planteada, adicionalmente se presentan los requerimientos de la arquitectura y se describen los módulos del sistema.

2.1 Arquitectura

La aplicación funciona en el entorno de un servidor de aplicaciones que provee diferentes servicios inherentes a la arquitectura J2EE como servicios de transacciones, de nombrado, etc. La aplicación esta dividida en tres capas de la siguiente forma:

CAPA	DESCRIPCION
Presentación	El frameWork Tapestry está encargado de la recepción de peticiones HTTP y respuesta por medio de la construcción archivos HTML[1]
Negocio	Los servicios del sistema implementan la lógica del negocio, estos servicios se encuentran en el paquete ***com.edesa.matri.service***
Persistencia	Esta capa esta implementada en los objetos del paquete ***com.edesa.matri.DAO***

En la Figura 1. se puede apreciar la distribución de las capas, de esta manera, Paginas contiene todos los archivos relacionados con clases utilitarias del sistema, clases utilitarias de tapestry, clases tapestry, archivos html, page, jwc, imágenes etc. Ahora, Servicios esta compuesta por todas las clases que se encuentran en el paquete ***com.edesa.matri.service*** y los DAOs se encuentran en el paquete ***com.edesa.matri.DAO***. Clases Persistentes está conformada por la abstracción de la Base de Datos siguiendo el frameWork Hibernate, estas clases se encuentran en el paquete ***com.edesa.matri.persistence***.

[1] Ver 2.1.1. Tapestry

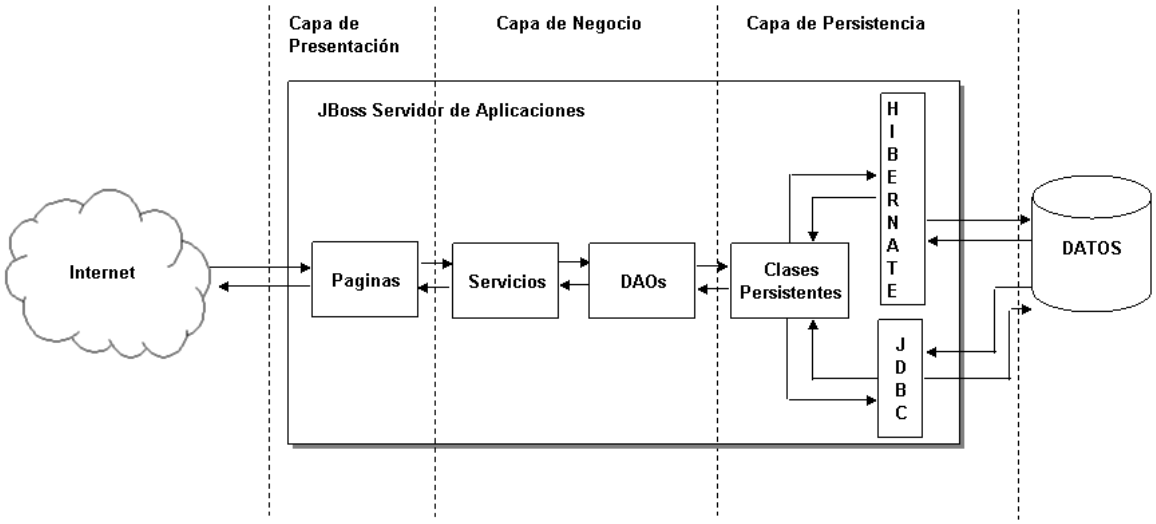

Figura 1. Diagrama de Arquitectura Software

2.2 Tapestry

Tapestry es un framework que trabaja sobre contenedores WEB o servidores de aplicación que soporten contenedores WEB. Esta construido bajo el Servlet API 2.2 y es compatible con versiones JDK 1.2 en adelante.

Tapestry se encarga de recibir las peticiones de lo usuarios, invocar la capa que implementa la lógica del negocio y enviar respuesta al usuario, lo cual hace de Tapestry un frameWork que permite implementar de forma clase la capa de presentación.

Tapestry es un framework basado en componentes donde cada página constituye un componente y a su vez un componente puede estar constituido por diferentes componentes.

Las páginas tapestry en general están compuestas de las siguientes partes:

Parte	DESCRIPCION
Plantilla	Constituido por un archivo HTML el cual puede tener inmersos tags especiales que son componentes sobre el framework y que en la renderización de la pagina se convierten en html.
Descriptor	Cada plantilla tiene asociado un descriptor en XML el cual tiene el mismo nombre único de su plantilla.
Clase	Constituido por una clase java provee un flujo para los diferentes casos de uso. La clase de una página es especificada en el descriptor de la misma.

SIMAT utiliza actualmente la version 3.0.3 de Tapestry.

2.2.1 Páginas Tapestry en SIMAT

Cada una de las páginas de SIMAT tienen la siguiente configuración de archivos:

pagina.html
pagina.page
referencia.java

donde pagina.page contiene una referencia a la clase de java (referencia.java) que controla el flujo en la misma. Esta referencia en el .page esta enmarcada con el siguiente tag:

```
<page-specification class="com.edesa.matri.*.referencia.java">
```

Si tomamos como ejemplo el modulo Adminitración/Permisos y Privilegios podemos saber cual es la página Tapestry asociada haciendo clic derecho. En el archivo podemos ubicar un tag similar al siguiente:

```
href="/app?service=direct/1/Accesos/$Border.$MenuItem$55.$DirectLink">
```

Con lo cual sabemos que la página asociada al módulo se llama Accesos, esta página la ubicamos en el directorio context/WEB-INF del proyecto donde podemos ubicar Accesos.html y Accesos.page y en este ultimo encontramos la referencia al archivo .java

```
<page-specification class="com.edesa.matri.roles.Accesos">
```

2.2.2 Componentes Tapestry en SIMAT

Cada uno de estos componentes tiene asociado un descriptor del componente con extensión .jwc por lo tanto el componente Border tiene asociado un archivo llamado Border.jwc, un archivo .java referenciado en el jwc

```
<component-specification class="com.edesa.matri.security.tapestry.NewBorder"
                allow-body="yes" allow-informal-parameters="yes">
```

y tiene asociado un archive Border.html, por supuesto este ultimo archivo no es requerido en la configuración de Tapestry ya que es posible que el html sea generado por el archivo .java, este caso tiene lugar el componente llamado ComboConstantes.

Los componentes en SIMAT son los siguientes:

Componente	Objetivo	Ejemplo de Uso

Border	Permite incluir tags comunes a todas las páginas de una app o módulo	Cualquier Página
ComboConstante	Combo que lista los hijos de una Cte padre	AdminReportes
ComboDivipola	Comboque lista los hijos de una Divipola padre	Consulta Alumnos
ComboJerarquia	Combo que lista los hijos de una Jerarquia padre	Vars
ComponentConstantes	Permite navegar a traves de las Ctes	Accesos
FinderInstituciones	Permite buscar Instituciones de una Jerarquia	MatriculaAlumnos
FinderInstituciones2	Permite buscar/seleccionar una institucion sin necesarimente buscar las sedes	LiberacionCupos
FinderSedes	Permite buscar Sedes a partir de una Institución	MatriculaAlumnos
LeerPersona	Permite ingresar y recibir datos basicos de una persona	Convenios
ListaOpcionesInstitucion	Tabla que permite seleccionar Instituciones junto con su modelo Educativo	InscripcionAlumnos
ToolBarComponent	Sirve para insertar una barra de herramientas, salvar,eliminar,etc	Cualquier Página
Wizard	Permite tener tabs en las páginas	MatriculaAlumnos

2.3 Hibernate

Hibernate tiene como objetivo facilitar y manejar la persistencia de objetos Java y a su vez permite realizar operaciones de consulta sobre los datos. Básicamente Hibernate se comunica con la fuente de datos a través de una capa intermedia que permite configurar, por supuesto, los parámetros que tienen que ver con la conexión como usuario, password, etc y adicional, y mas interesante, permite configurar el tipo de Base de Datos con quien se conecta, lo anterior se traduce en que una aplicación basada en Hibernate es prácticamente[2] independiente del motor de Base de Datos que se utilice.

Para utilizar el frameWork es necesario tener el archivo *.jar que se encuentra en la página de Hibernate, en SIMAT se utiliza la versión 2.1.8, además se requiere el(los) driver(s) de JDBC dependiendo del(los) motor(es) de Base de Datos al que nos conectemos. Hibernate requiere un archivo de propiedades llamado `hibernate.properties` en el cual se especifican las características de la conexión con el motor de base datos al igual que opciones adicionales del FrameWork[3].

Para la comunicación con la base de datos es necesario realizar mapeos de la misma en clases persitentes

[2] Esta independencia depende de las Bases de Datos que soporte Hibernate, para mayor información dirijase al sitio web http://www.hibernate.org/

[3] Para mayor información sobre la creación de aplicaciones con Hibernate diríjase al siguiente link http://www.hibernate.org/hib_docs/v3/reference/en/html/tutorial.html

java con lo que se conoce como Object/Relational Mapping (ORM) con lo cual simplemente se transforma la representación de los datos en la base de datos relacional a clases persistentes. En SIMAT los mapeos son generados por medio la traducción de comentarios XDOCLET en las clases Persistentes, estos objetos se encuentran en el paquete ***com.edea.matri.persistence***, los cuales generaran archivos de la forma nombreObjeto.hbm.xml, y son distribuidos, en SIMAT, en un archivo mappings-nombreAplicacion.jar.

En SIMAT, la tarea ANT que genera los mapeos (archivos hbm.xml) se llama ***hibernatedoclet***[4]

2.4 Servicios del Negocio

Los servicios del sistema se encuentran empaquetas, como se mencionó anteriormente, en ***com.edesa.matri.service***, estos servicios son los encargados de manejar la lógica del negocio. Los servicios pueden ser clasificados de tipo Consulta o Transaccional, cuando un servicio es de consulta simplemente realiza llamados a un DAO que ejecuta una consulta sobre los datos y en un bloque FINALLY (java) se cierra la sesión, ahora, si el servicio es de Tipo Transaccional su estructura básica es la siguiente:

Acción	Comando Sugerido
Abrir Transacción	*HibernateUtil.beginTransaction()*
Realizar los llamados a todos los DAOs requeridos por el servicio	*Instanciar los DAOs requeridos e invocar los métodos correspondientes*
Cerrar Transacción	*HibernateUtil.closeTransaction()*
En caso de error hacer un rollBack. En general, esta acción se lleva acabo en el segmento catch.	*HibernateUtil.rollBack()*
Cerrar la sesión, en general, esta acción tiene lugar en el segmento FINALLY del servicio	*HibernateUtil.closeSession()*

NOTAS:

Es muy importante resaltar que para efectos de Transaccionalidad un servicio solo llama DAOs, de esta forma la arquitectura evita abrir dos transacciones diferentes o cerrar sesiones de forma indeseada ya que los DAO del sistema no tiene manejo de sesiones asociado.

Un método de un servicio, por facilidad, se debe programar de manera que toda la transacción se lleve a cabo dentro del mismo, desde su inicio hasta su fin. Esto minimiza errores en la transacción y flush anticipados. Estos métodos solo deben llamar los métodos de los DAOs o métodos que no interfieran con la sesión o transacción.

[4] En el archivo de deploy build.xml se encuentra un target llamado *hibernate* en donde se realiza la invocación a la tarea ant que genera los mapeos.

Se puede abrir DAOs y llamar métodos de estos como sea necesario, mientras sea dentro del mismo Thread. Tanto la sesión como la transacción son exactamente una por Thread.

HibernateUtil, que es la clase que provee el manejo de sesiones y transacciones no permite que se ejecute más de una sesión por Thread. Esto es porque, por integridad de los datos, no se debe manejar mas de una sesión por operación. Es posible utilizar 2 sesiones en el mismo Thread, siempre y cuando se den por separado (hacer cerrado una antes de abrir la otra)

Commit y Rollback completan la transacción, debe hacerse una sola de las 2 por transacción. (no debe ejecutarse una después de la otra o se generara una excepción "No existe la transacción".

Al nivel de los servicios (Lógica del negocio) se deben evitar las operaciones directas sobre la base de datos (saves, updates, deletes, etc), esto es responsabilidad de la capa de los DAOs, que proveen operaciones puntuales sobre la base de datos a reutilizar por los servicios.

Si dentro del método se hacen consultas en la BD que hagan un flush automático y esto genera problemas en la transacción, el autoflush se puede desactivar para la sesión actual con la línea: HibernateUtil.getSession().setFlushMode(FlushMode.NEVER); Sin embargo, esto ocasionara que el commit no haga flush automáticamente, por lo cual se deberá hacer explicito (HibernateUtil.flushSession();) antes de hacer el commit.

En la siguiente tabla se listan las clases que implementan los servicios del negocio y se provee una breve descripción de los mismos[5]:

Para información mas especifica sobre los servicios del sistema recomendamos dirigirse al JAVADOC de la aplicación.

3. Requerimientos

3.1 Requerimientos soportados por la Arquitectura

La arquitectura representa una solución a los diferentes requerimientos planteados en el documento de Términos de Referencia del proyecto [TER 2004]. En especial la arquitectura soporta directamente los siguientes requerimientos:

REQUERIMIENTO	CÓMO LA ARQUITECTURA SATISFACE EL REQUERIMIENTO
010 - Transaccionalidad	Por medio de Hibernate (nivel de persistencia) se da soporte a las transacciones.

[5] Para conocer en detalle los métodos de estos servicios por favor remítase al javaDoc del Proyecto.

REQUERIMIENTO	CÓMO LA ARQUITECTURA SATISFACE EL REQUERIMIENTO
020 – Concurrencia	Tapestry + Hibernate aseguran acceso concurrente de múltiples usuarios a los mismos recursos de la aplicación.
040- Tiempos de respuesta	Con servidores y redes adecuadamente dimensionados, la arquitectura propuesta es perfectamente escalable horizontalmente en servidores de aplicaciones (clusters) para soportar eficientemente miles de usuarios concurrentes (sujeto a que el servidor de aplicaciones, tal como JBoss, soporte clusters). El escalamiento horizontal en el motor de base de datos requeriría soporte extra del manejador como lo da, por ejemplo, ORACLE 10g.
060 – Mantenibilidad	La arquitectura separa claramente las responsabilidades de los niveles de persistencia, servicios del negocio, e interfaz usuario. Esta separación de responsabilidades hace que el código sea más mantenible.
080 – Adaptabilidad	La arquitectura contempla que el sistema pueda funcionar con diferentes motores de bases de datos: Oracle, MS SQL Server, y PostgreSQL; diferentes servidores de aplicaciones J2EE tal como JBOSS; y diferentes sistemas operativos donde haya una implementación de JAVA (LINUX y MS Windows, por ejemplo).
Otros requerimientos solicitados por el Ministerio: • Usar Java y J2EE • Usar software abierto	• La arquitectura aplica patrones J2EE [ALUR 2003] y el lenguaje de desarrollo es JAVA. • Las herramientas de desarrollo y de deployment son open source tal como ECLIPSE, Tapestry, Hibernate y JBoss. • Si el motor de base de datos es PostgreSQL, todo el sistema sería Open Source y no se requiere pagar licenciamiento ni soporte.

3.2 Requerimientos de la Aplicación

REQUERIMIENTOS EN EL CLIENTE	
HARDWARE	**SOFTWARE**
Cualquier máquina donde corra Microsoft Internet Explorer V6 o Mozilla Firefox 1.5	JavaScript Habilitado en el Browser
Para información sobre requerimientos de Hardware para los browsers por favor remítase a la documentación de los mismos	Browser debe aceptar cookies
	Para requerimientos mínimos de software remítase a la documentación de los browsers

REQUERIMIENTOS EN EL SERVIDOR DE APLICACIONES	
HARDWARE	**SOFTWARE**
Cualquier máquina donde corra Java SDK 1.5	Cualquier sistema operativo donde corra Java SDK 1.5
Para los requerimientos mínimos de hardware de Java SDK 1.5 y JBoss 3.2.X refiérase a la documentación de cada producto	Java SDK 1.5 instalado
SIMAT requiere inicialmente para subir alrededor de 200MB en RAM y unos 2Mb por usuario concurrente	Instalador de SIMAT (incluye JBoss 3.2.X y las librerías requeridas para su funcionamiento)

REQUERIMIENTOS EN EL SERVIDOR DE BASE DE DATOS	
HARDWARE	**SOFTWARE**
Cualquier máquina soportada por Oracle Server 9i/10g, PostgreSQL 7/8 o MS SQL 6/7	Para los requerimientos mínimos de software refiérase a la documentación de Oracle Server 9i/10g, PostgreSQL 7/8 o MS SQL 6/7
Para los requerimientos mínimos de hardware refiérase a la documentación de Oracle Server 9i/10g, PostgreSQL 7/8 o MS SQL 6/7	SIMAT requiere el motor de base de datos previamente instalado y un esquema de usuario en blanco para crear las tablas del aplicativo
Se requiere de aproximadamente 5Gb de disco por cada millón de niños cargados por cada año lectivo	

4. HERRAMIENTAS Y SOFTWARE DE BASE

Las herramientas de desarrollo y software de base se escogieron en su mayoría Open Source y que además tuviesen soporte para construir y administrar sistemas J2EE:

TIPO	HERRAMIENTA	REFERENCIA
IDE (Integrated Development Environment)	ECLIPSE 3.1	http://www.eclipse.org/ Compite directamente en prestaciones con herramientas comerciales como Borland JBuilder® y Oracle JDeveloper®. Es extensible por medio de plugins
Framework de desarrollo J2EE.	Tapestry 3.0	http://jakarta.apache.org/tapestry/ Es un framework para J2EE que implementa, a parte de muchas otras cosas, un patrón MVC para la construcción y administración de interfaces usuario. Se puede montar sobre diversos contenedores y

TIPO	HERRAMIENTA	REFERENCIA
		servidores J2EE tales como JBoss, Jetty, y Tomcat.
ORM (Object/Relational Mapping) para manejo de persistencia.	Hibernate 2.1	http://www.hibernate.org/ Permite a una aplicación JAVA/J2EE manejar clases persistentes sobre bases de datos relacionales. También tiene soporte a transacciones y un lenguaje de consulta (HQL) orientado a objetos.
Motor de Base de Datos de prueba	Oracle 9iR2	Inicialmente se está utilizando en el ambiente de pruebas una base de datos Oracle. Más adelante se conducirán pruebas contra PostgreSQL y MS SQLServer.
Modelador UML	Enterprise Architect v4.1	http://www.sparxsystems.com.au/ Soporta UML 2 y varias extensiones para elaborar diagramas de Procesos, Bases de datos relacionales, y otros.
Manejador de Transacciones	Cualquiera soportando JTA	http://java.sun.com/products/jta/ Hibernate tiene una interfaz JTA para ser usada con cualquier manejador de transacciones que soporte JTA, tal como el de JBoss. También es posible configurar el sistema para ser usado con transacciones JDBC si no se dispone de una implementación JTA.

5. INTERFAZ DE USUARIO

El objetivo de este capítulo consiste en ilustrar la asociación de los módulos del sistema con las páginas que lo componen, de esta forma constituirse en una ayuda a la hora de realizar operaciones de mantenimiento sobre el sistema.

Cada uno de los módulos en la aplicación es compuesto al menos por una página la cual tiene asociado un nombre de archivo con extensión html, de igual forma cada página que pertenece a un módulo tiene asociado un identificador asociado con el código de la constante que lo representa, lo anterior con el fin de realizar validaciones de seguridad y acceso.

OPCION ADMINISTRACION			
MODULO	**PAGINA PRINCIPAL**	**ID MODULO**	**PAGINAS SECUNDARIAS**
Ajustes a Duplicados	AlumnosDuplicados	0223	NuevoAlumno InscripcionAlumnos Traslados
Ajuste Importaciones	AjusteImportaciones	0224	

OPCION ADMINISTRACION			
Calendario Default	CalendarioDefault	0213	
Cambiar Clave	CambioClave	0219	
Carga de Datos	UploadData	0210	
Constantes	AdminConstantes	Solo ROOT	
Cortes	Cortes	0204	
Divipola	AdminDivipola	Solo ROOT	
Estrategias	Estrategias	0225	
Jerarquia	Jerarquias	0203	
Niveles	Niveles	0205	
Novedad Traslados	Buscar Alumnos	0226	NuevoAlumno NovedadTrasladar InscripcionAlumnos
Parámetros X Secretaria	ParametrosAdministracion	0211	
Permisos y Privilegios	Accesos	0217	
Permisos para Reportes	PermisosReportes	0222	
Pesos Variables	PesoVars	0215	
Procesos Matricula	Procesos	0212	
Procesos por Lotes	ProcesosLotes	0220	
Programación de Tareas	ProgramacionTareas	0218	Frecuencia
Reglas de Estado	Transición	0208	
Reportes	AdminReportes	0209	
Roles	Roles	0206	
Usuarios	UsuariosBusqueda	0207	UsuariosAdministracion
Variables	Vars	0214	
Auditoria	ManejaHibernateUtilAudit	0221	

OPCION INSTITUCIONES			
MODULO	**PAGINA PRINCIPAL**	**ID MODULO**	**PAGINAS SECUNDARIAS**
Directorio	BuscarInstituciones	0301	Instituciones BuscarSedesInst Sedes
Grupos	AdminGrupos	0302	
Fusiones	Fusiones	0303	CrearFusion DetalleFusion
Convenios	Convenios	0304	
Modelos por Grado	InstAndGrupos	0305	

OPCION ESTUDIANTES			
MODULO	**PAGINA PRINCIPAL**	**ID MODULO**	**PAGINAS SECUNDARIAS**
Registro de Estudiantes	BusquedaAlumnosPorInstitucion	0401	SedesInstitucion BuscarAlumnosSede
Consulta de Alumnos	ConsultaAlumnos	0402	EstadosAlumno Novedades

OPCION PROYECCIONES			
MODULO	**PAGINA PRINCIPAL**	**ID MODULO**	**PAGINAS SECUNDARIAS**
Parámetros Grupos	PryPgrps	0501	
Proyectar Cupos	ProyeccionMatriz	0502	PlaneacionConvenios
Saldo Cupos	SaldoCupos	0503	

OPCION INSCRIPCIONES			
MODULO	**PAGINA PRINCIPAL**	**ID MODULO**	**PAGINAS SECUNDARIAS**
Nueva Inscripción	BuscarAlumnos	0602	NuevoAlumno NovedadTrasladar InscripcionAlumnos
Modificar Inscripción	BusquedaInscripciones	0601	InscripcionAlumnos NuevoAlumno
Cupos Rectores	AsignaCupoRector	0603	
Asignación de Cupos Nuevos	AsignacionCupos	0605	
Anular Asignaciones	AsignaCupoRector	0606	

OPCION MATRICULA			
MODULO	**PAGINA PRINCIPAL**	**ID MODULO**	**PAGINAS SECUNDARIAS**
Promocion	Promocion	0701	
Registrar Reprobados	BusqInstRep	0702	RegistrarRepitencia
Anular Reprobados	AnularRepitencia	0703	
Matricular	MatriculaAlumnos	0704	
Cambios a Matriculados	CambioGradoMatri	0705	
Retirar Estudiante	TrasladoMatri	0706	
Solicitar Traslado	Traslados	0707	NuevoAlumno
Novedades	ConsultaAlumnosNovedades	0708	NuevoAlumno Novedades
Asignación Masiva de Traslados	AsignacionTraslado	0709	
Prematrícula	Prematricula	0710	Traslados
Asignar Estrategias	AsignaEstrategias	0713	
Liberación de Cupos	LiberacionCupos	0711	
Transferencias	Transferencias	0712	

OPCION REPORTES			
MODULO	**PAGINA PRINCIPAL**	**ID MODULO**	**PAGINAS SECUNDARIAS**
Reportes	ExecReport	08	
Resolución 166	Resolucion166	0801	
Reportes Planos	ReportesAlumnosCriterios	0802	

Existen algunas páginas que en la columna ID MODULO dicen “SOLO ROOT” lo cual indica que son páginas accesibles únicamente por el usuario ROOT. La columna PAGINA PRINCIPAL indica el nombre del archivo html que es lanzado cuando se ingresa la módulo y las PAGINAS SECUNDARIAS son archivos html que pueden ser accesados desde la página principal.

6. DESCRIPCION DE LA PERSISTENCIA

En este capítulo se describe el modelo de persistencia que se esta utilizando por medio de la descripción de diagramas de clases (Clases Persistentes) por medio de las cuales el sistema hace referencia a los datos. Cabe recordar que algunas operaciones de acceso a los datos no tienen en cuanta la capa intermedia de HIBERNATE en su lugar utilizan JDBC directamente para este fin.

Para información sobre la documentación referente a la Base de Datos el debe se debe remitir al anexo llamado ***docDataBase*** en donde se realiza una descripción general de las tablas que soportan la aplicación y sus respectivos campos.

La descripción de los diagramas de clase se realizará por partes intentando describir de forma general y ordenada la disposición del modelo de persistencia.

En general, de la mayoría de los diagramas se excluyen algunas asociaciones que no ofrecen valor agregado a los diagramas dado el fin que tiene cada uno de ellos, de esta forma las asociaciones que tenga un clase con otra dentro de un contexto pueden ser o no relevantes. De igual manera, del diagrama se excluye la relación que tienen la mayoría de los objetos persistentes que la clase BasePersister de la cual heredan todos los objetos persistentes. BasePersister permite generalizar campos de auditoria para los objetos del negocio.

El lector puede consultar el JavaDoc de la aplicación en el link ***persistente*** en donde se detalla información sobre los mapeos y los atributos los cuales permiten identificar todas las asociaciones existentes en el modelo.

6.1 Estado de los Alumnos

La Figura 2. ilustra las asociaciones existentes que soportan los cambios de estado de los Alumnos lo cual constituye la funcionalidad mas importante de la aplicación. En el diagrama es claro que EstadoAlumno

constituye un historial de los estados de un alumno donde el estado está constituido por una serie de atributos como grupo, jornada, sede, institución, etc. Es claro que un conjunto de EstadoAlumno esta asociado a un Alumno lo cual constituye un historial, además, un Alumno tiene un EstadoAlumno que constituye la referencia al último estado de un alumno.

En el Diagrama, igualmente, se ilustran los cortes que se realizan al sistema, estos cortes constituyen una fotografía de los EstadoAlumno de una Jerarquia.

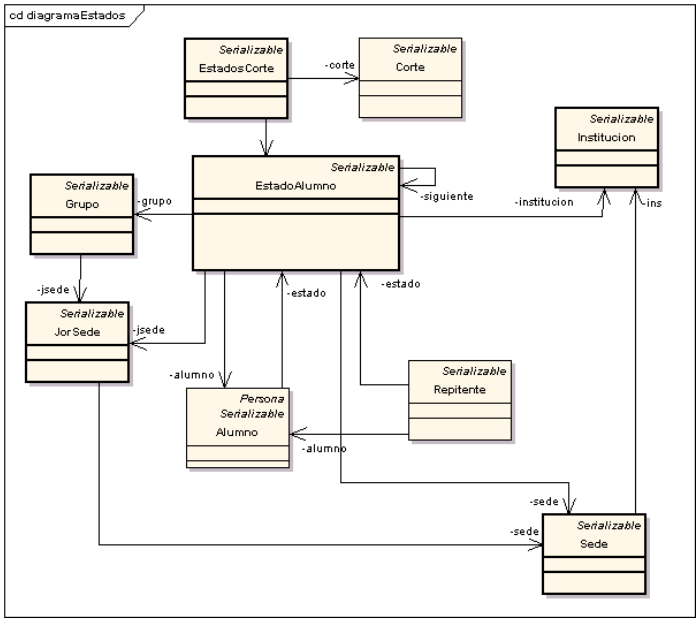

Figura 2. Diagrama de Clases. Contexto Estados de un Alumno

6.2 Convenios

En la Figura 3. se describen las asociaciones que modelan los convenios entre las instituciones y por supuesto, los alumnos que son asociados a un convenio.

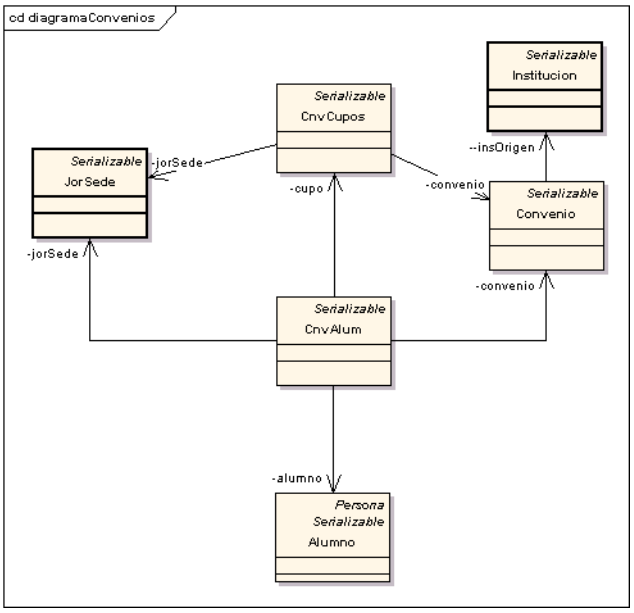

Figura 3. Diagrama de Clases. Contexto Convenios

6.3 Personas

Este punto se ilustra el uso de la clase Persona en el modelo de persistencia. En la Figura 4 en primer lugar es importante notar que un Alumno es una Persona[6], un usuario no es una Persona pero al igual que un Familiar están asociados con una Persona.

En la Figura 4. es claro que una Persona tiene asociado un Nombre, una Direccion y una Identificacion, a su vez es claro que una Identificacion y una Direccion estan asociadas con una Divipola, de donde es expedida y a donde pertence respectivamente.

[6] Es una Persona de Tipo Alumno

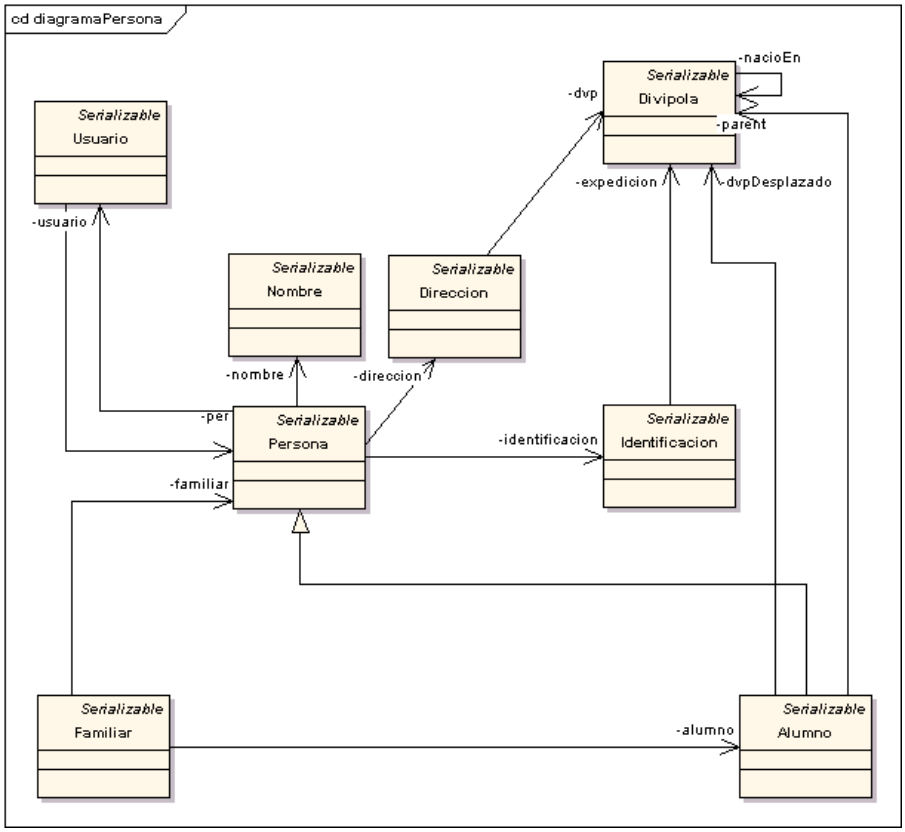

Figura 4. Diagrama de Clases. Contexto Persona

6.4 Seguridad

La seguridad de la aplicación esta basada en roles, donde cada uno de los usuarios tiene un rol, donde los roles son definidos por Jerarquia. A su vez, un rol tiene acceso a las opciones del menú y a los módulos del sistema y este acceso está enmarcado en términos de Creación, Lectura, Actualización y Eliminación y por supuesto en la aplicababilidad que cada uno de estos tenga en un módulo en particular. De la misma forma, un se generan permisos de generación con respecto a los reportes que tiene disponibles el sistema. La figura 5. ilustra el diagrama de clases en el contexto de la seguridad.

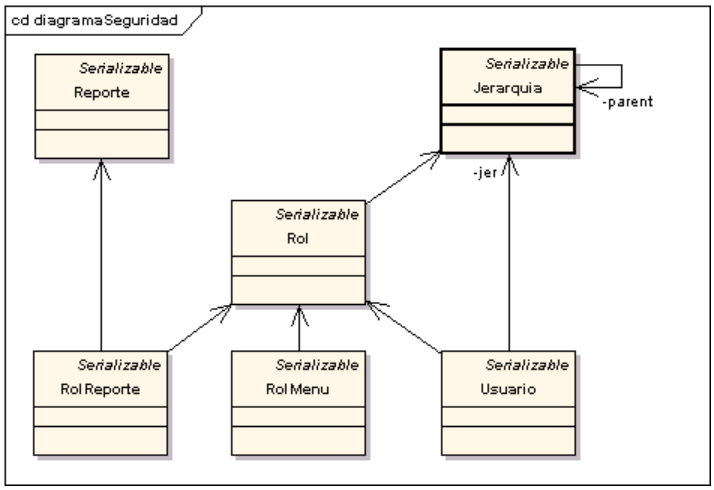

Figura 5. Diagrama de Clases. Contexto Seguridad

6.5 Estrategias

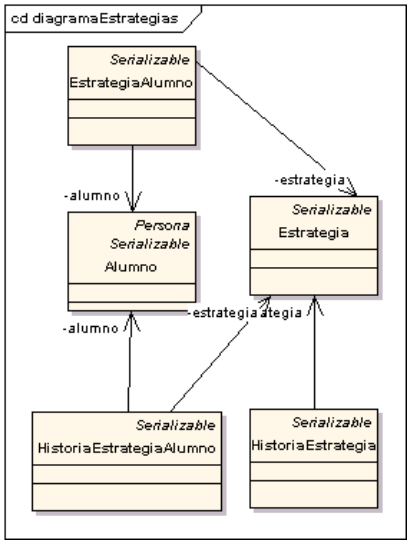

Figura 6. Diagrama de Clases. Contexto Estrategia para continuidad en la Educación

La Figura 6. ilustra el modelo que da lugar al manejo de las Estrategias asociadas a los Alumnos, también se ilustra el historial de estrategias asociadas a los alumnos y el historia de las operaciones que se realizan sobre las estrategias.

6.6 Inscripciones

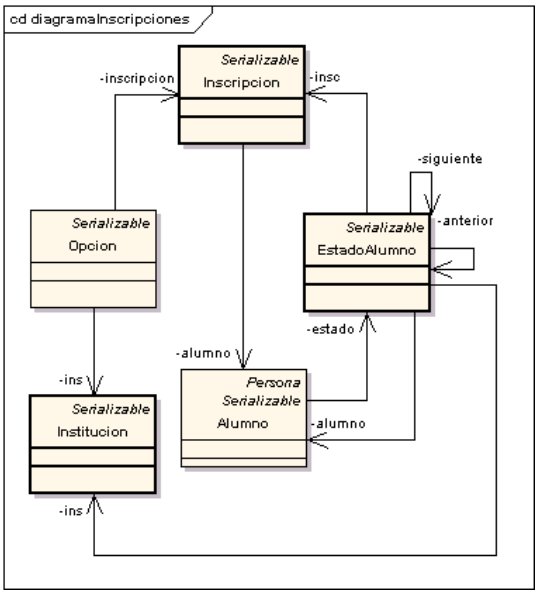

Figura 7. Diagrama de Clases. Contexto Inscripciones

La Figura 7. ilustra las clases persistentes involucradas en el proceso de Inscripción, donde una Opcion se traduce una Institucion a la cual se inscribe un Alumno.

6.7 Traslados

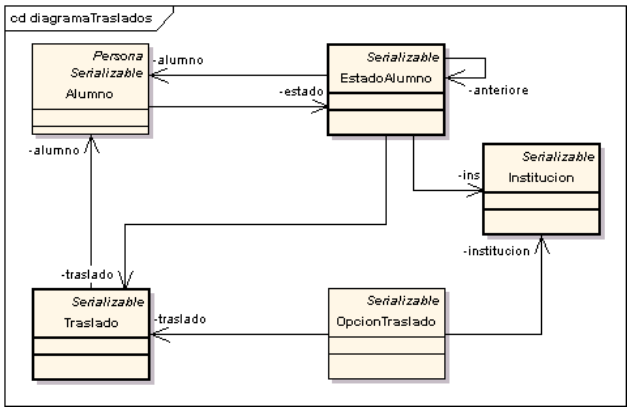

Figura 8. Diagrama de Clases. Contesto de Traslados

La Figura 8. ilustra las clases persistentes involucradas en el proceso de solicitud de Traslados, donde las Opciones de Traslados son opciones de Instituciones a la cuales un Alumno se desea trasladar.

6.8 Jerarquía Institucional

El diagrama de la Figura 9. ilustra la jerarquía de las organizaciones del Sistema, donde las organizaciones son las Instituciones, Sedes, JorSedes, Grados y Grupos. En el diagrama no se aprecian los grados dado que estos se representan con Ctes por lo cual no es práctico incluirlo. En General, un grupo está asociado a una JorSede y a un Grado, la JorSede representa una Jornada, también representada con una Cte, que pertence a una Sede y una Sede pertenece a una Institución, finalmente una Institución se encuentra en una Jerarquia.

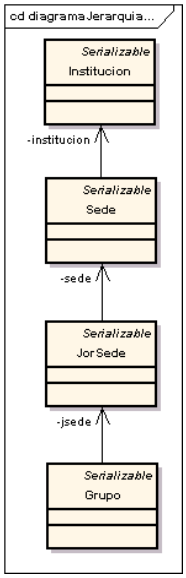

Figura 9. Diagrama de Clases. Contexto de la Jerarquía Institucional

6.9 Proyecciones

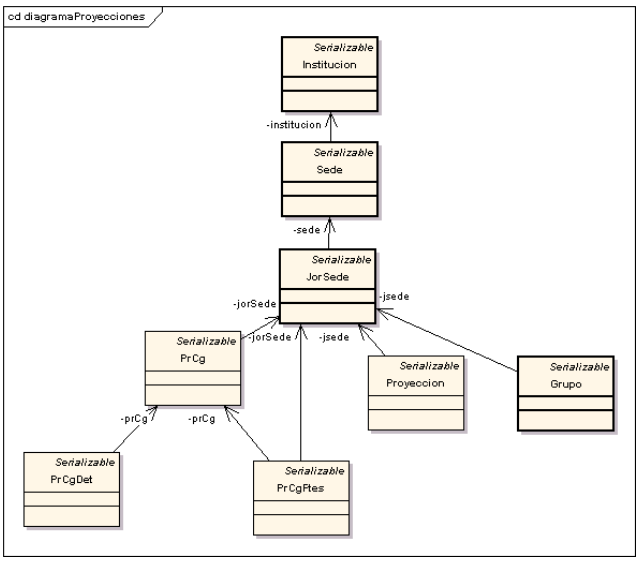

Figura 10. Diagrama de Clases. Contexto de Proyecciones, cupos y Grupos

En la Figura 10. se ilustran las relaciones que dan lugar a la realización de proyecciones de grados, grupos y cupos. Por medio de las clases PrCg, PrCgDet y PrCgFtes se mantienen las proyecciones del sistema con respecto a jornadas y grados. Ahora, por medio de las clases persistentes Proyeccion y Grupo tiene lugar tanto el mantenimiento de los grupos como su creación y cupos a nivel de grupos.

6.10 Otras Clases

En la Figura 11. se ilustran clases que son base, clases de configuración y además se incluyen las clases que soportan los procesos por lotes. Divipola abstrae la división política Colombiana que se requiere para la

configuración de sitios de expedición, ubicación, etc. Transicion abstrae las reglas que rigen los cambios de estado de los alumnos. Cte contiene todas las constantes de la aplicación como grados, metodologías, EPS, resguardos, Etnias, etc. Jerarquia abstrae el concepto de jerarquías y secretarias, de educación en la organización del Ministerio de Educación Nacional. Reporte constituye la configuración de reportes del sistema. Proceso parametriza lapsos para realización de los procesos del sistema como las matriculas, las inscripciones, novedades, etc. Constraint configura la traducción de validación de los Constraints de la base de datos. También se presenta la clase que mapea una vista materializada para la generación del Anexo 6A.

En General, estas clases son base tanto para la configuración de los objetos del negocio como para la realización de los procesos de la aplicación.

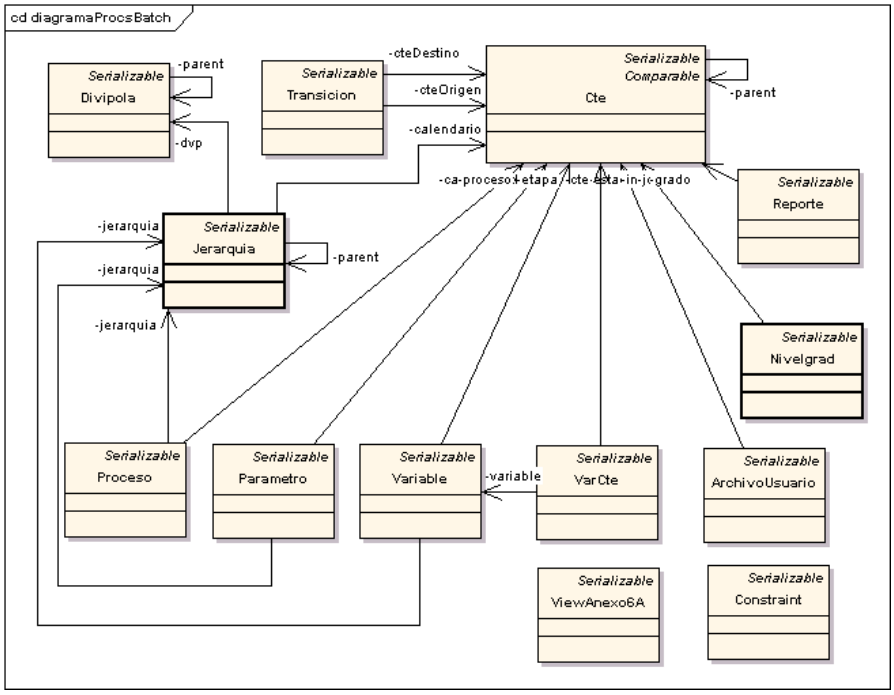

Figura 11. Diagrama de Clases. Contexto clases auxiliares y de configuración.

7. DIAGRAMA DE DEPLOYMENT

Dentro del servidor de aplicaciones el diagrama de deployment luce de la siguiente manera:

+ *nombreAplicacion.ear*
  + *META-INF*
    - *application.xml*
  + *nombreAplicacion.war*
    + *WEB-INF*
      +*classes*
        + *com.edesa.matri…*
        + *com.edesa.aspect…*
        + *com.edesa.audit…*
        + *com.edesa.men…*

+*lib*
+*web.xml*
+ *reports*
+ *css*
+ *img*
- *constantes.properties*
- *ehcache.xml*
- *matri.properties*
- *quartz.properties*
+ *driver-oracle.jar*
+ *hibernate-nombreAplicacion.har*
+ *quartz.jar*

8. CREACIÓN DE UNA APLICACIÓN

Para la creación de una aplicación como SIMAT se utilizan diferentes archivos de configuración generar el deployment. El esquema está basado en plantillas las cuales son configuradas por medio de un archivo de propiedades llamado *ant.properties* y ejecutadas por el archivo de generación del proyecto *build.xml*.

El archivo *ant.properties* y el *build.xml* permite configurar las diferentes plantillas del proyecto como el archivo *web.xml, hibernate.properties, hibernate-service.xml, log4j.xml, application.xml, jboss-app.xml, jboss-web.xml, mail-service.xml, matri.properties, oracle-ds.xml,sqlserver-ds.xml,postgres-ds, quartz.properties, treecache.xml.* Estas plantillas son eventualmente son publicadas en servidor de aplicaciones con el fin de configurar su comportamiento, las características que debe utilizar y, en general, la configuración de los servicios de la aplicación.

En el archivo *build.xml* tiene lugar la publicación de los servicios del motor de Base de Datos, la compilación de las clases de java por medio de la tarea de ant *iajc*, la generación de los archivos de mapeo que utiliza Hibernate el cual se realiza por medio de la tarea *xdoclet.modules.hibernate.HibernateDocletTask*, se publica el servicio de hibernate en el servidor de aplicaciones y crea los archivos de distribución WEB y empresarial WAR y EAR respectivamente.

```
ant.home=C:/apache-ant-1.6.2
xdoclet.home=C:/xdoclet-1.2
tapestry.home=C:/Tapestry-3.0.4
java.home=C:/jre1.5.0_06/
hibernate.home=C:/hibernate-2.1.8
izpack.home=D:/Java/IzPack

jboss.home= C:/jboss-3.2.7
jboss.configuration=default
```

Las variables anteriormente presentadas configuran la ubicación de las librerías requeridas para llevar a cabo la compilación del proyecto, la ubicación del servidor de aplicaciones y en que configuración se del servidor de aplicaciones jboss se realizará la publicación.

Ahora, el data-source puede incluirse en el archivo de distribución EAR o puede ubicarse fuera, esta ultima opción permitiendo actualizarlo en caliente. Esta característica es configurada con la siguiente variable:

```
application.datasource-ear=false
```

y se lleva a cabo en el build.xml por medio de la siguiente

```
<if>
        <equals arg1="${application.datasource-ear}" arg2="true" />
        <then>
                <replace file="${build.ear.dir}/jboss-app.xml">
                        <replacefilter token="INIT_DS_DEPLOY" value="--&gt;"/>
                        <replacefilter token="END_DS_DEPLOY" value="&lt;!--"/>
                </replace>
                <echo message="[jboss-app.xml] Se incluyo el datasource en el EAR." />
                <ear destfile="${build.deploy.dir}/${ear.file}" appxml="${build.ear.dir}/application.xml">
                        <fileset dir="${build.deploy.dir}" includes="*.war,*.har,${ds.file}"/>
                        <fileset dir="${build.ear.dir}" includes="*.jar"/>
                        <metainf dir="${build.ear.dir}" includes="jboss-app.xml"/>
                </ear>
                <copy todir="${local.jboss.deploy.dir}">
                  <fileset dir="${build.deploy.dir}" includes="${application.mappings.name},${ear.file}"/>
                </copy>
        </then>
         <else>
           <echo message="[jboss-app.xml] No se incluyo el datasource en el EAR. Deploy por separado." />
           <ear destfile="${build.deploy.dir}/${ear.file}" appxml="${build.ear.dir}/application.xml">
                        <fileset dir="${build.deploy.dir}" includes="*.war,*.har"/>
                        <fileset dir="${build.ear.dir}" includes="*.jar"/>
                        <metainf dir="${build.ear.dir}" includes="jboss-app.xml"/>
           </ear>
        <copy todir="${local.jboss.deploy.dir}">
         <fileset dir="${build.deploy.dir}" includes="${ds.file},${application.mappings.name},${ear.file}"/>
       </copy>
 </else>
</if>
```

De forma similar se configuran los nombres del archivo de mapeo mappings-AppName.JAR, archivos para importación y procesos en batch, la ubicación de los archivos plantilla para los reportes del sistema y ubicación del log del sistema.

```
application.imports.path=${jboss.home}/server/${jboss.configuration}/archivos-
${application.name}
application.reports.path=${jboss.home}/server/${jboss.configuration}/reports
application.logging.path=${jboss.home}/server/${jboss.configuration}/log
```

También se parametriza la conexión con la base de datos. Tenga en cuenta que la ip de la Base de Datos puede ser pública o privada dependiendo de si su servidor de aplicaciones se encuentra en la misma red del motor de base de datos o fuera de ella.

```
database.type=oracle
database.port=1521

database.username=user
database.password=pwd
database.name=nombre de la Base de Datos
```

```
database.address=ip de la base de datos
database.connstring=jdbc:oracle:thin:@${database.address}:${database.port}:${da
tabase.name}
database.driver=oracle.jdbc.driver.OracleDriver
```

La mensajería de la aplicación también debe ser configurada al igual de parámetros comunicación con los datos de Hibernate. Es importante recordar que la mensajería depende de que existe un servicio de SMTP disponible para la aplicación.

```
application.mail.jndiname = java:/Mail
application.mail.user =
application.mail.password =
application.mail.transport.protocol = smtp
application.mail.store.protocol = pop3
application.mail.pop3.host = localhost
application.mail.smtp.host = localhost
application.mail.from = simat@edesa.net

application.showsql=false
application.dialect=net.sf.hibernate.dialect.Oracle9Dialect
application.cache=net.sf.hibernate.cache.HashtableCacheProvider
```

Dado que el sistema utiliza Quartz con el fin de realizar procesamiento programado por lotes es necesario configurar la herramienta como tal y los procesos que se soportan en la aplicación, como la importación de archivos y los procesos en batch. En las siguientes lineas se configura Quartz para trabajar máximo con dos hilo y se le indica al archivo build.xml que ajuste la plantilla WEB:XML para configurar los servlets de ImportServlet y ProcBatchServlet ya que estos son quienes se encuentran con marca TRUE.

```
quartz.activo = true
quartz.threads = 2
quartz.dbpool.size = 2
quartz.clustered = false

application.import=true
application.import.cronexp=0 0/5 * * * ?

application.etapas=false
application.etapas.cronexp=0 0 0 * * ?

application.backup=false
application.backup.cronexp=0 0 0 * * ?

application.cortes=false
application.cortes.cronexp=0 0 0 * * ?

application.batch=true
application.batch.cronexp=0 0/3 * * * ?
```

Si se tiene una configuración de dos aplicaciones apuntando a la misma Base de Datos solo la primera que haya sido lanzada de ellas obtendrá el servicio de Quartz y configurará el SCHEDULER.

9. INSTALADOR

9.1 Alcances del Instalador

El Sistema de Matrículas del Ministerio de Educación Nacional, SIMAT, cuenta con un ejecutable para su instalación, y además con la capacidad de generar dicho archivo de instalación con parámetros personalizados. El objetivo de estas funcionalidades, es facilitar a usuarios administradores el proceso de despliegue de la aplicación. Este instalador lleva a cabo las siguientes operaciones:

- Instalación de Ant (Herramienta que genera la aplicación final)
- Instalación de JBoss (Servidor de aplicaciones J2EE de libre distribución)
- Generación de la aplicación usando Ant, a partir de los parámetros dados por el usuario.
- Configuración de JBoss y despliegue de la aplicación.
- Ejecución de *scripts* de base de datos que creen las tablas y los datos iniciales requeridos por el SIMAT. Los sistemas manejadores de Bases de Datos soportados por el SIMAT son Oracle 9i, Postgresql 8 o superior y SQL Server 6.5 o superior.

El instalador se ejecuta de la misma manera tanto en sistemas Windows como Linux (ver página 31), sin embargo no contempla las siguientes características soportadas por el SIMAT:

- Instalación de SIMAT en clúster usando servidores JBoss, con replicación de sesiones y de caché de objetos.
- Instalación en OC4J u otros servidores J2EE.
- Instalación de la aplicación en un servidor JBoss o utilizando un ant previamente instalados.
- Instalación del sistema manejador de base de datos. Este manual tampoco cubre esta información.

9.2 Instalación de una nueva instancia de SIMAT

9.2.1 Requerimientos

Para llevar a cabo el proceso de instalación y para la ejecución del SIMAT, se necesita tener instalado el J2SE (Java 2 Stardard Edition) versión 1.4.2 o superior. El instalador de java asegura que el *runtime* de dicho lenguaje quede disponible desde la línea de comando (queda declarado en la variable de entorno PATH del respectivo sistema operativo). Sin embargo, se recomienda verificar la versión ejecutando:

```
java -version
```

Esto, debido a que otras aplicaciones que utilicen Java pueden imponer su propia versión de java. Para arreglar esto, asegúrese que la ruta de la versión pertinente de java quede delante de las demás en la variable de ambiente PATH.

Como se había mencionado en la sección de Alcances (página 30), este manual no cubre la instalación del motor de base de datos. El usuario encargado de la instalación debe llevar a cabo este proceso y debe crear un usuario exclusivo y con permisos de administración en dicho motor para el SIMAT. Asimismo, deberá crear los respectivos *tablespaces* que soporten la cantidad de datos que va a manejar el SIMAT.

Aproximádamente se requieren unos 100Mb por cada 10,000 alumnos en el sistema. El crecimiento anual depende de las novedades de los niños, pero puede ser alrededor de un 15%.

9.2.2 Ejecución del Instalador

El archivo de instalación del Sistema de Matrículas consiste en un archivo de Java (Java Archive) con extensión .jar, que se ejecuta desde la línea de comando a través de la instrucción:

```
java -jar matri-install.jar
```

Luego de esto, el instalador se cargará en memoria y aparecerá la siguiente ventana:

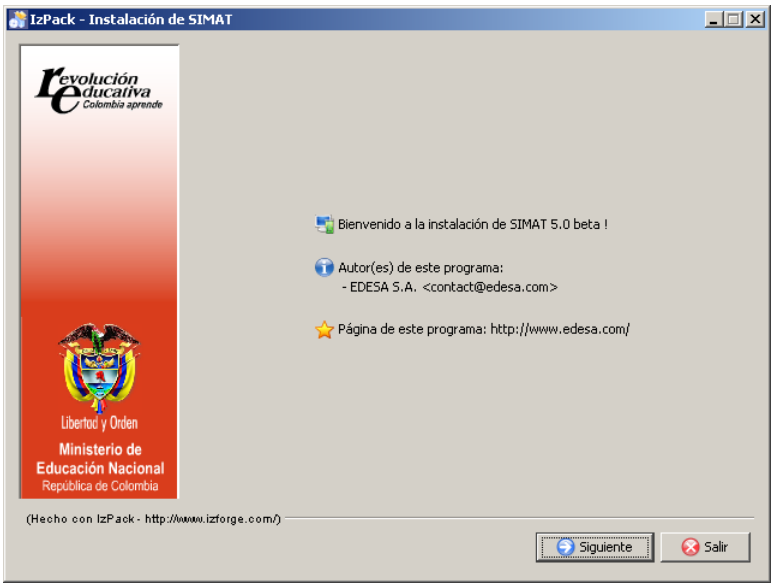

Figura 11. Pantalla de bienvenida del instalador de SIMAT

Al hacer clic en "Siguiente", se mostrará información de último momento (o *readme file*) que EDESA S.A. considera importante que el usuario conozca antes de instalar la aplicación. Al pulsar "Siguiente", se mostrará la pantalla con la licencia, es necesario aceptarla y hacer clic en "Siguiente" para continuar con el proceso. Luego de esta pantalla, comienza la captura de datos para la instalación:

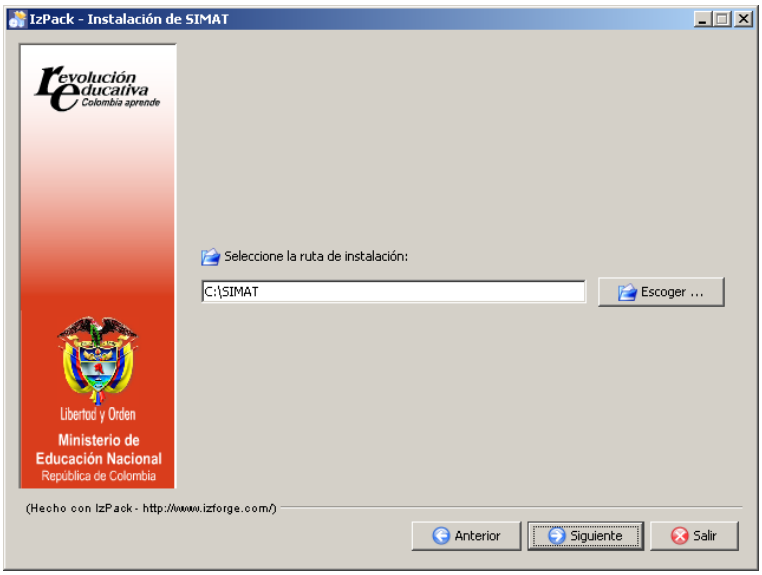

Figura 12. Pantalla de ruta de instalación

En esta pantalla deberá seleccionar la ruta final donde quedarán los archivos del sistema SIMAT. Dentro de esta ruta también quedará instalado el servidor de aplicaciones Jboss. Al pulsar "Siguiente" se le mostrarán al usuario los diferentes paquetes que conforman la instalación del sistema de matrículas. Por ahora, todos los paquetes son obligatorios.

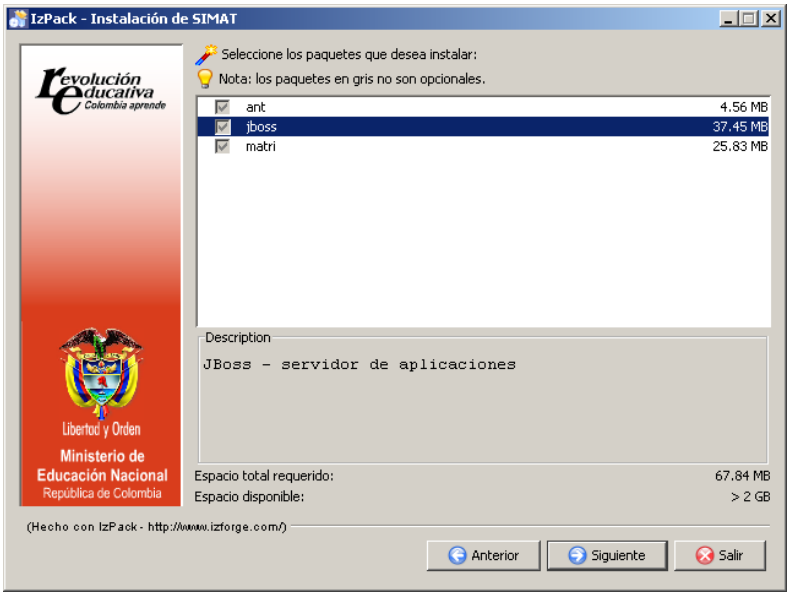

Figura 13. Pantalla de paquetes de archivos que conforman el SIMAT

Al hacer clic en "Siguiente", se mostrará la pantalla que captura los parámetros generales de la aplicación de matrículas:

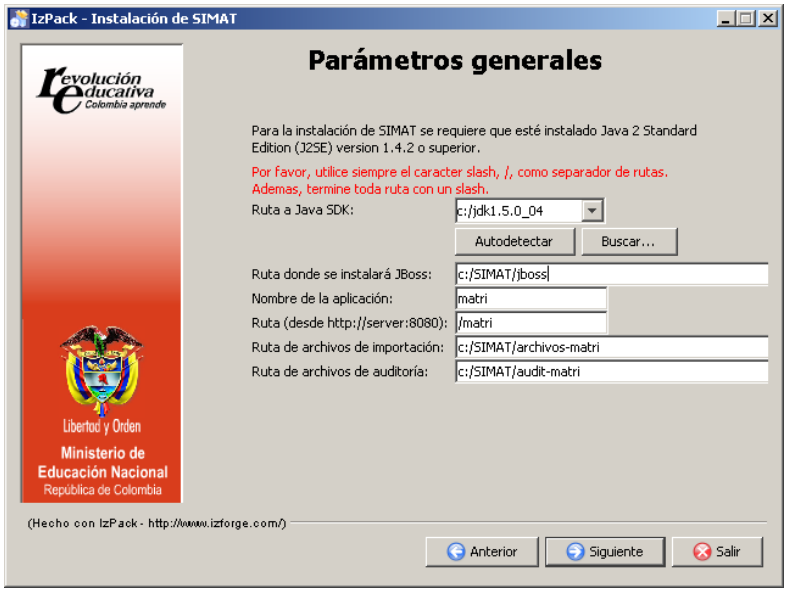

Figura 13. Pantalla de parámetros generales de instalación del SIMAT

Los parámetros que se piden en esta pantalla son los siguientes:

Ruta a Java SDK	Directorio raíz del Java 2 Standard Edition.
Ruta donde se instalará Jboss	Ruta donde se desea instalar Jboss, es necesario escribir la ruta completa, no se pueden especificar valores relativos a partir de la ruta de instalación.
Nombre de la aplicación	El nombre de la aplicación es vital para diferenciar esta aplicación de otras iguales que en un futuro pudieran estar desplegadas en el mismo servidor. Este proceso de instalación solo permite la creación de una sola aplicación.
Ruta de la aplicación	URL a través del cual se planea acceder a la aplicación. El usuario solo debe digitar el resto del URL desde http://<nombre_servidor>:8080, es decir, algo como por ejemplo: "/matri", o simplemente "/" (sin las comillas) para acceder desde la raíz del servidor. El *slash* ("/") al principio es absolutamente necesario.
Ruta de archivos de importación	Ruta en el sistema de archivos donde se desea almacenar los archivos que los usuarios publiquen para importación, y los archivos de salida de dicho proceso.
Ruta de archivos de auditoría	Ruta en el sistema de archivos donde se desea almacenar los archivos HTML de auditoría que genera la aplicación.

Al llenar la información del sistema y oprimir "Siguiente", el instalador le pide al usuario los datos acerca de la base de datos sobre la cual operará el SIMAT, como se puede ver en la siguiente figura:

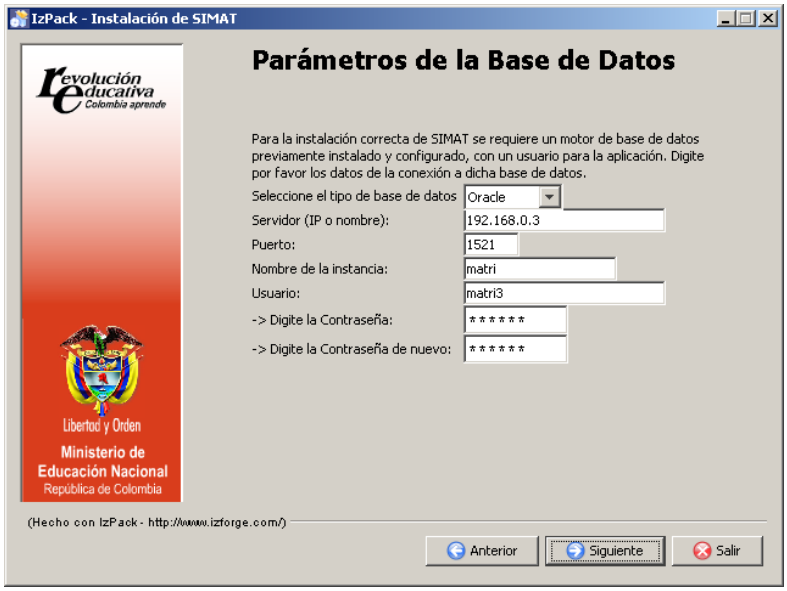

Figura 14. Pantalla de parámetros de la base de datos que usará el SIMAT

Los parámetros en detalle son los siguientes:

Tipo de base de datos	**SIMAT fue desarrollado para soportar Oracle 9i, Postgresql 8 o superior y MS SQLServer 6.5 o superior.**
Servidor (IP o nombre)	Nombre del servidor, o dirección IP donde se accede al servicio del motor de base de datos.
Puerto	Puerto en el servidor donde se encuentra disponible dicho servicio. Los puertos por defecto son 1521 para Oracle, 5432 para Postgresql y 1433 para SQLServer.
Nombre de la instancia	Instancia de la base de datos.
Usuario	Usuario de la base de datos que será *owner* de las tablas de SIMAT.
Contraseña	Palabra clave para autenticación del usuario de base de datos.

Al llenar los datos anteriores, el usuario debe hacer clic en “Siguiente”, lo cual iniciará el proceso de copiado de los archivos necesarios para la instalación:

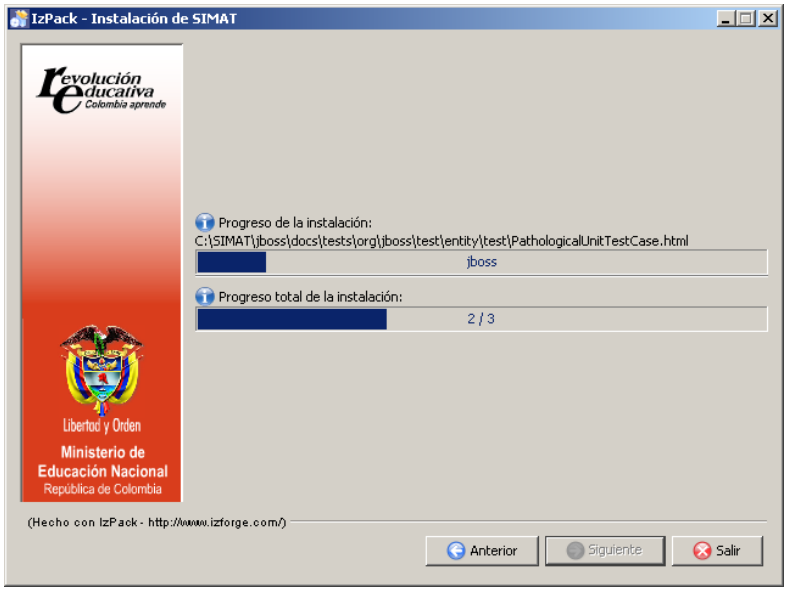

Figura 15. Pantalla de copiado de archivos

Al terminar este paso, el instalador informará que ha terminado de copiar los archivos. Presionar “Siguiente” para continuar con el proceso de instalación.

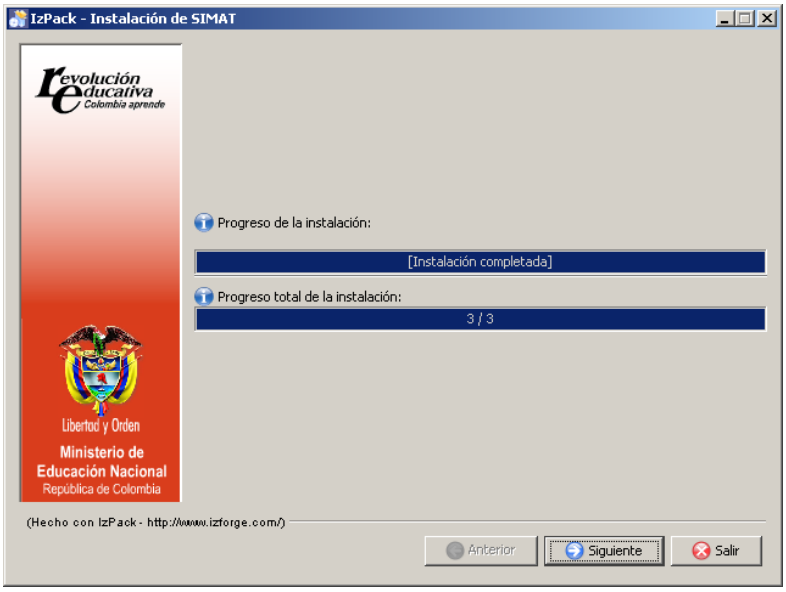

Figura 16. Copia de archivos finalizada

En este punto se ejecutará el proceso de generación de la aplicación SIMAT de acuerdo a los parámetros proporcionados por el usuario. En la consola se puede apreciar con detalle el desarrollo del mismo.

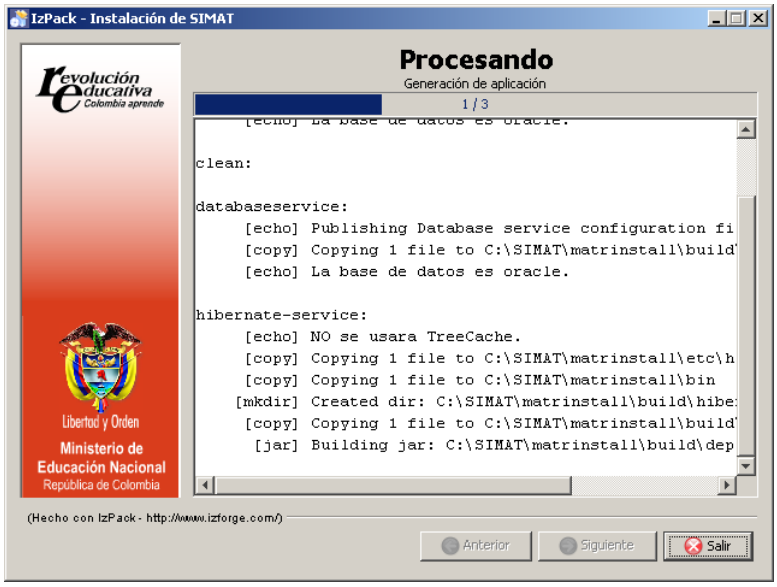

Figura 17. Pantalla del proceso de generación de la aplicación

Cuando termine este proceso, presionar “Siguiente”. Si este proceso presenta algún error, por favor revise lo siguiente:

- Que las rutas del sistema de archivos en la pantalla de parámetros generales (Figura 13) usen el caracter *slash* (/) como separador, aún en sistemas Windows.
- Que el servicio del motor de base de datos a utilizar para el SIMAT este iniciado y disponible.
- Que la información acerca de la base de datos suministrada en la pantalla de la Figura 14 corresponda con los datos reales.

En este punto el proceso habrá terminado satisfactoriamente la instalación.

Figura 19. Instalación Finalizada

9.2.3 Desinstalación

El instalador de SIMAT crea dentro del directorio de la instalación, especificado durante dicho proceso (ver Figura 12), un directorio llamado “Uninstaller”, que contiene un archivo llamado “uninstaller.jar”. Entonces, para desinstalar la aplicación, ejecute en la línea de comando:

```
java -jar uninstaller.jar
```

La siguiente ventana aparecerá en pantalla:

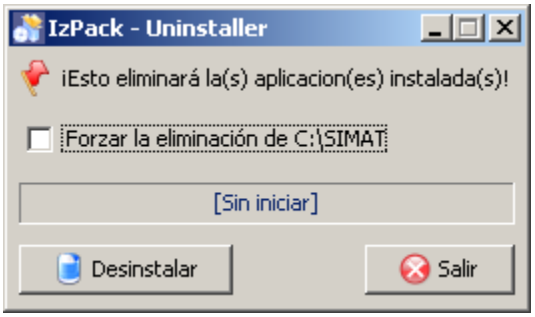

Figura 8. Ventana de desinstalación DE SIMAT

Para llevar a cabo el proceso, simplemente presione “Desinstalar”.

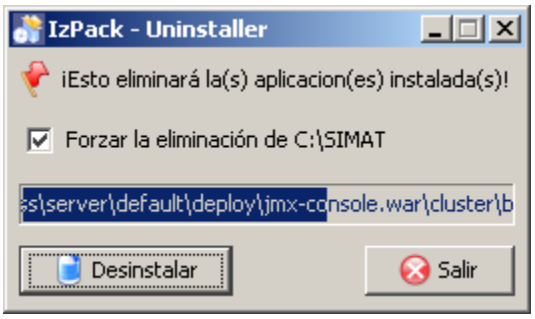

Figura 9. Ejecución de la desinstalación

Para terminar, presione el botón “Salir”. Es posible que al finalizar la desinstalación tenga que borrar manualmente el directorio donde se instaló el SIMAT.

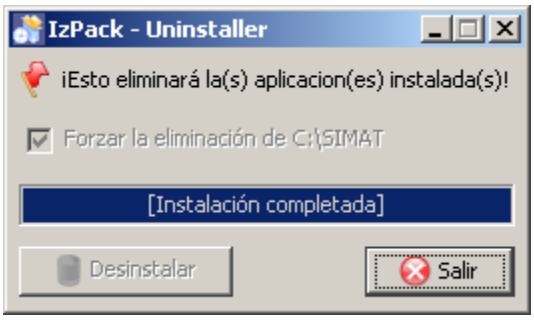

Figura 22. DesIntalación Finalizada

9.3 Construcción del Instalador

Cuando se necesita un instalador con propiedades especiales, o que utilice archivos fuentes actualizados, un usuario administrador con acceso al código fuente del SIMAT puede crear dicho instalador personalizado. Se recomienda sin embargo tener un buen nivel de conocimientos acerca de la estructura del proyecto y sus archivos fuentes, y del archivo “ant.properties” en la raíz del mismo, que son los lugares donde puede darse la personalización del sistema. El proceso se describe en las siguientes subsecciones.

9.3.1 Requerimientos y Consideraciones

Para generar un instalador se necesitan las siguientes herramientas *open source* debidamente instaladas:

- Java 2 Standard Edition, J2SE 1.4.2 o superior: Intérprete de programas Java (http://java.sun.com/j2se/).
- Ant 1.6.2 o superior: Herramienta *open source* para la automatización de procesos de construcción de *software* (http://ant.apache.org/).
- Xdoclet 1.2.2: Librería *open source* para generación de código. (http://xdoclet.sourceforge.net/)
- Tapestry 3.0.3: Juego de herramientas *open source* para el desarrollo de aplicaciones Web (http://jakarta.apache.org/tapestry/).
- Hibernate 2.1.8: Solución *open source* para mapeo objeto-relacional (http://www.hibernate.org/).
- IzPack 3.8 o superior: Herramienta *open source* para la creación de instaladores (http://www.izforge.com/izpack/).

La instalación de estas herramientas consiste simplemente en descomprimir el archivo descargable desde

los sitios Web respectivos, con las siguientes excepciones:

- J2SE cuenta con un instalador que no requiere parámetros especiales para los propósitos de este manual.
- Luego de descomprimir el archivo de Ant, se debe declarar la variable de entorno `ANT_HOME` que apunte al directorio raíz donde Ant quedó instalado. El directorio `bin` de Ant debería quedar configurado en el `PATH` del sistema operativo.

Para este proceso, no es necesario tener un motor de base de datos instalado, pero si para ejecutar el instalador generado.

9.3.2 Proceso de construcción del Instalador

Para construir un instalador de SIMAT es necesario compilar los archivos fuentes de Java, generar los mapeos necesitados por Hibernate y organizar los archivos necesarios para generar la aplicación con los parámetros específicos del usuario en tiempo de ejecución del instalador; este proceso se lleva a cabo utilizando Ant y los archivos "build.xml" y "ant.properties" del directorio del proyecto SIMAT (este último debe ser modificado. Estos archivos, junto con los de JBoss y Ant, necesarios para la instalación, son ubicados en la carpeta `install` del proyecto SIMAT.

El proceso para obtener un instalador de SIMAT es el siguiente:

a. Editar el archivo `ant.properties`. En las primeras líneas, sustituir los valores de las propiedades `ant.home`, `xdoclet.home`, `tapestry.home`, `java.home`, `hibernate.home`, `izpack.home`, colocando las rutas respectivas donde se encuentran instaladas las herramientas respectivas. Tener en cuenta utilizar el caracter *slash* ("/") como separador de rutas, aún en sistemas Windows.

b. En la línea de comando ejecutar:
```
ant deploy-installer
```

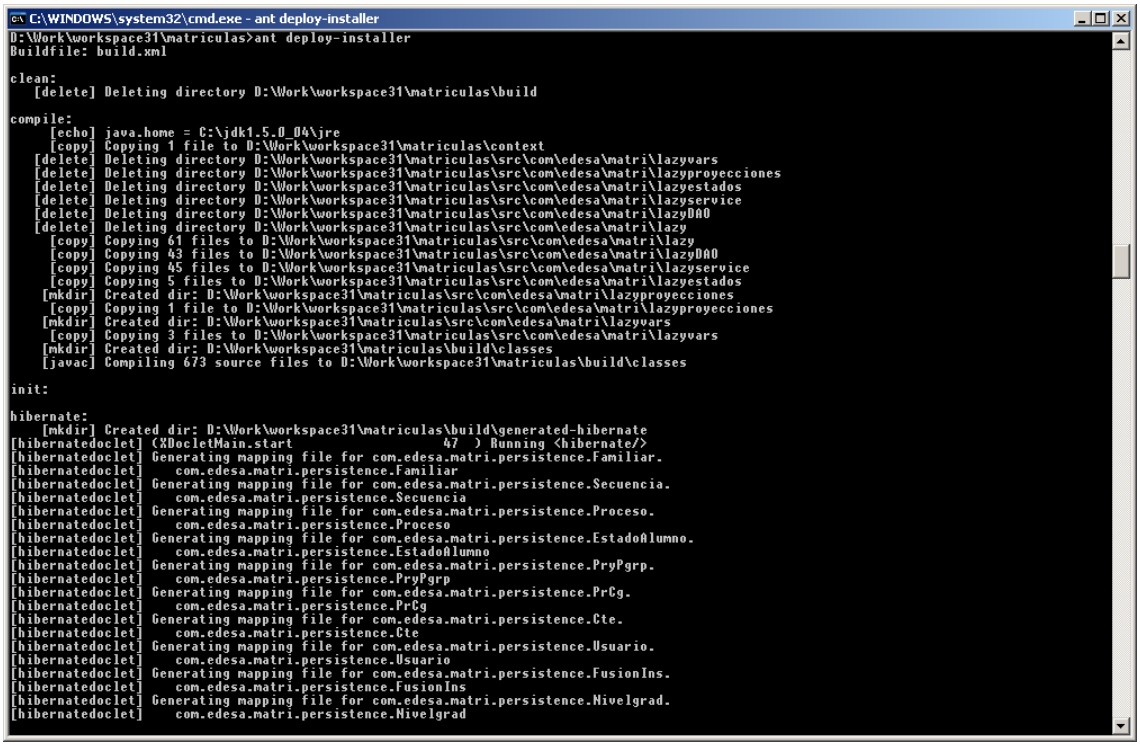

Figura 10. Generación del instalador

c. Si la ejecución de Ant no retorna errores, el instalador generado queda en la ruta `install/<nombre_de_aplicacion>-matri.jar`, donde nombre_de_aplicación corresponde al nombre de la aplicación en `ant.properties` (propiedad `application.name`). El archivo generado es un ejecutable de java (ver sección "**Ejecución del Instalador**").

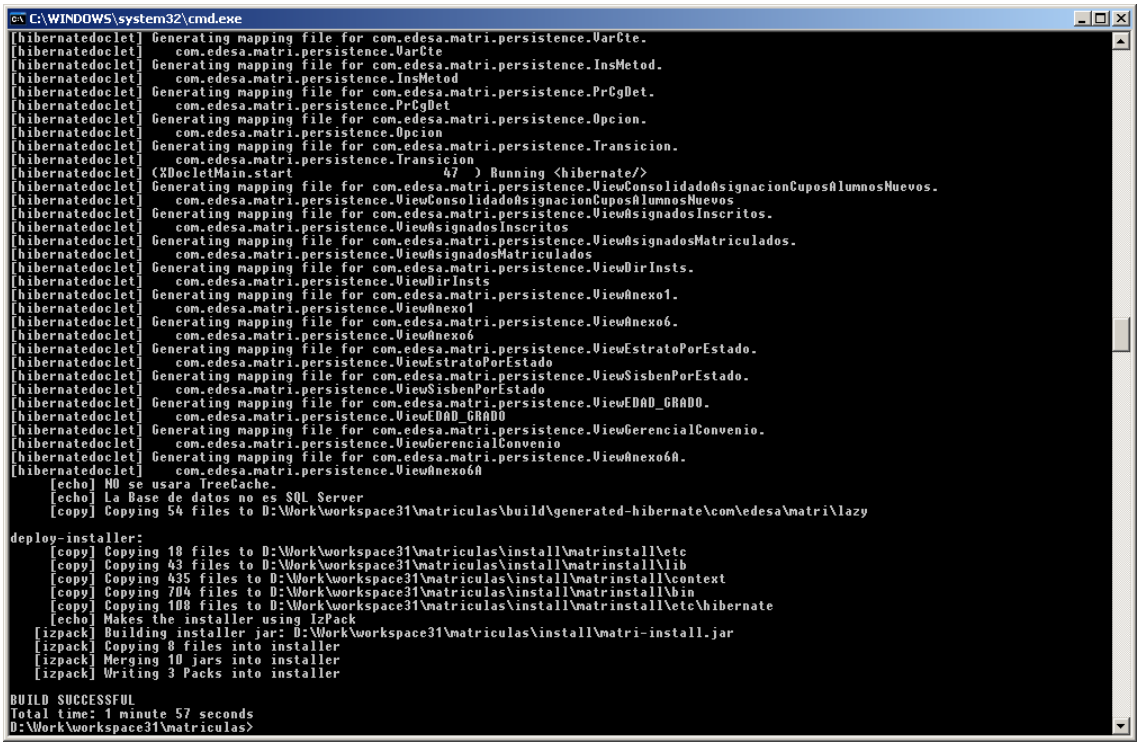

Figura 24. Build Exitoso