UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE QUITO

CARRERA: INGENIERÍA DE SISTEMAS

Tesis previa a la obtención del título de: INGENIERO DE SISTEMAS

TEMA:

ANÁLISIS, DISEÑO, DESARROLLO E INTEGRACIÓN DEL MÓDULO DE GESTIÓN DE DISTRIBUTIVOS QUE TRABAJE DE FORMA INTEGRADA PARA EL SISTEMA UPS SCHEDULE PARA LA UNIVERSIDAD POLITÉCNICA SALESIANA SEDE QUITO

AUTORES:

DENNIS FERNANDO REYES PÉREZ
MILTON AURELIO CHILIQUINGA IZA

DIRECTOR:

RODRIGO EFRAÍN TUFIÑO CÁRDENAS

Quito, mayo 2014

DECLARATORIA DE RESPONSABILIDAD Y AUTORIZACIÓN DE USO DEL TRABAJO DE TITULACIÓN

Nosotros, autorizamos a la Universidad Politécnica Salesiana la publicación total o parcial de este trabajo de titulación y su reproducción sin fines de lucro.

Además declaramos que los conceptos y análisis desarrollados y las conclusiones del presente trabajo son de exclusiva responsabilidad de los autores.
Quito, mayo 2014.

Milton Aurelio Chiliquinga Iza
CI: 050297962-8

Dennis Fernando Reyes Pérez
CI: 171804061-9

ÍNDICE

RESUMEN

La generación de distributivos en la Universidad Politécnica Salesiana, ha presentado un inconveniente sin resolver, por el mismo hecho de realizar los procesos manuales, utilizando recurso humano y sobre todo utilizando una fuerte cantidad de tiempo debido a la falta de tecnología que facilite la correcta realización de los procesos mencionados.

Por este motivo se ha planteado la elaboración de un sistema robusto denominado "UPS Schedule", mismo que por el gran alcance que dispone, ha sido divido en varios módulos, centrándose este documento en el módulo de gestión de distributivos para solventar el inconveniente que presenta la Universidad Politécnica Salesiana, Sede Quito.

El módulo mencionado anteriormente busca ofrecer una solución sencilla y efectiva al mismo tiempo, utilizando tecnología actual que permita al usuario una interacción ágil e intuitiva con el sistema, proporcionando una interfaz amigable al usuario para cumplir con el objetivo principal de gestionar de forma óptima los distributivos generados semestralmente.

ABSTRACT

The distributive generation in Salesian Polytechnic University has presented a problem unresolved by the very fact of manual processes, using human resources and above all using a large amount of time because of the lack of technology to facilitate the successful completion of the processes mentioned.

For this reason it has been suggested development of robust system called “UPS Schedule”, which by the powerful features, has been divided into several modules, focusing this document in the management distributive module to solve the drawback of the Salesian Polytechnic University.

The above mentioned module seeks to provide a simple and effective solution at the same time, using current technology that allows the user a flexible and intuitive interaction with the system, providing a user-friendly interface to meet the main objective of optimally manage distributional generated semiannually.

INTRODUCCIÓN

En el siguiente apartado, se presenta la razón por la cual se ha decidido la implementación del tema de distributivos de la Universidad Politécnica Salesiana, Sede Quito, así como también el problema actual con el desarrollo de ésta actividad tan importante dentro de la institución, además se detalla la metodología utilizada para la investigación y desarrollo.

El primer capítulo: **Análisis y diseño**, analiza todos los requerimientos recogidos a lo largo de la investigación, presenta el diseñado actual del sistema, analizando el alcance y proyección de sus objetos, así como también el diseño de la interfaz de usuario para cumplir con el propósito del proyecto, se detallan todos los requerimientos funcionales y no funcionales, presentando las correspondientes interfaces de hardware y software, interfaces de comunicación, de la misma manera muestra los requisitos de rendimiento y seguridad.

El segundo capítulo: **Desarrollo e integración**, describe todas las herramientas de implementación tanto para la programación, diseño y administración que se han utilizado en la etapa de construcción, realizando al mismo tiempo un proceso de análisis y presentación de las posibles ventajas y desventajas de la misma, para finalizar el capítulo, se presenta la integración en donde se detalla la manera en la que fue implementado y unificado los módulos de gestión de distributivos y de horarios del sistema UPS Schedule.

Por último, en el tercer capítulo Pruebas, presenta las pruebas respectivas de carga, usabilidad y las pruebas funcionales para la verificación del correcto funcionamiento del proyecto desarrollado.

Planteamiento del problema

El principal inconveniente que se ha detectado en la Universidad Politécnica Salesiana, Sede Quito, es que durante las últimas etapas de cada semestre, los Directores de Carrera, poseen una carga de trabajo redundante y al mismo tiempo

complejo al realizar los distributivos de cada carrera tanto de docentes como de los grupos.

La actividad de gestionar docentes, grupos y distributivos se ha convertido en una tarea tediosa debido a varios factores como son:

- ✓ Incremento del número de carreras
- ✓ Incremento de cursos ofertados
- ✓ Modificación y acoplamientos de mallas curriculares
- ✓ Número de grupos
- ✓ Números de docentes y su disponibilidad

Un factor primordial que se debe tomar en cuenta en el detalle de este inconveniente, es que todo el procedimiento se lo realiza manualmente. Al momento cada Director de Carrera debe obtener información para realizar esta tarea ya sea por mail o por otros medios de comunicación para poder cumplir con el proceso, en el mejor de los casos ayudado por hojas de cálculo y en el peor de los casos a través de hojas de papel.

Objetivo general

Obtener un módulo funcional para la gestión de distributivos que trabaje de forma integrada para el sistema UPS Schedule de la Universidad Politécnica Salesiana, Sede Quito.

Objetivos específicos

- ✓ Analizar el proceso del módulo de distributivos
- ✓ Definir los requerimientos del módulo de distributivos
- ✓ Establecer interfaces de comunicación con los otros módulos del sistema
- ✓ Diseñar el módulo en base a la arquitectura base del sistema
- ✓ Programar el módulo utilizando software libre
- ✓ Gestionar docentes, grupos y distributivos
- ✓ Imprimir y exportar los distributivos, grupos y docentes de la carrera

- ✓ Probar el funcionamiento del módulo y la correcta interacción con el resto del sistema
- ✓ Integrar el módulo al sistema UPS Schedule

Justificación del proyecto

La Universidad Politécnica Salesiana, al momento no posee un sistema capaz de gestionar tanto los docentes, grupos y distributivos que se generan cada semestre, motivo por el cual se va a desarrollar un módulo denominado "gestión de distributivos".

El módulo en mención proporcionará al sistema UPS Schedule una funcionalidad que beneficiará principalmente a cada Director de Carrera con sus tareas, pero con un sistema amigable, entendible y fácil de manipular para que pueda cumplir con sus procesos de manera sencilla y eficiente.

Para ofrecer a cada Director de Carrera un sistema fácil de manipular, se ha decidido desarrollar la aplicación con tecnologías de punta, en este caso se utilizará JSF (JavaServer Faces) cuyo framework de aplicaciones facilita el desarrollo de interfaces, proporcionando tanto al desarrollador como al cliente obtener interfaces amigables para el usuario, con la finalidad que éste pueda interactuar con la aplicación de forma eficiente.

Alcance del proyecto

El módulo de "gestión de distributivos", será el encargado de administrar tanto los docentes, grupos y distributivos de todas las carreras que posea la Universidad Politécnica Salesiana, Sede Quito, con el objetivo principal de ofrecer a cada Director de Carrera la administración de estos procesos, ofreciendo al usuario un panel gráfico, en el cual podrá seleccionar los docentes que trabajará en su carrera en el caso de la gestión de docentes, mientras que para los grupos poseerá un panel amigable para la respectiva creación y modificación de grupos.

Por otro lado en la gestión de distributivos, el Director de Carrera tendrá la capacidad de crear nuevos distributivos o de ser el caso, basarse en anteriores períodos para la correspondiente modificación.

CAPÍTULO 1

ANÁLISIS Y DISEÑO

El siguiente capítulo presenta el respectivo análisis al proyecto para verificar que su desarrollo sea exitoso, abarcando los requerimientos recolectados y a través de los mismos diseñar sus respectivas interfaces.

1.1 Análisis de viabilidad

A continuación se detalla el análisis de tres aspectos importantes dentro del desarrollo de un proyecto de software, los cuales son: viabilidad operativa, técnica y económica.
Cabe destacar que el sistema desarrollado no dispone de soluciones similares en el mercado.

1.1.1 Viabilidad operativa.

Desde el punto de vista operacional el sistema de gestión de distributivos se convertirá en una herramienta necesaria para los Directores de Carrera, con la existencia de los recursos tecnológicos necesarios y la correcta selección de la plataforma de desarrollo, ofrecerá facilidad de uso para cumplir con los procesos manuales existentes.

1.1.2 Viabilidad técnica.

- **Recursos humanos**

El análisis, diseño e implementación del sistema, será realizado por los integrantes de la tesis, en el cual el coordinador del proyecto es el tutor de tesis, encargado de supervisar el avance en cada fase de desarrollo.

Para el desarrollo del sistema será necesario de analistas y diseñadores, estos roles serán desempeñados por los tesistas.

La programación y correspondientes pruebas del módulo de distributivos serán desarrolladas por los tesistas.

- **Hardware**

La Universidad Politécnica Salesiana, Campus Sur, dispone con la infraestructura necesaria y equipos de última generación, en los cuales se podrá implementar el módulo de distributivos ofreciendo las debidas garantías necesarias de funcionalidad.

El hardware que se tiene a disposición para el proyecto tiene las siguientes características:

1 Gabinete para Servidores Blades, con los siguientes componentes:

- ✓ 1 Enclosure Blade c3000, con conexión eléctrica bifásica
- ✓ Switches de Lan 1GB E2 para Enclosure c3000
- ✓ Sistema de ventiladores (4 instaladas) y fuentes redundantes (2 instaladas)
- ✓ 1 Módulo On Board Administración

Servidores para aplicaciones:

Servidor BL460C con el siguiente detalle:

- ✓ 1 Procesador Intel Xeon QuadCore, 2.33 GHz 1x4MB Cache
- ✓ 4GB RAM
- ✓ 1 Smart Array RAID 1
- ✓ 2 Discos de 146GB 10k SAS HDD

Por consiguiente el módulo de distributivos se encontrará disponible en equipos de alto rendimiento operando 24/7, garantizando disponibilidad de información en todo momento.

1.1.3 Viabilidad económica.

Los gastos que tendrá que solventar la universidad no son excesivos, debido a que cuenta con una infraestructura informática adecuado para alojar el sistema, cabe mencionar que el proyecto será desarrollado por los tesistas por lo que no será considerado como un gasto adicional para la institución.

Con ello, el beneficio que dispondrá la universidad con la implementación del sistema son los siguientes:

- ✓ Mejor calidad en la prestación de servicios hacia los estudiantes
- ✓ Optimización de tiempo y recursos en el proceso de la generación de los distributivos
- ✓ Automatización de procesos

Tabla 1. Costos de implementación

Concepto	Valor	Responsable
Software • Eclipse Kepler • PrimeFaces • PosgreSQL • SQL Power Architect • Libre Office	 $0 $0 $0 $0 $0	✓ Milton Chiliquinga ✓ Dennis Reyes
Hardware • 2 Laptops	 $ 3000	✓ Milton Chiliquinga ✓ Dennis Reyes
Recursos humanos • Desarrolladores	 $ 2000	✓ Milton Chiliquinga. ✓ Dennis Reyes
Gastos indirectos • Alimentación • Transporte • Acceso Internet • Derecho de certificación • Matrícula en denuncia de tesis • Derecho de certificación de mención	 $ 600 $ 750 $ 350 $ 40 $ 400 $ 285	✓ Milton Chiliquinga. ✓ Dennis Reyes
TOTAL	$ 7425	

Elaborado por: Dennis Reyes y Milton Chiliquinga.

Tabla 2. Análisis de utilidad neta

Años de proyección de vida útil del proyecto	**1**	
UPDATE	0	
Ahorros de costos de inversión	7425	
Gastos de depreciación	0	
Utilidad antes de impuesto		7425
Impuesto del 36,25%	2691.56	
Utilidad neta	**4733.44**	

Años de proyección de vida útil del proyecto	**2**	
UPDATE	0	
Ahorros de costos de inversión	4733.44	
Gastos de depreciación	0	
Utilidad antes de impuesto		4733.44
Impuesto del 36,25%	1715.87	
Utilidad neta	**3017.57**	

Años de proyección de vida útil del proyecto	**3**	
UPDATE	0	
Ahorros de costos de inversión	3017.57	
Gastos de depreciación	0	
Utilidad antes de impuesto		3017.57
Impuesto del 36,25%	1093.87	
Utilidad neta	**1923.70**	

Años de proyección de vida útil del proyecto	**4**	
UPDATE	0	
Ahorros de costos de inversión	1923.70	
Gastos de depreciación	0	
Utilidad antes de impuesto		1923.70
Impuesto del 36,25%	697.3407	
Utilidad neta	**1226.36**	

Años de proyección de vida útil del proyecto	**5**	
UPDATE	0	
Ahorros de costos de inversión	1226.36	
Gastos de depreciación	0	
Utilidad antes de impuesto		1226.36
Impuesto del 36,25%	444.55	
Utilidad neta	**781.80**	

Elaborado por: Dennis Reyes y Milton Chiliquinga.

1.1.3.1 Análisis de factibilidad TIR / VAN.

VAN: Es la rentabilidad mínima pretendida por la persona que ha invertido en el proyecto, por de debajo de la cual estará dispuesto a efectuar su inversión.

TIR: Es la tasa de interés que hace que el VAN del proyecto sea igual a cero, es un criterio de rentabilidad, se la obtiene mediante la siguiente fórmula.

$$\textbf{TIR}= \sum_{T=1}^{n}[FCt/(1+i)^t] - Io$$

Dónde:

Io= Inversión inicial.

FC= Flujo de caja del proyecto

i= Tasa de descuento

t= Tiempo

n= Vida útil del proyecto

Tabla 3. Desarrollo TIR/VAN

Años de vida util del proyecto	1	2	3	4	5
UPDATE	0	0	0	0	0
Ahorros de costos de inversión	7425	4733.44	3017.57	1093.87	1226.36
Gastos de depreciación	0	0	0	0	0
Utilidad antes de impuesto	7425	4733.44	3017.57	1093.87	1226.36
Impuesto del 36,25%	2691.56	1715.87	1093.87	697.34	444.5
Utilidad neta	4733.44	3017.57	1923.7	1226.36	781.8
VAN	$2,391.88				
TIR	37%				
C/B	(1.02)				

Elaborado por: Dennis Reyes y Milton Chiliquinga.

En base a los parámetros de aceptación de un proyecto se puede concluir:

- ✓ **VAN=** 2,391.88: Indicador de aceptabilidad del proyecto.
- ✓ **TIR=** 37%: Cabe mencionar el valor de la TIR debe encontrarse en el rango de interpolación de tasas de actualización para que el proyecto sea factible.

- ✓ **C/B=** 1.02: El costo/beneficio es mayor a 1, por tanto el proyecto se considera rentable.

1.2 Requerimientos

El propósito de esta sección es describir en forma detallada los requerimientos de software de un sistema generador de distributivos para la Universidad Politécnica Salesiana en su Sede Quito.

En los siguientes apartados se mostrará las características del módulo a desarrollar en el sistema (gestión de distributivos), los perfiles de usuario, sus interfaces y principalmente se especificará en detalle su funcionalidad y las operaciones que realizará.

Este documento está enfocado al personal técnico que se encargará de diseñar, desarrollar y mantener el sistema propuesto.

Abreviaturas

- ✓ **UPS**

  Abreviatura de Universidad Politécnica Salesiana.

- ✓ **Campus universitario**

  Es el conjunto de terrenos y edificios que pertenecen a una universidad y en el cual funcionan las carreras.

- ✓ **Distributivo**

  Es la distribución de materias a los docentes que realiza el Director de Carrera antes de iniciar un nuevo período académico.

- ✓ **Carrera**

  Hace referencia a una opción de titulación que la universidad brinda a sus estudiantes, por ejemplo Ingeniería de Sistemas, Administración de Empresas, etc.

- ✓ **Nivel**

  Cada una de las carreras dentro de la UPS tiene dividida su malla curricular dentro de niveles los mismos que por lo general tienen una duración de un semestre y sirven para separar de manera estructurada a la carrera según el nivel de conocimiento que el estudiante va adquiriendo con el transcurso del tiempo.

- ✓ **Grupo**

  Es una forma de organización del alumnado (paralelo) que se encuentra cursando el mismo nivel, sirve principalmente para dividir en números manejables de estudiantes.

- ✓ **Materia**

  Dentro de cada uno de los niveles los estudiantes tiene que tomar determinadas materias, las mismas que consisten en el estudio de determinada área del conocimiento.

Este documento consta de tres secciones de acuerdo a la norma IEEE 830.

En primera instancia dispone de una sección de introducción al documento y proporciona una visión general de la especificación de requerimientos del software desarrollado.

El segundo apartado es una descripción general de la aplicación para conocer las principales funciones que debe realizar la misma.

Para culminar con el documento, la tercera instancia se define detalladamente los requerimientos que debe satisfacer la aplicación.

1.2.1 Perspectiva.

El software desarrollado tiene como perspectiva principal de ser de carácter privado, restringiendo el uso únicamente para los Directores de Carrera, los cuales dispondrán la información contenida en este aplicativo.

La meta principal del software es brindar un apoyo considerable hacia los Directores de Carrera en la instancia del desarrollo de los distributivos para la Universidad Politécnica Salesiana, Sede Quito.

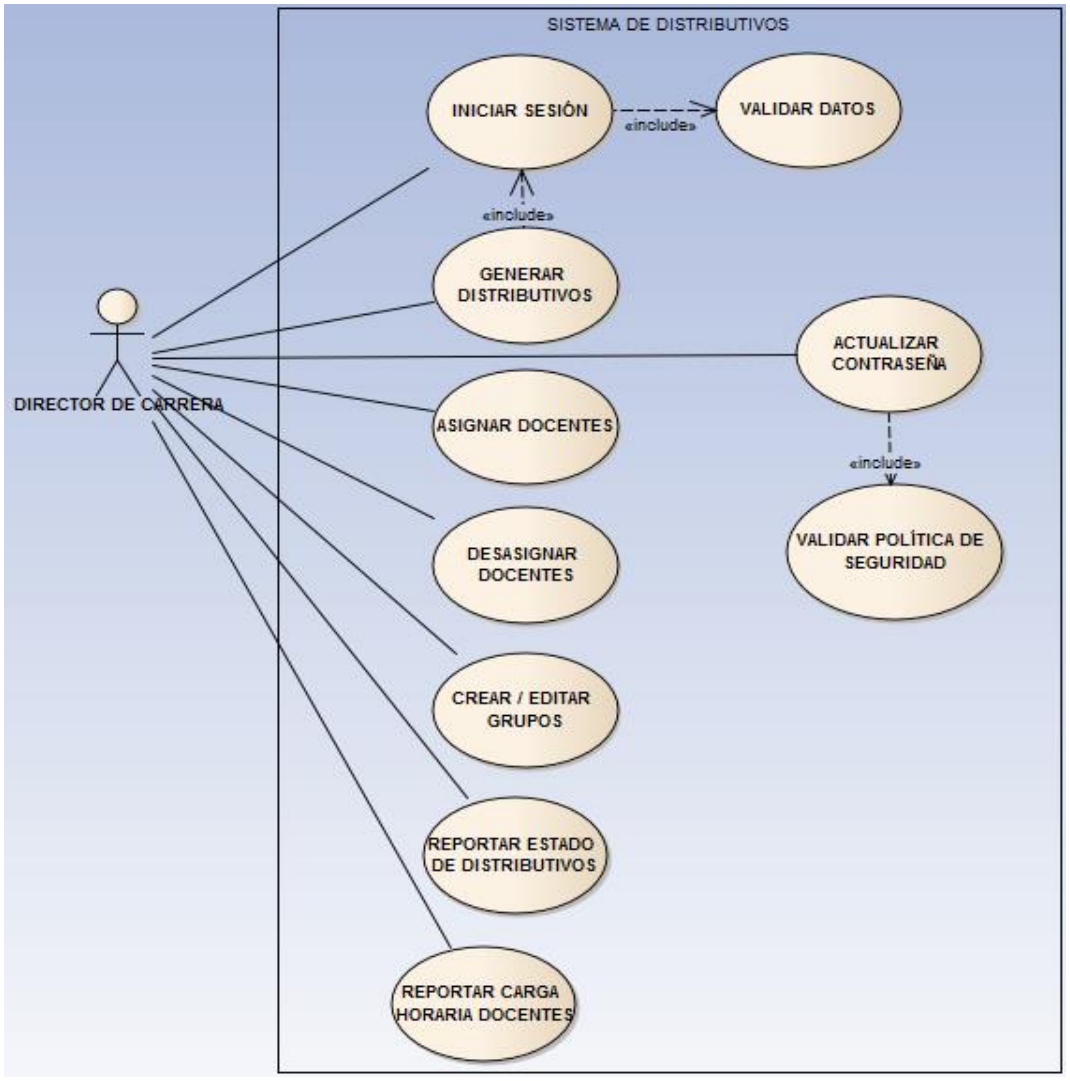


Figura 1. Diagrama de casos de uso, sistema de distributivos.

Elaborado por: Dennis Reyes y Milton Chiliquinga.

1.2.2 Funcionalidades.

- ✓ **Asignación de docentes**

Mediante esta funcionalidad cada Director de Carrera podrá observar a cada uno de sus colaboradores (docentes).

Este proceso se realizará a través de un filtrado por la cédula de identidad o por el nombre del docente, una vez realizado este proceso, dispondrá la opción de asignar al colaborador para la carrera.
Cabe mencionar que si el Director de Carrera no encontrara en primera instancia, el sistema dará la alerta de que posiblemente el docente ya este asignado a una carrera.

- ✓ **Gestión de grupos**

El Director de Carrera tendrá la capacidad de crear y modificar todos los grupos existentes de la Universidad Politécnica Salesiana, Sede Quito, según fuera el caso.

El Director de Carrera tendrá las siguientes funcionalidades:

- ✓ Crear un nuevo grupo
- ✓ Modificar un grupo existente
- ✓ Eliminar grupo existente

- ✓ **Gestión de distributivos**

Mediante esta funcionalidad cada Director de Carrera tendrá a su disposición la posibilidad de crear nuevos distributivos para el período correspondiente.

Para este proceso se creará un diseño muy amigable, sencillo y eficiente, en el cual el usuario logeado visualizará todas las materias a su cargo y tendrá la capacidad asignar una materia a un docente.
Cabe mencionar que el distributivo es uno de los insumos más importantes para generar los horarios.

- ✓ **Reporte**

Este proceso imprime y exporta en formato PDF las asignaciones tanto de docentes, grupos y de distributivos de cada una de las carreras de la Universidad Politécnica Salesiana, Sede Quito.

Los reportes disponibles son los siguientes:

- ✓ Asignaciones
- ✓ Carga de horarios

1.2.3 Características de usuarios.

El software fue desarrollado enfocándose únicamente para aquellos usuarios cuyo perfil sea el de Director de Carrera, quienes serán responsables de la correcta gestión de distributivos y administración de la información correspondiente a materias, docentes, grupos y reportes

1.2.4 Restricciones.

El sistema debe ser instalado en los servidores de la Universidad Politécnica Salesiana, utilizar un gestor de base de datos, un servidor de aplicaciones y su acceso deberá ser a través de la intranet institucional.

1.2.5 Suposiciones y dependencias.

El documento de requerimientos se basa en la estructura actual de las carreras, el distributivo generado por los Directores de Carrera, la disponibilidad horaria de los docentes y la hora de clases (definida en 60 minutos). Si alguno de estos factores cambia, se deberá revisar los requerimientos planteados.

1.2.6 Requisitos futuros.

Se podría pensar en una integración con el sistema actual de gestión académica de la Universidad, Sistema Nacional Académico (SNA), para que la información generada por el sistema propuesto se ingrese automáticamente.

1.2.7 Requerimientos funcionales.

Para definir los requerimientos funcionales se ha utilizado la plantilla organizada por clase de usuarios (Especificación de requerimientos de software IEEE Std 830-1998).

Tabla 4. Requisito funcional 1: Gestión de grupos

Descripción	**Agregar, modificar y eliminar grupos para cada nivel de la carrera en un período académico determinado**
Precondición	Carrera, niveles y período existentes en el sistema
Entrada	Datos del grupo (número, nivel asignado)
Proceso	Datos modificados en el sistema
Salida	Mensaje de aceptación / error

Elaborado por: Dennis Reyes y Milton Chiliquinga.

Tabla 5. Requisito funcional 2: Gestión de distributivos

Descripción	**Agregar, modificar y eliminar un distributivo dentro de un período académico**
Precondición	Materias, docentes y período académico existente en el sistema
Entrada	Datos del distributivo (docente y materia asignada)
Proceso	Datos modificados en el sistema
Salida	Mensaje de aceptación / error

Elaborado por: Dennis Reyes y Milton Chiliquinga.

Tabla 6. Requisito funcional 3: Reportes

Descripción	**Generación de reportes de distributivos**
Precondición	Distributivos generados en el sistema
Entrada	Datos del distributivo (docente y materia asignada)
Proceso	Datos recuperados en el sistema
Salida	Reporte de distributivos / error

Elaborado por: Dennis Reyes y Milton Chiliquinga.

Tabla 7. Requisito funcional 4: Autenticación de usuario

Descripción	**Validar el ingreso de usuarios registrados al aplicativo**
Precondición	Docentes y carreras existentes en el sistema
Entrada	Datos del usuario (username y password)
Proceso	Datos recuperados en el sistema
Salida	Ingreso al sistema de distributivos / error de inicio de sesión.

Elaborado por: Dennis Reyes y Milton Chiliquinga.

Tabla 8. Requisito funcional 5: Asignación de docentes

Descripción	**Asignar docentes a una carrera específica**
Precondición	Usuarios registrados en el sistema
Entrada	Datos del docente (nombres completos y cédula)
Proceso	Datos modificados en el sistema
Salida	Mensaje de aceptación / error

Elaborado por: Dennis Reyes y Milton Chiliquinga.

1.3 Diseño

El siguiente apartado detalla las funciones más destacadas que el usuario (Director de Carrera) tendrá a su completa disposición dentro del sistema, tomando en cuenta la metodología OOHDM se presentará el respectivo diseño conceptual, diseño navegacional y las interfaces abstractas.

1.3.1 Diseño conceptual.

El siguiente diagrama conceptual consta las clases más importantes para el correcto funcionamiento de los distributivos generados por los Directores de Carrera.

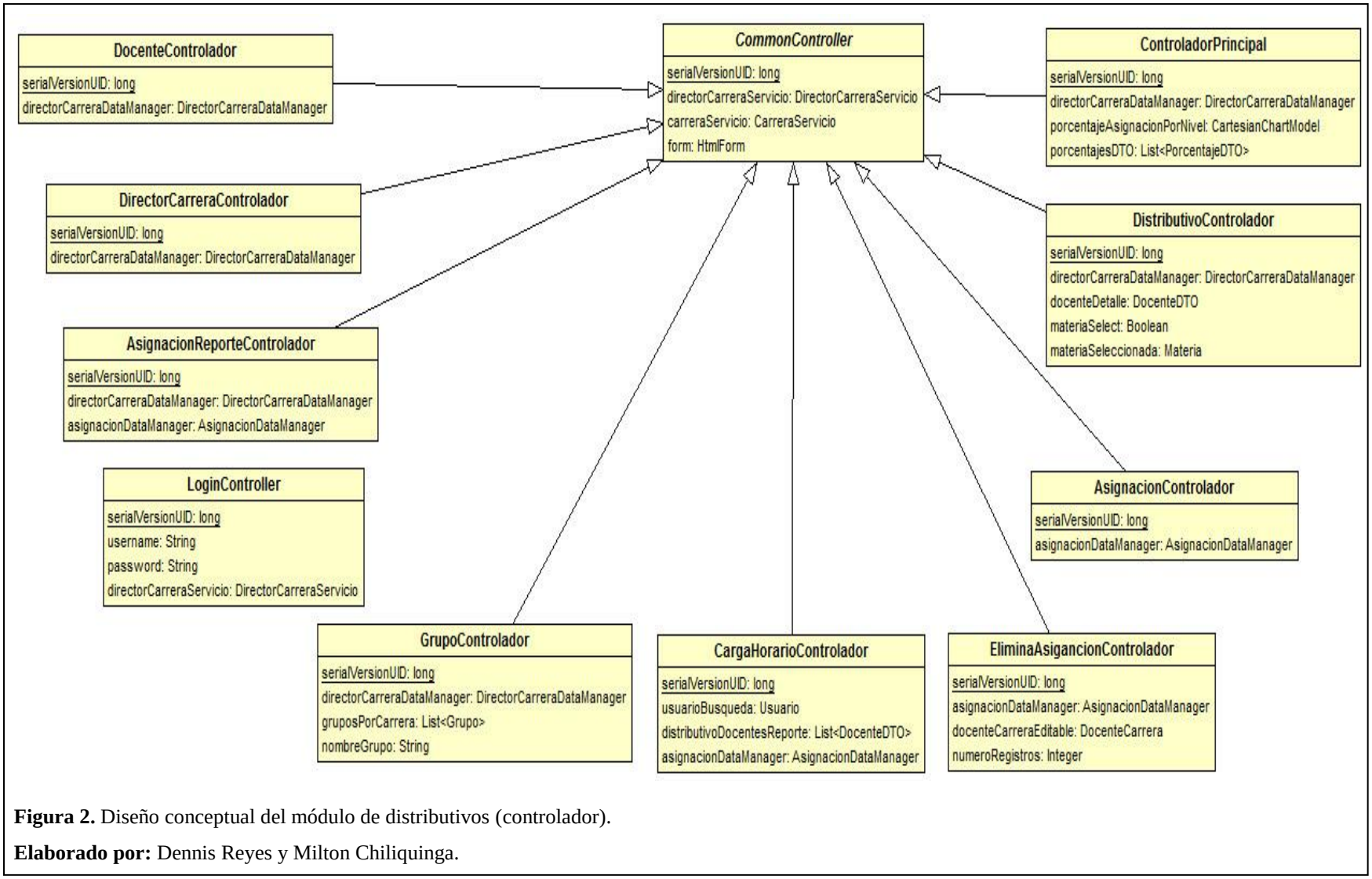


Figura 2. Diseño conceptual del módulo de distributivos (controlador).

Elaborado por: Dennis Reyes y Milton Chiliquinga.

1.3.2 Diseño navegacional.

A continuación, el diagrama navegacional nivel administrador correspondiente a los Directores de Carrera para mayor facilidad y mejor administración del sistema de distributivos.

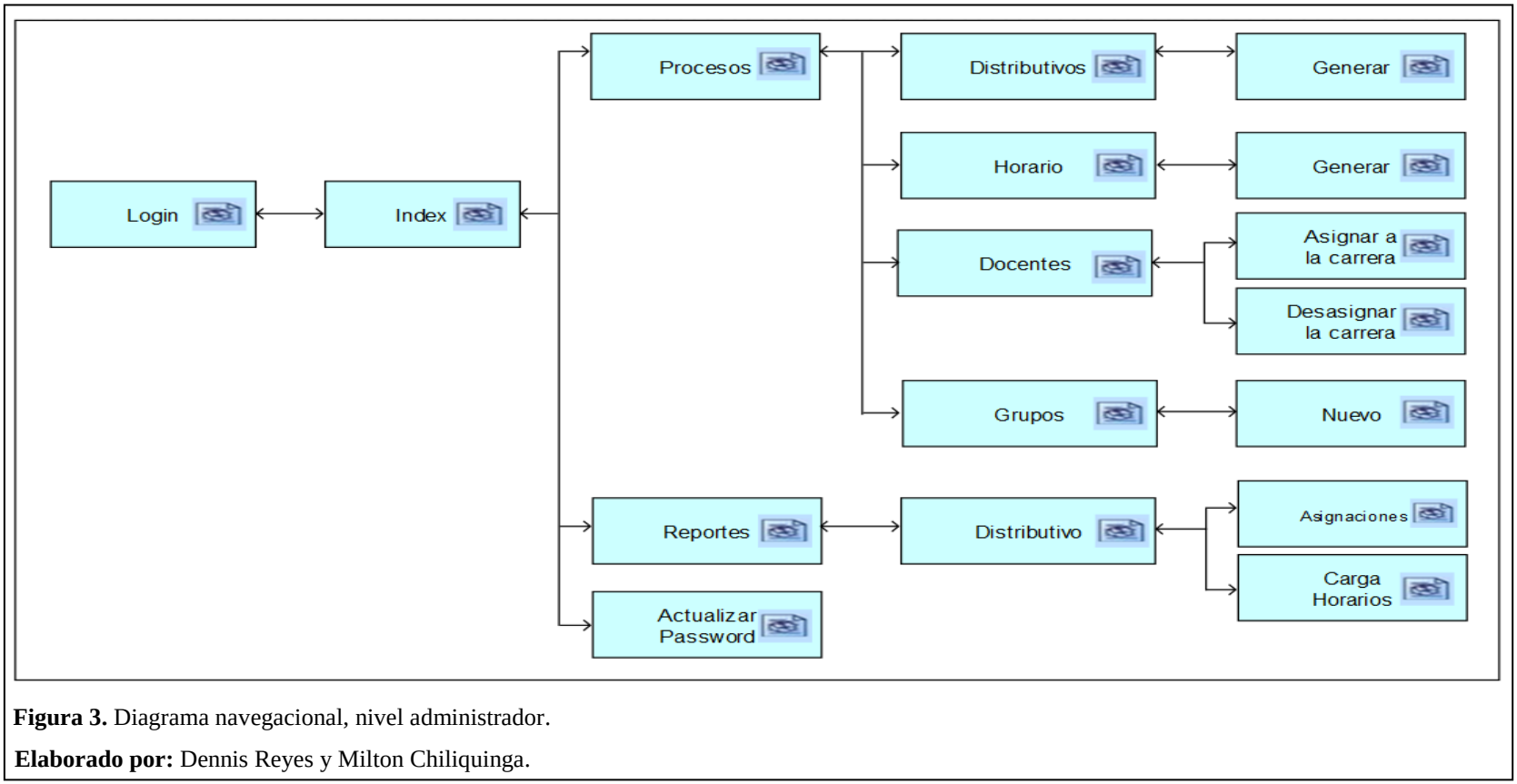


Figura 3. Diagrama navegacional, nivel administrador.

Elaborado por: Dennis Reyes y Milton Chiliquinga.

La anterior ilustración, muestra el diseño navegacional para el usuario, en la cual se puede observar la forma correcta que dispone dicho usuario para dirigirse de una página web hacia otra, optimizando de esta manera la gestión y correcta manipulación del módulo de distributivos.

Se divide en tres subprocesos, en "Reportes", donde el usuario tendrá la posibilidad de generar informes de las asignaciones y carga de horarios según lo requiera, en "Procesos", podrá efectivizar el distributivo como tal y en "Actualizar Password" por motivos de seguridad de ingreso al aplicativo.

Como se puede apreciar en la figura 3, el diseño de las páginas, es fácil de manipular y muy intuitivo al momento de navegar a través del sistema.

1.3.3 Interfaces abstractas.

Para el diseño de las interfaces de usuario se ha considerado ciertos parámetros importantes como son facilidad de uso y seguridad para la información que se está manipulando dentro del sistema de distributivos.

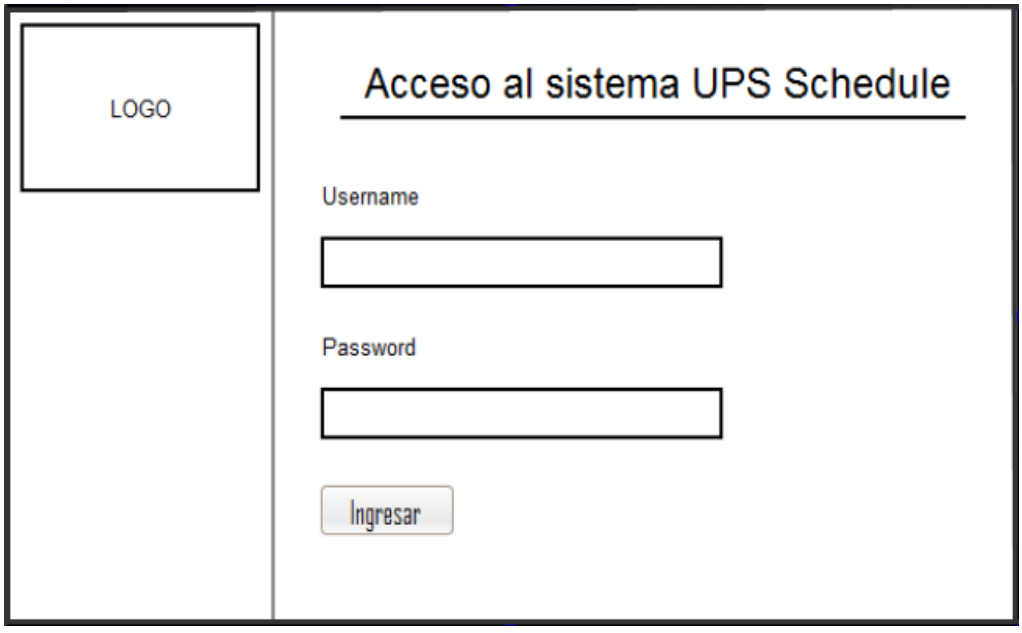


Figura 4. ADV: Login.
Elaborado por: Dennis Reyes y Milton Chiliquinga.

La página principal se encuentra divida en secciones muy bien diferenciadas, misma que será utilizada con el mismo estilo para los demás despliegues de información que dispone el aplicativo.

El banner consta del logo de la institución en este caso de la Universidad Politécnica Salesiana, incluyendo un espacio para la visualización del usuario que ingresará al sistema.

En la zona izquierda de la pantalla se encuentra el menú del aplicativo mismo que permanecerá visible durante toda la sesión del usuario registrado para facilitar la navegabilidad dentro del mismo.
La parte central de la pantalla es donde se puede visualizar todas las acciones que realice el usuario dentro del menú principal, generando facilidad de uso dentro de una misma ventana

Figura 5. ADV: Página principal.
Elaborado por: Dennis Reyes y Milton Chiliquinga.

Dentro del menú "Procesos", el usuario dispone la funcionalidad de distributivos, misma que posee varios filtros como son la selección de una malla y nivel, con el objetivo de evitar posibles errores al momento de generar los distributivos para una correspondiente carrera.

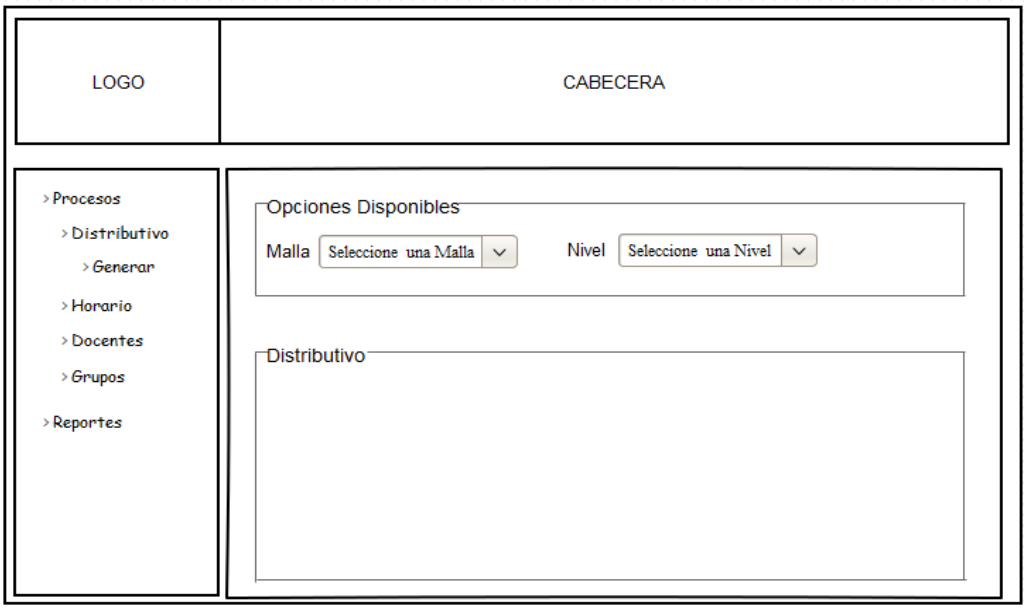


Figura 6. ADV: Distributivos.

Elaborado por: Dennis Reyes y Milton Chiliquinga.

Para la asignación de un docente a una carrera particular, el aplicativo ofrece la funcionalidad de buscar ya sea por cédula o por nombre a un colaborador en específico y un botón para su correspondiente asignación.

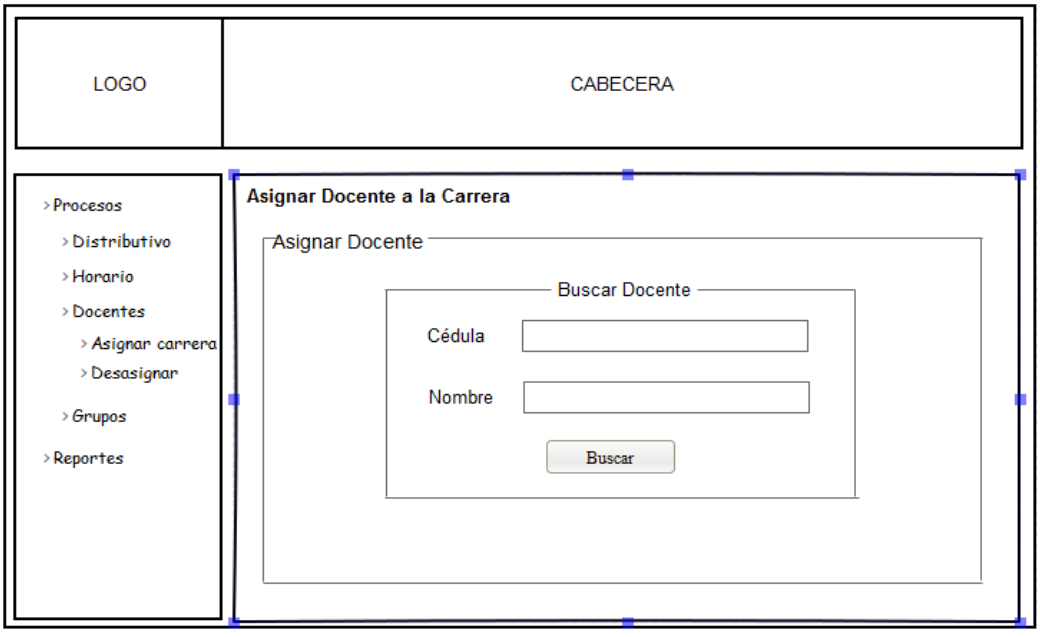


Figura 7. ADV: Buscar docente.

Elaborado por: Dennis Reyes y Milton Chiliquinga.

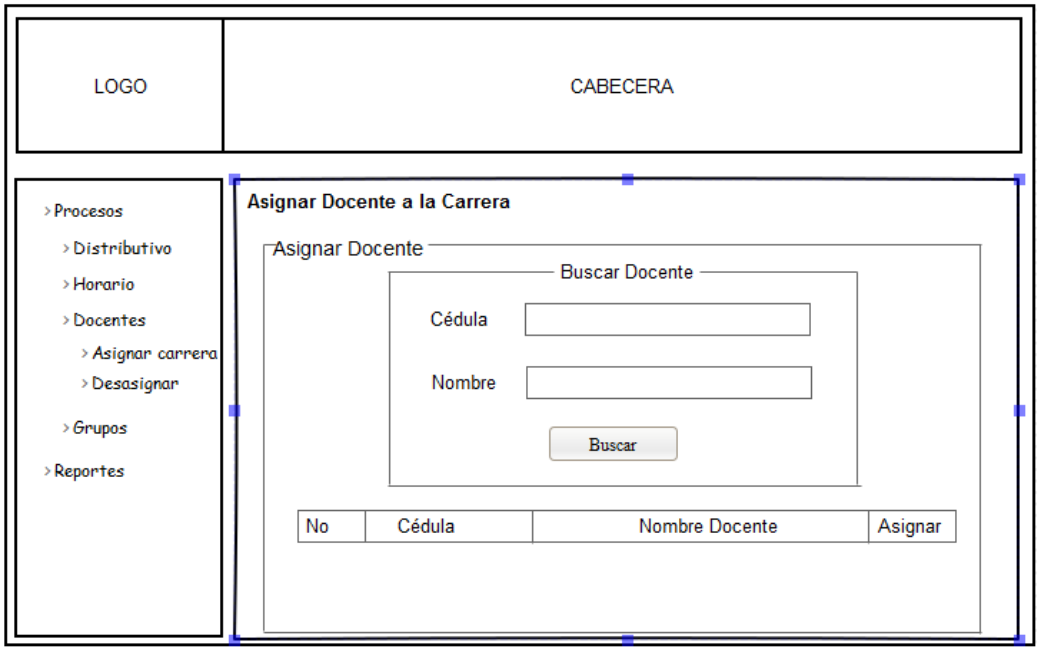


Figura 8. ADV: Asignar docente.

Elaborado por: Dennis Reyes y Milton Chiliquinga.

En el caso de desvincular un docente de una carrera, el sistema facilita este proceso, con tan solo buscar el docente y pinchar en el botón para eliminar la relación entre los mismos.

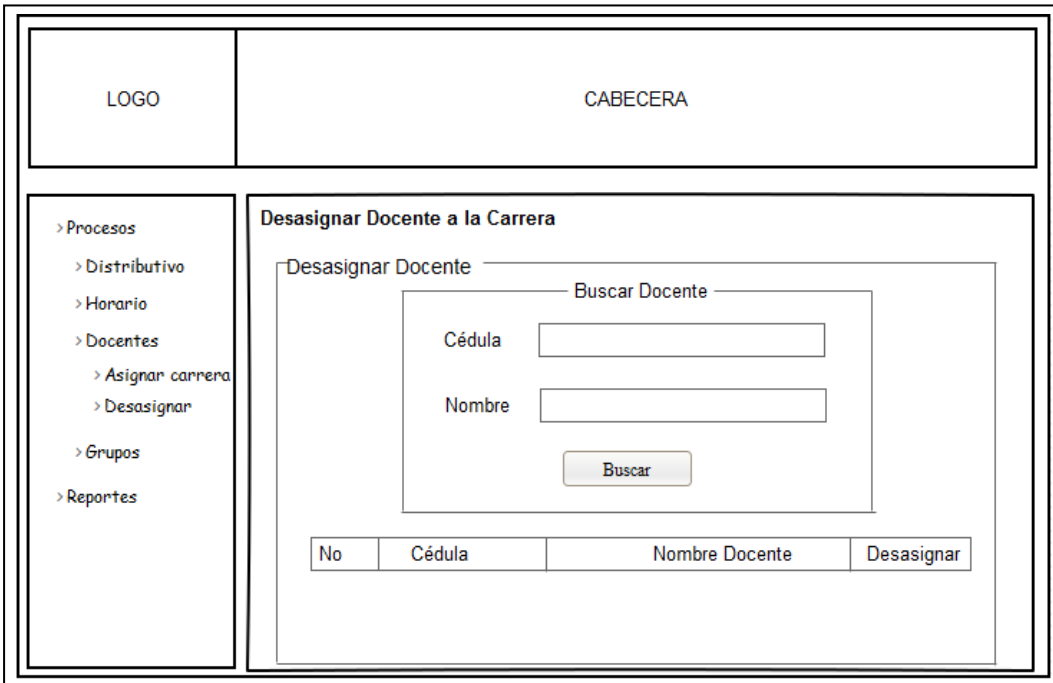


Figura 9. ADV: Desasignar docente.

Elaborado por: Dennis Reyes y Milton Chiliquinga.

En el menú “Grupos”, el usuario cuenta con la opción para buscar un grupo asignado a la carrera, posteriormente puede seleccionar entre los existentes y editarlo si fuera necesario, por otro lado, de ser el caso también puede crear un nuevo grupo el cual es asignado automáticamente a la Carrera del Director logeado.

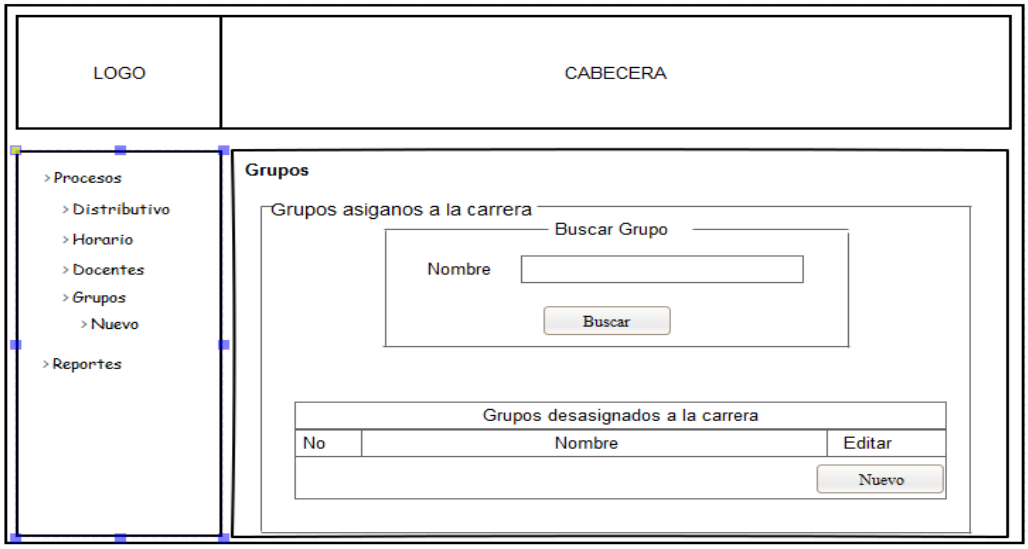


Figura 10. ADV: Nuevo grupo.

Elaborado por: Dennis Reyes y Milton Chiliquinga.

El reporte de asignaciones de los distributivos dispone de filtros tanto para seleccionar la malla, nivel y grupo al cual se desea conocer su información, brindando facilidad para extraer datos relevantes según sea el caso.

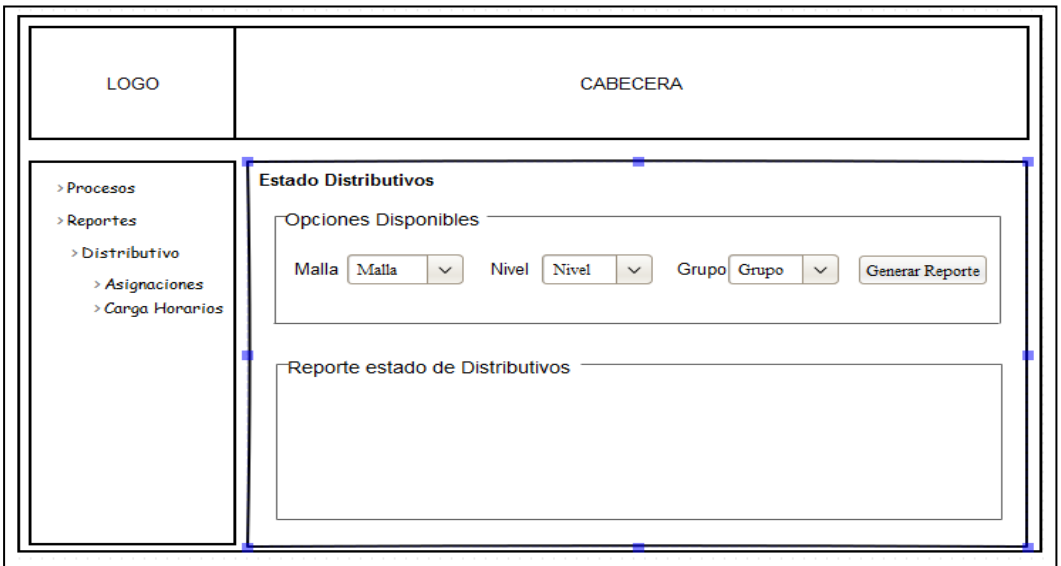


Figura 11. ADV: Reporte distributivos.

Elaborado por: Dennis Reyes y Milton Chiliquinga.

En el reporte de carga horario, el Director de Carrera buscará el docente que necesite, ya sea por el nombre o por la cédula y verificar el porcentaje de carga horaria que posee el mismo.

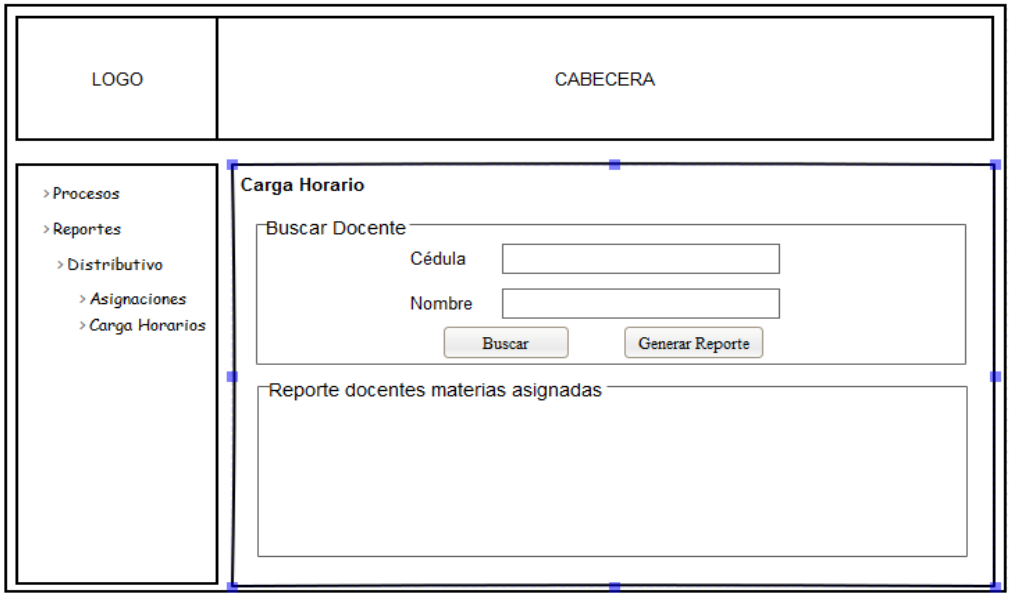


Figura 12. ADV: Carga horarios.

Elaborado por: Dennis Reyes y Milton Chiliquinga.

Por temas de seguridad, el aplicativo ofrece una ventana sencilla en la cual el usuario logeado podrá cambiar su contraseña, teniendo en cuenta ciertas especificaciones con el cambio de la misma.

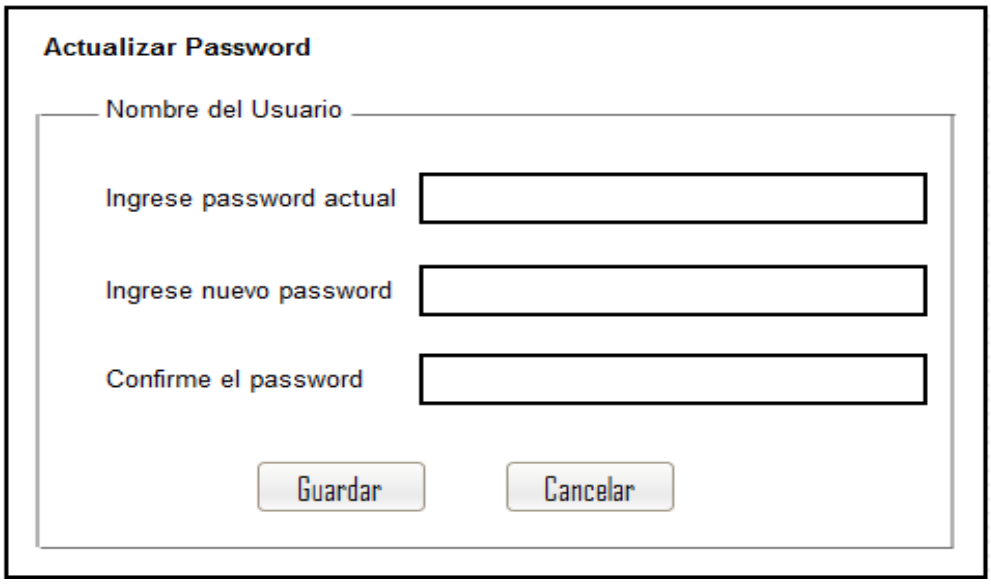


Figura 13. ADV: Actualizar password.

Elaborado por: Dennis Reyes y Milton Chiliquinga.

1.3.4 Diagrama conceptual de la base de datos.

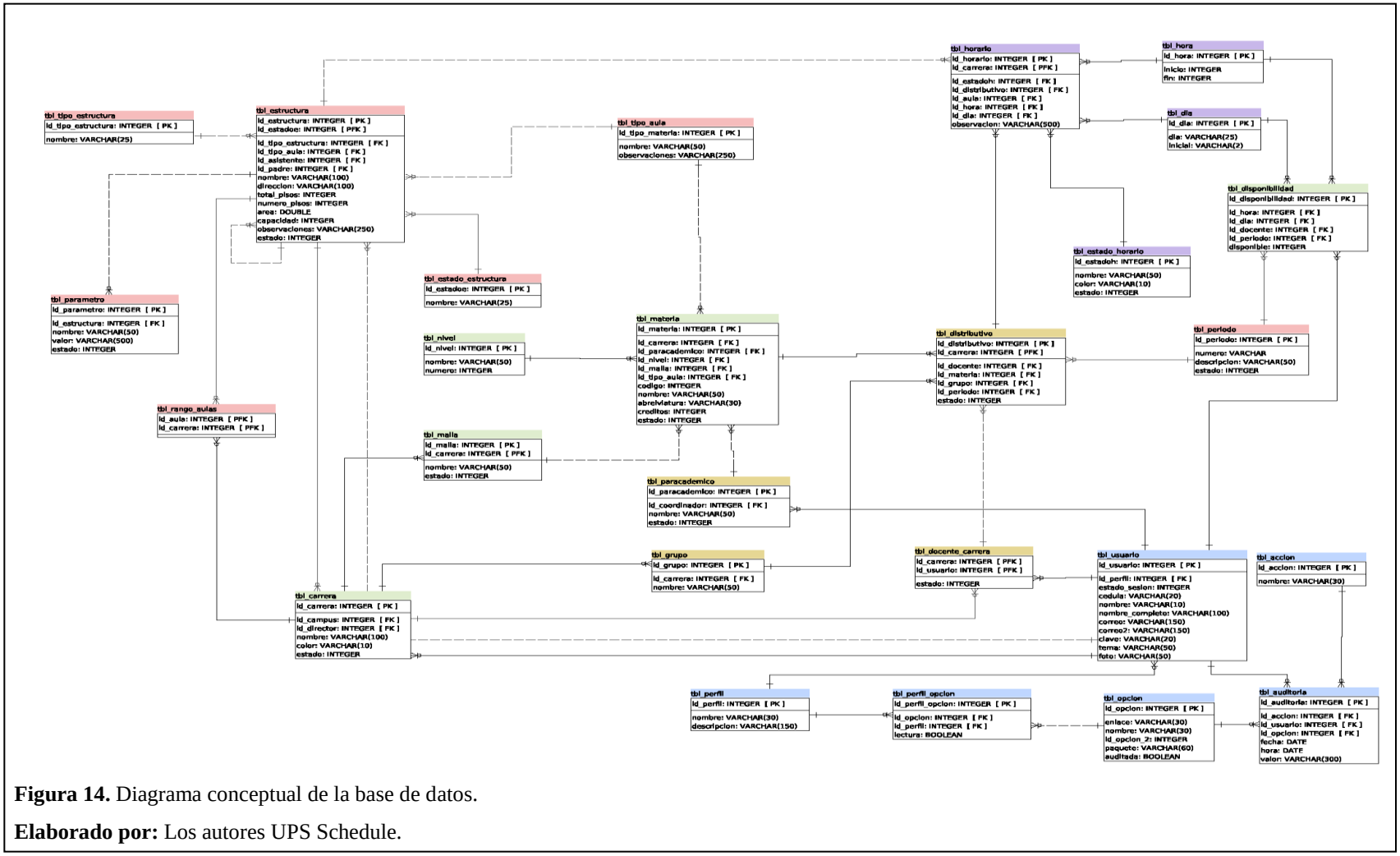


Figura 14. Diagrama conceptual de la base de datos.

Elaborado por: Los autores UPS Schedule.

El siguiente apartado presenta una breve descripción de las tablas que son esenciales para el correcto funcionamiento del módulo de distributivos, cabe mencionar que se ha seguido un estándar para la denominación de tablas dentro de la base de datos UPS Schedule, con el fin de obtener una inmediata diferenciación entre tablas y clases. Por ejemplo: tbl_carrera, en donde "tbl" corresponde a tabla y "carrera" una descripción de dicha tabla.

Tabla 9. Descripción de las tablas de la base de datos

Tabla	Descripción
tbl_carrera	Almacena toda la información de las carreras
tbl_docente_carrera	Almacena toda la información de los docentes por carreras
tbl_materia	Almacena toda la información de las materias
tbl_grupo	Almacena toda la información de los grupos
tbl_periodo	Almacena toda la información del periodo
bl_perfil	Almacena toda la información de perfiles
tbl_malla	Almacena toda la información de la malla académica
tbl_nivel	Almacena toda la información de los niveles
tbl_distributivo	Almacena toda la información de los distributivos
tbl_usuario	Almacena toda la información de los usuarios

Elaborado por: Dennis Reyes y Milton Chiliquinga.

CAPÍTULO 2

DESARROLLO E INTEGRACIÓN

2.1 Plataformas y herramientas

Para el desarrollo del sistema se analizará y aprovechará la gran variedad de ventajas que brinda la tecnología informática para dar solución a problemas planteados en este documento.

El sistema será implementado en un ambiente web, ofreciendo páginas dinámicas, mismas que permitirán el acceso a la aplicación desde cualquier dispositivo dentro de la Universidad Politécnica Salesiana, Campus Sur.

El aplicativo será desarrollado bajo las siguientes tecnologías:

2.1.1 Eclipse (IDE).

Eclipse es un conjunto de servicios para implementar un entorno de desarrollo a partir de componentes conectados.

Una vez compilado el código, el mismo puede ser ejecutado en cualquier máquina. Esto se debe a que el código se ejecuta sobre una máquina virtual, Java Virtual Machine, cuya función principal es la de interpretar el código fuente y convertirlos a código de la CPU que se esté utilizando.

Entre sus principales características, dispone de un editor de texto con resaltado de sintaxis, su compilación es en tiempo real, posee pruebas unitarias con JUnit, dispone de un control de versiones, mediante CVS y es posible añadir control de versiones con Subversión e integración con Hibernate.

2.1.2 Power Architect.

Es una herramienta de modelado de datos y tiene muchas características únicas dirigidas al almacenamiento de los mismos. Permite a los usuarios de la herramienta, realizar perfiles en la base de datos de origen y generar automáticamente los metadatos de ETL.

Entre sus principales características dispone de conexión hacia múltiples bases de datos simultáneamente, todos los proyectos se archivan en formato XML, Drag and Drop de las tablas origen y las columnas en el área de trabajo, es gratuita bajo licencia Open Source GPL v.3, permite la comparación de modelos de datos y estructuras de bases de datos e identifica sus discrepancias.

Es una herramienta diseñada esencialmente para grupos de desarrollo en la cual se puede realizar los respectivos modelados de datos y efectivizar mediante la documentación de los mismos en cada aplicación que se esté desarrollando.

2.1.3 PostgreSQL.

Es un sistema gestor de bases de datos Objeto-Relacional, desarrollado en la década de 1980. Actualmente es considerado como una de las alternativas de sistema de bases de datos de código abierto.

Entre sus principales ventajas:

- ✓ Ahorros considerables en costo de operación, conservando todas las características, estabilidad y rendimiento.
- ✓ Es Multiplataforma.
- ✓ Extensible al disponer del código fuente abierto.
- ✓ Diseñado para ambientes de alto volumen de manejo de información.

Entre sus características principales, permite el uso de transacciones manteniendo la integridad de los datos, soporta integridad referencial que garantiza la validez de los datos de las base de datos, ofrece varios métodos de bloqueo para controlar el acceso

concurrente a los datos de las tablas, permite conectividad TCP/IP, JDBC, ODBC e interfaz con diversos lenguajes de programación.

2.1.4 Tecnología JSF.

Java Server Faces es una tecnología y framework que realiza aplicaciones web, simplificando el desarrollo de interfaces de usuario. Utiliza Java Server Pages para desplegar las páginas web, entre sus principales funcionalidades incluye un conjunto de componentes para la interfaz de usuario, un modelo de eventos en el lado del servidor y administración de estados.

2.1.5 Hibernate.

Es un mapeo objeto relacional, el cual proporciona un marco que facilita el mapeo entre los atributos de una base de datos relacional y el modelo de objetos de una aplicación.

2.1.6 Glassfish.

Es un servidor de aplicaciones que al ser software libre, brinda asistencia gratuita de la comunidad, así como también, las mayores prestaciones en lo que refiere a fiabilidad y rendimiento.

Implementa tecnologías desarrolladas en la plataforma Java EE y entre sus principales características destaca su velocidad, alta escalabilidad, manejo centralizado de clúster e instancias y bajo consumo de memoria.

2.1.7 PrimeFaces.

Es una librería de componentes visuales para JSF, cuenta con gran variedad de componentes que facilitan la creación de páginas web, además es código abierto.

La principal ventaja es que cuenta con 100 componentes OpenSource, algunos con muy alta complejidad, ofreciendo una gran gama de herramientas para el correcto desarrollo.

2.1.8 Enterprise java beans (EJB).

La utilización de EJB disminuye el tiempo de desarrollo de aplicaciones web, permitiendo la construcción de sistemas escalables al ser gestionados por un contenedor del servidor de aplicaciones que permite la gestión de los accesos a los diferentes recursos como son:

- ✓ Bases de datos.
- ✓ Colas de mensajes
- ✓ Ficheros

Proporcionando servicios como:

- ✓ Seguridad
- ✓ Transacciones
- ✓ Mensajería.

2.1.9 AJAX.

Es el acrónimo de Asynchronous JavaScript + XML y trata de variedad de tecnologías entre las cuales las conforman las siguientes:

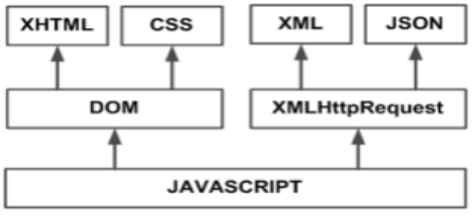


Figura 15. Componentes AJAX.
Elaborado por: Dennis Reyes y Milton Chiliquinga.

Las tecnologías:

- ✓ **XHTML y CSS**, sirven para crear una presentación basada en estándares.

- ✓ **DOM**, sirve para la interacción dinámica de la presentación.
- ✓ **XML, XSLT y JSON**, se las utiliza para el intercambio y la manipulación de información.
- ✓ **XMLHttpRequest**, sirve para el intercambio asíncrono de información.
- ✓ **JavaScript**, une todas las demás tecnologías.

2.1.10 Modelo vista controlador (MVC).

Modelo: Contiene los datos y la funcionalidad de la aplicación.
Vista: Gestiona como se muestra la información.
Controlador: Determina las modificaciones que se realizan en el modelo con la vista

Su principal ventaja es que proporciona un mecanismo de configuración para componentes complejos mucho más tratable que el puramente basado en eventos.

2.1.11 JNDI.

Es un sistema que interactúa con sistemas de nombres y directorios, todas las operaciones se realizan a través del directorio JNDI interfaz proporcionando un marco común, entre sus ventajas:

- ✓ Enlaza diferentes tipos de directorios como LDAP con un DNS
- ✓ Aísla a la aplicación del protocolo y detalles de implementación
- ✓ Proporciona un árbol de memoria almacenar y objetos de configuración del búsqueda

2.1.12 Hibernate query languaje (HQL).

Es un lenguaje de consulta potente, muy similar a SQL, que obtiene información realizando consultas a tablas y sus respectivas columnas. Entre sus principales características es orientado a objetos, dispone de herencia, polimorfismo y asociación.

Tabla 10. Descripción de las tablas de la base de datos

Cláusulas	Funciones escalares
From	avg (…)
Select	sum (…)
Where	min (…) / max (…)
Orden by	count (…)
Group by	count (distinct…)

Elaborado por: Dennis Reyes y Milton Chiliquinga.

2.1.13 Java authentication and authorization service (JAAS).

Es un API de Java que provee un framework con herramientas de autenticación y autenticación para aplicaciones desarrolladas en Java.

Ofrece una interfaz portable que autentica usuarios en Java y a su vez garantiza los accesos a varias aplicaciones, basándose en permisos y políticos de seguridad.

2.1.14 Jasper report.

Es una biblioteca de clases de código abierto que proporciona a los desarrolladores facilidades de generar reportes en aplicaciones Java, requiere de un Java Development KIT (JDK) 1.3 o superior para la exitosa ejecución de los sistemas.

Entre sus características principales cuenta con un diseño de informe flexible, capaz de generar reportes gráficos y exportar informes a una variedad de formatos.

2.2 Diagrama físico

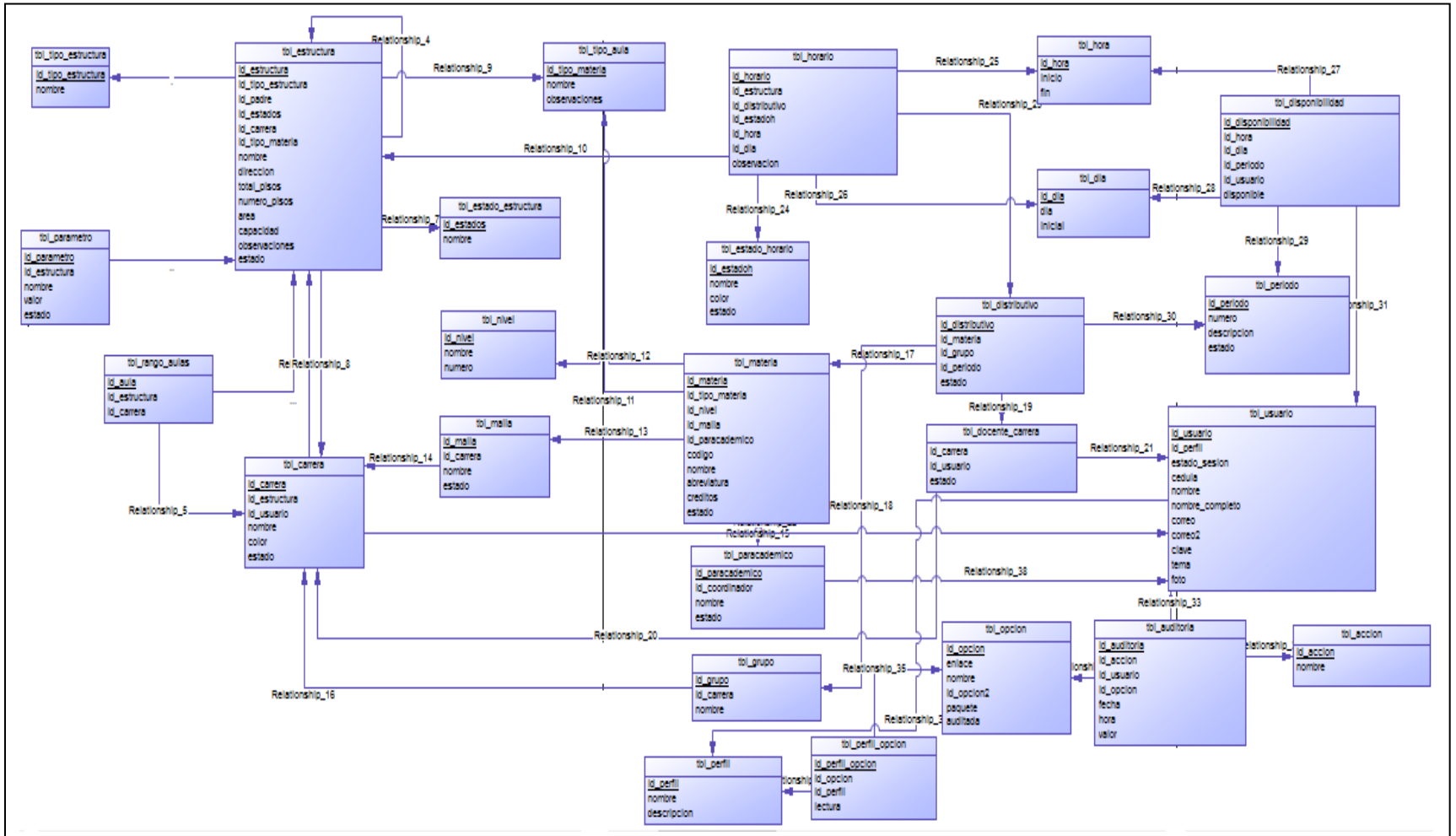


Figura 16. Diagrama físico de la base de datos.

Elaborado por: Dennis Reyes y Milton Chiliquinga.

2.2.1 Diccionario de datos.

A continuación se detalla el diccionario de datos de cada una de las tablas que son utilizadas por el sistema de distributivos, en el cual consta de una descripción de la tabla, sus respectivos campos, claves primarias y sus respectivas relaciones con las demás tablas de la base de datos.

Tabla 11. Tabla carrera, base de datos UPS Schedule

Tabla: tbl_carrera (permite almacenar toda la información de las carreras)						
Campo	**Descripción**	**Tipo dato**	**Ext.**	**Obligatorio**	**Restricción**	**Restricción validación**
id_carrera	Identificador de la Carrera	Integer		S	PK	NOT NULL
id_campus	Identificador del campus	Integer		S	FK	NOT NULL
id_director	Identificador del director de carrera	Integer		S	FK	NOT NULL
Nombre	Nombre de la Carrera	Character varying	100	S		NOT NULL
Color	Color de la Carrera	Character varying	10	S		NOT NULL
Estado	Estado de la Carrera	Integer		S		NOT NULL
Clave Primaria	tbl_carrera_pk (id_carrera)					
Clave Foráneas	tbl_estructura_tbl_carrera_fk (id_campus) ; tbl_usuario_tbl_carrera_fk (id_director)					

Elaborado por: Dennis Reyes y Milton Chiliquinga.

Tabla 12. Tabla docente_carrera, base de datos UPS Schedule

Tabla: tbl_docente_carrera (permite almacenar toda la información de los docentes por carreras)						
Campo	**Descripción**	**Tipo dato**	**Ext.**	**Obligatorio**	**Restricción**	**Restricción validación**
id_carrera	Identificador de la Carrera	Integer		S	PK	NOT NULL
id_usuario	Identificador del usuario	Integer		S	FK	NOT NULL
Estado	Estado del docente por dicha carrera	Integer		S		NOT NULL
Clave Primaria	tbl_docente_carrera_pk (id_carrera, id_usuario)					
Clave Foráneas	carrera_docente_carrera_fk (id_carrera) ; tbl_usuario_tbl_docente_carrera_fk (id_usuario)					

Elaborado por: Dennis Reyes y Milton Chiliquinga.

Tabla 13. Tabla materia, base de datos UPS Schedule

Tabla: tbl_materia (permite almacenar toda la información de los docentes por carreras)						
Campo	**Descripción**	**Tipo dato**	**Ext.**	**Obligatorio**	**Restricción**	**Resticción validación**
id_materia	Identificador del material	Integer		S	PK	NOT NULL
id_carrera	Identificador de la Carrera	Integer		S	FK	NOT NULL
id_paracademico	Identificador del paracadémico	Integer		S	FK	NULL
id_nivel	Identificador del nivel	Integer		S	FK	NOT NULL
id_malla	Identificador de la malla	Integer		S	FK	NOT NULL
id_tipo_aula	Identificador del tipo de aula	Integer		S	FK	NOT NULL
Codigo	Código de la materia	Integer		S		NOT NULL
Nombre	Nombre de la materia	Character varying	50	S		NOT NULL
Creditos	Créditos	Integer		S		NOT NULL
Estado	Estado de la materia	Integer		S		NOT NULL
Clave Primaria	tbl_materia_pk (id_materia)					
Clave Foráneas	tbl_malla_tb_materia_fk (id_malla, id_carrera) ; tbl_nivel_tb_materia_fk (id_nivel) ; tbl_paracademico_tb_materia_fk (id_paracademico) ; tbl_tipo_materia_tb_materia_fk (id_tipo_aula)					

Elaborado por: Dennis Reyes y Milton Chiliquinga.

Tabla 14. Tabla grupo, base de datos UPS Schedule

Tabla: tbl_grupo (permite almacenar toda la información de los grupos)						
Campo	**Descripción**	**Tipo dato**	**Ext.**	**Obligatorio**	**Restricción**	**Resticción validación**
id_grupo	Identificador del grupo	Integer		S	PK	NOT NULL
id_carrera	Identificador de la Carrera	Integer		S	FK	NOT NULL
nombre	Nombre del grupo	Character varying	50	S		NOT NULL
Clave Primaria	tbl_grupo_pk (id_grupo)					
Clave Foráneas	tbl_carrera_tb_dir_grupo_fk (id_carrera)					

Elaborado por: Dennis Reyes y Milton Chiliquinga.

Tabla 15. Tabla periodo, base de datos UPS Schedule

Tabla: tbl_periodo (permite almacenar toda la información del periodo)						
Campo	**Descripción**	**Tipo dato**	**Ext.**	**Obligatorio**	**Restricción**	**Restricción validación**
id_periodo	Identificador del periodo	Integer		S	PK	NOT NULL
numero	Número del periodo	Character varying		S		NOT NULL
descripcion	Descripción del periodo	Character varying	50	S		NOT NULL
estado	Estado del periodo	Integer		S		NOT NULL
Clave Primaria	tbl_periodo_pk (id_periodo)					

Elaborado por: Dennis Reyes y Milton Chiliquinga.

Tabla 16. Tabla perfil, base de datos UPS Schedule

Tabla: tbl_perfil (permite almacenar toda la información de perfiles)						
Campo	**Descripción**	**Tipo dato**	**Ext.**	**Obligatorio**	**Restricción**	**Restricción validación**
id_perfil	Identificador del perfil	Integer		S	PK	NOT NULL
nombre	Nombre del perfil	Character varying	30	S		NOT NULL
descripcion	Descripción del perfil	Character varying	150	S		NOT NULL
Clave Primaria	tb_sis_perfil_codigo_sisper_pk (id_perfil)					

Elaborado por: Dennis Reyes y Milton Chiliquinga.

Tabla 17. Tabla malla, base de datos UPS Schedule

Tabla: tbl_malla (permite almacenar toda la información de la malla académica)						
Campo	**Descripción**	**Tipo dato**	**Ext.**	**Obligatorio**	**Restricción**	**Restricción validación**
id_malla	Identificador de la malla	Integer		S	PK	NOT NULL
id_carrera	Identificador de la Carrera	Integer		S		NOT NULL
nombre	Nombre de la malla	Character varying	50	S		NOT NULL
Estado	Estado de la malla	Integer		S		NOT NULL
Clave Primaria	tbl_malla_pk (id_malla)					
Clave Foráneas	tbl_carrera_tb_dir_malla_fk (id_carrera)					

Elaborado por: Dennis Reyes y Milton Chiliquinga.

Tabla 18. Tabla nivel, base de datos UPS Schedule

Tabla: tbl_nivel (permite almacenar toda la información de los niveles)						
Campo	**Descripción**	**Tipo dato**	**Ext.**	**Obligatorio**	**Restricción**	**Restricción validación**
id_nivel	Identificador de la malla	Integer		S	PK	NOT NULL
Numero	identificador de la Carrera	Integer		S		NOT NULL
Nombre	Nombre de la malla	Character varying	50	S		NULL
Clave Primaria	tbl_nivel_pk (id_nivel)					

Elaborado por: Dennis Reyes y Milton Chiliquinga.

Tabla 19. Tabla distributivo, base de datos UPS Schedule

Tabla: tbl_distributivo (permite almacenar toda la información de los distributivos)						
Campo	**Descripción**	**Tipo dato**	**Ext.**	**Obligatorio**	**Restricción**	**Restricción validación**
id_distributivo	Identificador de la materia	Integer		S	PK	NOT NULL
id_carrera	Identificador de la Carrera	Integer		S	FK	NOT NULL
id_docente	Identificador del paracadémico	Integer		S	FK	NULL
id_materia	Identificador del nivel	Integer		S	FK	NOT NULL
id_grupo	Identificador de la malla	Integer		S	FK	NOT NULL
id_periodo	Identificador del tipo de aula	Integer		S	FK	NOT NULL
Estado	Estado de la materia	Integer		S		NOT NULL
Clave Primaria	tbl_distributivo_pk (id_distributivo)					
Clave Foráneas	periodo_distributivo_fk (id_periodo) ; tb_materia_tbl_distributivo_fk (id_materia) ; tbl_docente_carrera_tbl_distributivo_fk (id_docente,id_carrera) ; tbl_grupo__tbl_distributivo_fk (id_grupo)					

Elaborado por: Dennis Reyes y Milton Chiliquinga.

Tabla 20. Tabla usuario, base de datos UPS Schedule

Tabla: tbl_usuario (permite almacenar toda la información de los usuarios)						
Campo	**Descripción**	**Tipo dato**	**Ext.**	**Obligatorio**	**Restricción**	**Restricción validación**
id_usuario	Identificador del usuario	Integer		S	PK	NOT NULL
id_perfil	Identificador del perfil	Integer		S	FK	NOT NULL
estado_sesion	Estado de la sesión	Integer		S	FK	NOT NULL

(Continuación)

Tabla 20. Tabla usuario, base de datos UPS Schedule

Cedula	Número de cédula del usuario	Character varying	50	S		NOT NULL
Nombre	Nick	Character varying	50	S		NOT NULL
nombre_completo	Nombre completo del usuario	Character varying	50	S		NOT NULL
Correo	Dirección de correo electrónico	Character varying	50	S		NULL
correo2	Dirección de correo electrónico alterno	Character varying	50	S		NULL
Clave	clave	Character varying	50	S		NOT NULL
Tema	tema	Character varying	50	S		NOT NULL
Foto	foto	Character varying	50	S		NULL
Clave Primaria	tb_sis_usuario_codigo_pk (id_usuario)					
Clave Foráneas	tbl_perfil_tbl_usuario_fk (id_perfil)					

Elaborado por: Dennis Reyes y Milton Chiliquinga.

2.3 Codificación

El siguiente apartado detalla el código fuente que ha sido implementado para el sistema de distributivos. Cabe mencionar que se describirá los métodos más importantes para cumplir con el objetivo principal debido a que en el mismo existe líneas de código conocidos para los que están familiarizados con el mundo de la programación.

2.3.1 Estructura de paquetes y clases.

La aplicación web ha sido desarrollada utilizando el patrón MVC (Modelo – Vista – Controlador), las clases han sido agrupadas en distintos paquetes según la funcionalidad que cumplan cada uno de ellos.

Tomando en cuenta lo mencionado anteriormente, los paquetes son los siguientes:

- ✓ ec.edu.ups.schedule.common.validator
- ✓ ec.edu.ups.schedule.distributivo.controlador
- ✓ ec.edu.ups.schedule.distributivo.datamanager
- ✓ ec.edu.ups.schedule.distributivo.datamodel
- ✓ ec.edu.ups.schedule.horarios
- ✓ ec.edu.ups.schedule.horarios.converter
- ✓ ec.edu.ups.schedule.recursos
- ✓ ec.edu.ups.schedule.util

En el siguiente diagrama, se puede apreciar de mejor manera lo expuesto, donde muestra la relación existente entre paquetes.

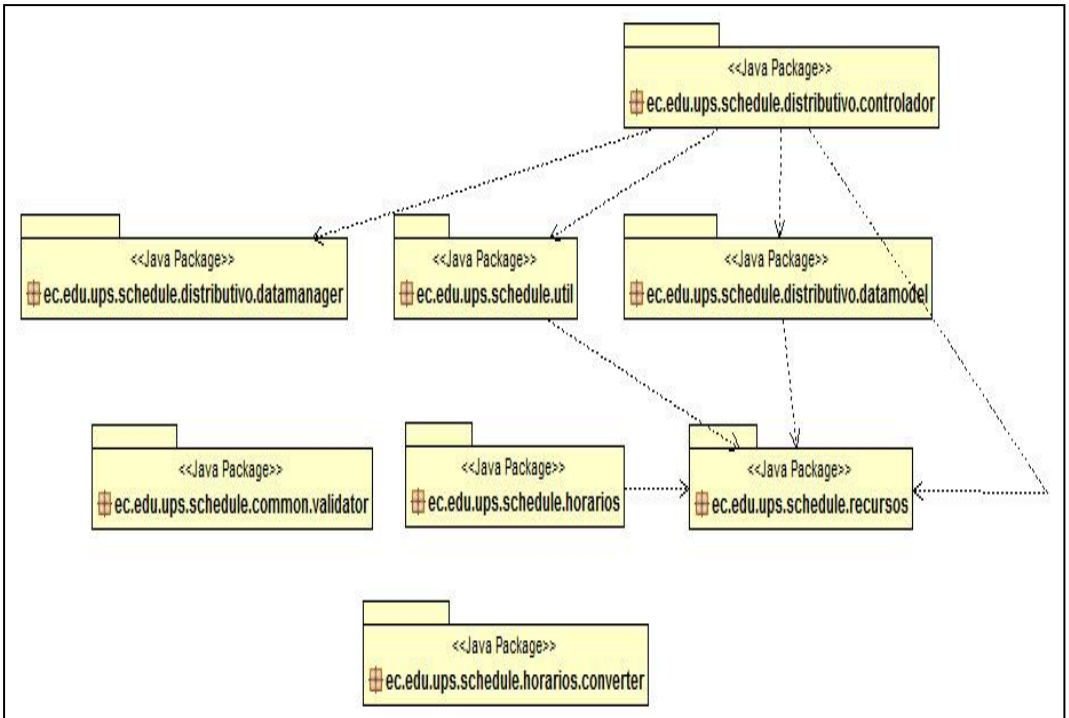


Figura 17. Datos ingresados, threads 20.

Elaborado por: Dennis Reyes y Milton Chiliquinga.

A continuación se profundizará con el paquete con mayor importancia, realizando una explicación del objetivo de sus clases y mostrando su respectiva codificación.

Paquete: ec.edu.ups.schedule.distributivo.controlador

Clase: Distributivo controlador.

En el siguiente apartado se presenta los métodos trascendentes de la clase "DistributivoControlador.java", entre los cuales son:

Método: Obtener distributivo por nivel grupo.

El objetivo es instanciar una lista en el datamanager de sesión, esta lista contiene todos los distributivos asociados al Director de Carrera que ingresó al sistema, agrupado por grupo, nivel y materia.

Código fuente:

```
Public void obtenerDistributivoPorNivelGrupo() {
      try {
            if(directorCarreraDataManager.getIdNivelSelecionad
            o(). equals(-1)&&
            directorCarreraDataManager.getIdMalla(). equals(-
            1))
directorCarreraDataManager.setDistributivoGrupos(null);
            else {
                  directorCarreraDataManager.setDistributivoGr
                  upos(director
                  CarreraServicio.obtenerDistributivoPorCarrer
                  aGrupo(
      directorCarreraDataManager.getCarrera().getIdCarrera(),
                  directorCarreraDataManager.getIdNivelSelecio
                  nado(),
                  directorCarreraDataManager.getIdMalla(),
            directorCarreraDataManager.getPeriodoActual().
                  getIdPeriodo()));
                  directorCarreraDataManager.setGruposPorCarre
                  ra(carrera        Servicio.
                  obtenerGgruposPorCarreraSinAsignar(

      directorCarreraDataManager.getCarrera().getIdCarrera(),
                  directorCarreraDataManager.getIdNivelSelecio
                  nado(),
                  directorCarreraDataManager.getIdMalla(),
      directorCarreraDataManager.getPeriodoActual().
                  getIdPeriodo()));
      }
      } catch (Exception e) {
            e.printStackTrace();
            MensajeControlador.addError(null, e.getMessage());
      }
}
```

Método: Calcular porcentaje asignado.

Tiene como fin obtener el porcentaje total de las materias asignadas en un nivel seleccionado, dentro de una carrera específica.

Código fuente:

```
private double calcularPorcentajeAsignado() throws Exception {
	int totalDistributivos = 0;
	Integer totalMateriasSinAsignar = 0;
	double respuesta = 0;
	if (null
	!=directorCarreraDataManager.getDistributivoGrupos()) {
		for (GrupoDTO grupoDto :
		directorCarreraDataManager.getDistributivoGrupos()
		) {
			totalDistributivos +=
			grupoDto.getDistributivo().size();
			totalMateriasSinAsignar +=
			grupoDto.getMateriasSinAsignar();
	}
	}
	respuesta = (totalMateriasSinAsignar * 100) /
	totalDistributivos;
	return respuesta;
}
```

Método: Asignar materia docente.

Su función es la de asignar una materia seleccionada por el usuario hacia un docente perteneciente a la Carrera del Director que ha ingresado al sistema.

Código fuente:

```
public void asignarMateriaDocente() {
	try {
		Distributivo distributivo
		=directorCarreraDataManager.getDistributivoMateria
		Selec();
		distributivo.setDocenteCarrera(directorCarreraData
		Manager. getDocenteSelecionado());
		directorCarreraServicio.guardarDistributivo(distri
		butivo);
		directorCarreraDataManager.setDistributivoGrupos(d
		irector
		CarreraServicio.obtenerDistributivoPorCarreraGrupo
		(
	directorCarreraDataManager.getCarrera().getIdCarrera(),
```

```
            directorCarreraDataManager.getIdNivelSelecionado()
            , directorCarreraDataManager.getIdMalla(),
      directorCarreraDataManager.getPeriodoActual().
            getIdPeriodo()));
            MensajeControlador.addInfo(null,
            UtilResources.getString("mensaje.exito.asignar.doc
            ente. materia",
            distributivo.getDocenteCarrera().getUsuario().
            getNombreCompleto(),distributivo.getMateria().
            getNombreDirmat(),distributivo.getGrupo().getNombr
            e(),
            distributivo.getMateria().getNivel().getNumero().t
            oString()));
      } catch (Exception e) {
      MensajeControlador.addError(null, e.getMessage());
      }
}
```

Método: Crear grupo sin asignar.

Crea un grupo para un nivel específico, cabe mencionar que dentro del nuevo grupo, todas las materias creadas no se encuentran asignadas a un docente.

Código fuente:

```
public void crearGrupoSinAsignar() {
      try {
      Distributivo distributivo;
            List<Materia> materias =
            carreraServicio.obtenerMateriasPor
            CarreraNivel(directorCarreraDataManager.getIdNivel
            Selecionado(),
            directorCarreraDataManager.getCarrera().
            getIdCarrera(),directorCarreraDataManager.getIdMal
            la());
            Grupo grupo = carreraServicio.obtenerGrupoPorId(
            directorCarrera
            DataManager.getIdGrupoSeleccionado());
            DocenteCarrera docenteCarrera = new
            DocenteCarrera();
      docenteCarrera.setId(new DocenteCarreraPK());
            docenteCarrera.getId().setIdCarrera(grupo.getCarre
            ra(). getIdCarrera());
      for (Materia materia : materias) {
            distributivo = new Distributivo();
      distributivo.setDocenteCarrera(docenteCarrera);
            distributivo.setEstado(ESTADO_ACTIVO);
            distributivo.setGrupo(grupo);
            distributivo.setMateria(materia);
                  distributivo.setPeriodo(directorCarreraDataM
                  anager. getPeriodoActual());
                  directorCarreraServicio.crearDistributivo(di
                  stributivo);
            }
```

```
                    directorCarreraDataManager.setDistributivoGr
                    upos(
                    directorCarreraServicio.obtenerDistributivoP
                    orCarreraGrupo(directorCarreraDataManager.ge
                    tCarrera().getIdCarrera(),directorCarreraDat
                    aManager.getIdNivelSelecionado(),directorCar
                    reraDataManager.getIdMalla(),directorCarrera
                    DataManager.getPeriodoActual().
                    getIdPeriodo()));
                    directorCarreraDataManager.setGruposPorCarre
                    ra(
                    carreraServicio.obtenerGgruposPorCarreraSinA
                    signar(directorCarreraDataManager.getCarrera
                    ().getIdCarrera(),directorCarreraDataManager
                    .getIdNivelSelecionado(),directorCarreraData
                    Manager.getIdMalla(),directorCarreraDataMana
                    ger.getPeriodoActual().getIdPeriodo()));
      } catch (Exception e) {
      MensajeControlador.addError(null, e.getMessage());
      }
}
```

Método: Agregar materia a distributivo.

Su finalidad es generar una materia hacia un distributivo específico que ha seleccionado el usuario logeado.

Código fuente:

```
public void agergarMateriaDistributivo() {
      try {
            Distributivo distributivo = new Distributivo();
            DocenteCarrera docenteCarrera = new
      DocenteCarrera();
            docenteCarrera.setId(new DocenteCarreraPK());
            docenteCarrera.getId().setIdCarrera(
           directorCarreraDataManager.getCarrera().getIdCarrer
           a());
            distributivo.setDocenteCarrera(docenteCarrera);
            distributivo.setEstado(ESTADO_ACTIVO);
            distributivo.setGrupo(directorCarreraDataManager.
            getGrupoSeleccionado().getGrupo());
            distributivo.setMateria(getMateriaSeleccionada());
            distributivo.setPeriodo(directorCarreraDataManager
            . getPeriodoActual());
            directorCarreraServicio.crearDistributivo
            (distributivo);
            directorCarreraDataManager.setDistributivoGrupos(
            directorCarreraServicio.obtenerDistributivoPorCarr
            eraGrupo(directorCarreraDataManager.getCarrera().
            getIdCarrera(),directorCarreraDataManager.getIdNiv
            elSelecionado(),
            directorCarreraDataManager.getIdMalla(),
```

```
            directorCarreraDataManager.getPeriodoActual().
            getIdPeriodo()));
            MensajeControlador.addInfo(null,UtilResources.getS
            tring( "mensaje.exito.agregar.materia",
            getMateriaSeleccionada().
            getNombreDirmat(),directorCarreraDataManager.getGr
            upo Seleccionado(). getGrupo().getNombre()));
      } catch (Exception e) {
            e.printStackTrace();
            MensajeControlador.addError(null, e.getMessage());
      }
}
```

Clase: Login controler.

A continuación se presenta el método más importante de la clase "LoginController.java".

Método: Login.

El objetivo principal de este método es el de realizar la autenticación del nombre de usuario y contraseña proporcionado por el Director de Carrera.

Código fuente:

```
public void login(ActionEvent actionEvent) {
      FacesContext context = null;
      try {
      context = FacesContext.getCurrentInstance();
            HttpServletRequest request = (HttpServletRequest)
            context.getExternalContext().getRequest();
      String redirectTo = "/paginas/privadas/principal.jsf";
      request.login(username, password);
            Usuario usuario =
            directorCarreraServicio.obtenerUsuarioPorUserName
            (FacesContext.getCurrentInstance()
      .getExternalContext().getRemoteUser());
      if (ESTADO_ACTIVO.equals(usuario.getEstadoSesion()))
            context.getExternalContext().redirect(request.
            getContextPath() + redirectTo);
      else
            context.getExternalContext().redirect(request.
            getContextPath() + "/paginas/error/403.jsf");
      } catch (Exception e) {
      e.printStackTrace();
            MensajeControlador.addError(null, "Las
            credenciales proporcionadas no son las
            correctas");
      }
}
```

Clase: Director carrera controlador.

Método: Actualizar password.

El método mencionado actualiza la contraseña del usuario que ha ingresado al sistema.

Código fuente:

```
public void actualizarPassword() {
	try {
		String passwordActual =
		MD5Encryptor.getInstancia().encryptPassword(
	directorCarreraDataManager.getPasswordActual());
		if
		(passwordActual.equals(directorCarreraDataManager.
		getUsuarioLogeado().getClave())) {
			if
			(directorCarreraDataManager.getNewPassword()
			. equals(directorCarreraDataManager.
			getConfirmPassword())) {
				directorCarreraDataManager.
				getUsuarioLogeado().
				setClave(MD5Encryptor.
				getInstancia().encryptPassword(directo
				rCarreraDataManager.
				getNewPassword()));
			directorCarreraServicio.guardarUsuario
			(directorCarreraDataManager.getUsuarioLogead
			o());
			MensajeControlador.addInfo(null, "Los datos
			del usuario "+
			directorCarreraDataManager.getUsuarioLogeado
			(). getNombreCompleto() + " se actualizarón
			correctamente");
			directorCarreraDataManager.setNewPassword
		(null);
				directorCarreraDataManager.
				setConfirmPassword(null);
				directorCarreraDataManager.setPasswordActual
				(null);
				} else {
					throw new Exception("Error: El nuevo
					password no coincide con su
					confirmación");
				}
		} else {
			throw new Exception("Error: El password
			actual no es el correcto");
		}
	  } catch (Exception e) {
		e.printStackTrace();
	MensajeControlador.addError(null, e.getMessage());
}}
```

Clase: Asignación reporte controlador.

Método: Generar reportes asignación PDF.

Genera un reporte con extensión .pdf con un diseño útil y fácil de interpretar para el usuario de acuerdo al filtrado realizado por el usuario que ingresa al sistema.

Código fuente:

```
public void generearReporteAsignacionPDF() {
try {
            Map<String, Object> parametros
            =inicializarParametros();
            parametros.put(REPORTE_PARAMETRO_PERIODO,
            directorCarreraDataManager. getPeriodoActual().
            getNumero());
      parametros.put(REPORTE_PARAMETRO_MALLA,
            directorCarreraServicio.obtenerMallaPorId(
            asignacionDataManager.getIdMalla()).getNombre());
            GeneradorReporte.imprimirReporte(parametros,
            REPORTE_ARCHIVO_JASPER_ASIGNACION,
      REPORTE_NOMBRE, GeneradorReporte.FORMATO_PDF,
      asignacionDataManager.getDistributivoGruposReporte());
} catch (Exception e) {

      e.printStackTrace();

      MensajeControlador.addError(null, e.getMessage());
}
}
```

Otros aspectos importantes son los DAOS, los mismos que son utilizados para acceder a la base de datos, realizar las correspondientes consultas hacia las tablas entre otras actividades. La principal diferencia entre un DTO y un DAO es que el objeto de transferencia de datos únicamente almacena información y entrega sus propios datos.

La imagen siguiente muestra el diagrama de clases de los DAOS utilizados en el sistema de distributivos, a partir del mencionado diagrama, iniciará con una breve explicación de los métodos más importantes utilizados para cumplir con el objetivo principal del sistema.

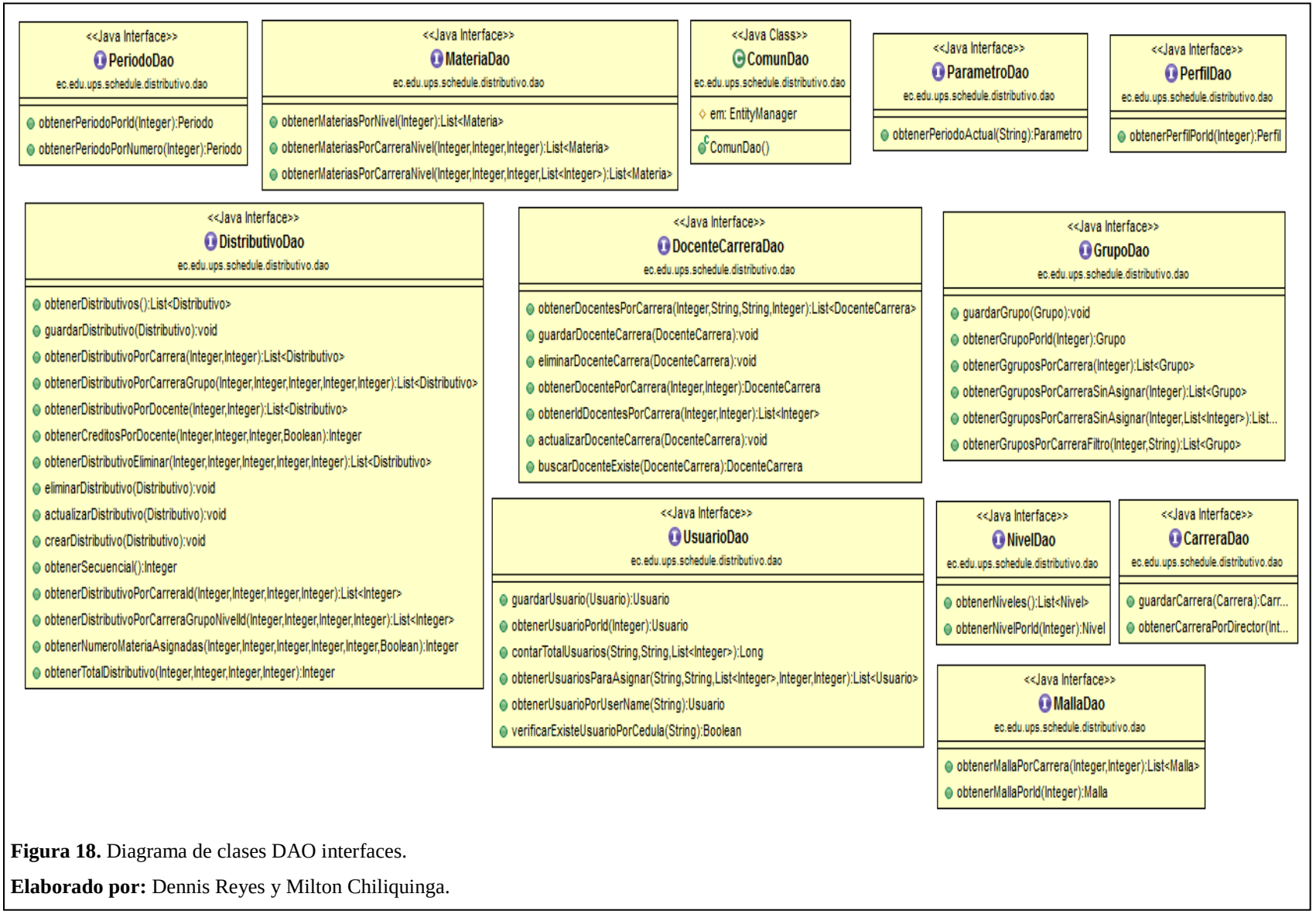


Figura 18. Diagrama de clases DAO interfaces.

Elaborado por: Dennis Reyes y Milton Chiliquinga.

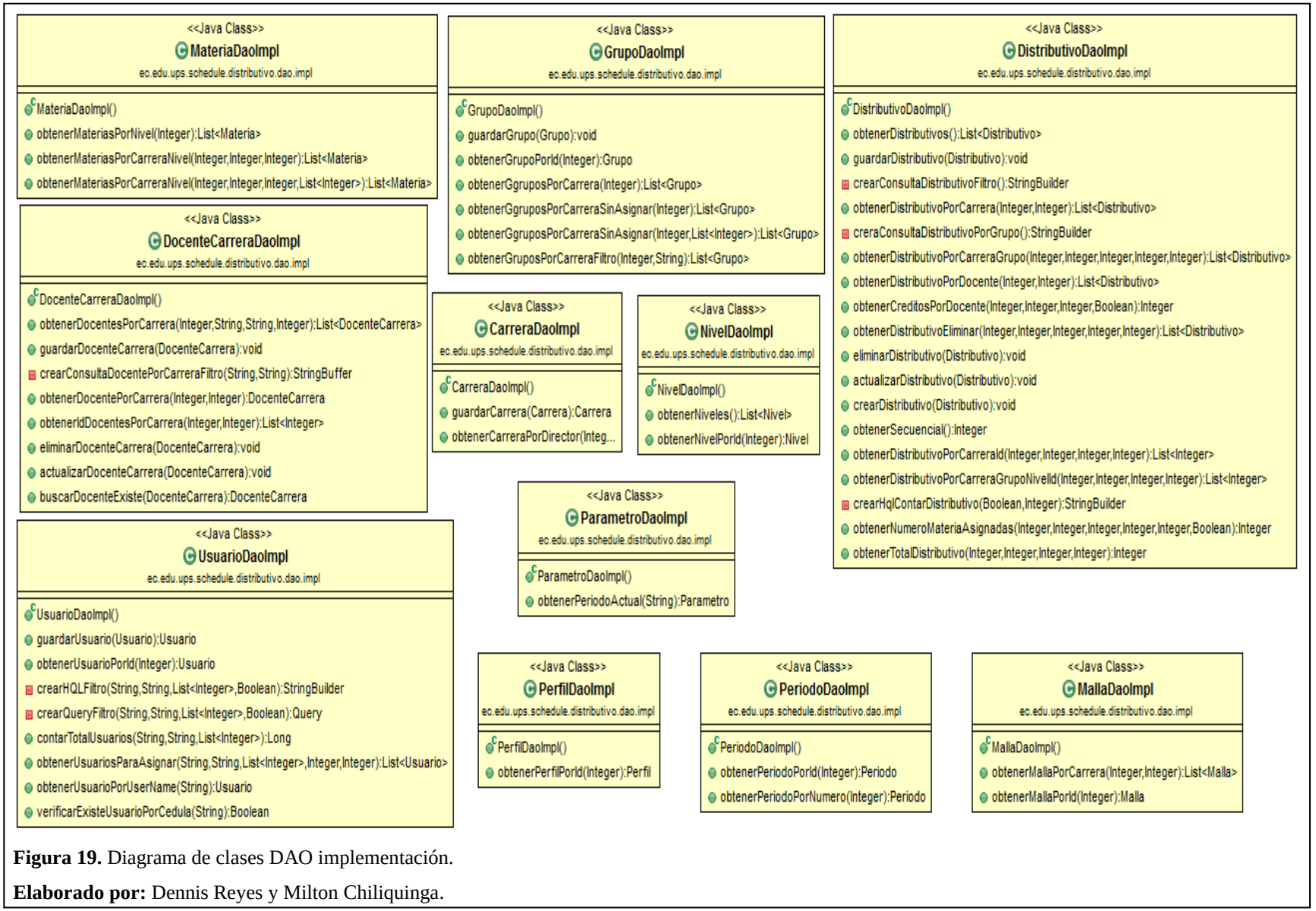


Figura 19. Diagrama de clases DAO implementación.

Elaborado por: Dennis Reyes y Milton Chiliquinga.

DistributivoDao: Se encuentra asociado a la tabla tbl_distributivos de la base de datos, entre sus principales métodos dispone:

- ✓ **Obtener distributivo por carrera grupo:** Su función es obtener los diferentes distributivos asociados a una carrera, nivel y un grupo en particular.

Código fuente:

```
List<Distributivo>   obtenerDistributivoPorCarreraGrupo(Integer
idCarrera, Integer idNivel, Integer idGrupo,Integer idMalla,
Integer idPeriodo);
```

Dónde:

- ✓ **idCarrera:** Identificador de la carrera.
- ✓ **idNivel:** Identificador del nivel.
- ✓ **idGrupo:** Identificar del grupo.
- ✓ **idMalla:** Identificador de la malla.
- ✓ **idPeriodo:** Identificador del periodo.

Implementación:

```
public List<Distributivo>
obtenerDistributivoPorCarreraGrupo(Integer idCarrera, Integer
idNivel, Integer idGrupo,Integer idMalla, Integer idPeriodo) {
List<Distributivo> listaDistributivo;
      Query query =
      em.createQuery(creraConsultaDistributivoPorGrupo().toStr
      ing());
query.setParameter("idCarrera", idCarrera);
query.setParameter("idNivel", idNivel);
query.setParameter("idGrupo", idGrupo);
query.setParameter("idMalla", idMalla);
query.setParameter("idPeriodo", idPeriodo);
query.setParameter("activo",
      ConstantesScheduleCore.ESTADO_ACTIVO);
listaDistributivo = (ArrayList<Distributivo>)
query.getResultList();
return listaDistributivo;
}
```

- ✓ **Obtener créditos por docente:** Consigue el número de créditos asignados a un docente en el periodo actual, en el caso de que el parámetro "isCreditosCarrera", es verdadero y obtiene los créditos asociados a la Carrera del Director asociado, caso contrario obtiene los créditos que tiene el docente fuera de la Carrera del Director.

Código fuente:

```
Integer obtenerCreditosPorDocente(Integer idDocente, Integer
idPeriodo, Integer idCarrera, Boolean isCreditosCarrera);
```

Dónde:

- ✓ **idDocente:** Identificador de la docente.
- ✓ **idPeriodo:** Identificador del periodo.
- ✓ **idCarrera:** Identificador de la carrera.
- ✓ **isCreditosCarrera:** Valor booleano.

Implementación:

```
public Integer obtenerCreditosPorDocente(Integer idDocente,
Integer idPeriodo, Integer idCarrera,Boolean
isCreditosCarrera) {
Long creditos;
StringBuilder hql = new StringBuilder();
      hql.append("SELECT SUM(d.materia.creditosDirmat) FROM
      Distributivo d ");
hql.append("WHERE d.docenteCarrera.id.idUsuario = :idDocente
");
hql.append("AND d.periodo.idPeriodo = :idPeriodo ");
hql.append("AND d.docenteCarrera.id.idCarrera ");
hql.append(isCreditosCarrera ? "=" : "!=");
hql.append(" :idCarrera ");
hql.append("AND d.estado = :activo ");
Query query = em.createQuery(hql.toString());
query.setParameter("idDocente", idDocente);
query.setParameter("idPeriodo", idPeriodo);
query.setParameter("idCarrera", idCarrera);
      query.setParameter("activo",
      ConstantesScheduleCore.ESTADO_ACTIVO);
creditos = (Long) query.getSingleResult();
if (creditos == null)
      return 0;
else
      return creditos.intValue();}
```

- ✓ **Obtener distributivo por carrera:** Obtiene una lista de claves primarias dentro de un distributivo en particular, filtrado por la carrera y el nivel, principalmente se la utiliza para encontrar los grupos ya asignados.

Código Fuente:

```
List<Integer>obtenerDistributivoPorCarreraId(Integer idCarrera,
Integer idNivel, Integer idMalla, Integer idPeriodo);
```

Dónde:

- ✓ **idCarrera:** Identificador de la carrera.
- ✓ **idNivel:** Identificador del nivel.
- ✓ **idMalla:** Identificador de la malla.
- ✓ **idPeriodo:** Identificador del periodo.

Implementación:

```
public List<Integer> obtenerDistributivoPorCarreraId(Integer
idCarrera, Integer idNivel, Integer idMalla,Integer idPeriodo)
{
final StringBuilder hql = new StringBuilder();
      hql.append("SELECT DISTINCT  d.grupo.idGrupo FROM
      Distributivo d ");
hql.append("WHERE d.docenteCarrera.id.idCarrera = :idCarrera
");
hql.append("AND d.materia.nivel.idNivel = :idNivel ");
hql.append("AND d.materia.malla.id.idMalla = :idMalla ");
hql.append("AND d.periodo.idPeriodo = :idPeriodo ");
hql.append("AND d.estado = :activo ");
Query query = em.createQuery(hql.toString());
query.setParameter("idCarrera", idCarrera);
query.setParameter("idNivel", idNivel);
query.setParameter("idMalla", idMalla);
query.setParameter("idPeriodo", idPeriodo);
      query.setParameter("activo",
      ConstantesScheduleCore.ESTADO_ACTIVO);
return query.getResultList();
}
```

- ✓ **Obtener materias asignadas:** Su objetivo es conseguir el número de materias asignadas a una carrera, esta información se encuentra filtrada por el nivel, la malla y el grupo específico.

Código Fuente:

```
Integer obtenerNumeroMateriaAsignadas(Integer idCarrera,
Integer idNivel, Integer idGrupo, Integer idMalla, Integer
idPeriodo, Boolean esCuentaMateria);
```

Dónde:

- ✓ **idCarrera:** Identificador de la carrera.
- ✓ **idNivel:** Identificador del nivel.
- ✓ **idGrupo:** Identificar del grupo.
- ✓ **idMalla:** Identificador de la malla.
- ✓ **idPeriodo:** Identificador del periodo.
- ✓ **esCuentaMateria:** Valor booleano.

Implementación:

```
public Integer obtenerNumeroMateriaAsignadas(Integer
idCarrera, Integer idNivel, Integer idGrupo, Integer
idMalla,Integer idPeriodo, Boolean esCuentaMateria) {
      Query query =
      em.createQuery(crearHqlContarDistributivo(esCuentaMateri
      a, idGrupo).toString());
query.setParameter("idCarrera", idCarrera);
query.setParameter("idNivel", idNivel);
if (idGrupo != null)
      query.setParameter("idGrupo", idGrupo);
query.setParameter("idMalla", idMalla);
query.setParameter("idPeriodo", idPeriodo);
      query.setParameter("activo",
      ConstantesScheduleCore.ESTADO_ACTIVO);
return ((Long) query.getSingleResult()).intValue();
}
```

CarreraDao: Asociado a la tabla tbl_carrera de la base de datos, su principal funcione es:

- ✓ **Obtener carrera por director:** La cual obtiene una carrera específica en función de un usuario (Director de Carrera) que ingresa al sistema.

Código fuente:

```
Carrera obtenerCarreraPorDirector(Integer idDirector);
```

Dónde:

- ✓ idDirector: Identificador del director.

Implementación:

```
public Carrera obtenerCarreraPorDirector(Integer  idDirector)
{
	List<Carrera> listaCarrera;
	Carrera carrera;
	Query query =
	em.createNamedQuery("Carrera.obtenerCarreraPorDirector")
	;
	query.setParameter("idDirector", idDirector);
	listaCarrera = query.getResultList();
	if (listaCarrera.isEmpty()) {
		carrera = null;
	} else {
		carrera = listaCarrera.iterator().next();
	}
	return carrera;
}
```

DocenteCarreraDao: Se encuentra ligado a la tabla tbl_docente_carrera de la base de datos, su método principal es:

- ✓ **Obtener docentes por carrera:** Consigue una lista de los diferentes docentes de la Universidad Politécnica Salesiana, información que se encuentra filtrada por la carrera, la cédula y nombre del docente.

Código fuente:

```
List<DocenteCarrera>          obtenerDocentesPorCarrera(Integer
idCarrera,    String    cedula,    String    nombre,    Integer
estadoActivo);
```

Dónde:

- ✓ **idCarrera:** Identificador de la carrera.
- ✓ **cedula:** Cédula del docente.
- ✓ **nombre:** Nombre del docente.
- ✓ **estadoActivo:** Estado del docente.

Implementación:

```
public List<DocenteCarrera> obtenerDocentesPorCarrera(Integer
idCarrera, String cedula, String nombre, Integer estadoActivo)
{
	cedula =
	UtilSchedule.getInstancia().validarParametroString(cedul
	a);
	nombre =
	UtilSchedule.getInstancia().validarParametroString(nombr
	e);
	Query query =
	em.createQuery(crearConsultaDocentePorCarreraFiltro(cedu
	la, nombre).toString());
	query.setParameter("idCarrera", idCarrera);
	query.setParameter("activo", estadoActivo);
	if (null != cedula)
		query.setParameter("cedulaDocente", cedula + "%");

	if (null != nombre)
		query.setParameter("nombreDocente",
		UtilSchedule.getInstancia().likeCompare(nombre));
	return (ArrayList<DocenteCarrera>)
	query.getResultList();
}
```

2.4 Integración

Como se había mencionado en capítulos anteriores, al módulo de horarios es necesario integrarlo al módulo de distributivos, esta decisión fue tomada debido a que los dos grupos que conformaban el proyecto "UPS Schedule" tenían una similitud en sus aplicaciones y era el tema de los usuarios que utilizarían los diferentes módulos, llegando a la conclusión que los Directores de Carrera serían los únicos en utilizar este sistema ya unificado.

Entre los principales inconvenientes presentados al momento de la integración de los dos módulos era la utilización de diferentes IDE de desarrollo, motivo por el cual se decide migrar los archivos de desarrollo de NetBeans (módulo de horarios) a Eclipse (módulo de distributivos) por la principal razón de que Eclipse facilitó la detección de errores al momento de la unificación de los aplicativos, así como también permitió un mejor control de flujo del sistema.

La siguiente ilustración evidencia la concepción principal para que los módulos de distributivos y horarios puedan interactuar de manera correcta.

2.4.1 Diagrama de integración sistemas distributivos – horarios.

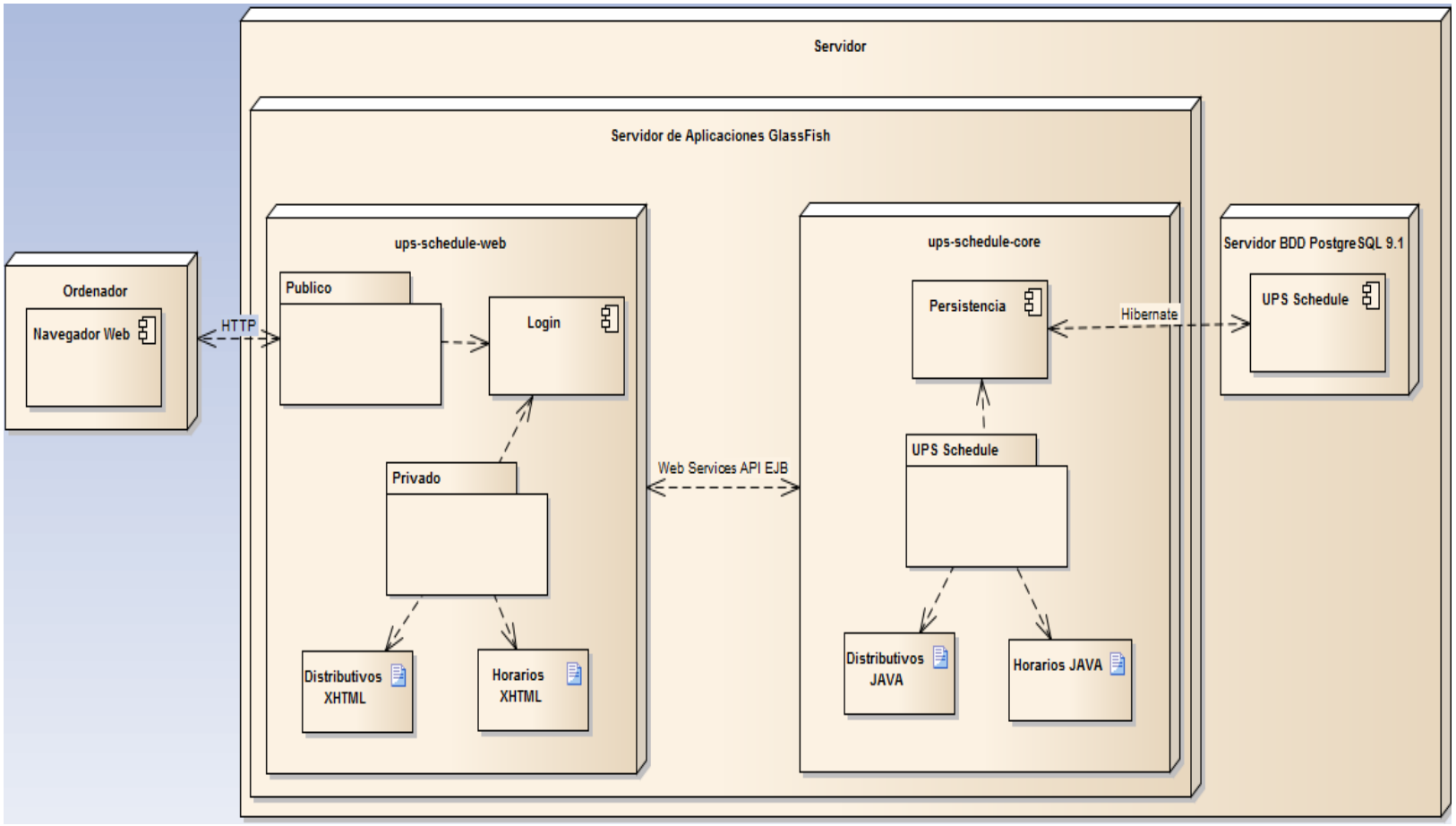


Figura 20. Diagrama de despliegue, integración distributivos - horarios.

Elaborado por: Dennis Reyes y Milton Chiliquinga.

2.4.2 Configuración de servidores para la integración.

Para la ejecución de los módulos se debe manipular tanto el servidor de aplicaciones como el servidor web, a continuación se detalla la configuración realizada al sistema para la correcta integración.

La siguiente imagen muestra la estructura del pool de direcciones que se ha creado para la conexión hacia la base de datos, en donde se instancia información como el nombre de la base de datos, el usuario y contraseña de conexión, así como también el driver que se utilizará y el puerto en el que se encuentra escuchando.

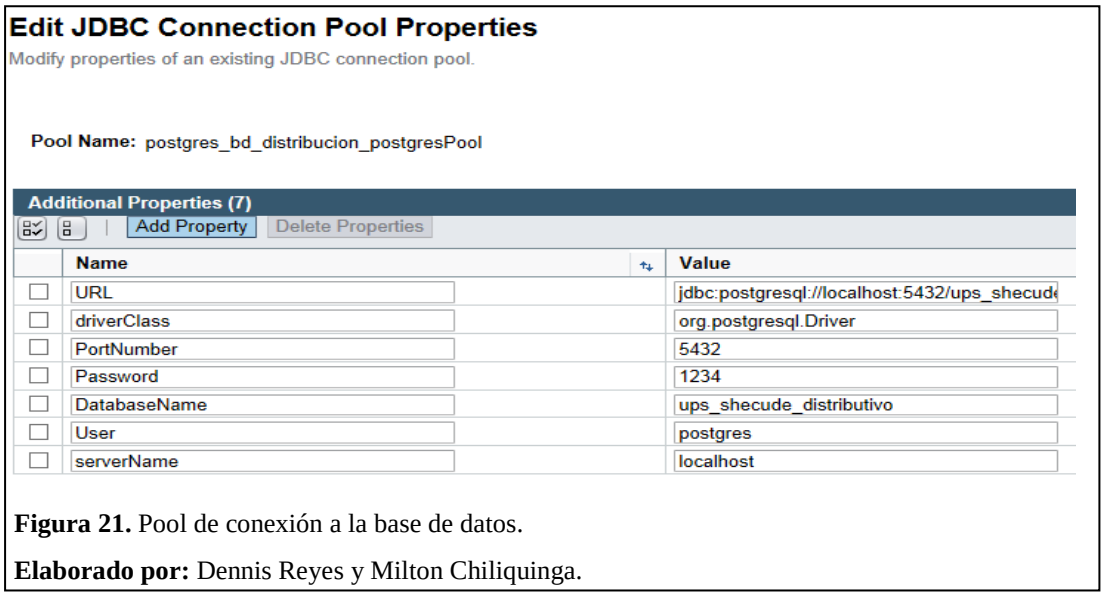


Figura 21. Pool de conexión a la base de datos.

Elaborado por: Dennis Reyes y Milton Chiliquinga.

Una vez realizada lo anteriormente expuesto, se puede verificar realizando un ping hacia la base de datos.

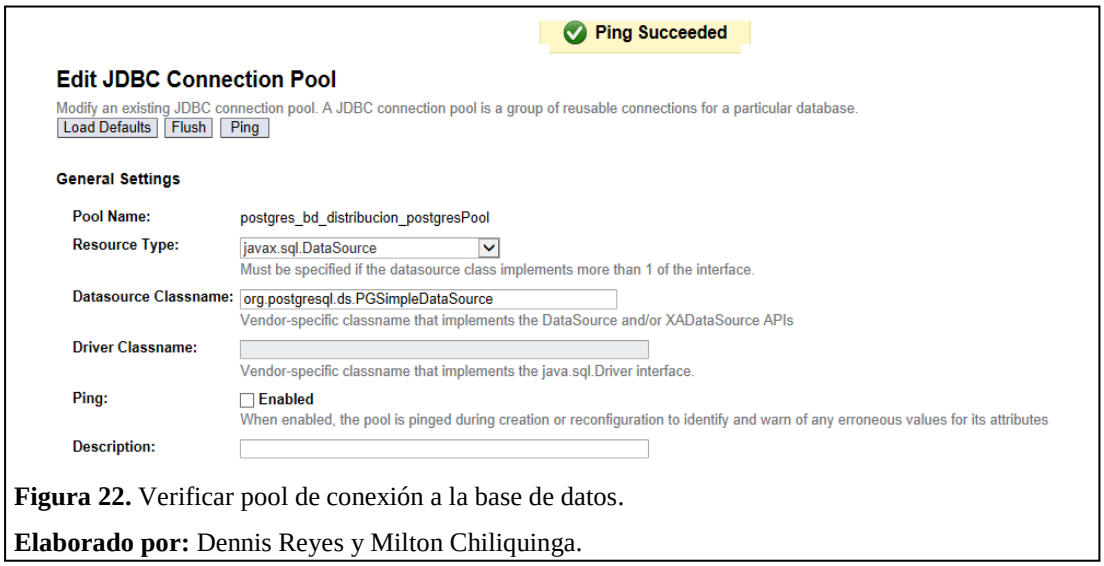


Figura 22. Verificar pool de conexión a la base de datos.

Elaborado por: Dennis Reyes y Milton Chiliquinga.

Uno de los aspectos a destacar en el desarrollo del módulo de distributivos es el manejo de JAAS para el tema de autenticación de usuarios, donde se especifica los campos que serán validados como son el username y password con su correspondiente algoritmo de encriptación MD5.

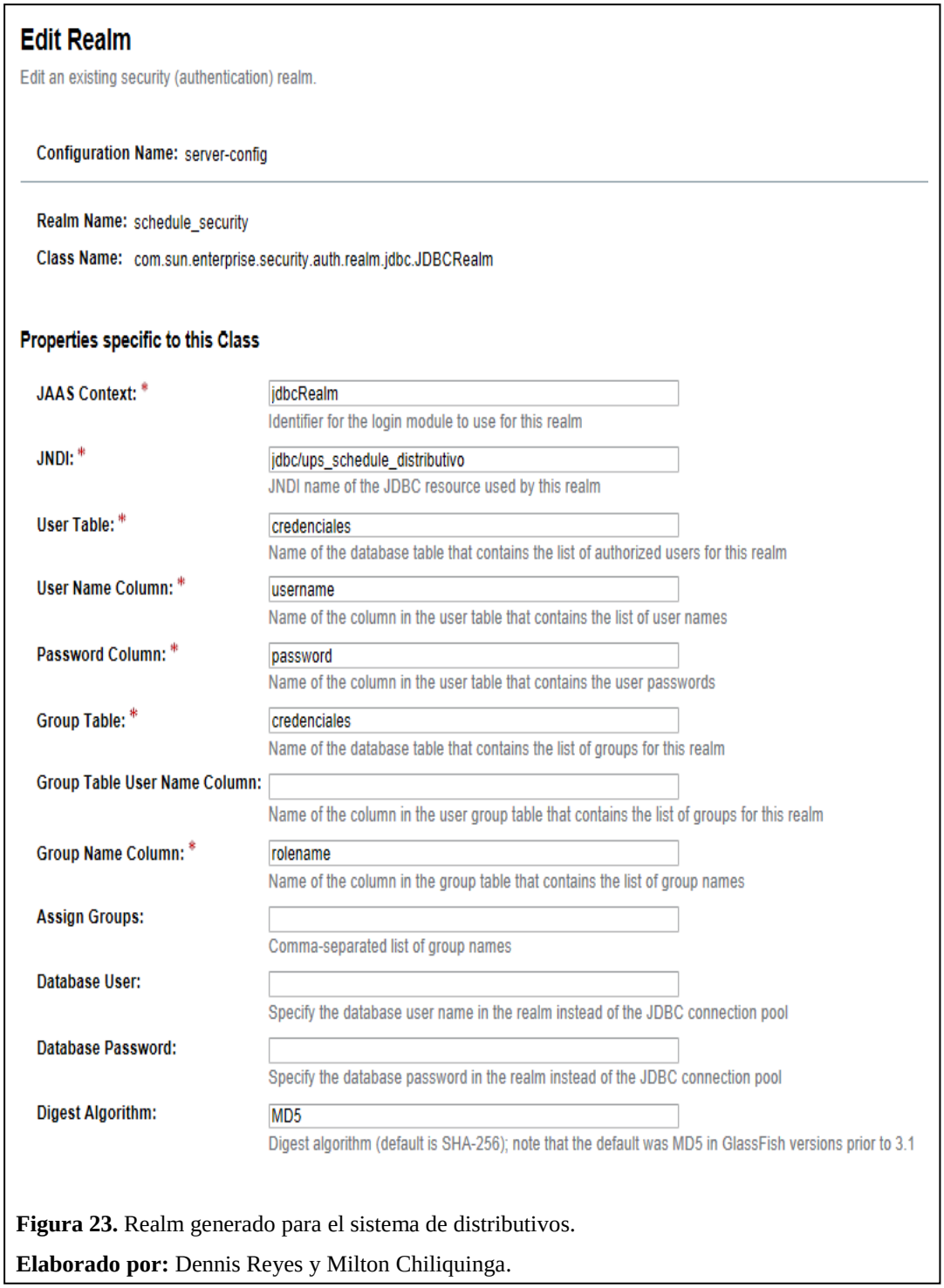


Figura 23. Realm generado para el sistema de distributivos.

Elaborado por: Dennis Reyes y Milton Chiliquinga.

2.4.3 Configuración de la aplicación web.

En el archivo web.xml se ha realizado las siguientes configuraciones para la aplicación web:

Tiempo límite de sesión

A continuación se muestra la estructura de la sesión, en donde si el usuario que ha ingresado al sistema no realiza operación alguna durante 30 min, la sesión caduca.

```
<session-config>
      <session-timeout>30</session-timeout>
</session-config>
```

Páginas de error

- ✓ Error 403, página de error cuando el usuario no tiene acceso hacia una página específica.

```
<error-page>
<error-code>403</error-code>
<location>/paginas/error/403.jsf</location>
</error-page>
```

- ✓ Error 404, se ha creado una página de error cuando no encuentra una URL a la que un usuario desea acceder.

```
<error-page>
<error-code>404</error-code>
<location>/paginas/error/404.jsf</location>
</error-page>
```

- ✓ Error 500, página generada cuando se ha producido un error interno en el servidor que le impidió completar la solicitud del cliente.

```
<error-page>
<error-code>500</error-code>
<location>/paginas/error/500.jsf</location>
</error-page>
```

Seguridades

Como se había mencionado anteriormente el módulo de distributivos utiliza Java authentication and authorization service (JAAS), a continuación se presenta la estructura del mismo.

En las siguientes líneas de código se especifica que el usuario ingresará al sistema mediante un formulario y es aquí donde es utilizado el realm creado en el servidor de aplicaciones “shedule_security”

```
<login-config>
      <auth-method>FORM</auth-method>
      <realm-name>schedule_security</realm-name>
      <form-login-config>
            <form-login-page>/paginas/publicas/login.jsf</form-
            login-page>
            <form-error-page>/paginas/error/errorLogin.jsf</form-
            error-page>
      </form-login-config>
</login-config>
```

Una vez definido el realm a utilizar, se especifica el rol para la aplicación en este caso serán los Directores de Carrera, el siguiente fragmento de código indica el proceso realizado.

```
<security-role>
      <description>Director de carrera</description>
      <role-name>directorAdmin</role-name>
</security-role>
```

Para finalizar, se definen todos los archivos a los cuales el usuario registrado tendrá acceso dentro aplicativo, en este caso se configura el acceso a todas las páginas con extensión “.jsf” con sus respectivos métodos “GET” y “POST” para el manejo de la información.

```
<security-constraint>
      <web-resource-collection>
            <web-resource-name>director-area</web-resource-name>
            <url-pattern>*.jsf</url-pattern>
            <http-method>POST</http-method>
            <http-method>GET</http-method>
      </web-resource-collection>
</security-constraint>
```

2.4.4 Configuración de la persistencia.

En el archivo de configuración persistence.xml se ha realizado las siguientes configuraciones principales.

Sistema gestor de base de datos

El fragmento de código especifica el dialecto que se utilizará para la base de datos, en este caso el sistema ha sido desarrollado en PostgreSQL.

```
<properties>
      <property name="hibernate.dialect"
      value="org.hibernate.dialect.PostgreSQLDialect" />
      <property name="hibernate.show_sql" value="true" />
      <property name="hibernate.format_sql" value="true" />
</properties>
```

Mapa de entidades

La siguiente configuración permite relacionar las tablas de la base de datos con las diferentes clases entidad, a continuación el código del mapa de entidades.

```
<class>ec.edu.ups.schedule.entidades.Accion</class>
<class>ec.edu.ups.schedule.entidades.Auditoria</class>
<class>ec.edu.ups.schedule.entidades.Carrera</class>
<class>ec.edu.ups.schedule.entidades.Dia</class>
<class>ec.edu.ups.schedule.entidades.Disponibilidad</class>
<class>ec.edu.ups.schedule.entidades.Distributivo</class>
<class>ec.edu.ups.schedule.entidades.DocenteCarrera</class>
<class>ec.edu.ups.schedule.entidades.EstadoHorario</class>
<class>ec.edu.ups.schedule.entidades.Grupo</class>
<class>ec.edu.ups.schedule.entidades.Hora</class>
<class>ec.edu.ups.schedule.entidades.Horario</class>
<class>ec.edu.ups.schedule.entidades.Malla</class>
<class>ec.edu.ups.schedule.entidades.Materia</class>
<class>ec.edu.ups.schedule.entidades.Modulo</class>
<class>ec.edu.ups.schedule.entidades.Nivel</class>
<class>ec.edu.ups.schedule.entidades.Opcion</class>
<class>ec.edu.ups.schedule.entidades.Paracademico</class>
<class>ec.edu.ups.schedule.entidades.Parametro</class>
<class>ec.edu.ups.schedule.entidades.Perfil</class>
<class>ec.edu.ups.schedule.entidades.PerfilOpcion</class>
<class>ec.edu.ups.schedule.entidades.Periodo</class>
<class>ec.edu.ups.schedule.entidades.UserCredential</class>
<class>ec.edu.ups.schedule.entidades.Usuario</class>
<class>ec.edu.ups.schedule.entidades.pk.DocenteCarreraPK </class>
<class>ec.edu.ups.schedule.entidades.pk.HorarioPK</class>
<class>ec.edu.ups.schedule.entidades.pk.MallaPK</class>
```

CAPÍTULO 3

PRUEBAS

3.1 Funcionalidad

Las pruebas de funcionalidad tienen como objetivo principal mostrar los resultados de la aplicación a través de la manipulación del mismo, identificando errores generados e identificar las soluciones a dichos errores.

Tabla 21. Pruebas de funcionalidad

No	Nombre de la prueba	Req.	Pasos	OK	Observaciones
1	Ingresar al sistema	1.1	1. Acceder a la página 2. Ingresar el usuario 3. Ingresar contraseña	SÍ	
2	Generar distributivo a una carrera	1.2	1. Acceder al menú "Procesos" 2. Seleccionar el ítem "Distributivos" 3. Seleccionar el ítem "Generar" 4. Escoger la malla 5. Escoger el nivel 6. Seleccionar materia a asignar docente 7. Busca el docente 8. Asignar docente a dicha materia	SÍ	
3	Quitar la asignación de una materia hacia un docente	1.2	1. Acceder al menú "Procesos" 2. Seleccionar el ítem "Distributivos" 3. Seleccionar el ítem "Generar" 4. Escoger la malla 5. Escoger el nivel	SÍ	

(Continuación)

Tabla 21. Pruebas de funcionalidad

			6. Seleccionar materia que se desea quitar la asignación 7. Presionar en “Quitar asignación”		
4	Eliminar una materia de un grupo específico de una carrera	1.2	1. Acceder al menú “Procesos” 2. Seleccionar el ítem “Distributivos” 3. Seleccionar el ítem “Generar” 4. Escoger la malla 5. Escoger el nivel 6. Seleccionar materia que se desea quitar la asignación 7. Presionar en “Eliminar materia”	SÍ	
5	Crear un grupo con sus respectivas materias dentro de una carrera	1.2	1. Acceder al menú “Procesos” 2. Seleccionar el ítem “Distributivos” 3. Seleccionar el ítem “Generar” 4. Escoger la malla 5. Escoger el nivel 6. Seleccionar el grupo 7. Presionar en el botón “Crear Grupo”	SÍ	
6	Eliminar un grupo con sus respectivas materias dentro de una carrera	1.2	1. Acceder al menú “Procesos” 2. Seleccionar el ítem “Distributivos” 3. Seleccionar el ítem “Generar” 4. Escoger la malla 5. Escoger el nivel 6. Presionar en “Eliminar Grupo”	SÍ	

(Continuación)

Tabla 21. Pruebas de funcionalidad

7	Asignar docentes a una carrera	1.3	1. Acceder al menú “Procesos” 2. Seleccionar el ítem “Docentes” 3. Seleccionar el ítem “Asignar a la carrera” 4. Buscar docente por cédula o nombre 5. Presionar en asignar el docente a dicha carrera 6. Confirmar asignación de dicho docente	SÍ	
8	Desasignar docentes a la carrera	1.4	1. Acceder al menú “Procesos” 2. Seleccionar el ítem “Docentes” 3. Seleccionar el ítem “Desasignar de la carrera” 4. Buscar docente por cédula o nombre 5. Presionar en desasignar el docente de dicha carrera 6. Confirmar la des asignación de dicho docente	SÍ	
9	Modificar un grupo asignado a la carrera	1.5	1. Acceder al menú “Procesos” 2. Seleccionar el ítem “Grupos” 3. Seleccionar el ítem “Nuevo” 4. Buscar nombre del grupo 4. Seleccionar editar dicho grupo 5. Confirmar cambios	SÍ	

(Continuación)

Tabla 21. Pruebas de funcionalidad

10	Crear un grupo a una carrera	1.5	1. Acceder al menú “Procesos” 2. Seleccionar el ítem “Grupos” 3. Seleccionar el ítem “Nuevo” 4. Buscar nombre del grupo 5. Pinchar el botón “Nuevo” 6. Ingresar nombre del nuevo grupo 7. Guardar grupo	SÍ	
11	Generar reporte del estado de los distributivos de una carrera	1.6	1. Acceder al menú “Reportes” 2. Seleccionar el ítem “Distributivos” 3. Seleccionar el ítem “Asignaciones” 4. Seleccionar la malla 5. Seleccionar el nivel 6. Seleccionar el grupo 7. Presionar en el botón “Generar Reporte ”	SÍ	
12	Exportar reporte del estado de los distributivos de una carrera	1.6	1. Acceder al menú “Reportes” 2. Seleccionar el ítem “Distributivos” 3. Seleccionar el ítem “Asignaciones” 4. Seleccionar la malla 5. Seleccionar el nivel 6. Seleccionar el grupo 7. Presionar en el botón “Generar Reporte ” 8. Presionar en la imagen “PDF”	SÍ	

(Continuación)

Tabla 21. Pruebas de funcionalidad

13	Generar reporte de las materias asignadas a un docente de una carrera	1.7	1. Acceder al menú “Reportes” 2. Seleccionar el ítem “Carga Horarios” 3. Buscar docente por cédula o por nombre 4. Presionar en el botón “Buscar”	SÍ	
14	Exportar reporte de las materias asignadas a un docente de una carrera	1.7	1. Acceder al menú “Reportes” 2. Seleccionar el ítem “Carga Horarios” 3. Buscar docente por cédula o por nombre 4. Presionar en el botón “Buscar” 5. Presionar en la imagen “PDF”	SÍ	

Elaborado por: Dennis Reyes y Milton Chiliquinga.

3.2 Usabilidad

El propósito de este tipo de prueba es mostrar los posibles problemas de usabilidad que posee el sistema, sugerir las respectivas recomendaciones para superar los inconvenientes encontrados, optimizando la interfaz y de esta manera mejorar la experiencia de navegación del usuario/cliente en el aplicativo.

Para llevar a cabo las correspondientes pruebas de usabilidad se ha seleccionado 10 usuarios al azar, los cuales manipularon el sistema de distributivos y al realizar el levantamiento de la información se obtuvo los siguientes resultados.

1. **¿Considera usted que el tiempo requerido para concluir con la actividad de generar los distributivos es el adecuado?**

    - Completamente satisfecho
    - Satisfecho
    - Insatisfecho

- Completamente insatisfecho

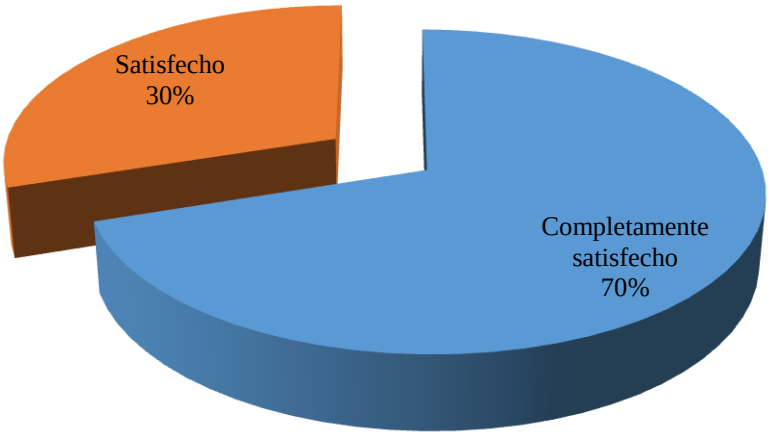


Figura 24. Pregunta 1.
Elaborado por: Dennis Reyes y Milton Chiliquinga.

2. Tras utilizar el software de distributivos, valore su grado de satisfacción en el aspecto da facilidad de uso / navegación.

- Completamente satisfecho
- Satisfecho
- Insatisfecho
- Completamente insatisfecho

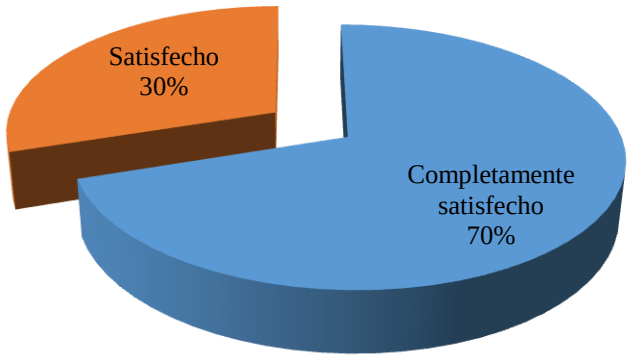


Figura 25. Pregunta 2.
Elaborado por: Dennis Reyes y Milton Chiliquinga.

3. **Por favor, puntúe su grado de acuerdo o desacuerdo acerca de la comprensión de la finalidad del sistema de distributivos.**

- Totalmente de acuerdo
- De acuerdo
- En desacuerdo
- Totalmente en desacuerdo.

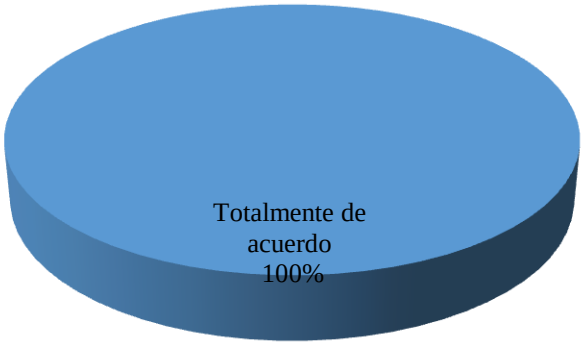


Figura 26. Pregunta 3.
Elaborado por: Dennis Reyes y Milton Chiliquinga.

4. **¿Los reportes generados por el sistema brinda información necesaria y relevante?**

- Totalmente de acuerdo
- De acuerdo
- En desacuerdo
- Totalmente en desacuerdo.

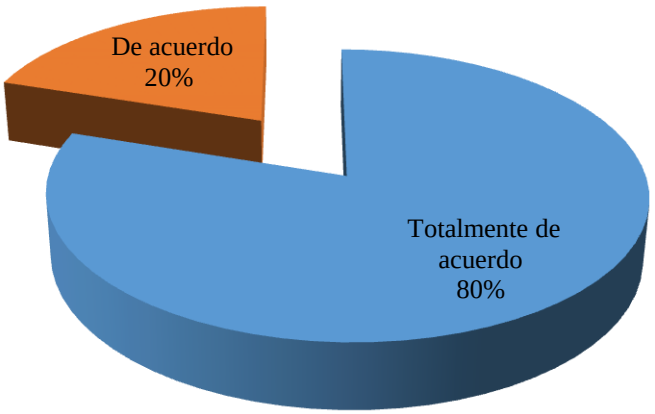


Figura 27. Pregunta 4.
Elaborado por: Dennis Reyes y Milton Chiliquinga.

5. Ha tenido algún tipo de error / problema al momento de manipular las páginas del sistema de distributivos.

- SÍ
- NO

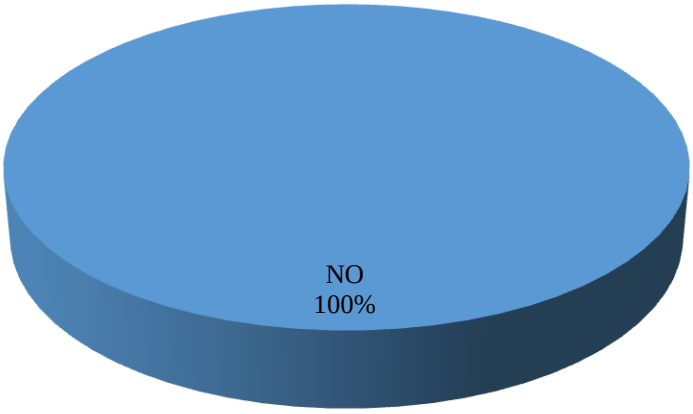


Figura 28. Pregunta 5.
Elaborado por: Dennis Reyes y Milton Chiliquinga.

6. **¿Tiene usted algún comentario adicional o sugerencia para ayudarnos a mejorar?**

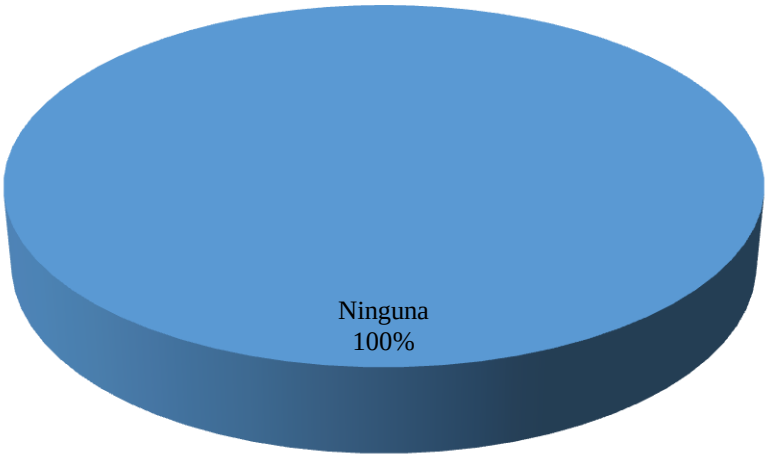


Figura 29. Pregunta 6.
Elaborado por: Dennis Reyes y Milton Chiliquinga.

Para el levantamiento de información se utilizó como recurso principal a la encuesta, de esta manera se observa que el sistema de distributivos dispone de una interfaz amigable, fácil de usar y que cumple con los requerimientos principales para la generación de distributivos.

3.3 Carga

Las pruebas de carga están diseñadas para verificar el comportamiento del sistema frente a situaciones extremas o anormales, consiste en hacer que el sistema demande recursos en alto volumen y a gran frecuencia.

Para ello se ha configurado la herramienta de generación de carga JMeter versión 2.11, implementada por Java que realiza un test del rendimiento y pruebas de estrés, monitoreando la página web http://localhost:8080/ups-schedule-web/paginas/privadas/principal.jsf del sistema de distributivos.

Para realizar las pruebas mencionadas, se ha configurado un servidor proxy que provee las herramientas Open Source y con ello simular la visita de un usuario al sistema.

El equipo utilizado como servidor dispone de las siguientes características:

- ✓ Intel Coreo i3, 4GB DDR3 de memoria RAM
- ✓ Sistema Operativo Windows 7 64 bits

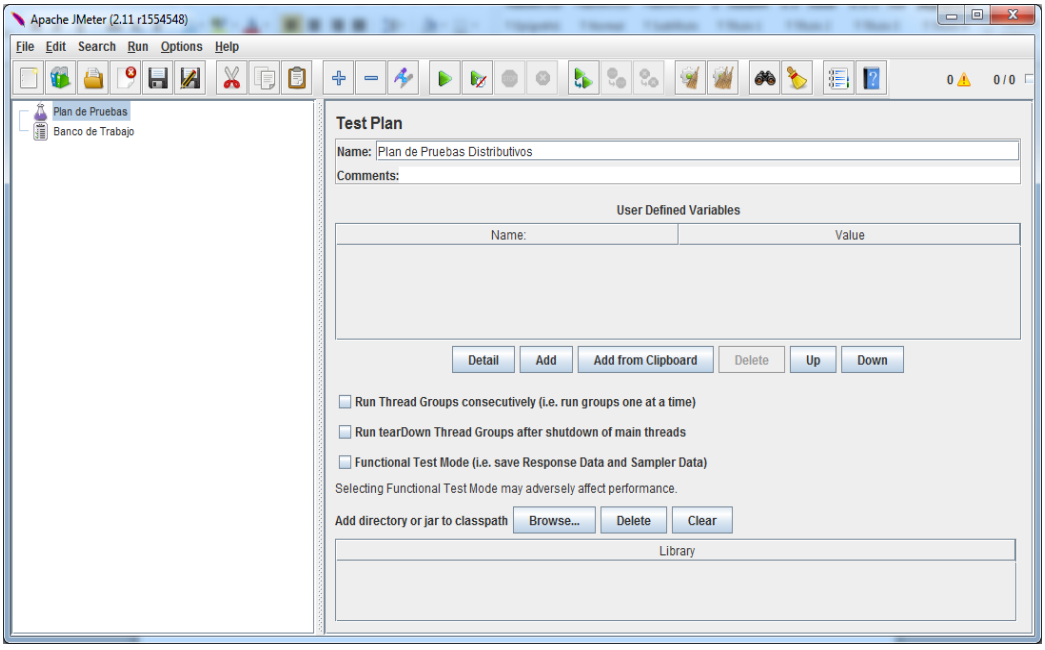


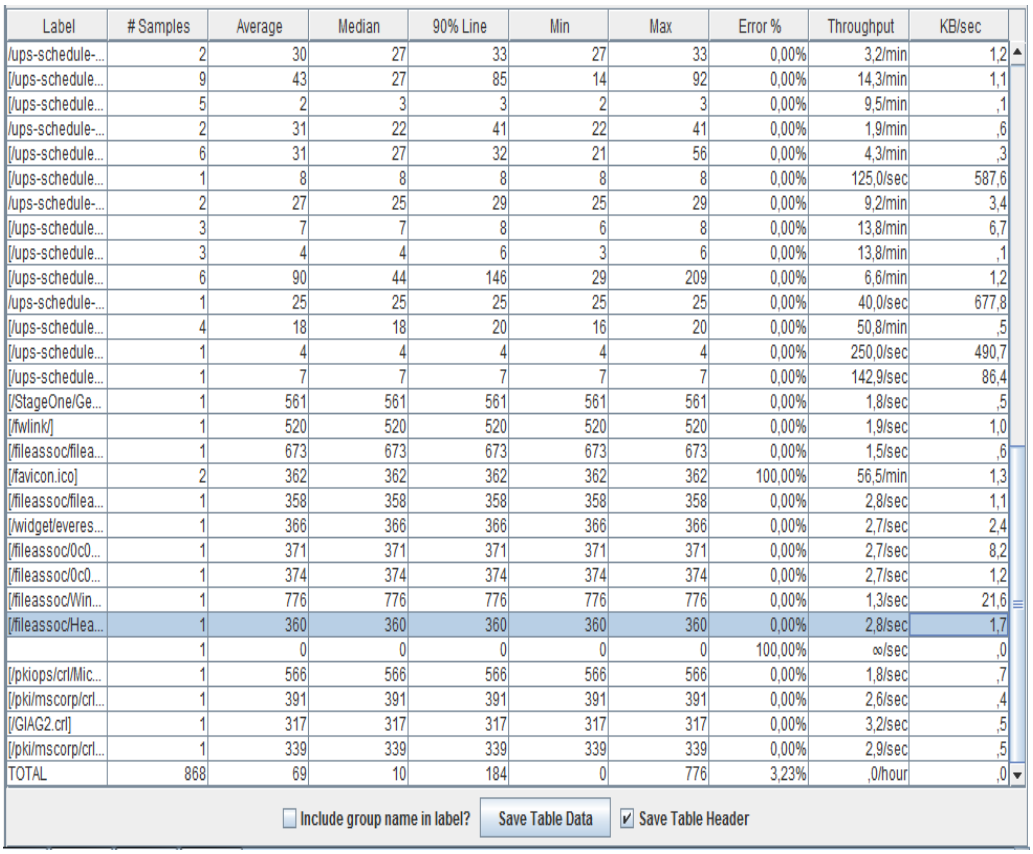
Figura 30. Plan de pruebas JMeter.

Elaborado por: Dennis Reyes y Milton Chiliquinga.

3.3.1 Análisis.

Cabe mencionar que el sistema de distributivos se encuentra dirigido hacia los Directores de Carrera, motivo por el cual, se ha decidido realizar una carga acercándonos a la realidad utilizando threads frecuente y simultánea de 20 usuarios.

Con los datos configurados previamente y mediante la ayuda del listener "Aggregate Report" o "Resultados Detallados" se ha obtenido la siguiente información.



Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/ups-schedule-...	2	30	27	33	27	33	0,00%	3,2/min	1,2
[/ups-schedule...	9	43	27	85	14	92	0,00%	14,3/min	1,1
[/ups-schedule...	5	2	3	3	2	3	0,00%	9,5/min	,1
/ups-schedule-...	2	31	22	41	22	41	0,00%	1,9/min	,6
[/ups-schedule...	6	31	27	32	21	56	0,00%	4,3/min	,3
[/ups-schedule...	1	8	8	8	8	8	0,00%	125,0/sec	587,6
/ups-schedule-...	2	27	25	29	25	29	0,00%	9,2/min	3,4
[/ups-schedule...	3	7	7	8	6	8	0,00%	13,8/min	6,7
[/ups-schedule...	3	4	4	6	3	6	0,00%	13,8/min	,1
[/ups-schedule...	6	90	44	146	29	209	0,00%	6,6/min	1,2
/ups-schedule-...	1	25	25	25	25	25	0,00%	40,0/sec	677,8
[/ups-schedule...	4	18	18	20	16	20	0,00%	50,8/min	,5
[/ups-schedule...	1	4	4	4	4	4	0,00%	250,0/sec	490,7
[/ups-schedule...	1	7	7	7	7	7	0,00%	142,9/sec	86,4
[/StageOne/Ge...	1	561	561	561	561	561	0,00%	1,8/sec	,5
[/fwlink/]	1	520	520	520	520	520	0,00%	1,9/sec	1,0
[/fileassoc/filea...	1	673	673	673	673	673	0,00%	1,5/sec	,6
[/favicon.ico]	2	362	362	362	362	362	100,00%	56,5/min	1,3
[/fileassoc/filea...	1	358	358	358	358	358	0,00%	2,8/sec	1,1
[/widget/everes...	1	366	366	366	366	366	0,00%	2,7/sec	2,4
[/fileassoc/0c0...	1	371	371	371	371	371	0,00%	2,7/sec	8,2
[/fileassoc/0c0...	1	374	374	374	374	374	0,00%	2,7/sec	1,2
[/fileassoc/Win...	1	776	776	776	776	776	0,00%	1,3/sec	21,6
[/fileassoc/Hea...	1	360	360	360	360	360	0,00%	2,8/sec	1,7
	1	0	0	0	0	0	100,00%	∞/sec	,0
[/pkiops/crl/Mic...	1	566	566	566	566	566	0,00%	1,8/sec	,7
[/pki/mscorp/crl...	1	391	391	391	391	391	0,00%	2,6/sec	,4
[/GIAG2.crl]	1	317	317	317	317	317	0,00%	3,2/sec	,5
[/pki/mscorp/crl...	1	339	339	339	339	339	0,00%	2,9/sec	,5
TOTAL	868	69	10	184	0	776	3,23%	,0/hour	,0

Figura 31. Resultado plan de pruebas JMeter.

Elaborado por: Dennis Reyes y Milton Chiliquinga.

Tabla 22. Resultados obtenidos, threads 20

Label	Samples	Average	Median	90% Line	Min	Max	Error %	KB/sec
Total	868	69	10	184	0	776	3,23	0

Elaborado por: Dennis Reyes y Milton Chiliquinga.

Como se puede apreciar, el tiempo promedio para acceder a una página es de 6,9 segundos, realizándose un total de 868 requerimientos al servidor.

Mediante la información proporcionada por esta herramienta se puede concluir que, el tiempo utilizado para los 20 threads es de 868 * 69 = 59892 milisegundos.

3.3.2 Gráfico.

La siguiente ilustración, muestra los resultados obtenidos anteriormente de forma gráfica, en el que se podrá apreciar el promedio de color azul, la mediana de color morado, su respectiva desviación con su distintivo en color rojo y su rendimiento de color verde.

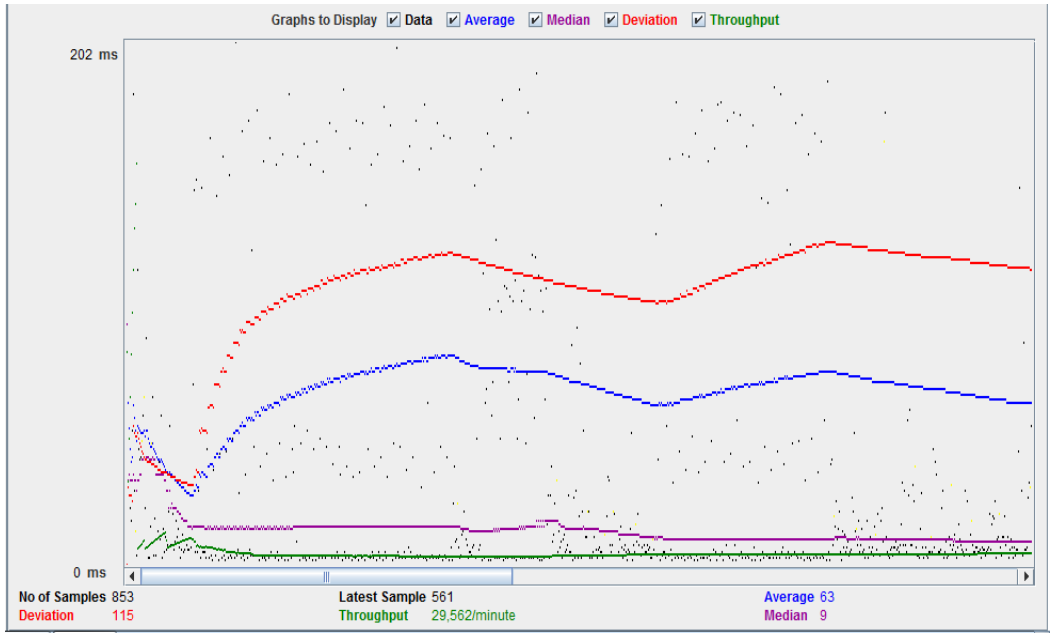


Figura 32. Valores correspondientes a la prueba, threads 20.

Elaborado por: Dennis Reyes y Milton Chiliquinga.

Tabla 23. Resultados obtenidos

# Threads	*#Muestras*	*#Rendimiento*	*Tiempo medio de Respuesta*	*Dispersión*	*%Error*
20	853	29.562/min	69	115	3.23%

Elaborado por: Dennis Reyes y Milton Chiliquinga.

CONCLUSIONES

Finalizando el sistema de gestión de distributivos, con un profundo análisis sobre toda la información obtenida, se puede concluir que:

- ✓ La base de datos seleccionada para el proyecto (PostgreSQL) al ser completamente ACID (Atomicity, Consistency, Isolation and Durability) permite que la información presentada al usuario sea enteramente veraz; por otra parte garantiza que los datos generados por la aplicación (distributivos) sean persistidos en la base, con lo cual se avala la disponibilidad de la información.

- ✓ Si bien no existe un software totalmente libre de fallos, las pruebas garantizan reducir a un mínimo los posibles errores que se puedan suscitar, en ese sentido las pruebas aplicadas al software en cuestión muestran que el propósito con el cual fue creado se cumple a cabalidad. En este punto hay que resaltar la eficiencia en la reducción de trabajo y tiempo que ofrece el aplicativo al generar un distributivo, en comparación a como se lo hace en la actualidad (manualmente).

- ✓ La decisión de trabajar con un mismo IDE (Eclipse Kepler) por parte del grupo de horarios y distributivos, obtuvo una gran ventaja obteniendo que los cambios fueron mínimos al realizar la integración de estos dos módulos y de momento se encuentran operando sin mayor problema.

- ✓ El uso del Modelo Vista Controlador (MVC) en la actualidad es de suma importancia debido a que permite la separación de la parte lógica del negocio con la parte de presentación del aplicativo, posibilitando la reutilización de código, facilitando el desarrollo de la aplicación y su correspondiente mantenimiento.

- ✓ La metodología utilizada en el diseño y la construcción del sistema de gestión de distributivos, permitió llevar a cabo un desarrollo ordenado con requerimientos claros al disponer el diseño navegacional y sus respectivas interfaces abstractas.

- ✓ El motor de persistencia utilizado en el desarrollo de la aplicación (Hibernate) permite facilitar la integración de forma rápida y eficiente de los módulos de Gestión de Horarios y Distributivos respectivamente independiente de la infraestructura de los clientes.

RECOMENDACIONES

- ✓ La principal recomendación es el de delegar la parte de seguridades y conectividad hacia la base de datos al servidor de aplicaciones que en nuestro caso es Glassfish, debido a que permite un mayor control del aplicativo y con ello brindar mayor seguridad a los usuarios que utilizarán el sistema.

- ✓ Se recomienda el uso de JAAS en el desarrollo de aplicaciones debido a su alto nivel de seguridad, al control total que se dispone sobre el sistema y la utilización de grupos para acceder al mismo.

- ✓ Uno de los puntos críticos de una aplicación es su mantenimiento y de ser el caso agregar nuevas funcionalidades, teniendo en cuenta que estos cambios no siempre serán realizados por quienes originalmente iniciaron el proyecto es muy importante seguir un patrón de diseño al momento de construir una aplicación; este aspecto es más valedero cuando varios grupos trabajan dentro de un mismo proyecto.

- ✓ En plena era de la comunicación las notificaciones son una parte importante en las aplicaciones que se construyen hoy en día, lamentablemente este módulo no cuenta con dicha funcionalidad; es por ello que sería muy interesante la implementación de esta funcionalidad a futuro; notificando a través de un e-mail cunado un docente fue asignado a una materia.

LISTA DE REFERENCIAS

- Pressman Roger. (2002). *Ingeniería de Software*, MC. Graw-Hill.
- Pressman Roger. (1997). *Ingeniería del software. Un enfoque práctico*, Madrid, McGraw-Hill / Interamericana de España
- Jacobi Jonas, Fallous Jhon R, *JSF and AJAX. Building Rich Internet Components*.
- Rozanki Uwe. *Enterprise Java Beans 3.0 con eclipse y Jboos*.
- Paul Deitel, Harvey Deitel. *Introducción a la programación orientada a objetos*.
- Luis Joyanes Aguilar; Ignacio Zahonero Martínez. *Programación en C, C++, Java y UML.*
- Antonio Martin Serra. *Programador científico Java.*
- Roldon Martínez David. *Aplicaciones web un enfoque práctico.*
- Antonio Martin Sierra. *Ajax en J2EE.*
- Ceballos Sierra. *Java, Interfaces gráficas y aplicaciones para internet.*
- Danny Ayers & Erik Bruhez. *Programación web 2.0*
- IEEE Recommended Practice for Software Requirements Specifications, 1998
- *Guía para la elaboración de tesis*. Recuperado el 20 de diciembre de 2013, desde, http://www.casalamm.com.mx/guia_tesis.pdf
- Fundación universitaria panamericana investigación, desarrollo e innovación. *Guía para la elaboración y presentación de proyectos de investigación.* Recuperado el 25 de diciembre de 2013, desde, http://exordio.qfb.umich.mx/archivos%20PDF%20de%20trabajo%20UMSNH/tesis/faa8636be38b1deaddf6023bf0f76d48.pdf

MANUAL DE USUARIO UPS SCHEDULE
MÓDULO DISTRIBUTIVO

Elaborado por: Milton Chiliquinga Dennis Reyes	**Revisado por:**
Firmas:	**Firmas:**
Fecha:	**Fecha:**

ÍNDICE

INTRODUCCIÓN

El siguiente módulo está orientado a los Directores de Carrera, quienes podrán realizar las siguientes tareas:

- ✓ Asignar una materia a un docente en particular
- ✓ Agregar materias al distributivo
- ✓ Agregar grupos al distributivo
- ✓ Eliminar la asignación que tiene un docente
- ✓ Eliminar materias del distributivo
- ✓ Eliminar grupos del distributivo
- ✓ Crear grupos
- ✓ Actualizar grupos
- ✓ Crear nuevos docentes
- ✓ Actualizar credenciales de usuario
- ✓ Generar reportes en pantalla y exportarlos en formato pdf

INGRESO AL SISTEMA

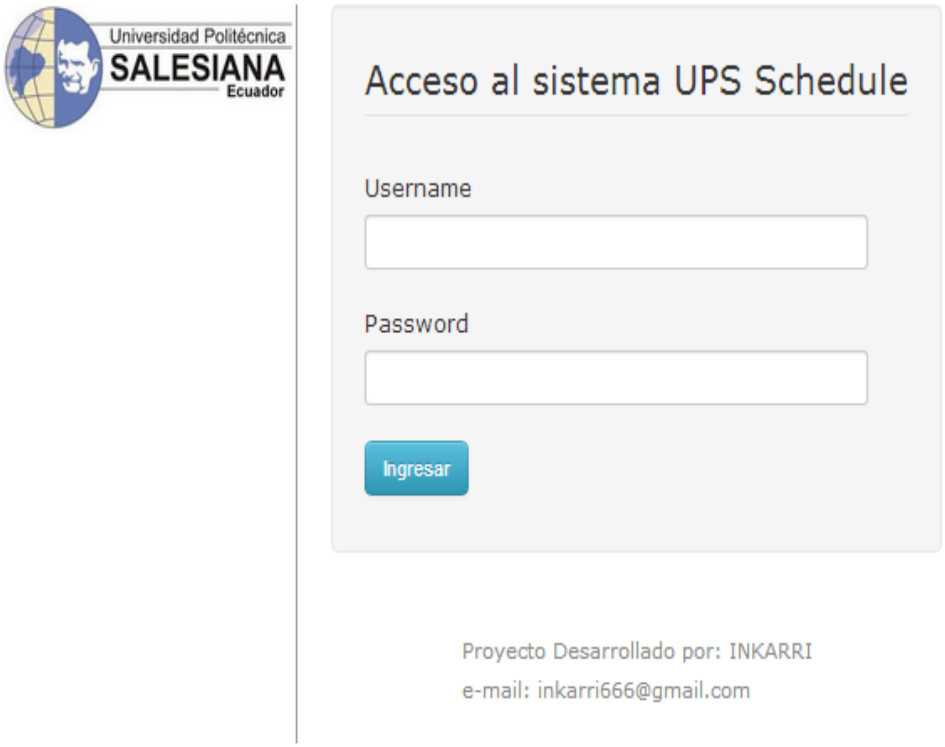


Figura 33. Ingreso al sistema.
Elaborado por: Dennis Reyes y Milton Chiliquinga.

La figura 33, muestra la pantalla de ingreso al sistema. Una vez proporcionado las credenciales de usuario validas, mostrara la siguiente pantalla.

Figura 34. Página inicial.
Elaborado por: Dennis Reyes y Milton Chiliquinga.

Esta página se muestra al iniciar el aplicativo, consta de una cabecera que contiene datos generales como son: carrera, nombre del director, la fecha actual, un link para actualizar el password, un link para salir del aplicativo y finalmente un hipervínculo para regresar a la página inicial. En la parte izquierda se muestra un menú de opciones y en la parte central el área de trabajo, esta área ira variando dependiendo de la opción que seleccione del menú.

OPCIONES DISPONIBLES DEL APLICATIVO

A continuación se detalla las opciones disponibles con las que cuenta el aplicativo.

2.1 Procesos

Figura 35. Menú de opciones "Procesos".
Elaborado por: Dennis Reyes y Milton Chiliquinga.

En esta opción se realizan todos procesos referente al módulo, consta de tres submenús, seguidamente se detalla cada opción disponible en los submenús.

2.1.1 Distributivo.

Figura 36. Menú de opciones "Generar".
Elaborado por: Dennis Reyes y Milton Chiliquinga.

Este submenú cuenta con una única opción que es “Generar”, es la opción más importante del módulo, donde se genera el distributivo. Una vez seleccionada dicha opción muestra la siguiente pantalla.

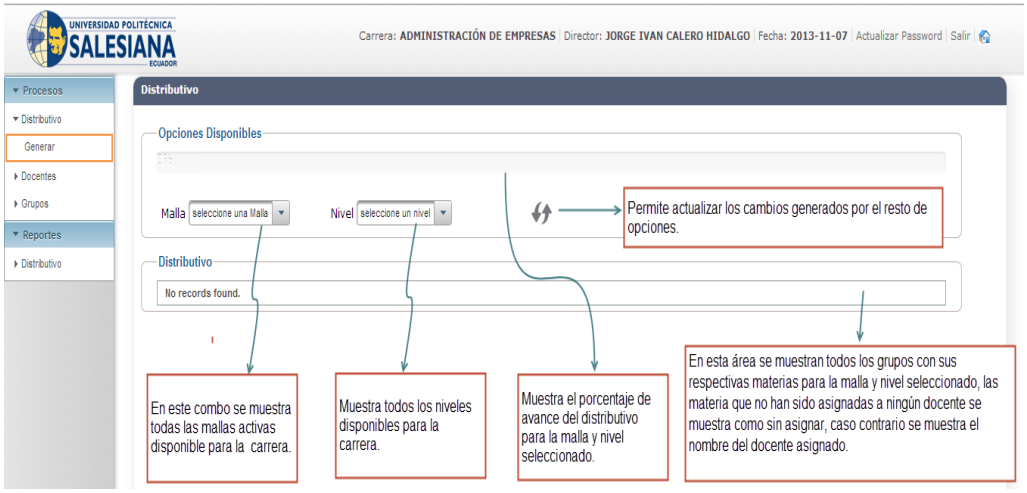


Figura 37. Pantalla para generar el distributivo.
Elaborado por: Dennis Reyes y Milton Chiliquinga.

En la figura anterior se genera el distributivo, para lo cual primero se debe seleccionar una malla y nivel antes de iniciar; una vez que se haya seleccionado una de las opciones disponibles en los combos se muestran todos los grupos con sus respectivas materias asociados al nivel seleccionado, en el caso de que existan grupos disponibles para la carrera pero que aún no han sido asignados al distributivo se mostrara un tercer combo con los grupos disponibles para agregar al distributivo; en la siguiente figura se ilustra mejor lo anterior mencionado.

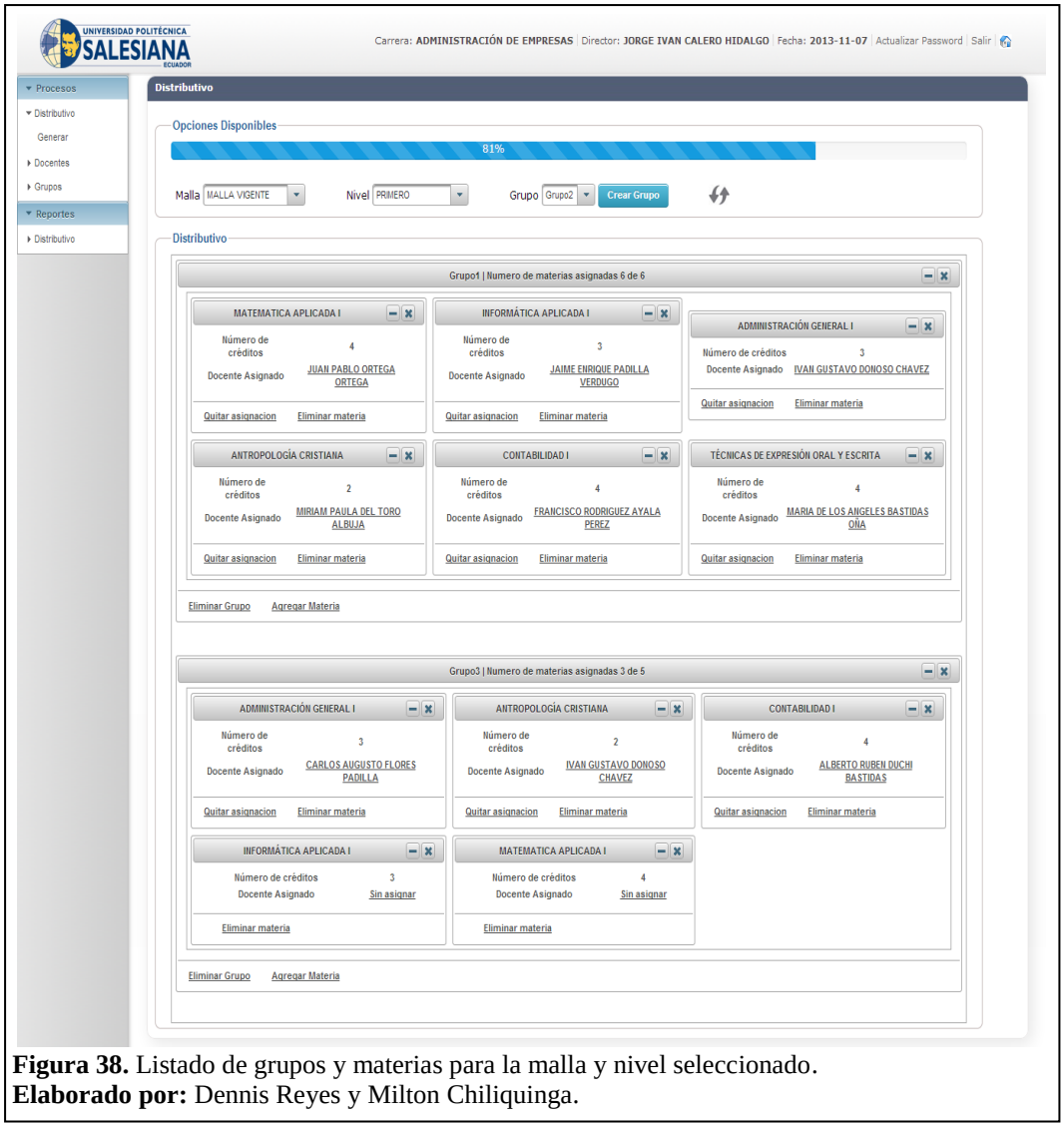


Figura 38. Listado de grupos y materias para la malla y nivel seleccionado.
Elaborado por: Dennis Reyes y Milton Chiliquinga.

La figura 38 muestra el resultado una vez seleccionado la malla y el nivel; para el ejemplo este nivel cuenta con tres grupos de los cual dos se encuentran ya en distributivo, de ser el caso se puede crear el grupo faltante con lo cual se agrega al distributivo el grupo y sus respectivas materias con la particularidad de que el docente asignado siempre será sin asignar.

Nótese que el primer grupo consta con seis materias mientras que el segundo tiene cinco, este es el caso cuando una materia está disponible en un grupo mientras que en los otros debido a la funcionalidad del distributivo.

2.1.2 Eliminar asignación.

A través de esta opción se puede quitar la asignación de un docente a una materia y grupo determinado. Para realizar esta acción clic en el siguiente link Quitar asignacion como muestra la siguiente figura.

Figura 39. Quitar la asignación de un docente.
Elaborado por: Dennis Reyes y Milton Chiliquinga.

Una vez seleccionada la opción el docente es desvinculado de la materia, los resultados se muestran en la siguiente figura.

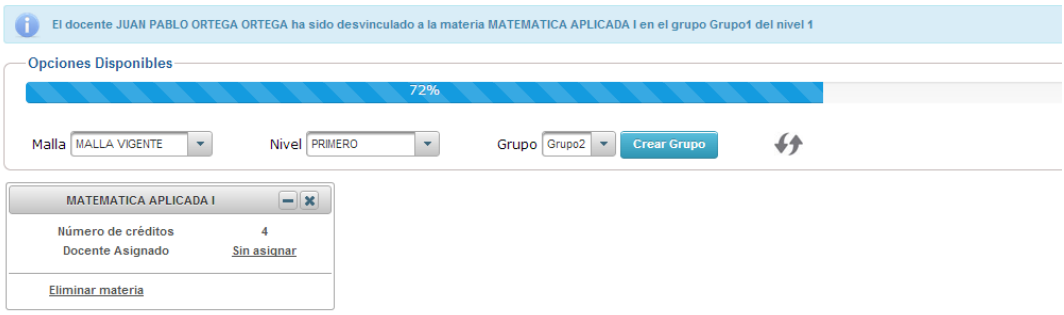


Figura 40. Docente desvinculado de la materia seleccionada.
Elaborado por: Dennis Reyes y Milton Chiliquinga.

Como muestra la imagen el docente fue desvinculado de la materia con lo cual en la parte de asignación ahora se muestra cómo Sin asignar.

2.1.3 Eliminar grupo.

Parte de la funcionalidad del módulo incluye la eliminación de todo un grupo del distributivo, para lo cual se dispone de la siguiente opción Eliminar Grupo una vez seleccionada esta opción se muestra una ventana emergente en la que se pide la confirmación de la acción, como muestra la siguiente figura.

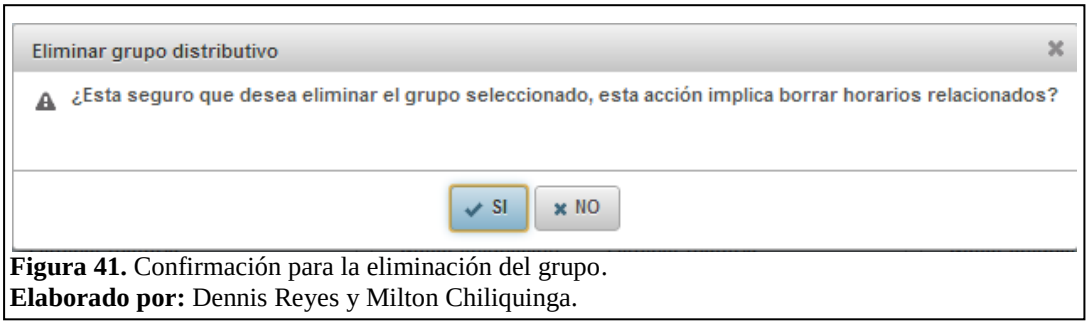


Figura 41. Confirmación para la eliminación del grupo.
Elaborado por: Dennis Reyes y Milton Chiliquinga.

Una vez aceptada la confirmación todo el grupo es eliminado del distributivo.

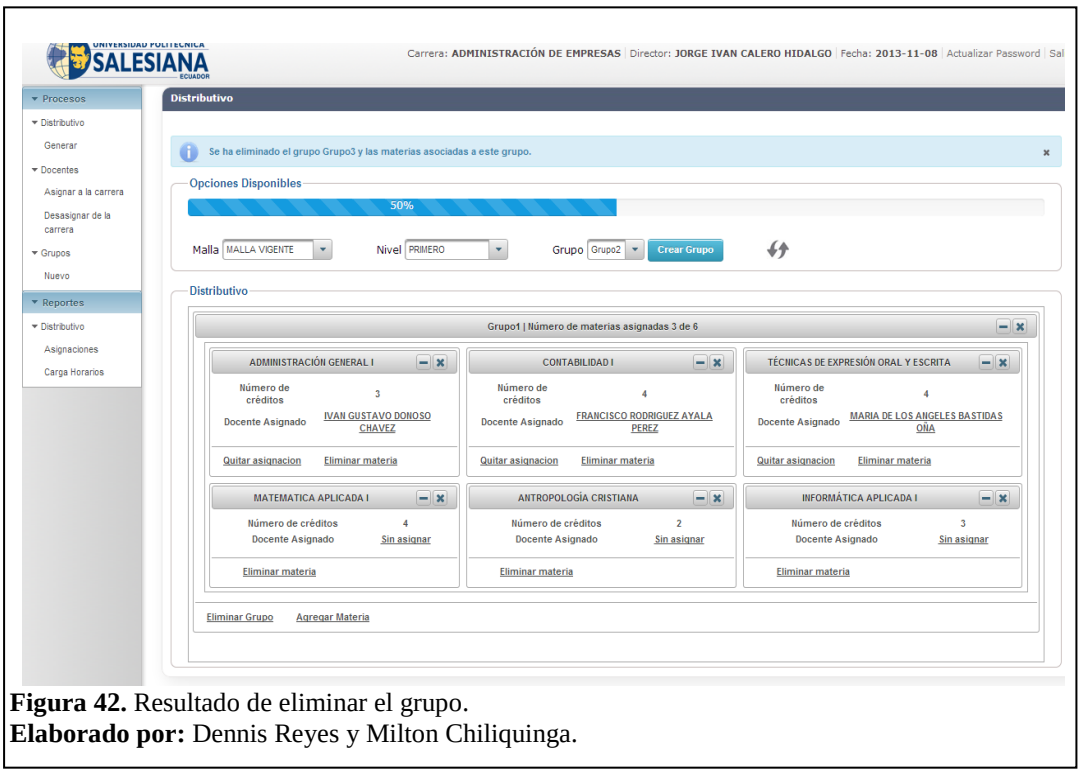


Figura 42. Resultado de eliminar el grupo.
Elaborado por: Dennis Reyes y Milton Chiliquinga.

2.1.4 Eliminar materia.

Elimina una o varias materias dentro de un grupo del distributivo, para realizar esta acción se dispone del siguiente link Eliminar materia después de seleccionar esta opción se muestra la siguiente ventana de confirmación:

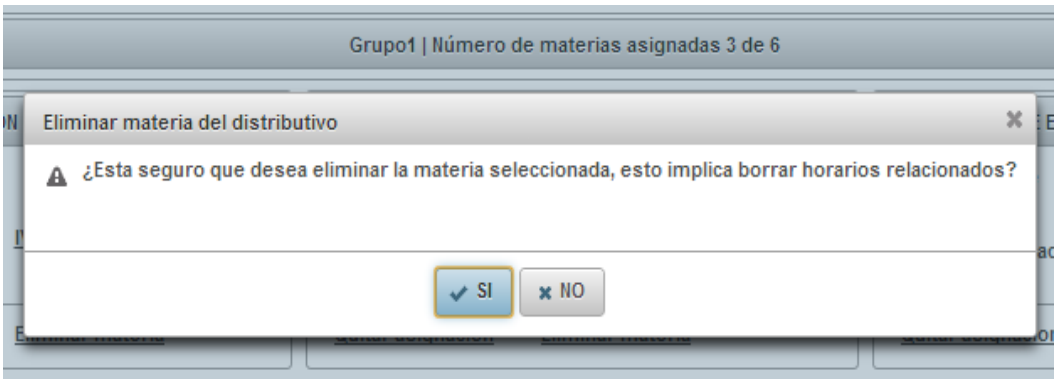


Figura 43. Confirmación para eliminar una materia.
Elaborado por: Dennis Reyes y Milton Chiliquinga.

Una vez que se acepte la confirmación la materia es eliminada del grupo seleccionado, como se muestra en la siguiente figura.

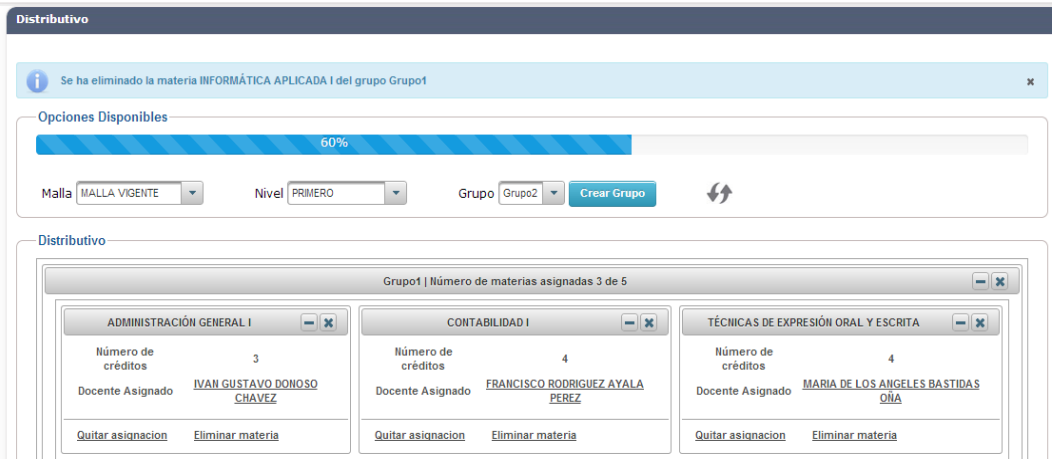


Figura 44. Eliminar una materia del distributivo.
Elaborado por: Dennis Reyes y Milton Chiliquinga.

2.2 Docentes

Figura 45. Sección correspondiente a los docentes.
Elaborado por: Dennis Reyes y Milton Chiliquinga.

Esta opción realiza la asignación de un docente a la carrera o de ser el caso desasignarlo de la carrera.

2.2.1 Asignar a la carrera.

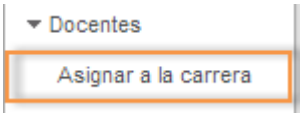


Figura 46. Asignar un docente a la carrera.
Elaborado por: Dennis Reyes y Milton Chiliquinga.

Una vez seleccionado la opción que muestra la figura anterior se despliega la siguiente pantalla.

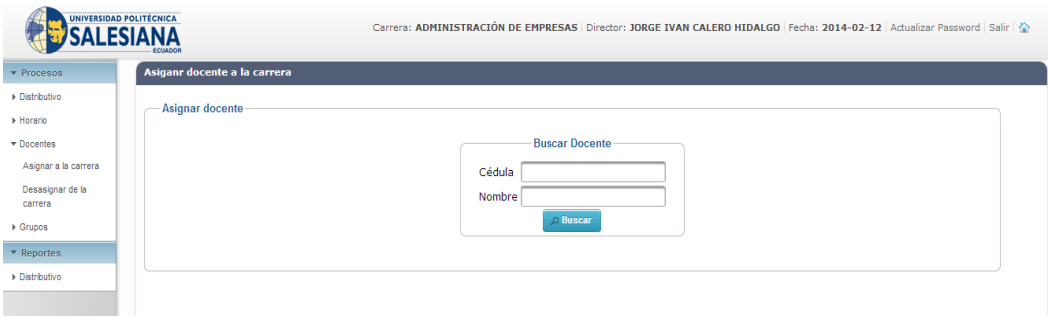


Figura 47. Panel para la búsqueda de un docente.
Elaborado por: Dennis Reyes y Milton Chiliquinga.

Para realizar la asignación de un docente primero se debe realizar la búsqueda del docente que se desea asignar para lo cual se dispone de dos criterios de búsqueda, la cédula o el nombre, los resultados de la búsqueda se despliegan en una tabla en la cual se puede seleccionar el docente que se desea asignar. Si el docente que busca no se encuentra registrado puede realizar el registro en este punto.

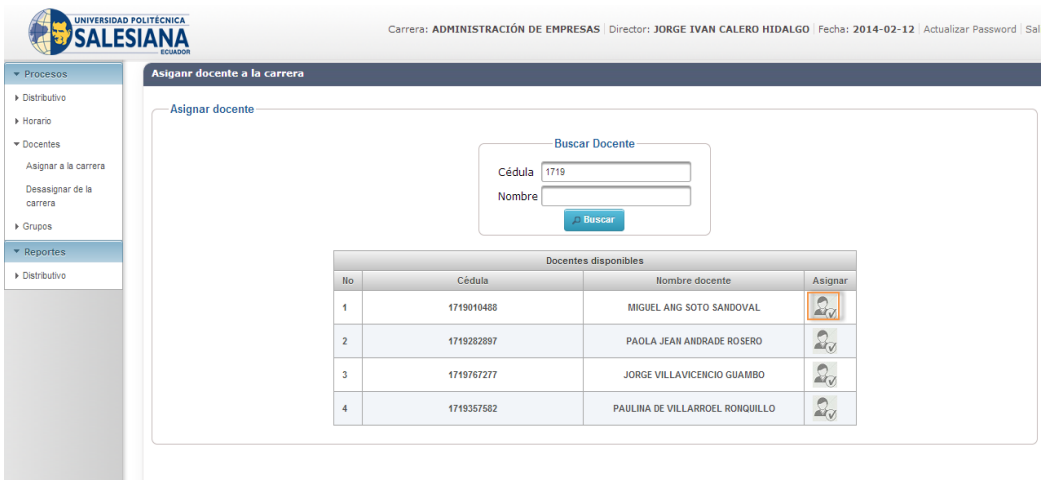


Figura 48. Resultado de la búsqueda del docente por cédula.
Elaborado por: Dennis Reyes y Milton Chiliquinga.

Para seleccionar el docente que se asignará a la carrera clic en el link  con lo cual se muestra un mensaje de confirmación una vez aceptada la confirmación el docente es asignado a la carrera y está disponible para agregarlo al distributivo.

2.2.2 Desasignar de la carrera.

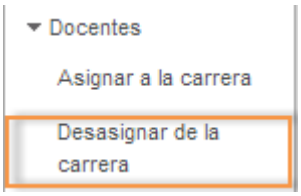


Figura 49. Desasignar un docente de la carrera.
Elaborado por: Dennis Reyes y Milton Chiliquinga.

Una vez seleccionada la opción que muestra la figura de despliega la siguiente pantalla.

Figura 50. Pantalla para la des asignación de un docente.
Elaborado por: Dennis Reyes y Milton Chiliquinga.

De la misma forma esta pantalla proporciona criterios de búsqueda para determinar el docente que será desasignado de la carrera.

Figura 51. Resultado de la búsqueda para deasignar un docente.
Elaborado por: Dennis Reyes y Milton Chiliquinga.

Para realizar la des asignación clic en el link  con lo cual se muestra un panel de confirmación en donde se informa de ser el caso que el docente tiene materias asignadas, estas materias automáticamente pasaran a ser desvinculadas del docente seleccionado e igualmente el docente ya no estará disponible para agregarlo al distributivo.

2.3 Grupos

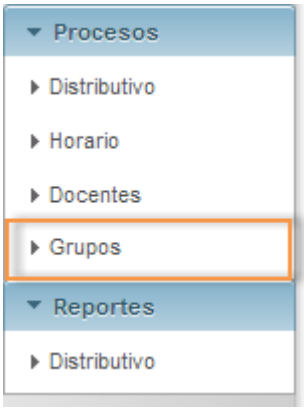


Figura 52. Grupos.
Elaborado por: Dennis Reyes y Milton Chiliquinga.

La siguiente opción permite crear o actualizar grupos para la carrera

2.3.1 Nuevo.

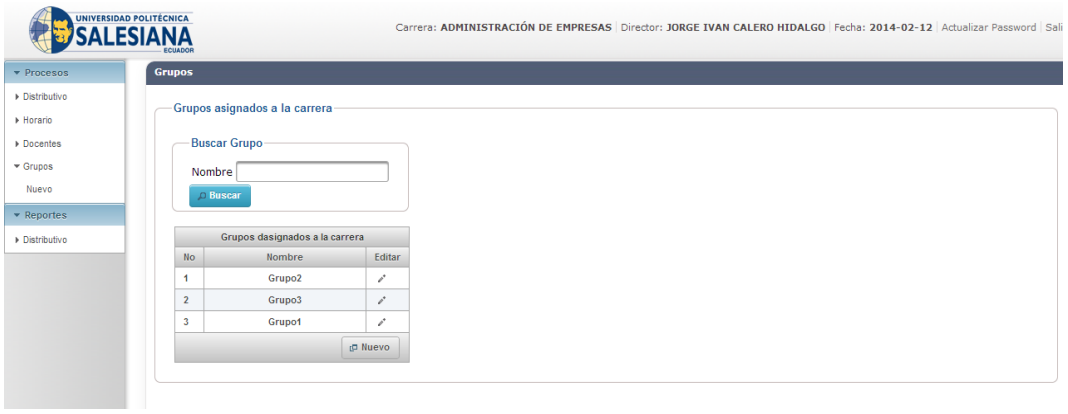


Figura 53. Pantalla para crear o actualizar un grupo.
Elaborado por: Dennis Reyes y Milton Chiliquinga.

Esta pantalla dispone de un panel de búsqueda con un criterio que es el nombre del grupo, este campo es opcional y en caso de estar en blanco busca todos los grupos asignados a la carrera como muestra el ejemplo de la figura anterior. Para editar el nombre del grupo clic en link en el caso de crear un nuevo grupo clic en link Nuevo este link abre una ventana emergente en la cual se puede crear el grupo, el mismo que será automáticamente asignado a la carrera.

REPORTES

Figura 54. Reportes del distributivo.
Elaborado por: Dennis Reyes y Milton Chiliquinga.

Esta sección contiene reportes del módulo de distributivo y horarios.

3.1 Distributivo

En la siguiente opción se pueden crear reportes en pantalla y posteriormente exportarlos a formato pdf. Los reportes disponibles son los siguientes.

3.1.1 Asignaciones.

Figura 55. Reporte de asignaciones del distributivo.
Elaborado por: Dennis Reyes y Milton Chiliquinga.

Esta opción despliega la siguiente pantalla:

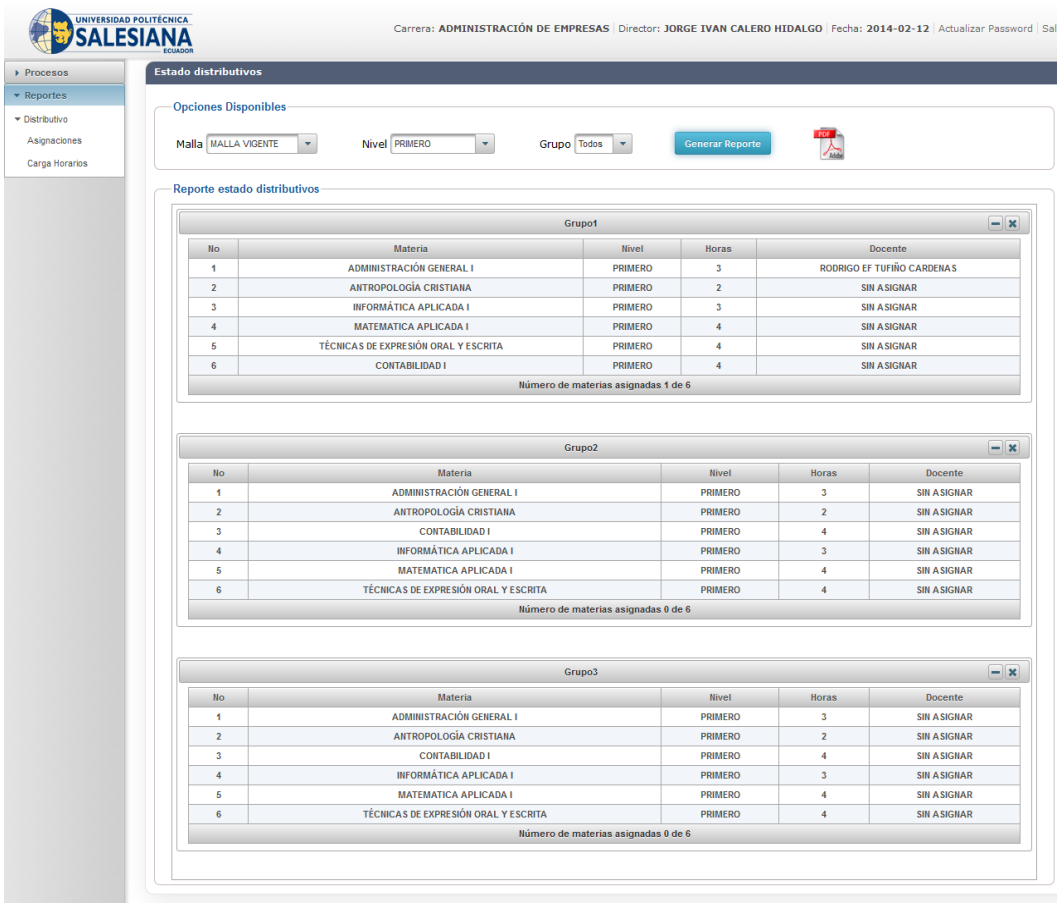
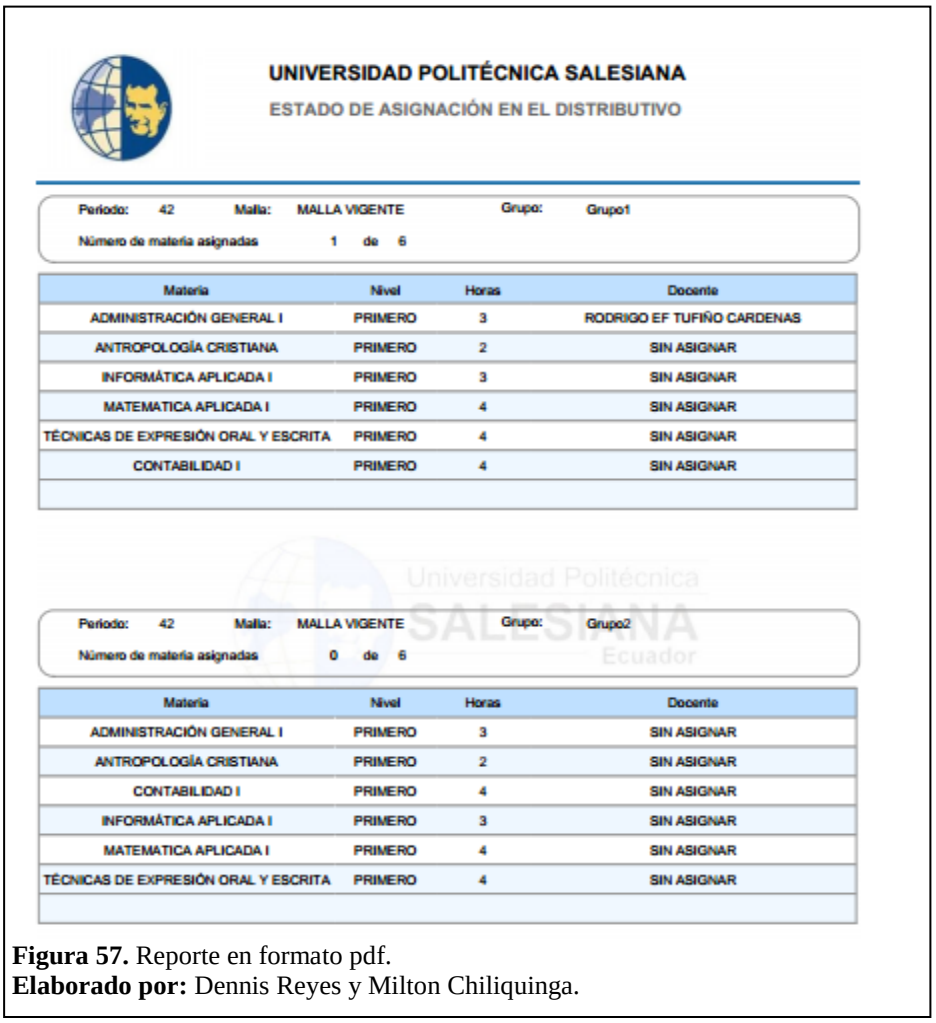


Figura 56. Reporte del estado de asignación dentro de la carrera.
Elaborado por: Dennis Reyes y Milton Chiliquinga.

Esta pantalla dispone de los siguientes filtros para generar el reporte: malla, nivel y grupo para el ejemplo se selecciona la malla vigente el primer nivel y en grupo todos; una vez completados los criterios del reporte clic en el siguiente botón 

para generar el reporte en pantalla y clic en el siguiente link para exportalo a formato pdf 

.



UNIVERSIDAD POLITÉCNICA SALESIANA

ESTADO DE ASIGNACIÓN EN EL DISTRIBUTIVO

Periodo: 42 Malla: MALLA VIGENTE Grupo: Grupo1

Número de materia asignadas 1 de 6

Materia	Nivel	Horas	Docente
ADMINISTRACIÓN GENERAL I	PRIMERO	3	RODRIGO EF TUFIÑO CARDENAS
ANTROPOLOGÍA CRISTIANA	PRIMERO	2	SIN ASIGNAR
INFORMÁTICA APLICADA I	PRIMERO	3	SIN ASIGNAR
MATEMATICA APLICADA I	PRIMERO	4	SIN ASIGNAR
TÉCNICAS DE EXPRESIÓN ORAL Y ESCRITA	PRIMERO	4	SIN ASIGNAR
CONTABILIDAD I	PRIMERO	4	SIN ASIGNAR

Periodo: 42 Malla: MALLA VIGENTE Grupo: Grupo2

Número de materia asignadas 0 de 6

Materia	Nivel	Horas	Docente
ADMINISTRACIÓN GENERAL I	PRIMERO	3	SIN ASIGNAR
ANTROPOLOGÍA CRISTIANA	PRIMERO	2	SIN ASIGNAR
CONTABILIDAD I	PRIMERO	4	SIN ASIGNAR
INFORMÁTICA APLICADA I	PRIMERO	3	SIN ASIGNAR
MATEMATICA APLICADA I	PRIMERO	4	SIN ASIGNAR
TÉCNICAS DE EXPRESIÓN ORAL Y ESCRITA	PRIMERO	4	SIN ASIGNAR

Figura 57. Reporte en formato pdf.
Elaborado por: Dennis Reyes y Milton Chiliquinga.

3.1.2 Carga horarios.

Figura 58. Reporte de la carga de horarios.
Elaborado por: Dennis Reyes y Milton Chiliquinga.

Esta opción despliega la siguiente pantalla.

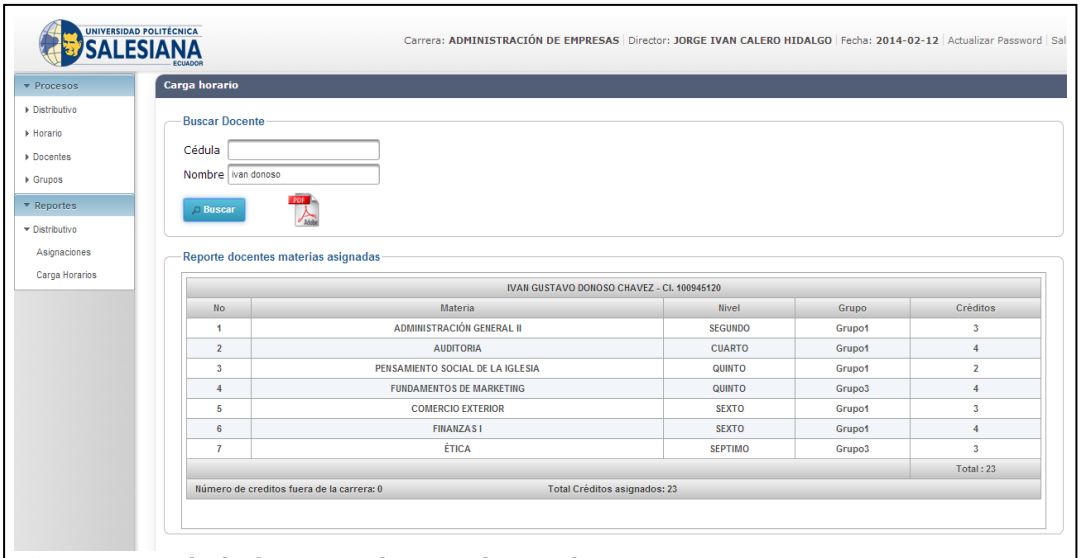


Figura 59. Resultado de generar el reporte de carga horarios.
Elaborado por: Dennis Reyes y Milton Chiliquinga.

Para generar este reporte se pone a disposición dos criterios de búsqueda, la cédula o el nombre del docente, en el caso de que estos campos estén vacíos el reporte se genera con todos los docentes asignados a la carrera.

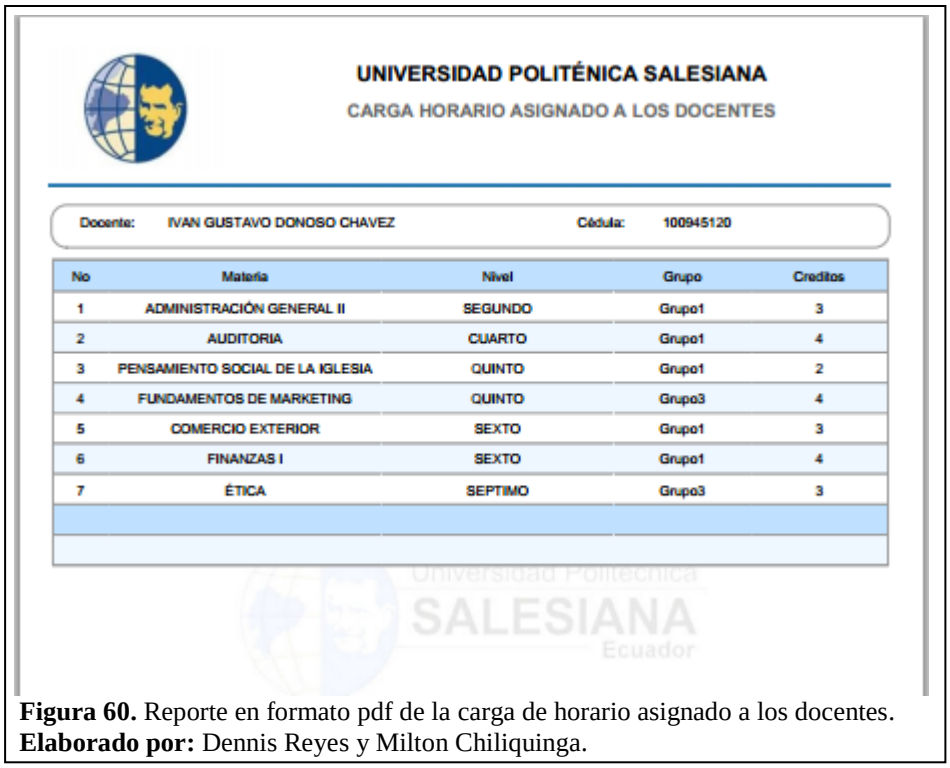


Figura 60. Reporte en formato pdf de la carga de horario asignado a los docentes.
Elaborado por: Dennis Reyes y Milton Chiliquinga.