UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
SEDE VIÑA DEL MAR - JOSÉ MIGUEL CARRERA

SISTEMA DE CONTROL DE PERSONAS Y ALUMNOS PARA JARDÍN INFANTIL

Trabajo de Titulación para optar al Título
de Técnico Universitario en
INFORMÁTICA

Alumnos:
José Roco Tapia
Héctor Sandoval Cuevas

Profesora Guía:
Catherine Gómez Barrera.

2018

RESUMEN

KEYWORD: SISTEMA INFORMÁTICO DE CONTROL DE PERSONAL – CONTROL DE ASISTENCIA – SISTEMA PARA JARDÍN– LENGUAJE DE PROGRAMACIÓN JAVA – BASE DE DATOS MARIADB

El jardín Infantil Brotes Nuevos se encuentra ubicado en la calle Calafquén en el sector de Achupallas, consta con más de 10 años de experiencia y ha construido una relación de confianza con la comunidad de vecinos y ex-alumnos que han salido del establecimiento, pero con el paso del tiempo se ha hecho necesario agilizar procesos que actualmente se realizan de forma manual, bajo esa meta se desarrolla este sistema informático para el apoyo de las actividades de matrículas generadas por el jardín, manejo de datos sobre personas relacionadas con el mismo siendo funcionarios, alumno, padres, apoderados y personas capacitadas para retirar al alumno del jardín, permitiendo a estos datos ser manipulados ingresando, eliminando o modificando cuando sea necesario, además de poder generar listados de información útil para el jardín, separando por cada tipo de las personas o cursos que se imparten dentro del jardín, y manteniendo actualizado el estado de los alumnos. Del mismo modo se mantiene un registro de las asistencias de los funcionarios con sus entradas y salidas del establecimiento, mientras que de los alumnos y apoderados se mantiene un registro de asistencia a las clases y reuniones respectivamente.

Control de personas y alumnos para jardín infantil es el nombre que recibe este sistema informático y fue desarrollado en lenguaje de programación Java v.8 con la herramienta NetBeans, con el gestor de base de datos fue MariaDB, este sistema reside en un único computador con sistema operativo Windows 10 que posee el jardín en su establecimiento, junto la base de datos ya nombrada.

A continuación se presenta lo que tratará cada capítulo de este trabajo:

Capítulo 1: Aspectos relevantes del diseño lógico. Aquí es donde se indica la organización de la empresa, la situación actual en que se encuentra el control de información que maneja el jardín, así como el proceso de matrícula, los problemas que se logran detectar y la proposición de una solución.

Capítulo 2: Medio ambiente computacional, herramientas de software y descripción de archivos. Se presentan las características del equipo que contendrá el sistema y detalla en totalidad las tablas que utiliza éste.

Capítulo 3: Descripción de programas. Muestra cómo funciona cada uno de los programas que posee el sistema informático.

Finalmente se presentan las conclusoines del trabajo.

ÍNDICE

ÍNDICE DE DIAGRAMAS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

El ser humano es un ser de sociedad, se une a otras personas y grupos para sobrevivir y convivir, organizándose y creando soluciones a los problemas o necesidades que tienen, y con el tiempo descubriendo nuevas mejorar y carencias para sus estilos de vida.

Una de las necesidades de la humanidad siempre ha sido el dejar descendencia y mantener una línea de conocimiento, y con el limitado tiempo y gran cantidad de tareas que se necesitan hacer, el cuidar y educar a la juventud se transforman en una acción de gran peso para prevalecer y ser un aporte a la comunidad en el futuro.

El jardín infantil “Brotes Nuevos” tiene la misión de cuidar y educar a niños entre 2 a 4 años, proporcionando una solución a la necesidad de mantener un ambiente propicio de desarrollo cognitivo y motor para niños que se ven inmersos en los primeros pasos de la vida educativa y cultural. Además el jardín no solo se centra en el niño, proporciona apoyo y guía a los padres manteniendo comunicación y haciéndolos parte del crecimiento del niño, mantiene una vista de comunidad, desarrollando propuestas de acción con los vecinos y otros jardines cercanos, así como enseñar sobre cultura.

La realización de la visión y misión que persigue esta organización, es necesario un proceso de burocrático de documentación y control de información para el establecimiento, no sólo manejan información de apoderados y alumno sino también de las personas que trabajan dentro de éste, de las actividades que se realizan no muy diferente a cualquier institución u organización que trate con personas, y es por eso que nace este proyecto, para dar una solución y agilizar estos procesos manteniendo un orden sobre las transacciones que se llevan a cabo día a día y se que mantendrán haciendo hasta el fin de la vida del jardín.

CAPÍTULO 1

ASPECTOS RELEVANTES DEL DISEÑO LÓGICO

1. Aspectos relevantes del diseño lógico.

1.1. DESCRIPCIÓN DE LA ORGANIZACIÓN.

El Jardín Infantil “Brotes Nuevos” se encuentra ubicado en Calafquen #446 en el sector de Achupallas, V región. Es dependiente del Centro Cultural Educativo del mismo nombre, su propósito es desarrollar un proyecto pedagógico alternativo, social, integral y culturalmente pertinente, siendo actualmente financiado por transferencia de fondos por la JUNJI.

Se construyó en el año 1992 con el objetivo de educar y cuidar niños, para que sus madres pudieran ingresar de forma más fácil al mundo laboral, todo esto gracias al apoyo ganado en un proyecto con la comunidad europea. En un inicio trabajaban en el lugar 4 personas y al igual que hoy consta con dos salas y cupos de 15 niños aproximadamente con intenciones de querer aumentar salas y el cupo de niños.

El Jardín Infantil entrega educación gratuita e integral a los niños y niñas desde los 2 años hasta los 3 años 11 meses de edad, además de brindarles: alimentación, descanso, protección y recreación; favoreciendo la relación con la familia y la comunidad.

Este espacio educativo busca potenciar y favorecer las habilidades, valores y actitudes en el manejo de estrategias de autoconocimiento y autoaprendizaje, estimulando a los educandos para que sean gestores en la construcción de sus propias estructuras y competencias cognitivas, afectivas y sociales.

El jardín cuenta actualmente con 8 personas, de las cuales una cumple el rol de educadora y directora del establecimiento, 3 son técnicos en párvulos, 2 practicantes cursando sus estudios, una manipuladora de alimentos encargada de la cocina y las comidas de los niños y una persona encargada del aseo. Este establecimiento funciona de 8:30 - 16:30 horas para los niños y desde 8:00 - 17:00 horas para sus funcionarios.

1.2. DESCRIPCIÓN DE LA SITUACIÓN ACTUAL.

El jardín infantil actualmente cuenta con un procedimiento manual que controla y almacena el personal que trabaja en él, asimismo el método para obtener información de los funcionarios o de los alumnos que se encuentran en el establecimiento es lento, ya que éstos se encuentran en carpetas y se hace tedioso localizar a una sola persona en

particular con todos sus datos. Se produce que en ocasiones los niños son retirados por otras personas de las cuales ni las encargadas de despacharlos conocen, teniendo que llamar al apoderado para confirmar, como también existe pérdida de tiempo por tener que ir al lugar en búsqueda de información.

Los procesos asociados al trabajo son los siguientes:

A. El proceso de matrícula ocurre en el mes de noviembre donde los apoderados presentan sus postulaciones en el establecimiento; los resultados de dichas postulaciones se dan a conocer en el mes de febrero del siguiente año. A los apoderados se les pide para la matrícula, una copia del certificado de nacimiento del niño, un certificado de estudio y/o trabajo del padre y/o la madre, datos de contacto, además de la ficha social de hogares. Todos estos papeles se guardan en una carpeta con documentos respecto al niño, la cual queda guardada en un estante con otras carpetas del mismo establecimiento.
B. La asignación de cursos ocurre de acuerdo con los cupos disponibles y el orden de llegada. Actualmente se cuenta con dos cursos los cuales son separados por edades en dos salas.
C. La generación de documentos, como listado de alumnos, cursos con sus respectivos alumnos, apoderados de los alumnos, se realiza recopilando información de los papeles que ya se tienen y creando un archivo Word de forma manual o en papel y lápiz el cual es pegado en un mural con varios listados más.
D. La asistencia es tomada por el profesional a cargo del curso durante la mañana, llenando una ficha previamente hecha que es dada por la JUNJI, de la misma forma se pide firmar a las personas que trabaja en el jardín al momento de llegar en un cuaderno de asistencia común. Durante las reuniones de apoderados se pide que éstos o la persona que represente al niño en la ocasión firme la asistencia.
E. El jardín cuenta con evaluaciones hacia los estudiantes las cuales constan de actividades realizadas a los alumnos en donde cada actividad es evaluada con una observación y es entregada al apoderado en reuniones.
F. El proceso de retirada del establecimiento para un niño es mediante un bus escolar o el apoderado responsable el cual habla con la encargada del curso o la directora para hacer el retiro efectivo del estudiante.

1.3. PROBLEMAS DETECTADOS.

A. Poca eficacia al momento de matricular a un alumno, ya que al momento de las postulaciones la encargada maneja todo el papeleo y en ocasiones tiende a perder algunos o simplemente no registra la postulación al jardín.
B. Lentitud en la generación de documentos tales como listado de alumnos, o de personal luego de algún cambio por estar esta información dispersa en papeles.
C. La asistencia al ser tomada en papel y no ser respaldada en algún lugar seguro, si ocurre algún percance, se tomaba nuevamente a lo cual en ocaciones se tiene que incurrir a usar la memoria de los funcionarios pudiendo ocurrir errores.
D. El proceso de creación de informes sobre las evaluaciones hechas a los niños se transforma en un proceso lento al tener que buscar las observaciones por alumno de forma independiente.
E. Nula constancia de personas que están autorizadas de retirar al niño.
F. Lentitud e ineficacia en la búsqueda de información sobre niños o personal mediante revisión de documentos en estantes y carpetas.
G. Redundancia de información repartida en documentos.
H. Inconsistencia de datos por documentos no actualizados pudiendo resultar en sobrecupos en las matrículas.

1.4. DESCRIPCIÓN DEL SISTEMA.

1.4.1. Objetivo Principal

Crear un sistema informático para el control, mantenimiento, almacenamiento de datos sobre personal, alumnos, matrículas, cursos, asistencia de personal, alumnos y apoderados, actividades realizadas, realizando creación de informes de utilidad para el jardín infantil.

1.4.2. Objetivos específicos

- Almacenar información detallada sobre las personas que tienen relación con el establecimiento ya sea nombre, previsión, número de contacto, cargo que desempeña, entre otros, esto incluye a padres, apoderados y retiradores del niño.

- Almacenar información detallada de los niños que asisten al jardín, como nombre, apoderado, curso, etc.
- Modificar, eliminar, consultar, agregar información sobre las personas involucradas con el jardín (niños, apoderados, funcionarios).
- Almacenar las actividades que se harán en el jardín en conjunto a registrar las observaciones que se harán a cada estudiante por actividad.
- Registrar datos sobre los cursos, salas y asistencia para usos de dirección.
- Control de usuarios.

1.4.3. Beneficios

- Disminución de redundancia de papeles rondando en las salas y dirección.
- Respaldo de información de uso recurrente.
- Uso menor de tiempo al generar documentos actualizados lo que genera tener acceso rápido a los datos de los estudiantes y de las personas.
- Mayor facilidad para generar listados dependiendo del caso necesario.
- Precisión en la información manejada respecto a las personas.
- Mejor orden y precisión al momento del proceso de matrículas para los alumnos.
- Mayor seguridad al momento de despachar a los niños.
- Respaldo de las actividades que se desarrollan con sus respectivas observaciones, las cuales serán impresas y entregadas a los apoderados de una forma más formal.
- Un mayor control sobre los datos de las personas y alumnos del establecimiento mediante el uso de niveles de acceso en el sistema.

1.4.4. Descripción general de la solución propuesta.

Mediante una pantalla interactiva se pedirá una autentificación de usuario solicitando nombre de usuario y contraseña para ingresar al sistema pudiendo obtener dos tipos de permisos, Administrador en caso de la directora y permisos de Profesor.

El usuario Administrador tendrá la capacidad exclusiva de eliminar, modificar o crear existencias del personal, los alumnos, apoderados, retiradores y usuarios. Para ambos tipos de usuarios será posible realizar cualquier tipo de consulta disponible a la base de datos, generar listados de alumnos, personal, apoderados, entre otros disponibles en las funcionalidades del sistema y realizar cambios a las actividades evaluadas agregando una descripción u observación sobre la actividad desarrollada; del mismo modo se puede hacer ingreso de la asistencia de los niños, apoderados o funcionarios.

El proceso de matrícula inicia con la intervención del apoderado, se solicitan los datos del alumno y se verifican en la tabla Alumnos para evitar sobrescribir o duplicar sus datos para luego ser almacenados, se ingresan los datos de su apoderado, revisando nuevamente su existencia pero en esta ocasión en la tabla Persona, para el caso de que no sea el único niño a cargo de ese apoderado. Se consulta al apoderado por la(s) persona(s) que recoge(n) al niño y se realiza su respectiva validación. El proceso termina registrando la matrícula en el sistema y creando un documento con los datos pertinentes de la matrícula.

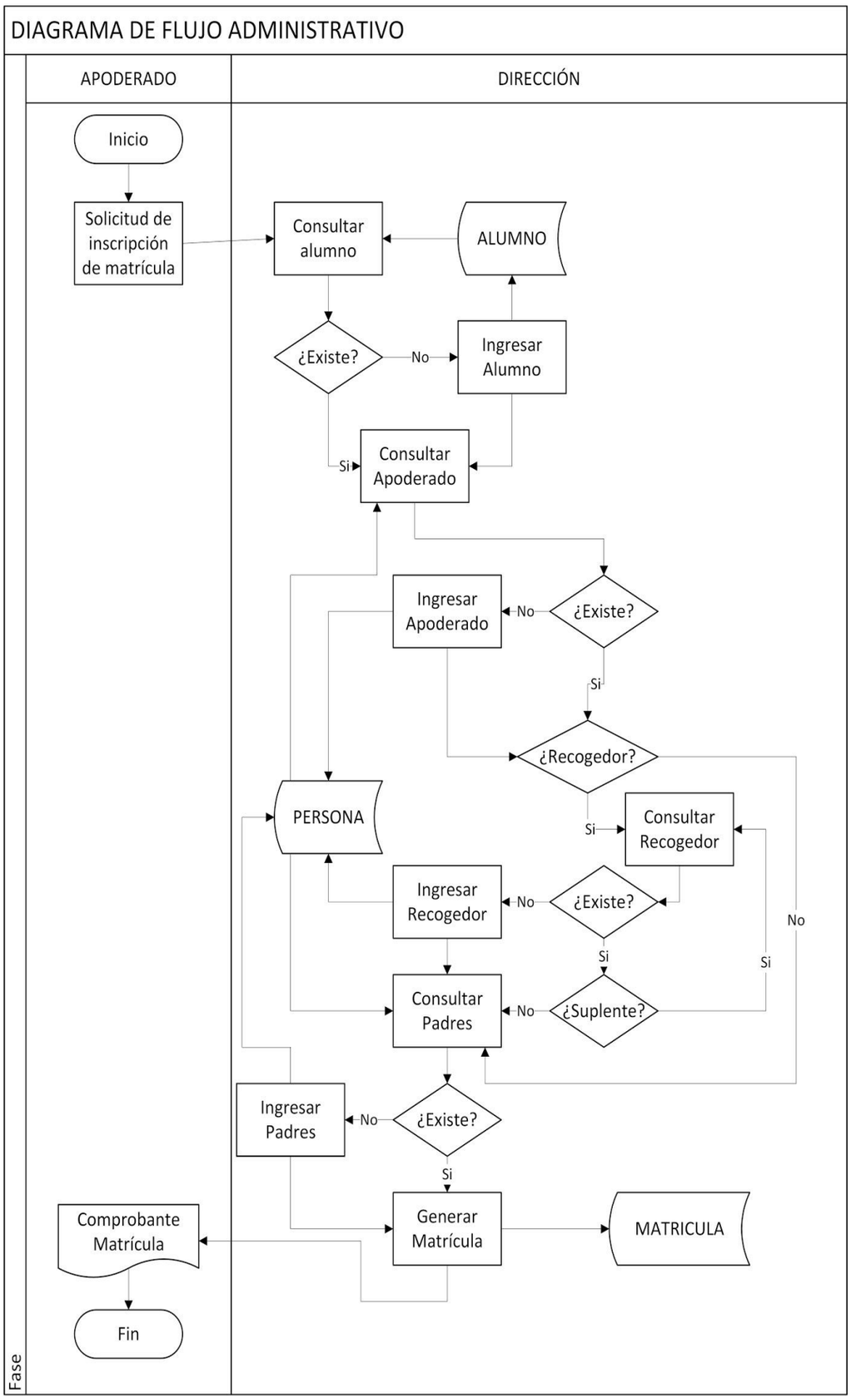
Antes de comenzar el periodo de clases o en ocasiones especiales, se actualizarán los datos de las actividades vía teclado almacenadas en la tabla actividades y se destinarán a los cursos.

Para el ingreso de las evaluaciones se buscará al alumno con el Rut, el código de dicha actividad y la fecha en la que se realizó la actividad con estos se dará la opción de colocar una observación al niño respecto a la actividad.

El proceso de asistencia se ingresará mediante teclado traspasando la información de la plantilla de asistencia de los alumnos que es entregada por la JUNJI.

El sistema con su respectiva base de datos estará en el computador disponible en el jardín localizado en la oficina de la directora. Poseerá una interfaz gráfica de fácil uso y manejabilidad.

1.4.5. Diagrama de flujo administrativo



Fuente: Elaborado en conjunto por José Roco y Héctor Sandoval.

Diagrama 1-1: Proceso de matrícula

1.5. FUNCIONALIDADES DEL SISTEMA.

- Acceso al sistema: Se pide el ingreso mediante un usuario y su respectiva contraseña para acceder al sistema permitiendo la validación de los datos ingresados consultándolos en la tabla de usuarios. El usuario administrador será creado por los programadores y podrá ser modificado en el primer ingreso del sistema, luego será modificado según lo requiera la directora.
- Mantenedor de alumnos inscritos: Permite agregar, modificar, eliminar y consultar los alumnos.
- Mantenedor de persona: Permite agregar, modificar, eliminar y consultar las personas.
- Mantenedor de previsiones: Permite agregar, modificar, eliminar y consultar las previsiones.
- Mantenedor de cursos: Permite agregar, modificar, eliminar y consultar los cursos que existen en la base de datos.
- Mantenedor de actividad: Permite agregar, modificar, eliminar y consultar las actividades que se tienen planificadas para el año.
- Mantenedor de sala: Permite agregar, modificar, eliminar y consultar las salas existentes en la base de datos.
- Mantenedor de usuarios: Permite agregar, modificar, eliminar y consultar los usuarios existentes en la base de datos.
- Generar, modificar y consultar matrícula: Permitirá generar un número de folio el cual será vinculado a un Rut y a un semestre que se esté dictando en el establecimiento, además, en caso de ser requerido, la modificación de los datos. Además, se podrá consultar por la matrícula del alumno.
- Ingresar, modificar y consultar evaluación: Permite ingresar una observación detallada de la evaluación hecha a un respectivo alumno. En caso de cometer algún error en el ingreso, permitirá modificar la observación de la evaluación o si por error confundió al alumno e ingresó un detalle a un alumno erróneo. Además, consultar la evaluación hecha y a que alumno fue esta misma.
- Ingresar, modificar y consultar asistencia: Permite ingresar y modificar la asistencia de los alumnos, apoderados y funcionarios cada caso con su respectiva pantalla. Además se podrá consultar la asistencia por cada persona existente en el sistema.
- Generar listado de cursos por sala: Entrega un listado de los cursos que posee el establecimiento y en qué sala están.

- Generar listado de salas: Muestra una lista con salas que existen en el establecimiento.
- Generar informe de alumnos evaluados con su detalle: Permite generar un informe por alumno con el detalle perteneciente, el día de la evaluación y qué evaluación fue hecha.
- Generar listado de matriculados: Entrega lista que contendrá los datos del alumno, año al cual pertenece, curso, semestre, etc.
- Generar listado de personas y clasificarlos dependiendo del tipo al cual esté asociado: Entrega un listado de las personas que posee la base de datos y gracias al atributo tipo, permitirá clasificarlos por segmentos para así agruparlos.
- Generar listado de alumnos con sus respectivos recogedores: Entrega lista que contendrá datos sobre la persona que está autorizada a recoger al niño del jardín, nombre, contacto, etc pudiendo ser el oficial o el suplente.
- Generar listado de asistencia: Entrega lista que contendrá los datos sobre la asistencia, de los alumnos, apoderados y funcionarios presentes, ausentes.
- Generar listado de actividades: Entrega una lista que contendrá las actividades a realizar en el semestre por curso.
- Generar listado de cursos: Entrega lista que contendrá los datos del curso, cantidad de alumnos, encargado, etc.
- Generar listado de previsiones: Entrega lista que contendrá los datos de las previsiones existentes en la base de datos.

1.6. INFORMACIÓN QUE SE MANEJARÁ

1.6.1. Entradas

- Datos de los alumnos: Datos como RUT, nombre, apellidos, fecha de nacimiento, curso, detalles sobre evaluaciones, número de ficha.
- Datos del personal: Datos como RUT, nombres, apellido materno, apellido paterno, dirección del hogar, número de teléfono, correo electrónico, tipo, nombre de la previsión a la cual pertenece.
- Datos de la previsión: Nombre, Descripción de ésta.
- Datos de matrícula: Año, Semestre, curso.
- Datos de sala: Capacidad y una breve descripción.
- Datos del curso: Nombre del curso y sala en la que se dicta.
- Datos de evaluación: Fecha de evaluación, código de la actividad realizada.

- Datos de actividad: Nombre de la actividad y descripción de la misma.
- Datos asistencia alumno: Información sobre si el alumno asistió o faltó a clases.
- Datos de usuario: Datos como usuario, contraseña.
- Datos asistencia personal: Información sobre si el funcionario llegó o no a trabajar.
- Datos asistencia apoderados: Información sobre si el apoderado acudió a reunión en una fecha específica.

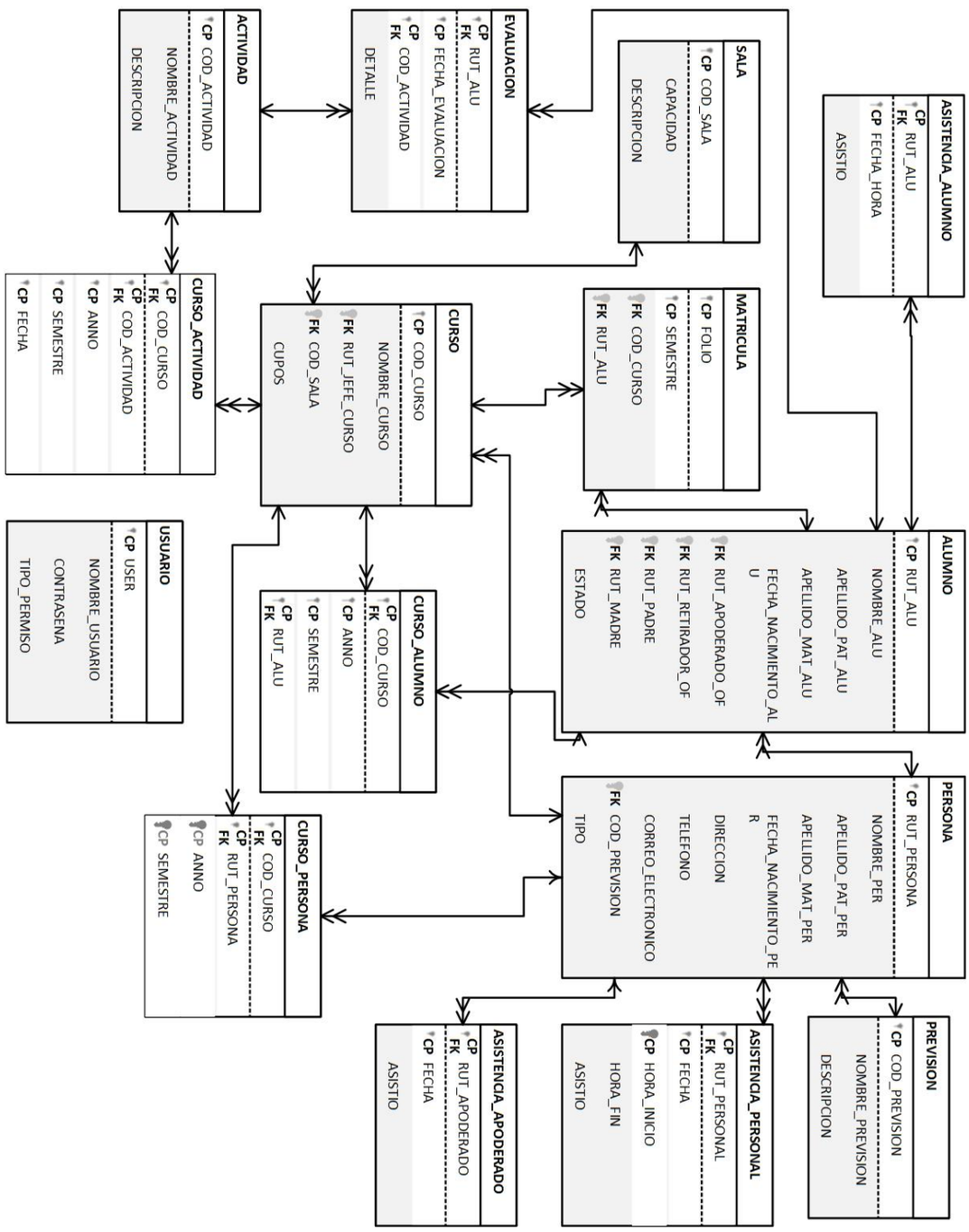
1.6.2. Salidas

- Listado de alumnos con su respectivo apoderado y recogedor: Se muestran datos como nombre del alumno, nombre del apoderado, nombre del recogedor, teléfono recogedor, teléfono apoderado. Pueden desplegarse datos sobre un alumno en específico usando los filtros disponibles, como nombre o Rut.
- Listado de personal con sus respectivos datos: Se muestran datos sobre el personal tales como, nombre, teléfono, dirección, correo, cumpleaños. Se podrá filtrar según nombre del funcionario. Pueden desplegarse datos sobre una persona en específico usando los filtros disponibles, como nombre o Rut.
- Listado de previsión con sus respectivos afiliados: Se muestran datos sobre la previsión señalando a qué persona corresponde.
- Listado de salas con sus respectivos profesores y curso pertenecientes a ésta: Se muestra información sobre las salas señalando la cantidad de alumnos y sus encargados.
- Listados de personas permitidas para retirar al niño: Se muestran los datos de las personas que pueden retirar al niño, nombre, teléfono, dirección. Pueden desplegarse datos sobre una persona en específico usando los filtros disponibles, como nombre o Rut.
- Listado de asistencia: Se muestra información sobre la asistencia según un día en específico. Pueden desplegarse datos en específico usando los filtros disponibles, Rut o nombre, de la misma forma se puede filtrar buscando una fecha específica.
- Listado de cursos: Se muestra información sobre los cursos, alumnos por cada uno, encargada, etc.
- Listado de usuarios: Se muestran los datos de todos los usuarios del sistema. Pueden desplegarse datos sobre un usuario en específico usando los filtros disponibles.

1.7. ENTIDADES DE INFORMACIÓN

1) Alumnos: En esta entidad se almacenarán los datos necesarios, como nombre, apellidos, etc. de todos los alumnos matriculados en el jardín.
2) Persona: Almacena los datos de las personas involucradas en el jardín como lo son sus datos personales, previsión, teléfono. Este archivo contendrá un campo tipo en el cual se especificará el tipo de persona ingresada al sistema ya sea funcionario, apoderado, director, practicante, etc.
3) Previsión: Contiene datos sobre las previsiones.
4) Matrícula: Registra los alumnos que están en el jardín en un año en particular y su respectivo semestre independiente si se retiró posteriormente del establecimiento.
5) Sala: Contiene la capacidad de la sala y una breve descripción de esta misma.
6) Curso: En esta entidad se almacenan los cursos disponibles en el jardín y la sala a cuál pertenece dicho curso.
7) Usuario: Esta entidad almacena los datos necesarios para la validación de ingreso al sistema.
8) Asistencia_Alumno: Se encarga de almacenar la asistencia de cada niño por fecha.
9) Asistencia_Personal: Se encarga de almacenar la asistencia del personal y su hora de ingreso y salida.
10) Asistencia_Apoderado: Se encarga de almacenar la asistencia de los apoderados en cada reunión.
11) Actividad: Se almacenan las actividades realizadas en el jardín con su descripción.
12) Evaluación: Se registran las observaciones según la actividad realizada a un respectivo alumno.
13) Curso_Persona: Si el tipo seleccionado en persona es practicante o funcionaria, indica cuál curso tiene a cargo guardando la información de forma histórica.
14) Curso_Alumno: Indica a qué curso está matriculado el alumno actualmente, guardando la información de forma histórica.
15) Curso_Actividad: Contiene información de la actividad a desarrollar y al curso correspondiente a tomarse durante el periodo del año.

1.8. MODELO DE DATOS



Fuente: Elaborado en conjunto por José Roco y Héctor Sandoval.

Diagrama 1-2: Modelo de datos

1.9. ESTRUCTURA DE CÓDIGOS

Para efectos de este sistema se utilizarán los siguientes códigos para identificar a previsión, sala, alumno, persona, curso, actividad, matrícula.

- Previsión: Se utilizará una sigla de un máximo de 6 carácteres correspondiente a cada previsión.

Tabla 1-1: Previsiones

Previsión	Código
Banmédica S.A.	BANMED
Chuquicamata Ltda.	CHUQUI
Colmena Golden Cross S.A.	COLGCS
Consalud S.A.	CONSAL
Cruz Blanca S.A.	CRUZBL
Cruz del Norte Ltda.	CRUZNO
Optima S.A.	OPTIMA
Fundación Ltda.	FUNDAC
Fusat Ltda.	FUSAT
Masvida S.A.	MASVID
Río Blanco Ltda.	RIOBLA
San Lorenzo Ltda.	SNLORE
Vida Tres S.A.	VIDTRE
Fonasa	FON_A FON_B FON_C FON_D

Fuente: Elaborado en conjunto por José Roco y Héctor Sandoval.

- Sala: Se utilizará un número correlativo.
- Curso: Se utilizará una abreviación del curso que contará con 7 carácteres como máximo, además de un número correlativo que indica el paralelo.

  Ejemplo:

  NVLME01 (Nivel Medio Extendido 01)
- Actividad: Su código es un número secuencial.
- Matrícula: Se utilizará un código el cual constará de tres partes, uno que contenga la palabra FO + dos últimos números del año + el número secuencial que indicará en la posición que se registró.

  Ejemplo:

  FO17001.

1.10. CONDICIONANTES DE DISEÑO

Para el sistema propuesto se tienen las siguientes consideraciones:

El lenguaje utilizado para crear el sistema será bajo Java y una base de datos en MariaDB.

La base de datos será implementada igual que el programa en el único computador que cuenta la institución, con respaldos anuales, los que serán subidos a la nube disponible por GoogleDrive.

CAPÍTULO 2

MEDIO AMBIENTE COMPUTACIONAL Y DESCRIPCIÓN DE ARCHIVOS

2. MEDIO AMBIENTE COMPUTACIONAL Y DESCRIPCIÓN DE ARCHIVOS

2.1. RECURSO COMPUTACIONAL

2.1.1. Configuración del sistema

El equipo donde será implementado el sistema será uno y se encuentra ubicado en la sala de dirección y sus especificaciones son:

- Procesador Intel Celeron CPU de 1,10 GHz de velocidad.
- Disco Duro: 500GB
- Memoria actual: 4 GB.
- Placa Madre: Lenovo C240
- Monitor: Pantalla 18.5" ancha LED
- Teclado: USB Marca Lenovo modelo lxh-ems-10ya
- Mouse: Óptico USB Marca Lenovo Modelo LXH-EMS-10ZA

Impresora:

- Impresora Canon PIXMA MP140

Cabe señalar que este equipo no cuenta con un sistema de respaldo. La base de datos será implantada en el equipo descrito anteriormente, siendo respaldada a finales de año en forma de informes generados por el sistema.

2.1.2. Equipo utilizado para el desarrollo

Notebook:

- Procesador: APU AMD A8-7410 2,2 GHz hasta 2,5 GHz
- Disco Duro: 1 TB
- Memoria actual: 4 GB
- Monitor: retroiluminación WLED de alta definición BrightView de 14’’ en diagonal
- Teclado: Teclado tipo isla de tamaño normal
- Mouse: TouchPad compatible con multigestos, sin botón activar/desactivar

2.1.3. Software utilizado

-Sistema operativo: Windows 10

El sistema gracias a su interfaz amigable apoya el desarrollo de un sistema ligero y entendible, además de permitir el funcionamiento de distintas tareas, proporcionando un ambiente seguro y estable.

-Herramientas de desarrollo de software: NetBeans 8.2

Esta herramienta es un entorno de desarrollo para diversos sistemas operativos. Es capaz de soportar el lenguaje Java pensado para aplicaciones, al ser de carácter multiplataforma es flexible para la programación esto sumado a su visión centrada a eventos lo hace expedito.

-Lenguajes: Java versión 8

Java es un lenguaje de programación orientado a objetos que permite usar conceptos propios como polimorfismo y herencia, sumado a NetBeans que ayuda a crear diversas pantallas interactivas que provocan eventos entre sí que permiten un desarrollo simplificado y sencillo de la interfaz provocando que el usuario comprenda y quede satisfecho con el sistema que él quiere.

2.2. DESCRIPCIÓN DE ARCHIVOS

2.2.1. Alumno

Tabla 2-1 Alumno

DESCRIPCIÓN DE LA TABLA				
Nombre Físico	ALUMNO			
Descripción	Tabla con los datos de los alumnos que han estado en el jardín			
Clave primaria	rut_alu			
Clave(s) Foránea(s)	rut_apoderado_of referencia a PERSONA rut_retirador_of referencia a PERSONA rut_padre referencia a PERSONA rut_madre referencia a PERSONA			
DESCRIPCIÓN DE REGISTRO				
N°	NOMBRE	TIPO	LARGO	DESCRIPCIÓN
1	rut_alu	INT	8	Rut del alumno
2	nombre_alu	VARCHAR	30	Corresponde al nombre del alumno
3	apellido_paterno_alu	VARCHAR	30	Corresponde al apellido paterno del alumno
4	apellido_materno_alu	VARCHAR	30	Corresponde al apellido materno del alumno
5	fecha_nacimiento_alu	DATE	11	Fecha de nacimiento del alumno
6	rut_apoderado_of	INT	8	Rut del apoderado oficial del alumno
8	rut_retirador_of	INT	8	Rut de la persona oficial con permiso para retirar al alumno
10	rut_madre	INT	8	Rut de la madre del niño

Tabla 2-1 Alumno (Continuacón)

N°	NOMBRE	TIPO	LARGO	DESCRIPCIÓN
11	rut_padre	INT	8	Rut del padre del alumno
12	estado	CHAR	1	Estado que indica si el alumno está matriculado. M=Matriculado, P=Pendiente.

Fuente: Elaborado en conjunto por José Roco y Héctor Sandoval.

2.2.2. Persona

Tabla 2-2 Persona

DESCRIPCIÓN DE LA TABLA				
Nombre Físico	PERSONA			
Descripción	Tabla con los datos de apoderados, retiradores y funcionarios			
Clave primaria	rut_persona			
Clave(s) Foránea(s)	cod_prevision referencia a PREVISION			
DESCRIPCIÓN DE REGISTRO				
N°	NOMBRE	TIPO	LARGO	DESCRIPCIÓN
1	rut_persona	INT	8	Corresponde al Rut de la persona
2	nombre_per	VARCHAR	30	Corresponde al nombre de la persona
3	apellido_pat_per	VARCHAR	30	Apellido paterno de la persona
4	apellido_mat_per	VARCHAR	30	Apellido materno de la persona
5	fecha_nacimiento_per	DATE	11	Fecha de nacimiento de la persona
6	direccion	VARCHAR	50	Dirección de la persona
7	telefono	VARCHAR	12	Teléfono de contacto de la persona

Tabla 2-2 Persona (Continuación)

N°	NOMBRE	TIPO	LARGO	DESCRIPCIÓN
8	correo_electronico	VARCHAR	30	Correo electrónico de contacto de la persona
9	cod_prevision	VARCHAR	6	Código correspondiente a una previsión.
10	tipo	VARCHAR	2	Corresponde al tipo de relación que tiene la persona con el jardín, pudiendo estar compuesta por maximo 2 tipos. A = apoderado. F = funcionario. R= recogedor.

Fuente: Elaborado en conjunto por José Roco y Héctor Sandoval.

2.2.3. Asistencia Alumno

Tabla 2-3 Asistencia Alumno

DESCRIPCIÓN DE LA TABLA				
Nombre Físico	ASISTENCIA_ALUMNO			
Descripción	Tabla que corresponde a la asistencia de los alumnos.			
Clave primaria	rut_alu + fecha_hora			
Clave(s) Foránea(s)	rut_alu referencia a ALUMNO			
DESCRIPCIÓN DE REGISTRO				
N°	NOMBRE	TIPO	LARGO	DESCRIPCIÓN
1	rut_alu	INT	8	Rut de un alumno.
2	fecha_hora	DATETIME	20	Fecha y hora en la que se tomó la asistencia.

Fuente: Elaborado en conjunto por José Roco y Héctor Sandoval.

2.2.4. Previsión

Tabla 2-4 Previsión

DESCRIPCIÓN DE LA TABLA				
Nombre Físico		PREVISION		
Descripción		Tabla que corresponde a la información de las previsiones.		
Clave primaria		cod_prevision.		
Clave(s) Foránea(s)		*no tiene*		
DESCRIPCIÓN DE REGISTRO				
N°	NOMBRE	TIPO	LARGO	DESCRIPCIÓN
1	cod_prevision	VARCHAR	6	Código correspondiente a una previsión.
2	nombre_prevision	VARCHAR	30	Nombre de una previsión.
3	descripcion	VARCHAR	50	Pequeña descripción de la previsión.

Fuente: Elaborado en conjunto por José Roco y Héctor Sandoval.

2.2.5. Sala

Tabla 2-5 Sala

DESCRIPCIÓN DE LA TABLA				
Nombre Físico		SALA		
Descripción		Corresponde a una sala existente en el jardín		
Clave primaria		cod_sala		
Clave(s) Foránea(s)		*no tiene*		
DESCRIPCIÓN DE REGISTRO				
N°	NOMBRE	TIPO	LARGO	DESCRIPCIÓN
1	cod_sala	TINYINT	1	Código correspondiente a una sala.
2	capacidad	TINYINT	2	Corresponde a la cantidad de niños que pueden estar en una sala.
3	descripcion	VARCHAR	50	Pequeña descripción de la sala.

Fuente: Elaborado en conjunto por José Roco y Héctor Sandoval.

2.2.6. Matricula

Tabla 2-6 Matricula

DESCRIPCIÓN DE LA TABLA				
Nombre Físico	MATRICULA			
Descripción	Información de la matrícula de un alumno del jardín			
Clave primaria	folio + semestre			
Clave(s) Foránea(s)	cod_curso referencia a CURSO rut_alu referencia a ALUMNO			
DESCRIPCIÓN DE REGISTRO				
N°	NOMBRE	TIPO	LARGO	DESCRIPCIÓN
1	folio	VARCHAR	7	Código único correspondiente a una matrícula.
2	semestre	TINYINT	1	Corresponde al número de semestre (1 ó 2).
3	cod_curso	VARCHAR	8	Código correspondiente al curso que se matriculó el alumno.
4	rut_alu	INT	8	Corresponde al RUT del alumno matriculado.

Fuente: Elaborado en conjunto por José Roco y Héctor Sandoval.

2.2.7. Asistencia Personal

Tabla 2-7 Asistencia Persona

DESCRIPCIÓN DE LA TABLA	
Nombre Físico	ASISTENCIA_PERSONAL
Descripción	Tabla con información del registro de asistencia del personal
Clave primaria	rut_personal + fecha + hora_inicio

Tabla 2-7 Asistencia Persona (Continuación)

Clave(s) Foránea(s)	rut_personal referencia a PERSONA			
DESCRIPCIÓN DE REGISTRO				
N°	NOMBRE	TIPO	LARGO	DESCRIPCIÓN
1	rut_personal	INT	8	Rut de una persona que trabaja en el jardín
2	fecha	DATE	11	Fecha en la que fue tomada la asistencia
3	hora_inicio	TIME	4	Hora en que la persona inicia su día de trabajo.
4	hora_fin	TIME	4	Hora en que la persona termina su día de trabajo.
5	asistio	CHAR	1	SI=S NO=N

Fuente: Elaborado en conjunto por José Roco y Héctor Sandoval.

2.2.8. Asistencia Apoderado

Tabla 2-8 Asistencia Apoderado

Nombre Físico	ASISTENCIA_APODERADO			
Descripción	Tabla que guarda información sobre la asistencia de los apoderados a reunión.			
Clave primaria	rut_apoderado + fecha			
Clave(s) Foránea(s)	rut_apoderado referencia a PERSONA			
DESCRIPCIÓN DE REGISTRO				
N°	NOMBRE	TIPO	LARGO	DESCRIPCIÓN
1	rut_apoderado	INT	8	Rut de un apoderado del jardín
2	fecha	DATE	11	Almacena el día, mes, año en la que se toma asistencia a los apoderados en una reunión
3	asistio	CHAR	1	SI=S NO=N

Fuente: Elaborado en conjunto por José Roco y Héctor Sandoval.

2.2.9. Evaluación

Tabla 2-9 Evaluación

DESCRIPCIÓN DE LA TABLA				
Nombre Físico		EVALUACION		
Descripción		Tabla sobre las evaluaciones que se realizaron a los alumnos		
Clave primaria		rut_alu + fecha_evaluacion + cod_actividad		
Clave(s) Foránea(s)		rut_alu referencia a ALUMNO cod_actividad referencia a ACTIVIDAD		
DESCRIPCIÓN DE REGISTRO				
N°	NOMBRE	TIPO	LARGO	DESCRIPCIÓN
1	rut_alu	INT	8	Corresponde al Rut del alumno al cual se le toma la evaluación.
2	fecha_evaluacion	DATE	11	Fecha en la cual fue llevada a cabo la evaluación.
3	cod_actividad	VARCHAR	10	Código de la actividad evaluada.
4	detalle	VARCHAR	50	Breve descripción sobre la evaluación hecha respecto al alumno.

Fuente: Elaborado en conjunto por José Roco y Héctor Sandoval.

2.2.10. Curso

Tabla 2-10 Curso

DESCRIPCIÓN DE LA TABLA				
Nombre Físico	CURSO			
Descripción	Tabla que contiene los cursos dictados en el jardín			
Clave primaria	cod_curso			
Clave(s) Foránea(s)	rut_jefe_curso referencia a PERSONA cod_sala referencia a SALA			
DESCRIPCIÓN DE REGISTRO				
N°	NOMBRE	TIPO	LARGO	DESCRIPCIÓN
1	cod_curso	VARCHAR	7	Código que identifica el curso
2	nombre_curso	VARCHAR	30	Nombre del curso
3	rut_jefe_curso	INT	8	Rut de la persona encargada del curso
4	cod_sala	TINYINT	7	Sala en la cual se imparte el curso
5	cupos	TINYINT	2	Cantidad de alumnos permitidos por curso

Fuente: Elaborado en conjunto por José Roco y Héctor Sandoval.

2.2.11. Curso Alumno

Tabla 2-11 Curso Alumno

DESCRIPCIÓN DE LA TABLA				
Nombre Físico		CURSO_ALUMNO		
Descripción		Histórico que permite ver a los alumnos en un determinado curso, en un año en particular.		
Clave primaria		cod_curso + anno + semestre + rut_alu		
Clave(s) Foránea(s)		cod_curso referencia a CURSO. rut_alu referencia a ALUMNO.		
DESCRIPCIÓN DE REGISTRO				
N°	NOMBRE	TIPO	LARGO	DESCRIPCIÓN
1	cod_curso	VARCHAR	7	Código del curso
2	anno	YEAR	4	Año
3	semestre	TINYINT	1	Corresponde al número de semestre (1 ó 2).
4	rut_alu	INT	8	Rut del alumno

Fuente: Elaborado en conjunto por José Roco y Héctor Sandoval.

2.2.12. Curso Persona

Tabla 2-12 Curso Persona

Nombre Físico	CURSO_PERSONA
Descripción	Tabla con información histórica de los profesores que han tenido un curso a cargo.
Clave primaria	cod_curso + rut_persona + anno + semestre
Clave(s) Foránea(s)	cod_curso referencia a CURSO rut_persona referencia a PERSONA

Tabla 2-12 Curso Persona (Continuación)

DESCRIPCIÓN DE REGISTRO				
N°	NOMBRE	TIPO	LARGO	DESCRIPCIÓN
1	cod_curso	VARCHAR	7	Código del curso.
2	rut_persona	INT	8	Rut del profesor encargado del curso
3	anno	YEAR	4	Año en que tomó el curso como profesor jefe.
4	semestre	TINYINT	1	Corresponde al número de semestre (1 ó 2).

Fuente: Elaborado en conjunto por José Roco y Héctor Sandoval.

2.2.13. Actividad

Tabla 2-13 Actividad

DESCRIPCIÓN DE LA TABLA				
Nombre Físico	ACTIVIDAD			
Descripción	Contiene la actividad a realizarse			
Clave primaria	cod_actividad			
Clave(s) Foránea(s)	*no tiene*			
DESCRIPCIÓN DE REGISTRO				
N°	NOMBRE	TIPO	LARGO	DESCRIPCIÓN
1	cod_actividad	VARCHAR	10	Código de la actividad.
2	nombre_actividad	VARCHAR	30	Nombre de la actividad.
3	descripcion	VARCHAR	50	Breve descripción de la actividad.

Fuente: Elaborado en conjunto por José Roco y Héctor Sandoval.

2.2.14. Curso Actividad

Tabla 2-14 Curso Actividad

DESCRIPCIÓN DE LA TABLA				
Nombre Físico	CURSO_ACTIVIDAD			
Descripción	Contiene la actividad dictada para un curso en particular.			
Clave primaria	cod_curso + cod_actividad + anno + semestre + fecha			
Clave(s) Foránea(s)	cod_curso referencia a CURSO. cod_actividad referencia a ACTIVIDAD.			
DESCRIPCIÓN DE REGISTRO				
N°	NOMBRE	TIPO	LARGO	DESCRIPCIÓN
1	cod_curso	VARCHAR	8	Código del curso.
2	cod_actividad	VARCHAR	10	Código de la actividad.
3	anno	YEAR	4	Año en que se realizó la actividad en el curso
4	semestre	TINYINT	1	Corresponde al número de semestre (1 ó 2).
5	fecha	DATE	11	Fecha en la que debería realizarse la actividad al curso

Fuente: Elaborado en conjunto por José Roco y Héctor Sandoval.

2.2.15. Usuarios

Tabla 2-15 Usuarios

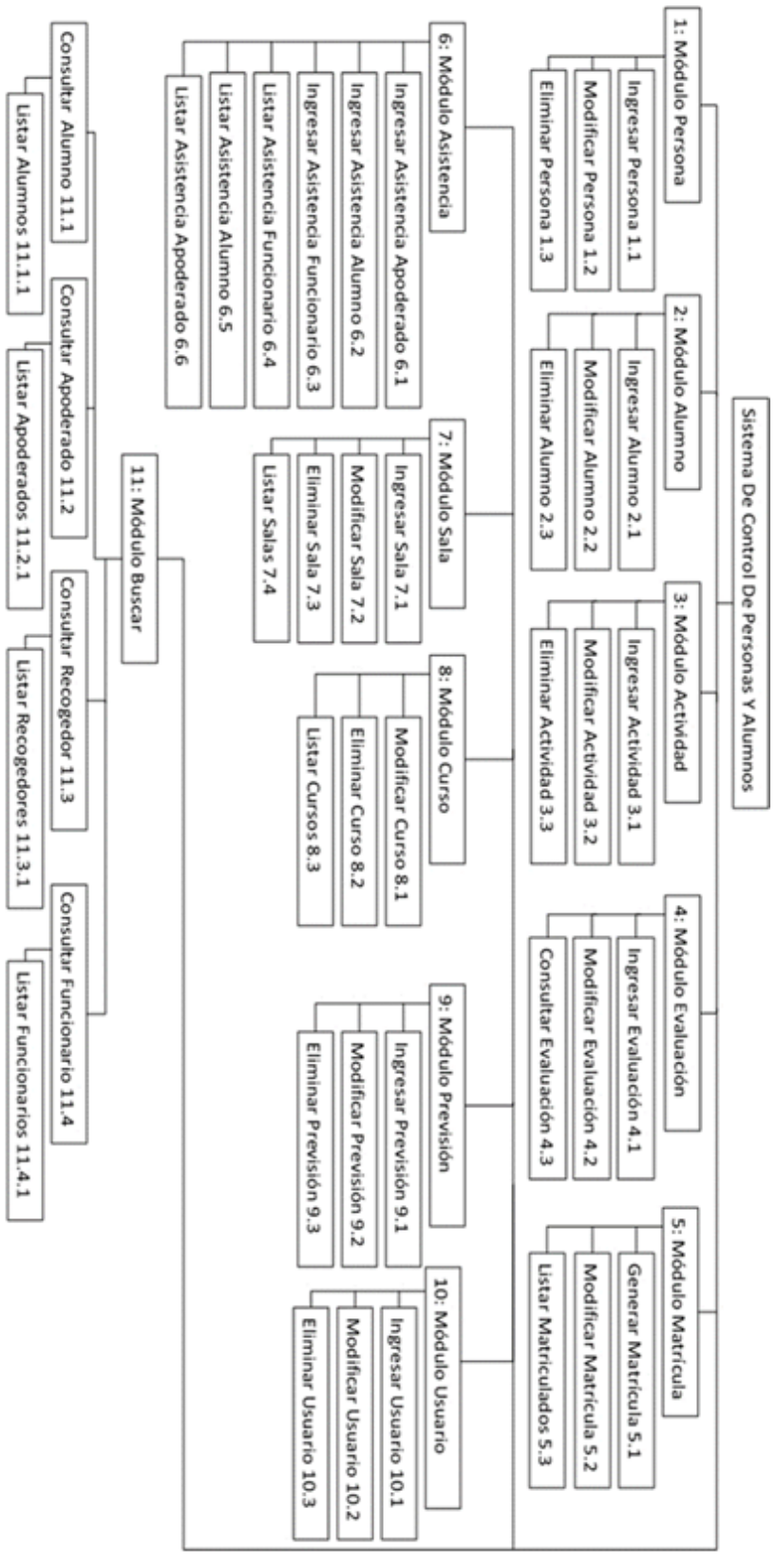
DESCRIPCIÓN DE LA TABLA				
Nombre Físico		USUARIOS		
Descripción		Tabla con los datos de los usuarios creados para el sistema.		
Clave primaria		user		
Clave(s) Foránea(s)		*no tiene*		
DESCRIPCIÓN DE REGISTRO				
N°	NOMBRE	TIPO	LARGO	DESCRIPCIÓN
1	user	VARCHAR	10	Usuario
2	nombre_usuario	VARCHAR	30	Nombre del Usuario
3	contrasena	VARCHAR	20	Contraseña
4	tipo_permisido	CHAR	1	Tipo permiso al acceder. A=Administrador / P=Profesor.

Fuente: Elaborado en conjunto por José Roco y Héctor Sandoval.

CAPÍTULO 3
DESCRIPCIÓN DE PROGRAMAS

3. DESCRIPCIÓN DE PROGRAMAS

3 .1. DIAGRAMA MODULAR

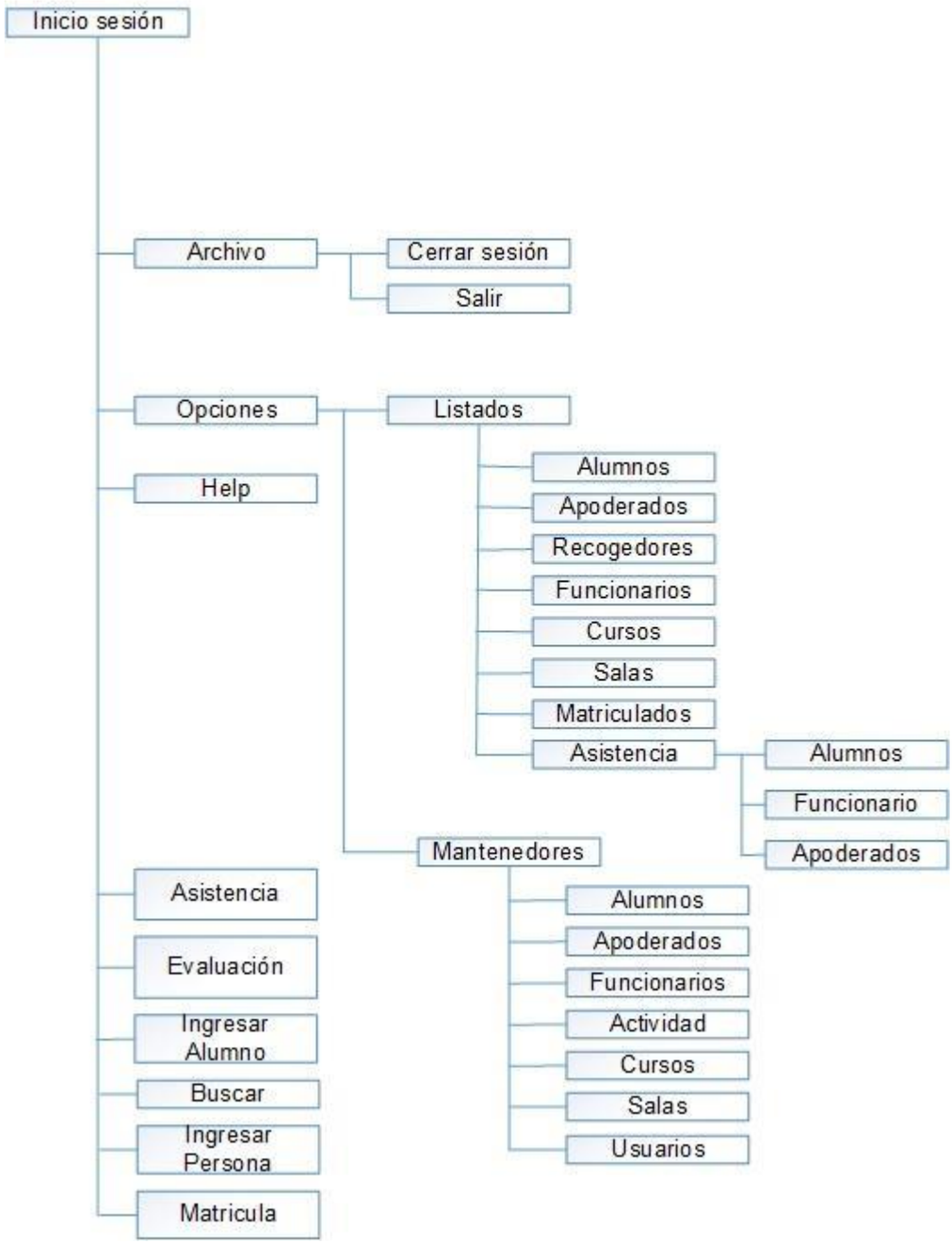


Fuente: Elaborado en conjunto por José Roco y Héctor Sandoval.

Diagrama 3-1: Diagrama Modular

3.2. DIAGRAMA DE MENÚS

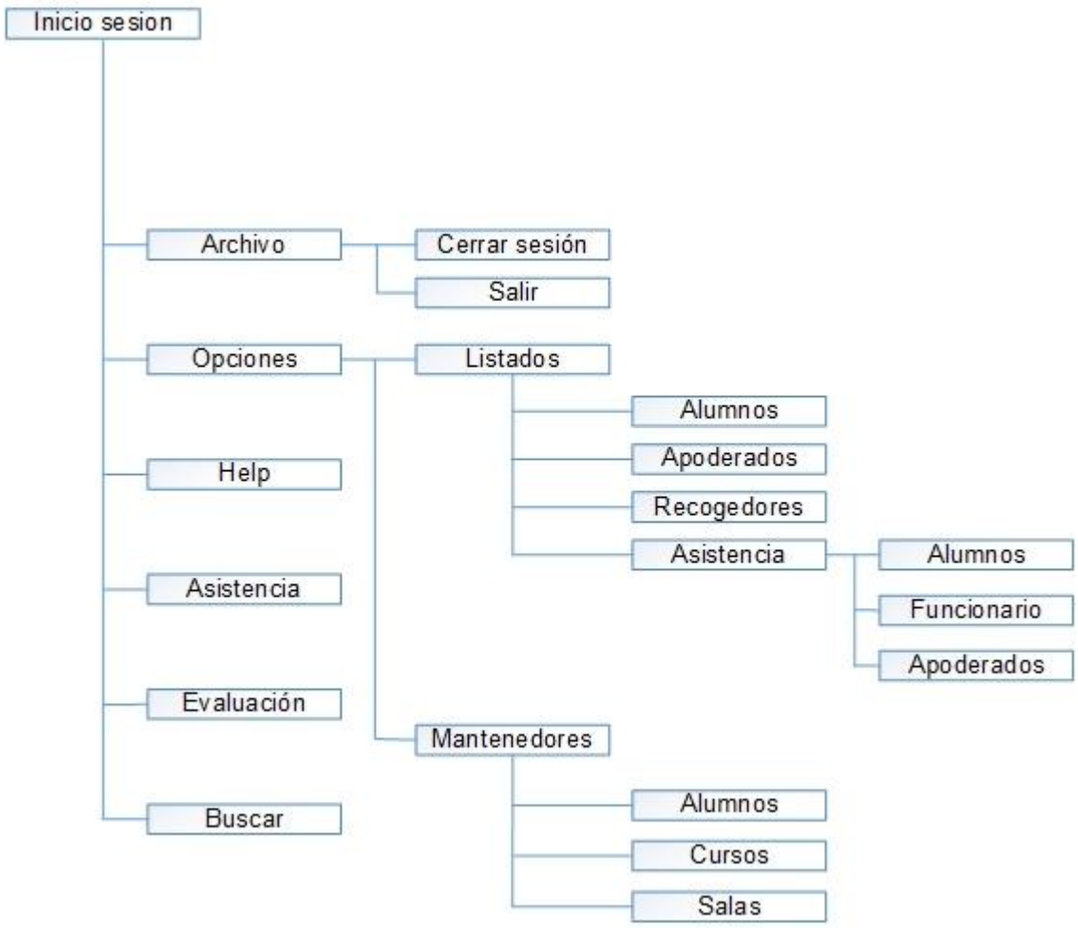
3.2.1. Menú Director



Fuente: Elaborado en conjunto por José Roco y Héctor Sandoval.

Diagrama 3-2 Menú Director

3.2.2. Menú Profesor



Fuente: Elaborado en conjunto por José Roco y Héctor Sandoval.

Diagrama 3-3: Menú Profesor

3.3. DESCRIPCIÓN DETALLADA DE PROGRAMAS

Tabla 3-1 Programas del sistema

	NOMBRE	OBJETIVO
1	Asistencia.java	Permite seleccionar cuál tipo de asistencia se ingresará en el sistema.
2	BD.java	Clase de conexión para la base de datos.
3	Buscar.java	Buscar según criterio (RUT, NOMBRE, APELLIDO PATERNO, APELLIDO MATERNO) seleccionando a una persona involucrada en el Jardín ya sea apoderado, funcionario, recogedor, padre, madre o alumno.
4	IngModActividad.java	Permite ingresar o modificar una Actividad en especial.
5	IngModCurAct.java	Permite ingresar o modificar las actividades hechas a un curso en especial.
6	IngModCurso.java	Permite ingresar o modificar los cursos que existen en el sistema.
7	IngModPrevision.java	Permite ingresar o modificar las previsiones registradas o no en el sistema.
8	IngModSala.java	Permite ingresar o modificar las salas que existen en el establecimiento.
9	IngModUsuario.java	Permite crear o modificar usuarios que ingresen al sistema.
10	IngresarAlumno.java*	Agregar un alumno al sistema.
11	IngresarAsistenciaAlumno.java*	Ingresar Asistencia de los alumnos.
12	IngresarAsistenciaApoderado.java*	Ingresar Asistencia de los apoderados.
13	IngresarAsistenciaFuncionario.java*	Ingresar Asistencia de los funcionarios.
14	IngresarModificarListarEvaluacion.java*	Ingresar, modificar y listar una evaluación para un alumno en particular.
15	IngresarPersona.java*	Agregar una persona (funcionario, apoderado, recogedor, padre o madre) al sistema.
16	InicioSesion.java*	Discrimina el acceso al sistema de control del Jardín Brotes Nuevos.
17	ListadoAlumno.java	Lista todos los alumnos por curso del establecimiento.
18	ListadoAsistenciaAlumno.java	Lista la asistencia a clases de los alumnos de un curso en una fecha y hora en especial.

Tabla 3-1 Programas del sistema (Continuación)

19	ListadoAsistenciaApoderado.java	Lista la asistencia de los apoderados a reuniones de un curso en una fecha en especial.
20	ListadoAsistenciaFuncionario.java	Lista la asistencia al trabajo de los funcionarios en una fecha y hora en especial.
21	ListadoCurso.java	Lista todos los cursos que existen en el establecimiento.
22	ListadoMatriculados.java	Lista todos los matriculados en un año y de un curso en especial del establecimiento.
23	ListadoPersona.java	Lista todas las personas existentes en el establecimiento (funcionarios, apoderados, recogedores, padres y madres).
24	ListadoSala.java	Lista todas las salas existentes y registradas en el establecimiento.
25	MantenedorActividad.java*	Permite ingresar, modificar, eliminar y consultar una actividad.
26	MantenedorAlumno.java*	Permite ingresar, modificar, eliminar y consultar un alumno.
27	MantenedorCurso.java	Permite ingresar, modificar, eliminar y consultar cursos.
28	MantenedorPersona.java*	Permite ingresar, modificar, eliminar y consultar las personas en el sistema.
29	MantenedorPrevision.java	Permite ingresar, modificar, eliminar y consultar una previsión.
30	MantenedorSala.java	Permite ingresar, modificar, eliminar y consultar salas.
31	MantenedorUsuario.java	Permite ingresar, modificar, eliminar y consultar usuarios.
32	MatriculaFrm.java*	Permite Generar matrículas, Modificarlas y listarlas.
33	ModEva.java	Permite modificar una matrícula.
34	PantallaInicio.java	Módulo principal que da las opciones a los demás módulos del sistema.
35	SistemaJardinBrotesNuevos.java	Clase Principal que da funcionamiento a la pantalla principal del sistema (InicioSesion.java).

Fuente: Elaborado en conjunto por José Roco y Héctor Sandoval

Los programas indicados con (*) se describirán detalladamente a continuación:

3.3.1. Iniciar Sesión

Nombre lógico: Inicio Sesión.

Nombre físico: InicioSesion.java

Objetivo: Permite ingresar al sistema mediante un usuario y contraseña discriminando el nivel de acceso.

Diagrama de Bloque:

Fuente: Elaborado en conjunto por José Roco y Héctor Sandoval.

Diagrama 3-4 Inicio Sesión

Reglas de proceso:

- Se presenta pantalla figura 3-1.
- Se ingresa nombre de usuario y contraseña.
- Al presionar enter se busca en la tabla Usuarios, comprobando la existencia de los datos y diferenciando el tipo de permiso que tiene el usuario para la visualización de las funcionalidades.
- Si los datos son correctos se ingresa al menú del sistema.
- Anexo, página 63.

Diseño de pantalla

Fuente: Elaborado en conjunto por José Roco y Héctor Sandoval.

Figura 3-1 Inicio Sesión

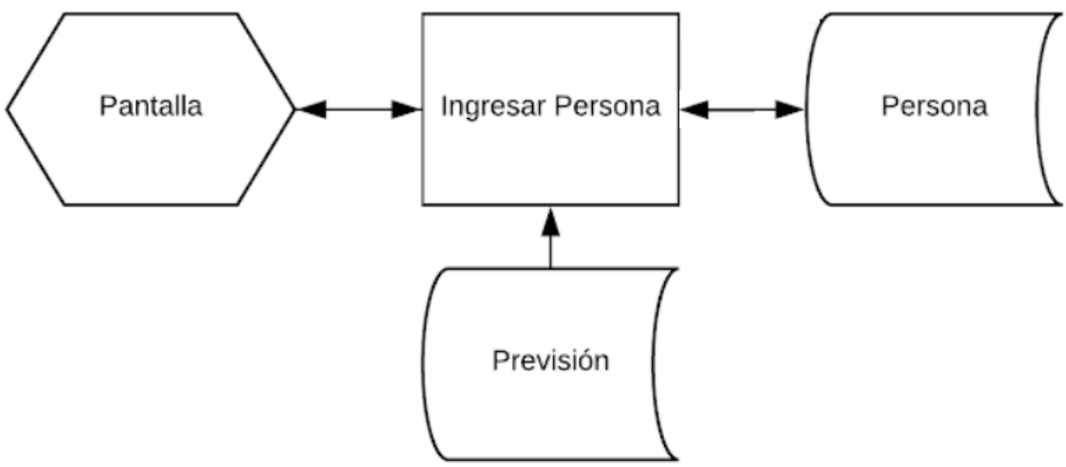
3.3.2. Ingresar Persona

Nombre lógico: Ingresar Persona.

Nombre físico: IngresarPersona.java

Objetivo: Ingresar una persona al sistema (Apoderado, Funcionario, Recogedor).

Diagrama de bloques:



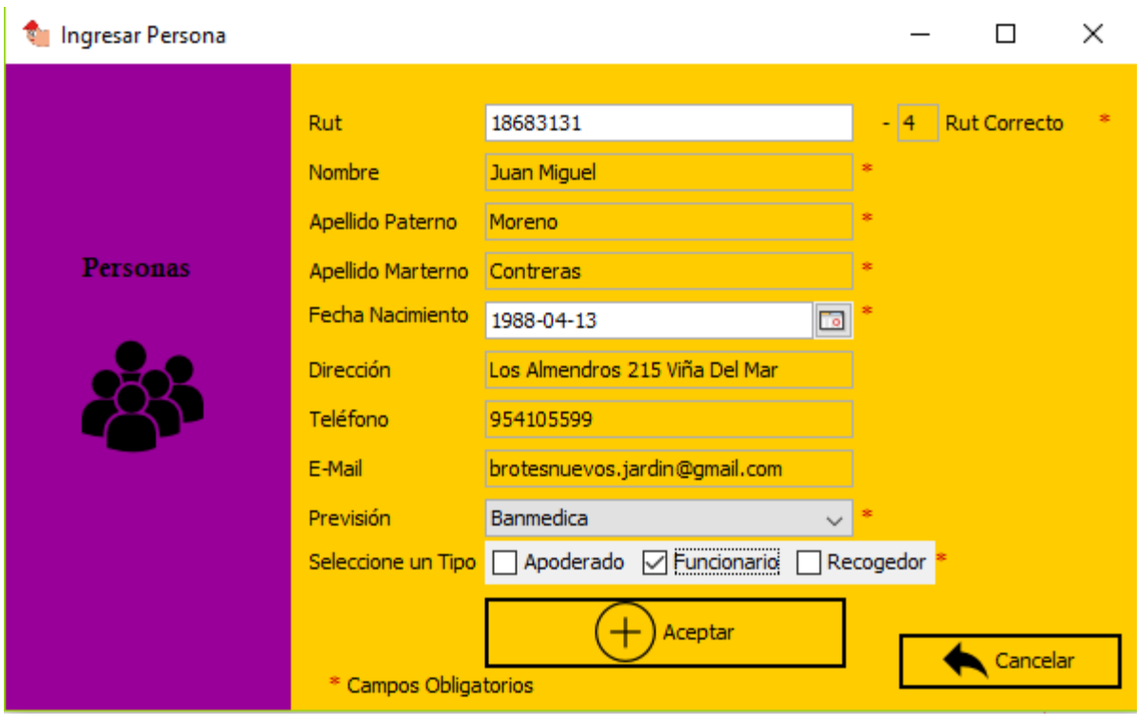
Fuente: Elaborado en conjunto por José Roco y Héctor Sandoval.

Diagrama 3-5 Ingresar Persona

Reglas de proceso:

- Se presenta pantalla figura 3-2
- Se carga la lista desplegable con la información de la tabla Previsión.
- Despliega estructura para completar datos de una persona.
- Se ingresa rut, al momento de ganar el foco el cuadro del dígito verificador se corrobora que el rut ingresado sin dígito no esté en la tabla Persona.
- Se ingresa el dígito verificador, al momento de perder el foco se valida si es correcto.
- Se ingresa nombre, apellido paterno, apellido materno, fecha de nacimiento, dirección, teléfono, e-mail.
- Se selecciona un tipo de previsión de la lista y se selecciona un tipo de persona.
- Al momento de presionar aceptar se valida que su fecha de nacimiento no sea mayor a la actual, el correo electrónico sea semejante a example@ejemplo.com, se haya seleccionado un solo tipo de función y que los campos obligatorios no estén vacíos.
- Se guarda el registro en la tabla Persona
- Anexo, página 65.

Diseño de pantalla:



Fuente: Elaborado en conjunto por José Roco y Héctor Sandoval.

Figura 3-2 Ingresar Persona

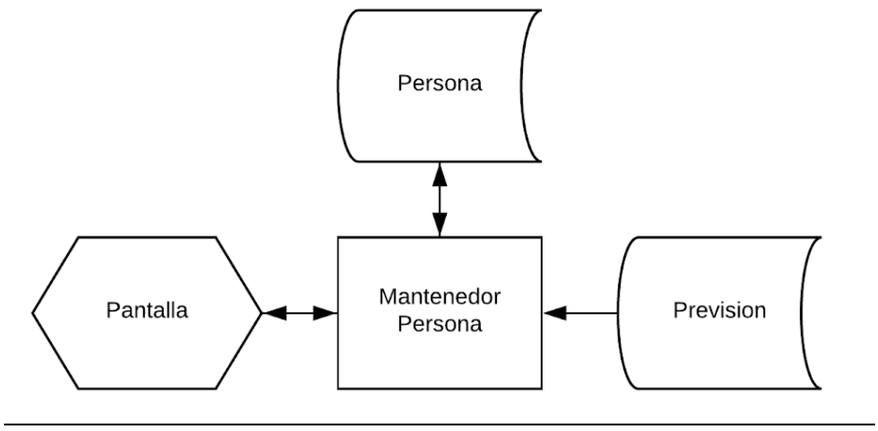
3.3.3. Mantenedor Persona

Nombre lógico: Mantenedor Persona.

Nombre físico: MantendorPersona.java

Objetivo: Modificar, eliminar y consultar una persona (Exceptuando alumnos) del jardín.

Diagrama de bloques:



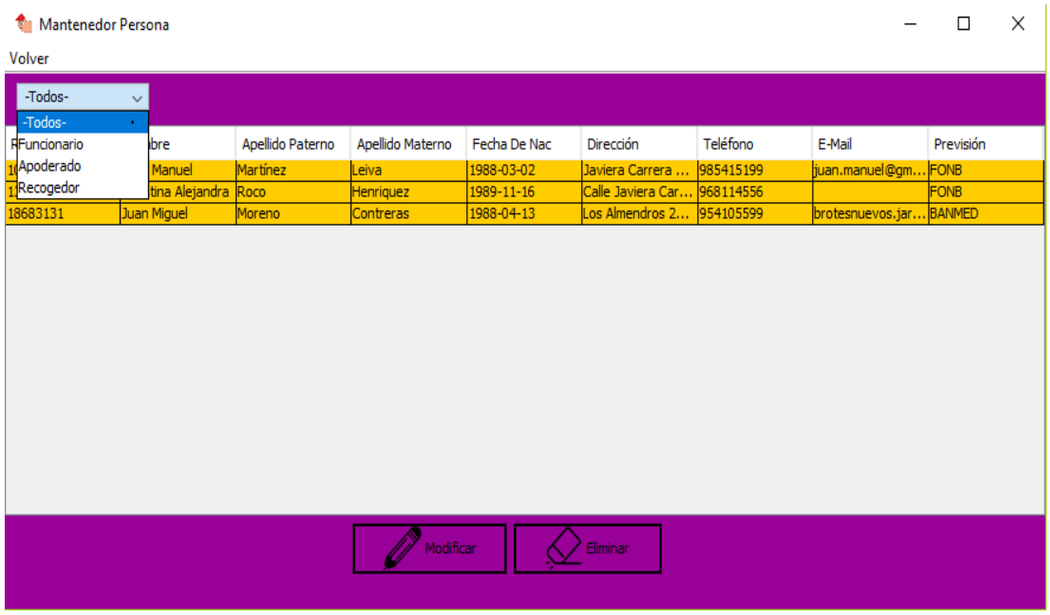
Fuente: Elaborado en conjunto por José Roco y Héctor Sandoval.

Diagrama 3-6 Mantenedor Persona

Reglas de proceso:

- Se presenta pantalla figura 3-3.
- Se consulta la tabla Persona y se listan todas las personas ingresadas en el sitema con sus respectivos datos.
- Al seleccionar del filtro una opción la lista se vuelve a cargar mostrando sólo los datos de las personas discriminadas por el filtro.
- Al presionar el botón modificar se valida que antes se haya seleccionado un registro de la lista y de ser asi se procede la ventana de modificación (figura 3-2) con los datos de la persona.
- Al hacer clic sobre una persona de la lista y presionar eliminar, se pregunta si realmente desea borrar el registro para proceder a eliminar. Si el registro se está siendo utilizando en otra sección del sistema se muestra el mensaje de error impidiéndolo.
- Anexo, página 73.

Diseño de pantalla:



Fuente: Elaborado en conjunto por José Roco y Héctor Sandoval.

Figura 3-3 Mantenedor Persona

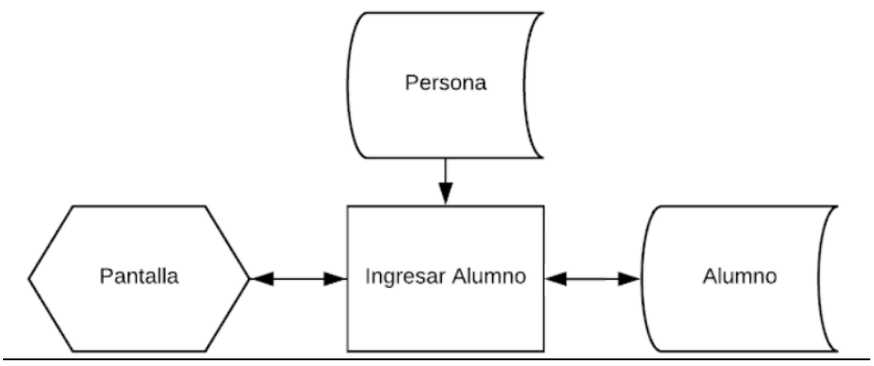
3.3.4. Ingresar Alumno

Nombre lógico: Ingresar Alumno.

Nombre físico: IngresarAlumno.java

Objetivo: Ingresar un alumno al sistema.

Diagrama de bloques:



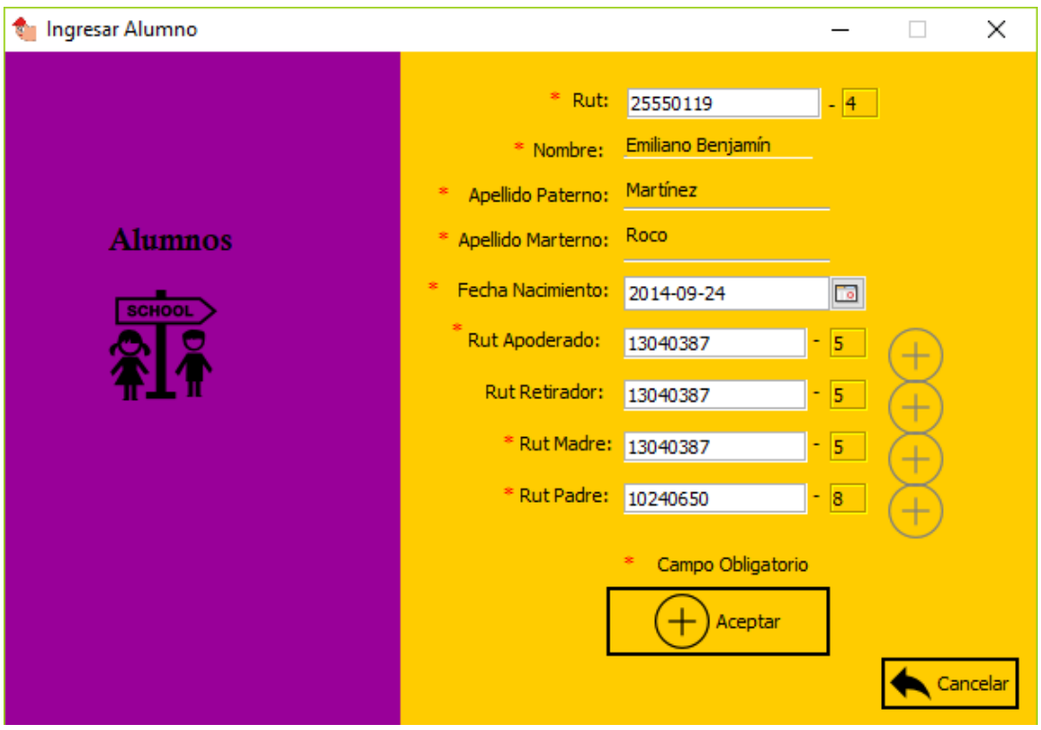
Fuente: Elaborado en conjunto por José Roco y Héctor Sandoval.

Diagrama 3-7 Ingresar Alumno

Reglas de proceso:

- Se presenta pantalla figura 3-4.
- Despliega estructura para completar datos de los alumnos.
- Se ingresar rut del alumno, al momento de perder el foco el rut sin dv se valida que éste no exista en la tabla Alumno. Se digita el dígito verificador y se valida que este sea correcto.
- Se ingresa Nombre, Apellido Paterno, Apellido Paterno, Fecha Ncimiento.
- Se ingresa Rut Apoderado, Rut Retirador, Rut Madre, Rut Padre, los cuales al momento de ser digitados se valida que existan en el sistema consultando la tabla Persona, de no ser así se permite ingresarlos presionando el boton con simbolo "+" correspondiente a cada uno para mostrar la pantalla correspondiente (Figura 3-2). Se validan los Rut's de las personas indicadas del mismo modo que Rut alumno.
- Al presionar aceptar se valida que la Fecha Nacimiento sea menor a la actual, que el Rut Padre y Rut Madre sean distintos y que los campos obligatorios no sean vacíos.
- Se guarda el registro en la tabla Alumno.
- Anexo, página 75.

Diseño de pantalla:



Fuente: Elaborado en conjunto por José Roco y Héctor Sandoval.

Figura 3-4 Ingresar Alumno

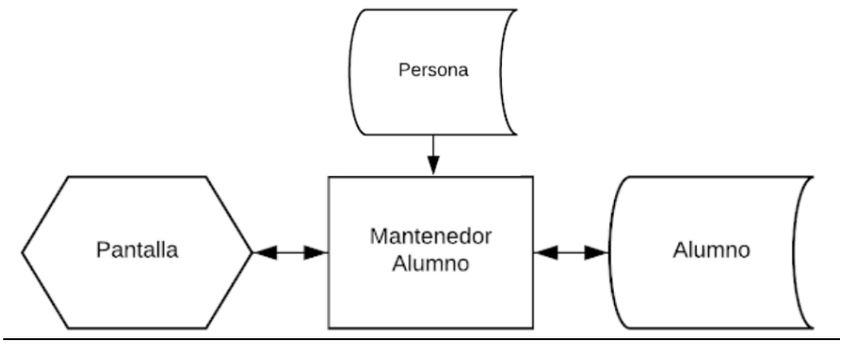
3.3.5. Mantenedor Alumno

Nombre lógico: Mantenedor Alumno.

Nombre físico: MantenedorAlumno.java

Objetivo: Modificar, consultar y eliminar un alumno previamente ingresado en el sistema.

Diagrama de bloques:



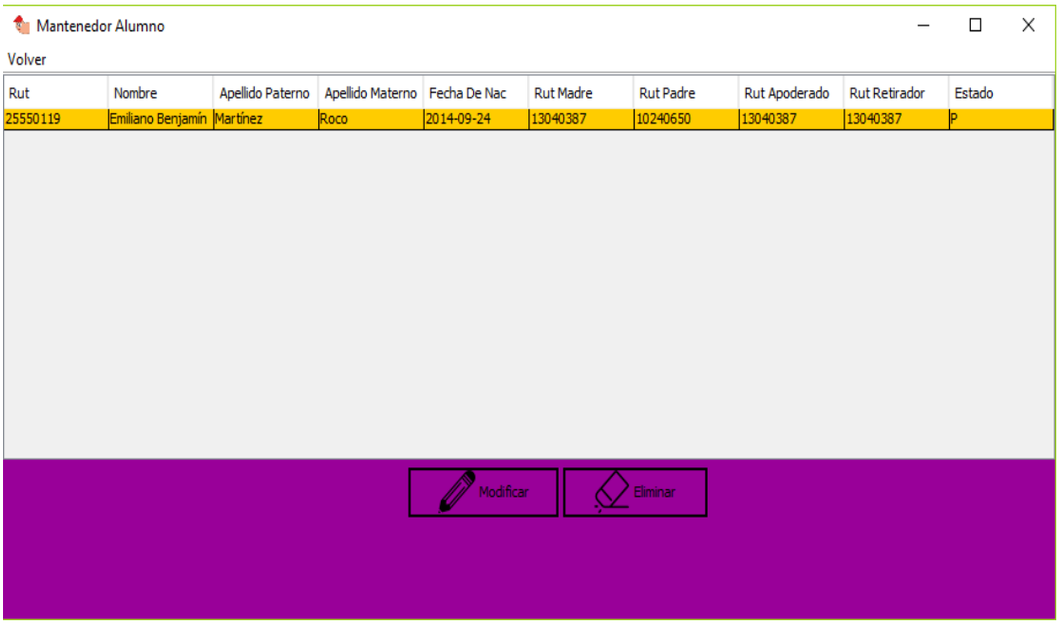
Fuente: Elaborado en conjunto por José Roco y Héctor Sandoval.

Diagrama 3-8 Mantenedor Alumno

Reglas de proceso:

- Se presenta pantalla figura 3-5.
- Se consulta la tabla Alumno y Persona y se listan todos los alumnos ingresados en el sistema con sus respectivos datos.
- Se selecciona un registro y al presionar modificar, se valida que se haya seleccionado un registro de la lista y se procede a la pantalla de modificar (figura 3-4).
- Al momento de seleccionar un registro y presionar eliminar, se valida que se haya seleccionado un registro de la lista, y luego se consulta si realmente desea eliminar el registro.
- Anexo, página 75.

Diseño de pantalla:



Fuente: Elaborado en conjunto por José Roco y Héctor Sandoval.

Figura 3-5 Mantenedor Alumno

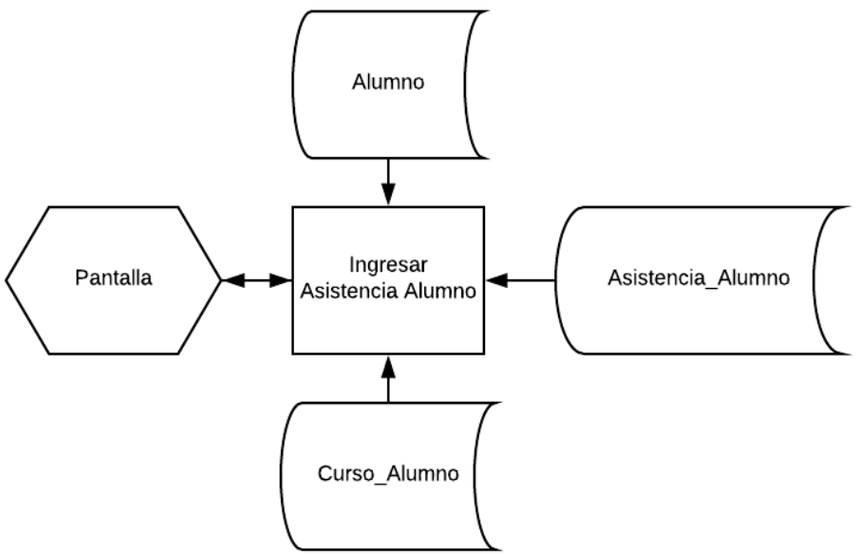
3.3.6. Ingresar Asistencia Alumno

Nombre lógico: Ingresar Asistencia Alumno.

Nombre físico: IngresarAsistenciaAlumno.java

Objetivo: Ingresar la asistencia de un alumno en una fecha determinada.

Diagrama de bloques:



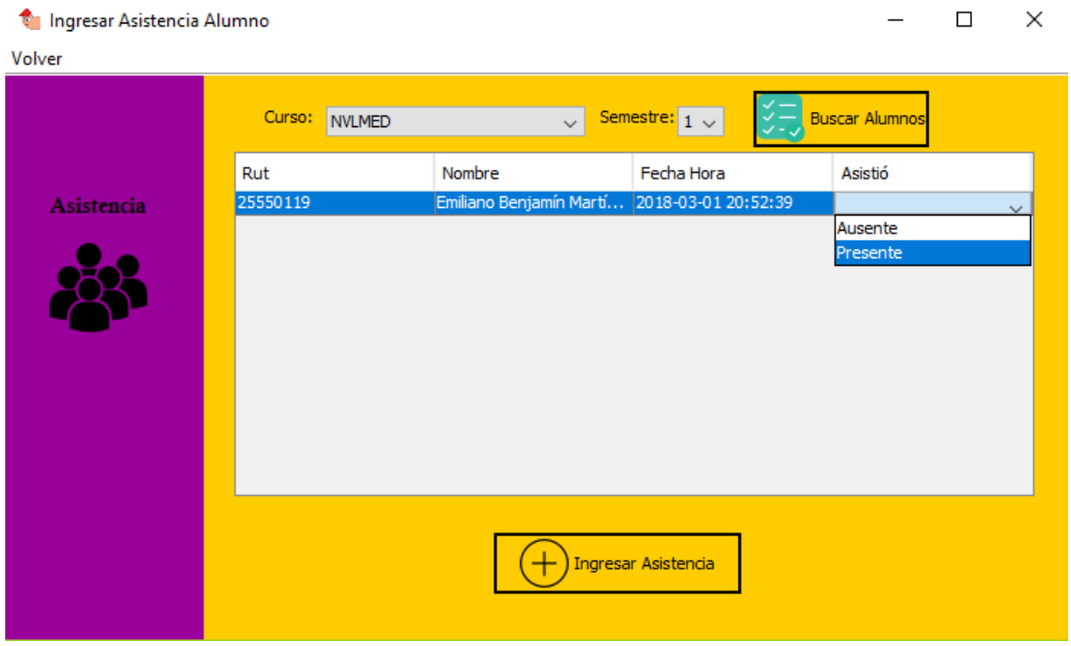
Fuente: Elaborado en conjunto por José Roco y Héctor Sandoval.

Diagrama 3-9 Ingresar Asistencia Alumno

Reglas de proceso:

- Se presenta pantalla figura 3-6.
- Se carga la lista despleganble con todos los cursos existentes en la tabla Curso.
- Se selecciona un curso y un semestre en particular para posteriormente presionar Buscar Alumnos. Se consulta la tabla Curso_Alumno, Alumno y se despliega la lista con los datos de los alumnos de dicho curso.
- Se procede a seleccionar de la lista desplegable si asistió el alumno o no, se debe hacer alumno por alumno discriminando Ausente o Presente (Cargados automáticamente al momento de generarse el listado).
- Presionando el botón Ingresar Asistencia se valida previamente que todos los alumnos estén con su asistencia discriminada.
- Se registra la asistencia de los alumnos quedando guardadas en la tabla Asistencia_Alumno.
- Anexo, página 77.

Diseño de pantalla:



Fuente: Elaborado en conjunto por José Roco y Héctor Sandoval.

Figura 3-6. Ingresar Asistencia Alumno

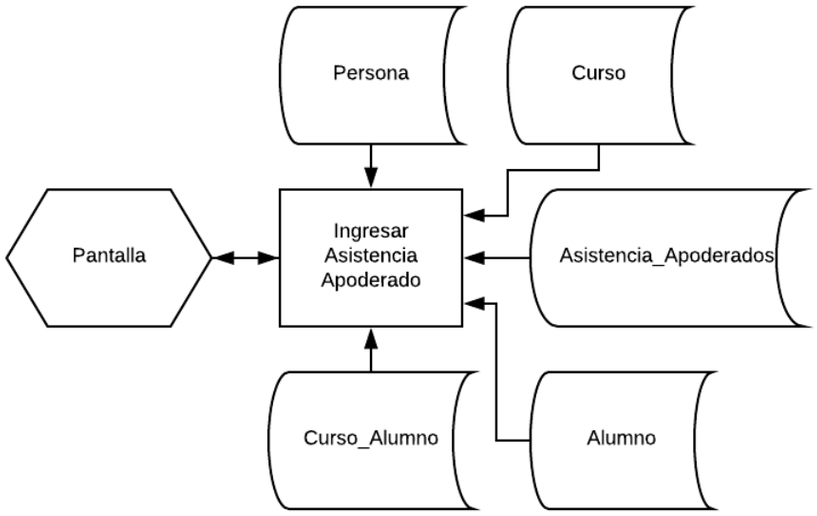
3.3.7. Ingresar Asistencia Apoderado

Nombre lógico: Ingresar Asistencia Apoderado.

Nombre físico: IngresarAsistenciaApoderado.java

Objetivo: Ingresar la asistencia de un apoderado en una fecha determinada.

Diagrama de bloques:



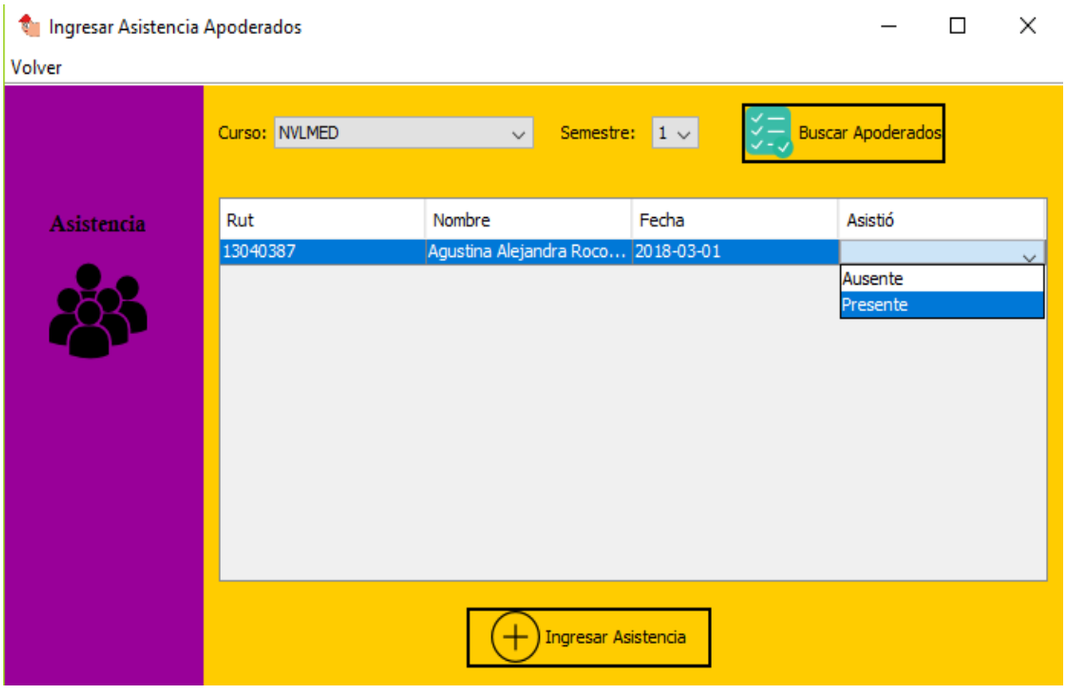
Fuente: Elaborado en conjunto por José Roco y Héctor Sandoval.

Diagrama 3-10 Ingresar Asistencia Apoderado

Reglas de proceso:

- Se presenta pantalla figura 3-7.
- Se consulta la tabla Curso y se cargan todos los cursos disponibles en la lista desplegable correspondiente.
- Se selecciona un semestre en particular y posteriormente se cliquea Buscar Apoderados esto consulta las tablas Curso_Alumno, Alumno, Persona según los criterios y los lista.
- Se selecciona Ausente o Presente de la lista desplegable de Asistió (cargada al momento de generarse la lista).
- Al momento de presionar Ingresar Asistencia se valida que no hayan apoderados sin ser discriminados por Ausente Presente.
- Se resgistra la asistencia en la tabla Asistencia_Apoderados.
- Anexo, página 80.

Diseño de pantalla:



Fuente: Elaborado en conjunto por José Roco y Héctor Sandoval.

Figura 3-7. Ingresar Asistencia Apoderados

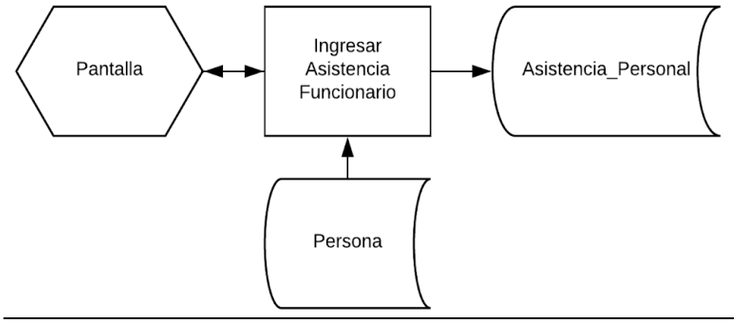
3.3.8. Ingresar Asistencia Funcionarios

Nombre lógico: Ingresar Asistencia Funcionarios.

Nombre físico: IngresarAsistenciaFuncionario.java

Objetivo: Ingresar la asistencia de un funcionario en una fecha determinada.

Diagrama de bloques:



Fuente: Elaborado en conjunto por José Roco y Héctor Sandoval.

Diagrama 3-11 Ingresar Asistencia Funcionario

Reglas de proceso:

- Se presenta pantalla figura 3-8.
- Se digita el Rut de la persona, validando que el dígito verificador sea correcto.
- Al momento de presionar el boton con el simbolo “+” se consulta la tabla Persona validando que exita el funcionario del establecimiento.
- Se consulta la tabla Asistencia_Personal para poder asignar si es una entrada o salida. Si es una entrada se registra sólo la hora de entrada, de no ser el registro correspondiente es modificado guardando la hora de salida.
- Se registra la entrada o salida.
- Anexo, página 83.

Diseño de pantalla:

Fuente: Elaborado en conjunto por José Roco y Héctor Sandoval.

Figura 3-8. Ingresar Asistencia Funcionario

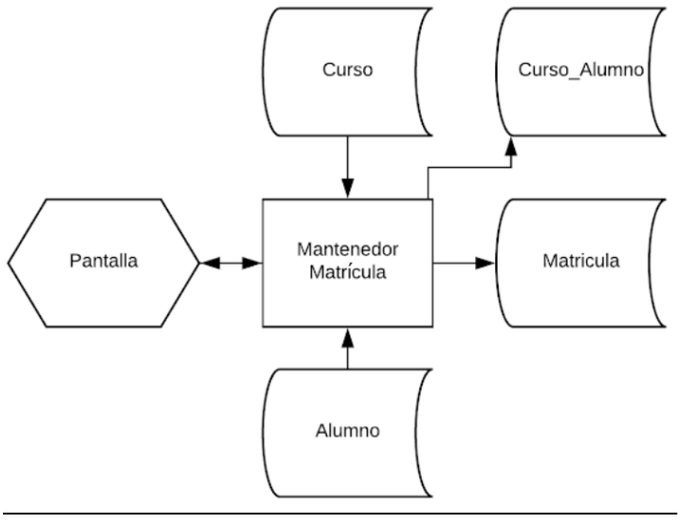
3.3.9. Generar Matrícula

Nombre lógico: Generar matrícula.

Nombre físico: MatriculaFrm.java

Objetivo: Ingresar al sistema una nueva matrícula a partir de un alumno.

Diagrama de bloques:



Fuente: Elaborado en conjunto por José Roco y Héctor Sandoval.

Diagrama 3-12 Generar Matrícula

Reglas de proceso:

- Se presenta pantalla figura 3-9.
- Se consulta la tabla Curso para poder cargar todos los cursos existentes en la lista desplegable.
- Se ingresa el Rut del alumno validando que exista previemante y el número verificador sea correcto y se despliega el nombre del alumno correspondiente.
- Se selecciona el curso y semestre correspondiente al que se quiera matricular al alumno.
- Al presionar Generar se valida, que no se matricule al alumno en un mismo curso y semestre en el mismo año.
- Permite generar la matrícula y se guarda en la tabla histórica (curso_alumno) de los alumnos matriculados con los datos correspondientes.
- Anexo, página 88.

Diseño de pantalla

Fuente: Elaborado en conjunto por José Roco y Héctor Sandoval.

Figura 3-9. Generar Matrícula

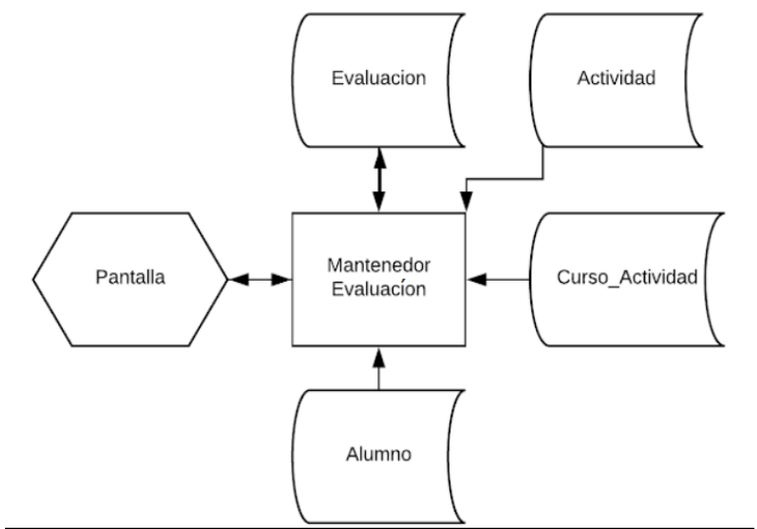
3.3.10. Mantenedor Evaluación

Nombre lógico: Ingresar Modificar Listar Evaluación alumnos.

Nombre físico: IngresarModificarListarEvaluacion.java

Objetivo: Permite ingresar o modificar una evaluación del sistema.

Diagrama de bloques:



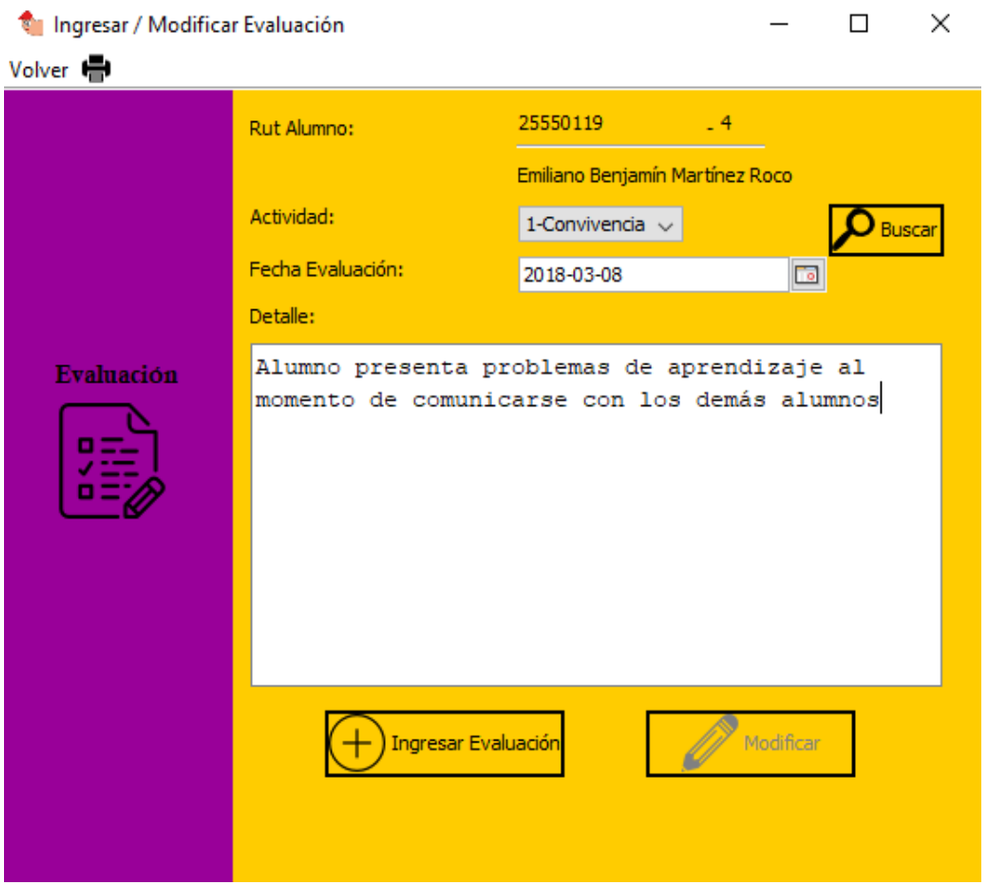
Fuente: Elaborado en conjunto por José Roco y Héctor Sandoval.

Diagrama 3-13 Mantenedor Evaluación

Reglas de proceso:

- Se presenta pantalla figura 3-10.
- Se ingresar el rut del alumno con su dígito verificador.
- Se valida que el alumno exista consultando la tabla Alumno y se despliega el nombre correspondiente si éste existe.
- Se selecciona la actividad a evaluar, cargada desde la ta union de tablas Curso_actividad y Curso.
- Se presiona el botón buscar para desplegar posibles evaluaciones ya existentes desde la tabla Evaluacion.
- Si no existe una evaluación para esa actividad se desbloquea la fecha para poder ingresar la fecha en que se realizó dicha actividad permitiendo también ingresar un detalle u observación sobre la actividad respecto al alumno.
- Al Guardar el registro se valida que la actividad sea correspondiente al curso donde el alumno está matriculado.
- Si el alumno ya fue evaluado para los criterios de busqueda, se despliegan los datos de la actividad, fecha evaluación y detalle permitiendo modificar estos útlimos dos.
- Permite ingresar la evaluación del alumno si todos los datos están correctos almacenandolos en la tabla Evaluacion.
- Anexo, página 92.

Diseño de pantalla:



Fuente: Elaborado en conjunto por José Roco y Héctor Sandoval.

Figura 3-10. Mantenedor Evaluación

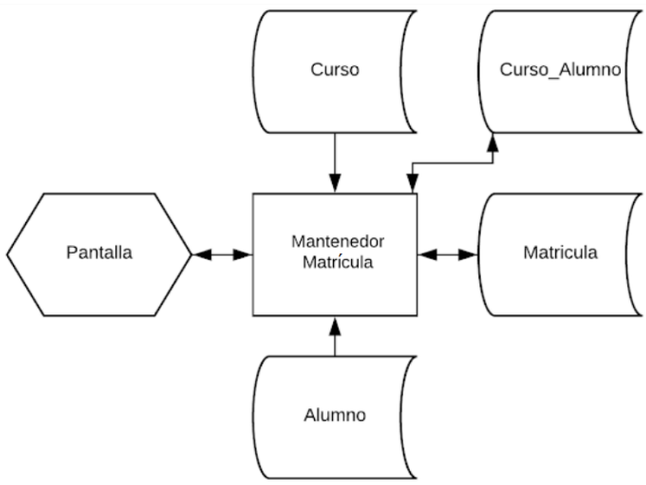
3.3.11. Mantenedor Matrícula

Nombre lógico: Mantenedor matrícula.

Nombre físico: MatriculaFrm.java

Objetivo: Modificar una matrícula existente en el sistema.

Diagrama de bloques:



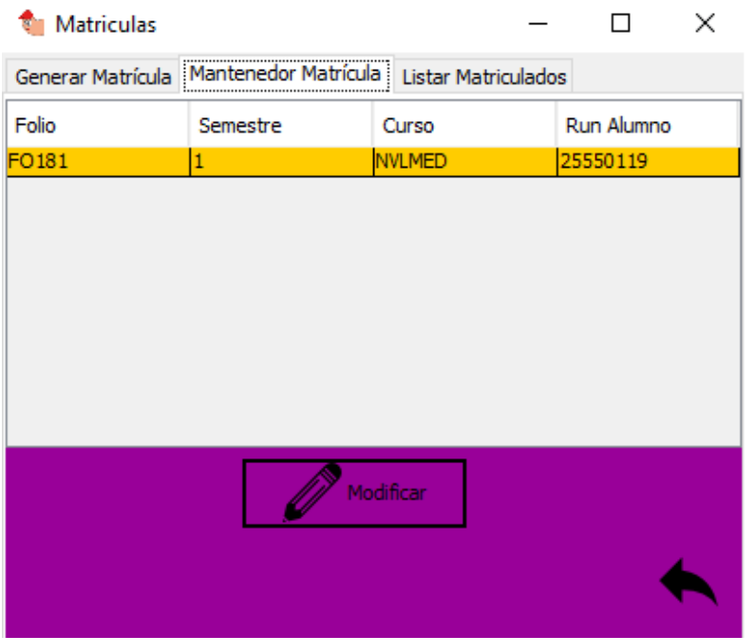
Fuente: Elaborado en conjunto por José Roco y Héctor Sandoval.

Diagrama 3-14. Mantenedor Matrícula

Reglas de proceso:

- Se presenta pantalla figura 3-11.
- Se cargan todos los datos de la tabla Matricula, Alumno, Curso y Curso_Alumno.
- Despliega estructura y lista todas las matrículas con sus respectivos datos.
- Al dar clic sobre Modificar permite el ingreso a la pantalla modal para modificar una matrícula validando que se haya seleccionado una de la lista mostrada.
- Anexo, página 88.

Diseño de pantalla:



Fuente: Elaborado en conjunto por José Roco y Héctor Sandoval.

Figura 3-11. Mantenedor Matrícula

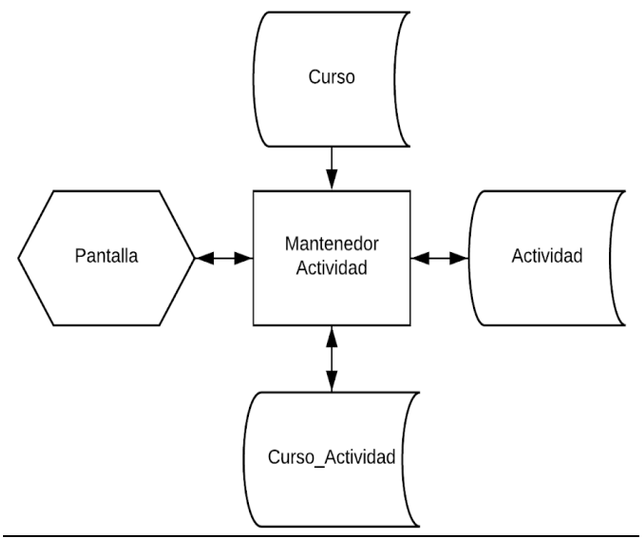
3.3.12. Mantenedor Actividad

Nombre lógico: Mantenedor actividad.

Nombre físico: MantenedorActividad.java

Objetivo: Permite modificar o eliminar las actividades ingresadas al sistema.

Diagrama de bloques:



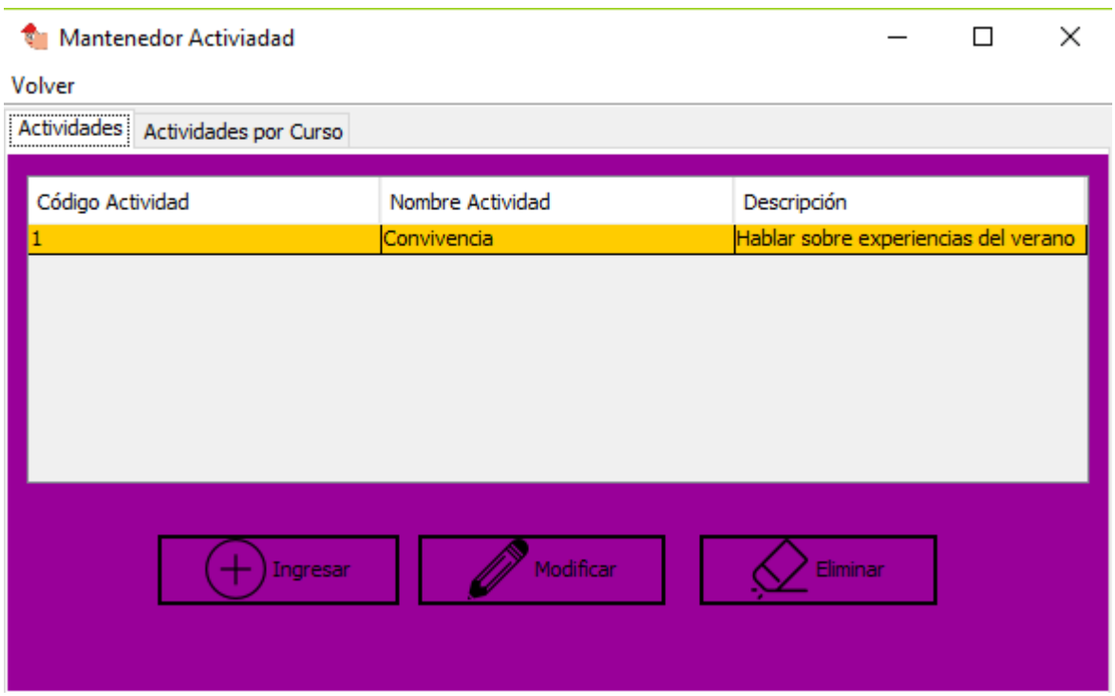
Fuente: Elaborado en conjunto por José Roco y Héctor Sandoval.

Diagrama 3-15. Mantenedor Actividad

Reglas de proceso:

- Se presenta pantalla figura 3-12.
- Despliega estructura y lista las actividades ingresadas en el sistema con sus respectivos datos extraidos desde Curso, Actividad y Curso Actividad.
- Al hacer clic sobre Ingresar permite acceder a la pantalla modal para ingresar una nueva actividad.
- Al hacer clic sobre Modificar permite acceder a la pantalla modal para modificar a la actividad que se elija desde la tabla validanto que previamente se haya seleccionado un registro desde la lista.
- Permite eliminar una actividad seleccionandola previemente desde la tabla, se valida que realmente se haya selccionado un registro.
- Anexo, página 97

Diseño de pantalla:



Fuente: Elaborado en conjunto por José Roco y Héctor Sandoval.

Figura 3-12. Mantenedor Actividad

CONCLUSIONES Y OBSERVACIONES

El campo de la crianza y educación cambia y se remodela continuamente, y de igual manera lo hacen las organizaciones para mantener una competencia de calidad, y para esto las metodologías cambian, las tecnologías avanzan y la gente se adapta, por esto se crean sistemas que den soporte, solucionen las necesidades y aceleren metodologías de trabajo.

Este trabajo nace con ese propósito, solucionar necesidades que aún durante su ciclo de vida se vieron afectados por el tiempo, por esto durante el desarrollo los estudios de requerimientos, se vieron pulidos, aclarados y conversados por las partes involucradas. Este método de retroalimentación de información se aplica a lo largo del desarrollo del sistema, agregando o eliminando elementos y replanteando formas para abordar una necesidad.

A lo largo de este trabajo se desarrollaron distintos tipos de documentos que en un inicio no se ven muy relevantes y que al pasar los procesos de trabajo toman más y más importancia al ser el soporte de todo el trabajo futuro, y la base de las decisiones que se toman, para completar el sistema, desde creando un modelo de datos conteniendo los datos con lo que se tendra que trabajar, hasta un diagrama de bloques de un proceso para mostrar cómo interactúa un proceso en el sistema, toman importancia no solo en el momento que realizamos este proyecto sino que también en el futuro para una mejor comprensión del trabajo que se realizó y que se quiere mejorar o ampliar.

El desarrollo de la programación se llevó a cabo en un periodo aproximado de 3 meses, en el cual, se tuvo que aprender cosas nuevas sobre el lenguaje y las herramientas que utilizamos para dar un mejor desempeño al programa. Aunque existieron ciertas dificultades en esta etapa, se logró desarrollar un sistema completo y funcional, con entrega de varios informes de diferentes características y los requerimientos descritos por el usuario.

BIBLIOGRAFÍA

Biblioteca Universidad Federico Santa María Sede José Miguel Carrera
trabajos de títulos varios.

Comunidad de Desarrollo Stack Overflow. [en línea] [Consulta: Septiembre 2017]. Disponible en https://stackoverflow.com/questions/1988570/how-to-catch-a-specific-exception-in-jdbc

ANEXOS

Inicio Sesión

```
public class InicioSesion extends javax.swing.JFrame {

    public InicioSesion() {
        initComponents();
        this.setLocationRelativeTo(null);
        RestrictedTextField rContrasena = new RestrictedTextField(TXT_PASSWORD);//,
"1234567890qwertyuiopasdfghjklÒzxcvbnmQWERTYUIOPASDFGHJKL—
ZXCVBNM");
        rContrasena.setLimit(20);
        RestrictedTextField rUsuario = new RestrictedTextField(TXT_USER);
        rUsuario.setLimit(10);
        //rUsuario.setOnlyText(true);
    }

    private void BTN_ACCEDERActionPerformed(java.awt.event.ActionEvent evt) {
        String sql = "";
        if (!(TXT_USER.getText().equals(""))) {
            try {
                char[] psw = TXT_PASSWORD.getPassword();
                String psw2 = "";
                for (int i = 0; i < psw.length; i++) {
                    psw2 = psw2 + psw[i];
                }
                if (psw2.equals("")) {
                    sql = "select user,tipo_permitido from usuario where user = '" +
TXT_USER.getText() + "' and contrasena = '" + psw2 + "';;
                } else {
                    sql = "select user,contrasena,tipo_permitido from usuario where user = '" +
TXT_USER.getText() + "' and "
                            + "contrasena = '" + psw2.trim() + "';
                }
                BD conectar = new BD();
                ResultSet lista = conectar.buscrAlgo(sql);
                if (lista.next()) {

                    if (lista.getString("tipo_permitido").equals("2")) {
```

```
                    PantallaInicio inicio = new PantallaInicio();
                    inicio.setVisible(true);
                } else {
                    System.out.println(lista.getString("tipo_permitido"));
                    PantallaInicio inicio = new PantallaInicio("2");
                    inicio.setVisible(true);
                }
                this.dispose();
            } else {
                JOptionPane.showMessageDialog(null, "ContraseÒa errÛnea", "Alerta",
JOptionPane.INFORMATION_MESSAGE);
            }

        } catch (Exception e) {
            JOptionPane.showMessageDialog(null, "Error en contraseÒa y/o usuario",
"Alerta", JOptionPane.INFORMATION_MESSAGE);
        }
    } else {
        if (TXT_USER.getText().equals("")) {
            JOptionPane.showMessageDialog(null, "Debe ingresar usuario", "Alerta",
JOptionPane.INFORMATION_MESSAGE);
        }
    }
  }


  private void TXT_USERKeyTyped(java.awt.event.KeyEvent evt) {
Character s;
    s = evt.getKeyChar();
    if(!Character.isLetterOrDigit(s)  && s != KeyEvent.VK_BACK_SPACE){
        System.out.println(evt.getExtendedKeyCode());
        evt.consume();
    }
  }

  private void TXT_PASSWORDKeyTyped(java.awt.event.KeyEvent evt) {
Character s;
    s = evt.getKeyChar();
```

```
        if(!Character.isLetterOrDigit(s)  && s != KeyEvent.VK_BACK_SPACE){
            System.out.println(evt.getExtendedKeyCode());
            evt.consume();
        }
    }
```

Ingresar Persona

```
public class IngresarPersona extends javax.swing.JFrame {

    public IngresarPersona(Persona persona,String algo) {
        initComponents();
        tipo=algo;
        this.setLocationRelativeTo(null);
        llenarPrevision();

        RestrictedTextField rRut = new RestrictedTextField(TXT_RUT);
        rRut.setOnlyNums(true);
        rRut.setLimit(8);

        RestrictedTextField rDv = new RestrictedTextField(TXT_DV, "1234567890kK");
        rDv.setLimit(1);

        RestrictedTextField rApellidoPaterno = new
RestrictedTextField(TXT_APEL_PAT);
        rApellidoPaterno.setLimit(30);
        rApellidoPaterno.setOnlyText(true);

        RestrictedTextField rApellidoMaterno = new
RestrictedTextField(TXT_APEL_MAT);
        rApellidoMaterno.setLimit(30);
        rApellidoMaterno.setOnlyText(true);
```

```
        RestrictedTextField rDireccion = new RestrictedTextField(TXT_DIRECCION);
        rDireccion.setLimit(100);
        //rDireccion.setAcceptSpace(true);

        RestrictedTextField rTelefono = new RestrictedTextField(TXT_TELEFONO);
        rTelefono.setOnlyNums(true);
        rTelefono.setLimit(9);

        BTN_CANCELAR.setText("Cancelar");
        TXT_RUT.setText(String.valueOf(persona.getRut_persona()));
        Calculos cal = new Calculos(String.valueOf(persona.getRut_persona()));
        TXT_DV.setText(String.valueOf(cal.CalcularDV()));
        if(cal.CalcularDV()==11){
          TXT_DV.setText("0");
        }
        if(cal.CalcularDV()==10){
          TXT_DV.setText("k");
        }
        TXT_NOMBRE.setText(persona.getNombre_per());
        TXT_APEL_PAT.setText(persona.getApellido_pat_per());
        TXT_APEL_MAT.setText(persona.getApellido_mat_per());
        DATE_FORM.setDate(persona.getFecha_nacimiento_per());
        TXT_DIRECCION.setText(persona.getDireccion());
        TXT_TELEFONO.setText(persona.getTelefono());
        TXT_CORREO.setText(persona.getCorreo());
        CMB_PREV.setSelectedItem(persona.getCod_prevision());
        String sql = "Select tipo from persona where rut_persona = " +
persona.getRut_persona();
        String tipo = "";
        try {
            ResultSet lista = conectar.buscarAlgo(sql);
            if (lista.next()) {
                tipo = lista.getString(1);
            }
        } catch (Exception e) {
            System.out.println("NoposeeTipo");
        }
        if (tipo.equals("A")) {
```

```
            CHK_APODERADO.setSelected(true);
        } else if (tipo.equals("F")) {
            CHK_FUNCIONARIO.setSelected(true);
        } else if (tipo.equals("R")) {
            CHK_RECOGEDOR.setSelected(true);
        }
        TXT_RUT.setEnabled(false);
        TXT_DV.setEnabled(false);
    }

    private void llenarPrevision() {
        String sql = "select * from prevision";
        try {
            ResultSet lista = conectar.buscarAlgo(sql);
            while (lista.next()) {
                CMB_PREV.addItem(lista.getString("nombre_prevision"));
            }
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(null, "Problemas de ConexiÛn",
"B⁄SQUEDA", JOptionPane.INFORMATION_MESSAGE);
        }
    }

    private void BTN_CANCELARActionPerformed(java.awt.event.ActionEvent evt) {
        if(tipo=="per"){
            this.dispose();
            Buscar pant=new Buscar();
            pant.setVisible(true);
            return;
        }
        if (BTN_CANCELAR.getText().equals("Cancelar")) {
            if (TXT_RUT.isEnabled()) {
                this.dispose();
                PantallaInicio inicio = new PantallaInicio("2");
                inicio.setVisible(true);
            } else {
                this.dispose();
                MantenedorPersona pant = new MantenedorPersona();
```

```
            pant.setVisible(true);
        }
    }
    if (BTN_CANCELAR.getText().equals("Volver")) {
        this.dispose();
        obj2.setVisible(true);
    }
  }

  private void TXT_RUTFocusLost(java.awt.event.FocusEvent evt) {
    String sql = "select rut_persona from persona where rut_persona=" + TXT_RUT.getText();
    try {
        ResultSet lista = conectar.buscarAlgo(sql);
        if (lista.next()) {
            LBL_VER.setText("Rut Existente");
            TXT_RUT.setBackground(Color.red);
        } else {
            LBL_VER.setText("Rut Correcto");
            TXT_RUT.setBackground(Color.white);
        }
    } catch (SQLException e) {
    }
    Calculos calculos = new Calculos(TXT_RUT.getText());
    if (!TXT_RUT.getText().isEmpty()) {
        if (calculos.CalcularDV() == -1) {
            JOptionPane.showMessageDialog(null, "Largo incorrecto", "DV - Incorrecto", JOptionPane.INFORMATION_MESSAGE);
            TXT_RUT.setText("");
            return;
        }
    }

    if (!TXT_DV.getText().isEmpty() && !TXT_RUT.getText().isEmpty()) {
        if (TXT_DV.getText().toUpperCase().equals("K")) {
            if (calculos.CalcularDV() != 10) {
                JOptionPane.showMessageDialog(null, "Error, Numero Verificador Incorrecto", "RUT", JOptionPane.INFORMATION_MESSAGE);
```

```
                TXT_DV.setText("");
                return;
            }
        }
        if (TXT_DV.getText().toUpperCase().equals("0")) {
            if (calculos.CalcularDV() != 11) {
                JOptionPane.showMessageDialog(null, "Error, Numero Verificador Incorrecto", "RUT", JOptionPane.INFORMATION_MESSAGE);
                TXT_DV.setText("");
                return;
            }
        }
        if (!calculos.ValidarDV(TXT_DV.getText(), calculos.CalcularDV())) {
            JOptionPane.showMessageDialog(null, "Error, Numero Verificador Incorrecto", "RUT", JOptionPane.INFORMATION_MESSAGE);
            TXT_DV.setText("");
        }

    }
  }


  private boolean camposVacios() {
    if (TXT_RUT.getText().equals("") || TXT_NOMBRE.getText().equals("") || TXT_APEL_PAT.getText().equals("") || TXT_APEL_MAT.getText().equals("") || TXT_DV.getText().equals("")
            || !(CHK_APODERADO.isSelected() || CHK_FUNCIONARIO.isSelected() || CHK_RECOGEDOR.isSelected())) {
        return false;
    } else {
        return true;
    }
  }
  private boolean validarFecha() {
    String sql = "select curdate() from dual";
    Boolean val = true;
    if (!(DATE_FORM.getDate() == null)) {
        try {
```

```
            //Fecha Del TXT
            String formato = DATE_FORM.getDateFormatString();
            Date date = DATE_FORM.getDate();
            SimpleDateFormat sdf = new SimpleDateFormat(formato);
            String fnacim = String.valueOf(sdf.format(date));
            //Fecha BD
            ResultSet lista = conectar.buscarAlgo(sql);
            lista.next();
            SimpleDateFormat formatoDelTexto = new SimpleDateFormat(formato);
            String strFecha = lista.getString("curdate()");
            Date fecha = null;
            try {
               fecha = formatoDelTexto.parse(strFecha);
               if (fecha.compareTo(date) == -1) {
                  JOptionPane.showMessageDialog(null, "Fecha No Puede Ser Mayor A
La Actual", "Fecha Incorrecta", JOptionPane.INFORMATION_MESSAGE);
                  DATE_FORM.setDate(null);
                  val = false;
               } else {
                  val = true;
               }
            } catch (ParseException ex) {
            }
         } catch (SQLException e) {
         }
      }
      return val;
   }
   private void BTN_ACEPTARActionPerformed(java.awt.event.ActionEvent evt) {
      String pre = "", tipo = "";
      pre = (String) CMB_PREV.getSelectedItem();
      if (!validarFecha()) {
         return;
      }
      if (camposVacios()) {
         try {
            if                    (!((CHK_APODERADO.isSelected()                    &&
CHK_FUNCIONARIO.isSelected())
```

```
                || (CHK_FUNCIONARIO.isSelected() && CHK_RECOGEDOR.isSelected())
                || (CHK_APODERADO.isSelected() && CHK_RECOGEDOR.isSelected()))) {
            if (CHK_APODERADO.isSelected()) {
               tipo = "A";
            }
            if (CHK_FUNCIONARIO.isSelected()) {
               tipo = "F";
            }
            if (CHK_RECOGEDOR.isSelected()) {
               tipo = "R";
            }
         } else {
            JOptionPane.showMessageDialog(null, "Debe Seleccionar Un Tipo Almenos", "Ingresar Persona", JOptionPane.INFORMATION_MESSAGE);
            return;
         }
         String formato = DATE_FORM.getDateFormatString();
         Date date = DATE_FORM.getDate();
         SimpleDateFormat sdf = new SimpleDateFormat(formato);
         String fnacim = String.valueOf(sdf.format(date));
         if (TXT_RUT.isEnabled()) {
            String sql = "INSERT INTO persona (`rut_persona`, `nombre_per`, `apellido_pat_per`, `apellido_mat_per`, `fecha_nacimiento_per`, `direccion`, `telefono`, `correo_electronico`, `cod_prevision`, `tipo`) "
                  + "VALUES ('" + TXT_RUT.getText() + "', "
                  + "'" + TXT_NOMBRE.getText() + "', "
                  + "'" + TXT_APEL_PAT.getText() + "', "
                  + "'" + TXT_APEL_MAT.getText() + "', "
                  + "'" + fnacim + "', "
                  + "'" + TXT_DIRECCION.getText() + "', "
                  + "'" + TXT_TELEFONO.getText() + "', "
                  + "'" + TXT_CORREO.getText() + "', "
                  + "(select cod_prevision from prevision where nombre_prevision='" + pre + "'), "
                  + "'" + tipo + "')";
            conectar.insertar(sql);
```

```
                vaciarCampos();
            } else {
                String sql = "UPDATE persona SET"
                        + " nombre_per='" + TXT_NOMBRE.getText()
                        + "' ,apellido_pat_per='" + TXT_APEL_PAT.getText()
                        + "' ,apellido_mat_per='" + TXT_APEL_MAT.getText()
                        + "' ,fecha_nacimiento_per='" + fnacim
                        + "' ,direccion='" + TXT_DIRECCION.getText()
                        + "' ,telefono='" + TXT_TELEFONO.getText()
                        + "' ,correo_electronico='" + TXT_CORREO.getText()
                        + "' ,cod_prevision = (select cod_prevision from prevision where nombre_prevision='" + pre + "'),"
                        + " tipo='" + tipo
                        + "' WHERE  rut_persona= " + TXT_RUT.getText();
                //System.out.println(sql);
                conectar.modificar(sql);
                vaciarCampos();
            }

            if (BTN_CANCELAR.getText().equals("Cancelar")) {
                this.initComponents();
            }
        } catch (Exception e) {
            JOptionPane.showMessageDialog(null, "Rellene campos Obligatorios", "Ingresar", JOptionPane.INFORMATION_MESSAGE);
        }
    } else {
        JOptionPane.showMessageDialog(null, "Rellene Campos Obligatorios", "Ingresar", JOptionPane.INFORMATION_MESSAGE);
    }
  }
  private void vaciarCampos(){
    TXT_APEL_MAT.setText("");
    TXT_APEL_PAT.setText("");
    TXT_CORREO.setText("");
    TXT_DIRECCION.setText("");
    TXT_NOMBRE.setText("");
    TXT_DV.setText("");
```

```
        TXT_RUT.setText("");
        TXT_TELEFONO.setText("");
    }

    private void TXT_CORREOFocusLost(java.awt.event.FocusEvent evt) {
        // Compiles the given regular expression into a pattern.
        Matcher mather = pattern.matcher(TXT_CORREO.getText());
        if (mather.find() == true) {
            //System.out.println("El email ingresado es v·lido.");
            BTN_ACEPTAR.setEnabled(true);
        } else {
            if (!TXT_CORREO.getText().equals("")) {
                BTN_ACEPTAR.setEnabled(false);
                JOptionPane.showMessageDialog(null, "Error, Formato correo
algo@example.com", "Formato E-mail", JOptionPane.INFORMATION_MESSAGE);
                //System.out.println("El email ingresado es inv·lido.");
            } else {
                BTN_ACEPTAR.setEnabled(true);
            }
        }
    }

    private void TXT_DVFocusLost(java.awt.event.FocusEvent evt) {
        if (TXT_DV.getText().isEmpty()) {
            return;
        }
        Calculos calculos = new Calculos(TXT_RUT.getText());
        if (calculos.CalcularDV() == -1) {
            return;
        }
        if (TXT_DV.getText().toUpperCase().equals("K")) {
            if (calculos.CalcularDV() != 10) {
                JOptionPane.showMessageDialog(null, "Error, Numero Verificador
Incorrecto", "RUT", JOptionPane.INFORMATION_MESSAGE);
                TXT_DV.setText("");
                return;
            }
        }
```

```
        if (TXT_DV.getText().toUpperCase().equals("0")) {
            if (calculos.CalcularDV() != 11) {
                JOptionPane.showMessageDialog(null, "Error, Numero Verificador Incorrecto", "RUT", JOptionPane.INFORMATION_MESSAGE);
                TXT_DV.setText("");
                return;
            }
        }
        if (!calculos.ValidarDV(TXT_DV.getText(), calculos.CalcularDV())) {
            JOptionPane.showMessageDialog(null, "Error, Numero Verificador Incorrecto", "RUT", JOptionPane.INFORMATION_MESSAGE);
            TXT_DV.setText("");
        }
    }

    private void TXT_DIRECCIONKeyTyped(java.awt.event.KeyEvent evt) {
       Character s;
        s = evt.getKeyChar();
        if(!Character.isLetterOrDigit(s)  && s != KeyEvent.VK_BACK_SPACE && s != KeyEvent.VK_SPACE){
            System.out.println(evt.getExtendedKeyCode());
            evt.consume();
        }
    }
```

Mantenedor Persona

```
public class MantenedorPersona extends javax.swing.JFrame {

    private void llenarTabla(String tipo) {
        String sql = "select * from persona where tipo like '" + tipo + "%'";
        System.out.println(sql);
        DefaultTableModel modelo = (DefaultTableModel) TblPersona.getModel();
        modelo.getDataVector().removeAllElements();
        TblPersona.clearSelection();
        try {
            ResultSet lista = conectar.buscarAlgo(sql);
            while (lista.next()) {
                Vector per = new Vector();
```

```
                per.addElement(lista.getInt("rut_persona"));
                per.addElement(lista.getString("nombre_per"));
                per.addElement(lista.getString("apellido_pat_per"));
                per.addElement(lista.getString("apellido_mat_per"));
                per.addElement(lista.getString("fecha_nacimiento_per"));
                per.addElement(lista.getString("direccion"));
                per.addElement(lista.getString("telefono"));
                per.addElement(lista.getString("correo_electronico"));
                per.addElement(lista.getString("cod_prevision"));
                modelo.addRow(per);
            }
            TblPersona.setModel(modelo);
        } catch (IllegalArgumentException e) {
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(null, "Problemas de B˙squeda",
"B⁄SQUEDA", JOptionPane.INFORMATION_MESSAGE);
        }
    }

    private void BTN_MODIFICARActionPerformed(java.awt.event.ActionEvent evt) {
        if (TblPersona.getSelectedRow() != -1) {
            this.dispose();
            IngresarPersona pantalla = new IngresarPersona(persona);
            pantalla.setVisible(true);
        } else {
            JOptionPane.showMessageDialog(null, "Seleccione una curso a Modificar",
"MODIFICAR", JOptionPane.INFORMATION_MESSAGE);
        }
    }

    private void CmbTiposItemStateChanged(java.awt.event.ItemEvent evt) {
        String tipo = CmbTipos.getSelectedItem().toString();
        tipo = tipo.substring(0, 1).trim();
        System.out.println(tipo);
        llenarTabla(tipo);
        TblPersona.repaint();
    }
```

```
    private void TblPersonaFocusLost(java.awt.event.FocusEvent evt) {
        if (TblPersona.getSelectedRow() != -1) {
            int posicion = TblPersona.getSelectedRow();
            persona.setRut_persona(Integer.parseInt(TblPersona.getValueAt(posicion,
0).toString()));
            persona.setNombre_per(String.valueOf(TblPersona.getValueAt(posicion, 1)));
            persona.setApellido_pat_per(String.valueOf(TblPersona.getValueAt(posicion,
2)));
            persona.setApellido_mat_per((String.valueOf(TblPersona.getValueAt(posicion,
3))));

persona.setFecha_nacimiento_per((Date.valueOf(TblPersona.getValueAt(posicion,
4).toString())));
            persona.setDireccion(String.valueOf(TblPersona.getValueAt(posicion, 5)));
            persona.setTelefono(String.valueOf(TblPersona.getValueAt(posicion, 6)));
            persona.setCorreo(String.valueOf(TblPersona.getValueAt(posicion, 7)));
            persona.setCod_prevision(String.valueOf(TblPersona.getValueAt(posicion, 8)));
            persona.setTipo(CmbTipos.getSelectedItem().toString().substring(0, 1));
        }
    }

    private void BTN_ELIMINARActionPerformed(java.awt.event.ActionEvent evt) {
        if (TblPersona.getSelectedRow() != -1) {
            int posicion = TblPersona.getSelectedRow();
            int confirmado = JOptionPane.showConfirmDialog(null, "øEst·s Seguro De
Eliminar?","Eliminar", JOptionPane.OK_CANCEL_OPTION);
            if (JOptionPane.OK_OPTION == confirmado) {
                String sql = "delete from persona where rut_persona = " +
Integer.parseInt(TblPersona.getValueAt(posicion, 0).toString()) ;
                conectar.eliminar(sql);
                TblPersona.repaint();
                llenarTabla("");
                CmbTipos.setSelectedIndex(0);

            }
        } else {
            JOptionPane.showMessageDialog(null, "Seleccione una persona a eliminar",
"ELIMINAR", JOptionPane.INFORMATION_MESSAGE);
```

```
        }
    }
```

Mantenedor Alumno

```
public class MantenedorAlumno extends javax.swing.JFrame {

    private void llenarTabla() {
        String sql = "select * from alumno";
        DefaultTableModel modelo = (DefaultTableModel) TBL_MOD_ALU.getModel();
        modelo.getDataVector().removeAllElements();
        TBL_MOD_ALU.clearSelection();
        try {
            ResultSet lista = conectar.buscarAlgo(sql);
            while (lista.next()) {
                Vector per = new Vector();
                per.addElement(lista.getInt("rut_alu"));
                per.addElement(lista.getString("nombre_alu"));
                per.addElement(lista.getString("apellido_paterno_alu"));
                per.addElement(lista.getString("apellido_materno_alu"));
                per.addElement(lista.getString("fecha_nacimiento_alu"));
                per.addElement(lista.getInt("rut_madre"));
                per.addElement(lista.getInt("rut_padre"));
                per.addElement(lista.getInt("rut_apoderado_of"));
                if (lista.getInt("rut_retirador_of") == 0) {
                    per.addElement("null");
                } else {
                    per.addElement(lista.getInt("rut_retirador_of"));
                }
                per.addElement(lista.getString("estado"));
                modelo.addRow(per);
            }
            TBL_MOD_ALU.setModel(modelo);
        } catch (IllegalArgumentException e) {
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(null, "Problemas de B˙squeda",
"B⁄SQUEDA", JOptionPane.INFORMATION_MESSAGE);
        }
```

```
    }

    private void BTN_ELIMINARActionPerformed(java.awt.event.ActionEvent evt) {
        if (TBL_MOD_ALU.getSelectedRow() != -1) {
            String sql = "delete from alumno where rut_alu = ";
            String rut = TBL_MOD_ALU.getValueAt(TBL_MOD_ALU.getSelectedRow(),
0).toString();
            sql = sql + (String) rut;
            conectar.eliminar(sql);
            TBL_MOD_ALU.clearSelection();
            llenarTabla();
        } else {
            JOptionPane.showMessageDialog(null, "Seleccione un alumno a eliminar",
"ELIMINAR", JOptionPane.INFORMATION_MESSAGE);
        }
    }

    private void BTN_MODIFICARActionPerformed(java.awt.event.ActionEvent evt) {
        if (TBL_MOD_ALU.getSelectedRow() != -1) {
            Alumno alumno = new Alumno();

alumno.setRut_alu(Integer.parseInt(TBL_MOD_ALU.getValueAt(TBL_MOD_ALU.ge
tSelectedRow(), 0).toString()));

alumno.setNombre_alu(TBL_MOD_ALU.getValueAt(TBL_MOD_ALU.getSelectedRo
w(), 1).toString());

alumno.setApellido_paterno(TBL_MOD_ALU.getValueAt(TBL_MOD_ALU.getSelect
edRow(), 2).toString());

alumno.setApellido_materno(TBL_MOD_ALU.getValueAt(TBL_MOD_ALU.getSelect
edRow(), 3).toString());

alumno.setFecha_nacimiento(Date.valueOf(TBL_MOD_ALU.getValueAt(TBL_MOD_
ALU.getSelectedRow(), 4).toString()));

alumno.setRut_apoderado_of(Integer.parseInt(TBL_MOD_ALU.getValueAt(TBL_MO
D_ALU.getSelectedRow(), 7).toString()));
```

```
        if(TBL_MOD_ALU.getValueAt(TBL_MOD_ALU.getSelectedRow(),
8).toString()!="null"){

alumno.setRut_retirador_of(Integer.parseInt(TBL_MOD_ALU.getValueAt(TBL_MOD
_ALU.getSelectedRow(), 8).toString()));
        }

alumno.setRut_padre(Integer.parseInt(TBL_MOD_ALU.getValueAt(TBL_MOD_ALU.
getSelectedRow(), 6).toString()));

alumno.setRut_madre(Integer.parseInt(TBL_MOD_ALU.getValueAt(TBL_MOD_ALU
.getSelectedRow(), 5).toString()));

alumno.setEstado(TBL_MOD_ALU.getValueAt(TBL_MOD_ALU.getSelectedRow(),
9).toString().charAt(0));
        IngresarAlumno pant = new IngresarAlumno(alumno, this);
        pant.setVisible(true);
        this.dispose();
        TBL_MOD_ALU.clearSelection();
        llenarTabla();
      } else {
        JOptionPane.showMessageDialog(null, "Seleccione un alumno a Modificar",
"Modificar", JOptionPane.INFORMATION_MESSAGE);
      }
   }
```

Asistencia Alumno

```
public class IngresarAsistenciaAlumno extends javax.swing.JFrame {
    public IngresarAsistenciaAlumno() {
        initComponents();
        this.setLocationRelativeTo(null);
        llenarCursos();
    }
    public IngresarAsistenciaAlumno(String tipo) {
        permiso=tipo;
        initComponents();
        this.setLocationRelativeTo(null);
        llenarCursos();
    }
    private void llenarCursos(){
        try {
            String sql="select * from curso";
            ResultSet lista = conectar.buscarAlgo(sql);
            while(lista.next()){
                cbmcurso.addItem(lista.getString(1));
            }
        } catch (Exception e) {
            JOptionPane.showMessageDialog(null, "Problemas de B˙squeda",
"B⁄SQUEDA", JOptionPane.INFORMATION_MESSAGE);
        }
    }
private void llenartabla(String curso) {
        String sql = "select
e.rut_alu,a.nombre_alu,a.apellido_paterno_alu,a.apellido_materno_alu from
curso_alumno e,alumno a "
                + "where cod_curso like '" + curso + "%' and anno = (select YEAR(NOW()))
and semestre = " +
                cbxsemestre.getSelectedItem() + "  and e.rut_alu=a.rut_alu" ;
        DefaultTableModel modelo = (DefaultTableModel) jTable1.getModel();
        modelo.getDataVector().removeAllElements();
        jTable1.clearSelection();
        Date date = new Date();
        DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
        String date2 = String.valueOf(dateFormat.format(date));
```

```
        try {

            ResultSet lista = conectar.buscarAlgo(sql);

            while (lista.next()) {
                Vector per = new Vector();
                per.addElement(lista.getInt("rut_alu"));
                per.addElement(lista.getString("nombre_alu")+"                "                +
lista.getString("apellido_paterno_alu")+" "+lista.getString("apellido_materno_alu"));
                per.addElement(date2);
                modelo.addRow(per);
            }
            jTable1.setModel(modelo);
        } catch (IllegalArgumentException e) {
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(null,        "Problemas        de        B˙squeda",
"B⁄SQUEDA", JOptionPane.INFORMATION_MESSAGE);
        }
}

    private void BTN_VOLVERMouseClicked(java.awt.event.MouseEvent evt) {

        if(permiso.equals("")){
        this.dispose();
        Asistencia inicio = new Asistencia("2");
        inicio.setVisible(true);
      }else{
         Asistencia inicio = new Asistencia();
        inicio.setVisible(true);
        this.dispose();
      }
    }

    private void BTN_GUARDARActionPerformed(java.awt.event.ActionEvent evt) {

        int fila=0;
        boolean val=false;
        //Reviso que no haya personas sin su asistencia...
```

```
        while(fila <= jTable1.getRowCount()-1) {
           if(!(jTable1.getValueAt(fila, 3)==null)){
            fila++;
           }else{
            JOptionPane.showMessageDialog(null, "El Alumno " + jTable1.getValueAt(fila, 0) + " no se ha ingresado su asistencia", "Asistencia", JOptionPane.INFORMATION_MESSAGE);
            fila=jTable1.getRowCount()+1;
            val=true;
           }
        }
        //Inserto la asistencia...
        fila=0;
        String m;
        while((fila <= jTable1.getRowCount()-1) && (val==false)) {
           if(jTable1.getValueAt(fila, 3)=="Presente"){
              m="S";
           }else{m="N";}
           String sql = "INSERT INTO asistencia_alumno(rut_alu,fecha_hora,asistio) Values"
                   + "( " + jTable1.getValueAt(fila, 0)
                   + " ,'" + jTable1.getValueAt(fila, 2)
                   + "' ,'" + m + "')";
           conectar.insertar2(sql);
           fila++;
        }
        if(val==false){
           JOptionPane.showMessageDialog(null, "Asistencia Ingresada Correctamente", "Asistencia", JOptionPane.INFORMATION_MESSAGE);
        }
     }

     private void btndesplegarActionPerformed(java.awt.event.ActionEvent evt) {
        try {
        if(cbmcurso.getSelectedItem().equals("Seleccione Curso")){
           JOptionPane.showMessageDialog(null, "Debe seleccionar curso", "Asistencia", JOptionPane.INFORMATION_MESSAGE);
           return;
```

```
        }
        String curso="";
        curso = (String) cbmcurso.getSelectedItem();
        llenartabla(curso);
        } catch (Exception e) {
           JOptionPane.showMessageDialog(null, "Error al desplegar Alumnos",
"Asistencia", JOptionPane.INFORMATION_MESSAGE);
        }
    }
```

Asistencia Apoderado

```
public class IngresarAsistenciaApoderado extends javax.swing.JFrame {

    public IngresarAsistenciaApoderado() {
        initComponents();
        this.setLocationRelativeTo(null);
        llenarCursos();
    }
    public IngresarAsistenciaApoderado(String tipo) {
        permiso=tipo;
        initComponents();
        this.setLocationRelativeTo(null);
        llenarCursos();
    }

    private void llenartabla(String curso) {
        String sql = "select
q.rut_persona,q.nombre_per,q.apellido_pat_per,q.apellido_mat_per from curso_alumno
e,alumno a,persona q "
                + "where cod_curso like '" + curso + "%' and anno = (select YEAR(NOW()))
and semestre = " +
                cbxsemestre.getSelectedItem() + " and e.rut_alu=a.rut_alu and
a.rut_apoderado_of=q.rut_persona" ;
        DefaultTableModel modelo = (DefaultTableModel) jTable1.getModel();
```

```
        modelo.getDataVector().removeAllElements();
        jTable1.clearSelection();
        Date date = new Date();
        DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
        String date2 = String.valueOf(dateFormat.format(date));
        try {

            ResultSet lista = conectar.buscarAlgo(sql);

            while (lista.next()) {
                Vector per = new Vector();
                per.addElement(lista.getInt("rut_persona"));
                per.addElement(lista.getString("nombre_per")+"                "                +
lista.getString("apellido_pat_per")+" "+lista.getString("apellido_mat_per"));
                per.addElement(date2);
                modelo.addRow(per);
            }
            jTable1.setModel(modelo);
        } catch (IllegalArgumentException e) {
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(null,        "Problemas        de        B´squeda",
"B⁄SQUEDA", JOptionPane.INFORMATION_MESSAGE);
        }
    }
     private void llenarCursos(){
        try {
            String sql="select * from curso";
            ResultSet lista = conectar.buscarAlgo(sql);
            while(lista.next()){
                cbmcurso.addItem(lista.getString(1));
            }
        } catch (Exception e) {
            JOptionPane.showMessageDialog(null,        "Problemas        de        B´squeda",
"B⁄SQUEDA", JOptionPane.INFORMATION_MESSAGE);
        }
    }

    private void BTN_VOLVERMouseClicked(java.awt.event.MouseEvent evt) {
```

```
        if(permiso.equals("")){
        this.dispose();
        Asistencia inicio = new Asistencia("2");
        inicio.setVisible(true);
      }else{
        Asistencia inicio = new Asistencia();
        inicio.setVisible(true);
        this.dispose();
      }
    }

    private void btndesplegarActionPerformed(java.awt.event.ActionEvent evt) {
        try {
            if(cbmcurso.getSelectedItem().equals("Seleccione Curso")){
            JOptionPane.showMessageDialog(null, "Debe seleccionar curso", "Asistencia",
JOptionPane.INFORMATION_MESSAGE);
            return;
        }
            //Sacar de aquÌ
            String curso="";
            curso = (String) cbmcurso.getSelectedItem();
            llenartabla(curso);
        } catch (Exception e) {
            JOptionPane.showMessageDialog(null, "Error al desplegar Alumnos",
"Asistencia", JOptionPane.INFORMATION_MESSAGE);
        }
    }

    private void BTN_GUARDARActionPerformed(java.awt.event.ActionEvent evt) {
        int fila=0;
        boolean val=false;
        //Reviso que no haya personas sin su asistencia...
        while(fila <= jTable1.getRowCount()-1) {
            if(!(jTable1.getValueAt(fila, 3)==null)){
             fila++;
            }else{
```

```
            JOptionPane.showMessageDialog(null, "Al apoderado " +
jTable1.getValueAt(fila, 1) + " no se le ha seleccionado su asistencia", "Asistencia",
JOptionPane.INFORMATION_MESSAGE);
             fila=jTable1.getRowCount()+1;
             val=true;
           }
        }
        //Inserto la asistencia...
        fila=0;
        String m;
        while((fila <= jTable1.getRowCount()-1) && (val==false)) {
           if(jTable1.getValueAt(fila, 3)=="Presente"){
              m="S";
           }else{m="N";}
           String sql = "INSERT INTO asistencia_apoderado(rut_apoderado,fecha,asistio)
Values"
                  + "( " + jTable1.getValueAt(fila, 0)
                  + " ,'" + jTable1.getValueAt(fila, 2)
                  + "' ,'" + m + "')";
           conectar.insertar2(sql);
           fila++;
        }
        if(val==false){
           JOptionPane.showMessageDialog(null, "Asistencia Ingresada Correctamente",
"Asistencia", JOptionPane.INFORMATION_MESSAGE);
        }
     }
```

Asistencia Funcionario

```
public class IngresarAsistenciaFuncionario extends javax.swing.JFrame {

    public IngresarAsistenciaFuncionario(String tipo) {
        permiso=tipo;
        initComponents();
```

```
        this.setLocationRelativeTo(null);
        llenarTabla();

        RestrictedTextField rRut = new RestrictedTextField(txtrut);
        rRut.setOnlyNums(true);
        rRut.setLimit(8);

        RestrictedTextField rDv = new RestrictedTextField(txtdv,"1234567890kK");
        rDv.setLimit(1);
    }

    private void llenarTabla() {
        String sql = "select e.nombre_per,e.apellido_pat_per,e.apellido_mat_per,a.hora_inicio,a.hora_fin "
                + "from asistencia_personal a, persona e "
                + "where e.rut_persona = a.rut_personal and fecha = (select date(now())) ";
        try {
            ResultSet lista = conectar.buscarAlgo(sql);
            DefaultTableModel modelo = (DefaultTableModel) tblasistencia.getModel();
            modelo.getDataVector().removeAllElements();
            tblasistencia.clearSelection();
            while (lista.next()) {
                Vector per = new Vector();
                per.addElement(lista.getString("nombre_per") + " " + lista.getString("apellido_pat_per") + " " + lista.getString("apellido_mat_per"));
                per.addElement(lista.getString("hora_inicio"));
                per.addElement(lista.getString("hora_fin"));
                modelo.addRow(per);
            }
        } catch (Exception e) {
        }
    }

    private void BTN_VOLVERMouseClicked(java.awt.event.MouseEvent evt) {
        if(permiso.equals("")){
        this.dispose();
        Asistencia inicio = new Asistencia("2");
        inicio.setVisible(true);
```

```
      }else{
        Asistencia inicio = new Asistencia();
       inicio.setVisible(true);
       this.dispose();
      }
    }

    private void BTN_GUARDARActionPerformed(java.awt.event.ActionEvent evt) {
       AsistenciaPersonal asis = new AsistenciaPersonal();
       if(txtrut.getText().equals("")){
          JOptionPane.showMessageDialog(null, "Debe Ingresar Un Rut", "°Error!", JOptionPane.ERROR_MESSAGE);
          return;
       }
       if(txtdv.getText().equals("")){
          JOptionPane.showMessageDialog(null, "Debe Ingresar DÌgito Verificador", "°Error!", JOptionPane.ERROR_MESSAGE);
          return;
       }
       asis.setRut_personal(Integer.parseInt(txtrut.getText()));

       try {
          String sql = "select * from persona where rut_persona = " + asis.getRut_personal() + " and tipo='F'";
          ResultSet lista = conectar.buscarAlgo(sql);
          if (!lista.next()) {
             JOptionPane.showMessageDialog(null, "Funcionario No Ingresado", "°Error!", JOptionPane.ERROR_MESSAGE);
             txtdv.setText("");
             txtrut.setText("");
             return;
          }
       } catch (Exception e) {
          JOptionPane.showMessageDialog(null, "Buscar ErrÛneo", "°Error!", JOptionPane.ERROR_MESSAGE);
       }
       try {
```

```
        String sql = "select * from asistencia_personal where rut_personal= " + asis.getRut_personal() + " and fecha = (select date(now())) and hora_fin is null";
        ResultSet lista = conectar.buscarAlgo(sql);
        if (lista.next()) {
            if (lista.getString("asistio") == null) {
                sql = "UPDATE asistencia_personal SET "
                        + " hora_fin = (select time(now()))"
                        + ", asistio = 'S' where rut_personal = " + lista.getString("rut_personal")
                        + " and fecha = '" + lista.getString("fecha") + "'";
                conectar.modificar2(sql);
                JOptionPane.showMessageDialog(null, "Salida Marcada", "°AdiÛs!", JOptionPane.INFORMATION_MESSAGE);
            } else {
                JOptionPane.showMessageDialog(null, "Asistencia Ya Marcada Para Salir", "°Adios!", JOptionPane.INFORMATION_MESSAGE);
                sql = "INSERT INTO asistencia_personal (rut_personal,fecha,hora_inicio) VALUES "
                    + "(" + asis.getRut_personal() + ","
                    + "(select date(now())),"
                    + "(select time(now())))";
                conectar.insertar2(sql);
                JOptionPane.showMessageDialog(null, "°A Trabajar Con Una Sonrisa!", "°Buen DÌa!", JOptionPane.INFORMATION_MESSAGE);
            }
        } else {
            sql = "INSERT INTO asistencia_personal (rut_personal,fecha,hora_inicio) VALUES "
                    + "(" + asis.getRut_personal() + ","
                    + "(select date(now())),"
                    + "(select time(now())))";
            conectar.insertar2(sql);
            JOptionPane.showMessageDialog(null, "°A Trabajar Con Una Sonrisa!", "°Buen DÌa!", JOptionPane.INFORMATION_MESSAGE);
        }
        txtrut.setText("");
        txtdv.setText("");
        llenarTabla();
```

```
        } catch (Exception e) {
            JOptionPane.showMessageDialog(null, "Error Al Marcar Asistencia " + e,
"°Error!", JOptionPane.ERROR_MESSAGE);
        }
    }

    private void txtdvFocusLost(java.awt.event.FocusEvent evt) {
        if(txtrut.getText().isEmpty()){
            return;
        }
        if(txtdv.getText().isEmpty()){
            return;
        }
        Calculos cal = new Calculos(txtrut.getText());
        int dv_v = cal.CalcularDV();
        System.out.println();
        if (!cal.ValidarDV(txtdv.getText(), dv_v)) {
            JOptionPane.showMessageDialog(null, "DÌgito Verificador Incorrecto",
"°Error!", JOptionPane.ERROR_MESSAGE);
            txtdv.selectAll();
            BTN_GUARDAR.setEnabled(false);
        } else {
            BTN_GUARDAR.setEnabled(true);}}

public class IngModActividad extends javax.swing.JFrame {

    private void TxtCodigoFocusLost(java.awt.event.FocusEvent evt) {
        actividad.setCod_actividad(TxtCodigo.getText());
    }

    private void TxtNombreFocusLost(java.awt.event.FocusEvent evt) {
        actividad.setNombre_actividad(TxtNombre.getText());
    }

    private void BtnAceptarActionPerformed(java.awt.event.ActionEvent evt) {
            String cod_act="";
            actividad.setDescripcion(TxtDetalle.getText());
            System.out.println("3");
```

```
        if(actividad.getNombre_actividad().equals("")){
          JOptionPane.showMessageDialog(null, "Ingrese Nombre De La Actividad",
"Actividad", JOptionPane.ERROR_MESSAGE);
          return;
        }
        if(actividad.getDescripcion().equals("")){
          JOptionPane.showMessageDialog(null, "Ingrese Descripcion De La Actividad",
"Actividad", JOptionPane.ERROR_MESSAGE);
          return;
        }
        try {
          ResultSet lista = conectar.buscarAlgo("select count(*)+1 from actividad");
          if(lista.next()){
            cod_act = lista.getString("count(*)+1");
            actividad.setCod_actividad(cod_act);
          }
      } catch (Exception e) {
        JOptionPane.showMessageDialog(null, "Error autoincremento", "Actividad",
JOptionPane.ERROR_MESSAGE);
      }
        if (actividad.getCod_actividad()!= ""
            || actividad.getDescripcion() != ""
            || actividad.getNombre_actividad()!= "") {
          if (TxtCodigo.isVisible()) {
            String sql = "UPDATE actividad SET"
                + " nombre_actividad='" + actividad.getNombre_actividad()
                + "' ,descripcion='" + actividad.getDescripcion()
                + "' WHERE  cod_actividad= '" + actividad.getCod_actividad()+ "'";
            conectar.modificar2(sql);
            mensaje = "Modificacion exitosa";
          } else {
            String sql = "INSERT INTO actividad (`cod_actividad`,
`nombre_actividad`, `descripcion`) VALUES ("
                + cod_act + ",'" + actividad.getNombre_actividad()+ "','" +
actividad.getDescripcion() + "')";
            mensaje = "Ingreso Realizado";
            conectar.insertar2(sql);
          }
```

```
            } else {
                if (mensaje == "") {
                    mensaje = "Campos obligatorios";
                }
            }


        JOptionPane.showMessageDialog(null, mensaje, "Actividad",
JOptionPane.INFORMATION_MESSAGE);
    }

    private void TxtDetalleKeyTyped(java.awt.event.KeyEvent evt) {
        Character s;
        s = evt.getKeyChar();
        if(!Character.isLetterOrDigit(s)  && s != KeyEvent.VK_BACK_SPACE && s !=
KeyEvent.VK_SPACE){
            System.out.println(evt.getExtendedKeyCode());
            evt.consume();
        }
    }
```

Matricula

```
public class MatriculaFrm extends javax.swing.JFrame {

    private void llenarcbxListados(){
        String sql="";
        try {
            sql = "select anno from curso_persona";
            ResultSet lista = conectar.buscarAlgo(sql);
            String formato="";
            while(lista.next()){
                formato = lista.getString(1);
                cbx_ano.addItem(formato.substring(0,4));
            }
           cbx_semestre.addItem("1");
```

```
            cbx_semestre.addItem("2");
        } catch (Exception e) {
            JOptionPane.showMessageDialog(null, "Problemas de B˙squeda",
"B⁄SQUEDA", JOptionPane.INFORMATION_MESSAGE);
        }

    }
    private void llenartabla() {
        String sql = "select * from matricula";
        DefaultTableModel modelo = (DefaultTableModel)
TBL_MATRICULA.getModel();
        modelo.getDataVector().removeAllElements();
        TBL_MATRICULA.clearSelection();
        try {
            ResultSet lista = conectar.buscarAlgo(sql);
            while (lista.next()) {
                Vector per = new Vector();
                per.addElement(lista.getString("folio"));
                per.addElement(lista.getInt("semestre"));
                per.addElement(lista.getString("cod_curso"));
                per.addElement(lista.getInt("rut_alu"));
                modelo.addRow(per);
            }
            TBL_MATRICULA.setModel(modelo);
        } catch (IllegalArgumentException e) {
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(null, "Problemas de B˙squeda",
"B⁄SQUEDA", JOptionPane.INFORMATION_MESSAGE);
        }
    }

    private void llenarCurso() {
        String sql = "select * from curso";
        try {
            ResultSet lista = conectar.buscarAlgo(sql);
            while (lista.next()) {
                CMB_CURSO.addItem(lista.getString("nombre_curso"));
            }
```

```
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(null, "Problemas de ConexiÛn",
"B⁄SQUEDA", JOptionPane.INFORMATION_MESSAGE);
        }
    }

    private void BTN_BUSCARActionPerformed(java.awt.event.ActionEvent evt) {
        //Buscar Alumno en la BD
        String sql = "select * from alumno where (estado='P' or estado='M') and rut_alu = "
+ TXT_RUT.getText();
        try {
            ResultSet lista = conectar.buscarAlgo(sql);
            if (lista.next()) {
                TXT_NOMBRE_COM.setText(lista.getString("nombre_alu") + ' ' +
lista.getString("apellido_paterno_alu") + ' '
                        + lista.getString("apellido_materno_alu"));
                BTN_GENERAR.setEnabled(true);
            } else {
                JOptionPane.showMessageDialog(null, "Error Alumno Ya Matriculado o No
Existe", "MatrÌcula", JOptionPane.INFORMATION_MESSAGE);

            }
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(null, "Error Alumno No Existe",
"B⁄SQUEDA", JOptionPane.INFORMATION_MESSAGE);
        }
    }

    private void BTN_GENERARActionPerformed(java.awt.event.ActionEvent evt) {
        //Generar La MatrÌcula
        String folio = "FO";
        String anno = "";
        //Preparar Folio

        //Obtener anno
        Date date = new Date();
        anno = date.toString().substring(date.toString().length() - 4,
date.toString().length());
```

```
        folio = folio + date.toString().substring(date.toString().length() - 2,
date.toString().length());
        String sql = "select count(folio) from matricula ";
        try {
            ResultSet lista = conectar.buscarAlgo(sql);
            int numeroMatriculado;
            if (lista.next()) {
                numeroMatriculado = Integer.parseInt(lista.getString("count(folio)"));
                numeroMatriculado = numeroMatriculado + 1;
                folio = folio + numeroMatriculado;
            }
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(null, "ERROR ", "MatrÌcula",
JOptionPane.INFORMATION_MESSAGE);
        }
        sql = "INSERT INTO matricula (`folio`, `semestre`, `cod_curso`, `rut_alu`) "
                + "VALUES ('" + folio + "', "
                + cbxSemestre.getSelectedItem() + ", "
                + "(select cod_curso from curso where nombre_curso= '" + (String)
CMB_CURSO.getSelectedItem() + "'), "
                + TXT_RUT.getText()
                + ")";
        if (validarCupos()) {
            conectar.insertarMat(sql);
            llenartabla();
        } else {
            JOptionPane.showMessageDialog(null, "No hay cupos Disponibles",
"MatrÌcula", JOptionPane.INFORMATION_MESSAGE);
        }
    }
    public boolean validarCupos() {
        boolean val = true;
        String sql = "select cupos from curso where cod_curso = " + "(select cod_curso
from curso where nombre_curso= '" + (String) CMB_CURSO.getSelectedItem() + "')";
        try {
            ResultSet lista = conectar.buscarAlgo(sql);
            if (lista.next()) {
                if (lista.getInt("cupos") == 0) {
```

```
                val = false;
            } else {
                int modificar;
                modificar = lista.getInt("cupos") - 1;
                String cur = "select cod_curso from curso where nombre_curso= '" +
(String) CMB_CURSO.getSelectedItem() + "'";
                ResultSet lista2 = conectar.buscarAlgo(cur);
                if (lista2.next()) {
                    String sql2 = "UPDATE curso SET"
                            + " cupos=" + modificar
                            + " WHERE  cod_curso= '" + lista2.getString("cod_curso") + "'";
                    System.out.println(sql2);
                    conectar.modificar2(sql2);
                }
                val = true;
            }
        }
    } catch (SQLException e) {
        JOptionPane.showMessageDialog(null, "ERROR ", "MatrÌcula",
JOptionPane.INFORMATION_MESSAGE);
    }
    return val;
  }

  private void BTN_CANCELARActionPerformed(java.awt.event.ActionEvent evt) {
    BTN_GENERAR.setEnabled(false);
    TXT_DV.setText("");
    TXT_RUT.setText("");
    TXT_NOMBRE_COM.setText("");
  }

  private void BTN_MODIFICARActionPerformed(java.awt.event.ActionEvent evt) {
    if (TBL_MATRICULA.getSelectedRow() != -1) {
      Matricula mat = new Matricula();
       int posicion = TBL_MATRICULA.getSelectedRow();
       String folio="";
       mat.setFolio(String.valueOf(TBL_MATRICULA.getValueAt(posicion,
0).toString()));
```

```
            mat.setSemestre(Byte.parseByte(TBL_MATRICULA.getValueAt(posicion,
1).toString()));
            mat.setCod_curso(String.valueOf(TBL_MATRICULA.getValueAt(posicion,
2)));
            mat.setRut_alu(Integer.parseInt(TBL_MATRICULA.getValueAt(posicion,
3).toString()));
            this.dispose();
            ModEva pantalla = new ModEva(mat);
            pantalla.setVisible(true);
        } else{
            JOptionPane.showMessageDialog(null, "Seleccione una curso a Modificar",
"MODIFICAR", JOptionPane.INFORMATION_MESSAGE);
        }
    }
```

Evaluación

```
public class IngresarModificarListarEvaluacion extends javax.swing.JFrame {

    private void BtnIngresarActionPerformed(java.awt.event.ActionEvent evt) {
        evalu.setDetalle(TxtDetalle.getText());
        String formato = date_eva.getDateFormatString();
        java.util.Date date = date_eva.getDate();
        SimpleDateFormat sdf = new SimpleDateFormat(formato);
        String fnacim = String.valueOf(sdf.format(date));
        if (evalu.getRut_alu() != 0 || evalu.getDetalle() != "" || evalu.getCod_actividad() !=
"") {
            String sql = "INSERT INTO `jardin`.`evaluacion` (`rut_alu`, `fecha_evaluacion`,
`cod_actividad`, `detalle`) VALUES ('"
                    + evalu.getRut_alu() + "', '" + fnacim + "', '" + evalu.getCod_actividad() +
"', '" + evalu.getDetalle() + "')";
            conectar.insertar(sql);
        }
    }

    private void TxtRutFocusLost(java.awt.event.FocusEvent evt) {
        try {
```

```
            evalu.setRut_alu(Integer.parseInt(TxtRut.getText()));
        } catch (NumberFormatException e) {
            evalu.setRut_alu(0);
        }
    }

    private void CmbCodigoFocusLost(java.awt.event.FocusEvent evt) {
        String activ = String.valueOf(CmbCodigo.getSelectedItem());
        activ = activ.substring(0, activ.indexOf("-"));
        evalu.setCod_actividad(activ.trim());
    }

    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
        if (TxtRut.getText().equals("")) {
            JOptionPane.showMessageDialog(null, "Debe Ingresar Un Rut", "°Error!",
JOptionPane.ERROR_MESSAGE);
            BtnIngresar.setEnabled(false);
            BTN_MODIFICAR.setEnabled(false);
            txtDV.setText("");
            TxtDetalle.setText("");
                    TxtDetalle.setEnabled(false);
                    BtnIngresar.setEnabled(false);
                    BTN_MODIFICAR.setEnabled(false);
                    date_eva.setDate(null);
                    date_eva.setEnabled(false);
            return;
        }
        if (txtDV.getText().equals("")) {
            JOptionPane.showMessageDialog(null, "DÌgito Verificador No Ingresado",
"°Error!", JOptionPane.ERROR_MESSAGE);
            BtnIngresar.setEnabled(false);
            BTN_MODIFICAR.setEnabled(false);
            TxtDetalle.setText("");
                    TxtDetalle.setEnabled(false);
                    BtnIngresar.setEnabled(false);
                    BTN_MODIFICAR.setEnabled(false);
                    date_eva.setDate(null);
                    date_eva.setEnabled(false);
```

```
            return;
        }
        if (CmbCodigo.getSelectedItem() == "--") {
            JOptionPane.showMessageDialog(null, "Debe Seleccionar Una Actividad",
"°Error!", JOptionPane.ERROR_MESSAGE);
            BtnIngresar.setEnabled(false);
            BTN_MODIFICAR.setEnabled(false);
            TxtDetalle.setText("");
                    TxtDetalle.setEnabled(false);
                    BtnIngresar.setEnabled(false);
                    BTN_MODIFICAR.setEnabled(false);
                    date_eva.setDate(null);
                    date_eva.setEnabled(false);
            return;
        }
        String sql = "Select * from alumno where rut_alu=" + evalu.getRut_alu();
        try {
            ResultSet lista = conectar.buscarAlgo(sql);
            if (lista.next()) {
              sql="select * from curso_actividad a,curso_alumno q where q.rut_alu = " +
evalu.getRut_alu() + " and q.cod_curso=a.cod_curso and a.cod_actividad = " +
evalu.getCod_actividad()+ " group by q.cod_curso" ;
                ResultSet lista2 = conectar.buscarAlgo(sql);
                if (lista2.next()) {
                    sql="";
                    sql = "select * from evaluacion where rut_alu = " + evalu.getRut_alu() + "
and cod_actividad = " + evalu.getCod_actividad();
                    ResultSet list = conectar.buscarAlgo(sql);
                    if(list.next()){
                      evalu.setFecha_evaluacion(list.getDate("fecha_evaluacion"));
                      evalu.setDetalle(list.getString("detalle"));
                      TxtDetalle.setText(evalu.getDetalle());
                      date_eva.setDate(evalu.getFecha_evaluacion());
                      BtnIngresar.setEnabled(false);
                      BTN_MODIFICAR.setEnabled(true);
                      date_eva.setEnabled(true);
                      TxtDetalle.setEnabled(true);
                    }else{
```

```
                BTN_MODIFICAR.setEnabled(false);
                BtnIngresar.setEnabled(true);
                date_eva.setEnabled(true);
                TxtDetalle.setEnabled(true);
            }
        } else {
            JOptionPane.showMessageDialog(null, "Actividad no puede realizarse para este alumno", "InformaciÛn", JOptionPane.INFORMATION_MESSAGE);
        }
      } else {
        JOptionPane.showMessageDialog(null, "Alumno No Existe", "InformaciÛn", JOptionPane.INFORMATION_MESSAGE);
        txtDV.setText("");
        TxtRut.setText("");
        BtnIngresar.setEnabled(false);
        BTN_MODIFICAR.setEnabled(false);
      }
    } catch (SQLException e) {
      JOptionPane.showMessageDialog(null, "Error Alumno No Existe", "B⁄SQUEDA", JOptionPane.INFORMATION_MESSAGE);
    }
  }

  private void BTN_MODIFICARActionPerformed(java.awt.event.ActionEvent evt) {
    try {
      evalu.setDetalle(TxtDetalle.getText());
      String formato = date_eva.getDateFormatString();
      java.util.Date date = date_eva.getDate();
      SimpleDateFormat sdf = new SimpleDateFormat(formato);
      String fnacim = String.valueOf(sdf.format(date));
      if (evalu.getRut_alu() != 0 || evalu.getDetalle() != "" || evalu.getCod_actividad() != "") {
        String activ = String.valueOf(CmbCodigo.getSelectedItem());
        activ = activ.substring(0, activ.indexOf("-"));
        String sql = "UPDATE evaluacion SET"
            + " rut_alu = " + TxtRut.getText()
            + ", fecha_evaluacion = '" + fnacim
            + "', detalle = '" + TxtDetalle.getText()
```

```
                    + "', cod_actividad = '" + activ + "'"
                    + " WHERE  rut_alu = " + evalu.getRut_alu() + " and fecha_evaluacion =
'" + evalu.getFecha_evaluacion() + "' and cod_actividad= '" + evalu.getCod_actividad()
+ "'";
                conectar.modificar(sql);
            }
        } catch (Exception e) {
            System.out.println("Fecha erronea " + e);
        }
    }

    private void txtDVFocusLost(java.awt.event.FocusEvent evt) {
        Calculos cal = new Calculos(TxtRut.getText());
        int d_v = cal.CalcularDV();
        if (!cal.ValidarDV(txtDV.getText(), d_v)) {
            JOptionPane.showMessageDialog(null, "DÌgito Verificador Incorrecto",
"°Error!", JOptionPane.ERROR_MESSAGE);
            txtDV.selectAll();
            txtDV.setText("");
            jButton3.setEnabled(false);
        } else {
            llencarlbl();
            jButton3.setEnabled(true);
        }
    }

    private void TxtDetalleKeyTyped(java.awt.event.KeyEvent evt) {
       Character s;
        s = evt.getKeyChar();
        if(!Character.isLetterOrDigit(s)  && s != KeyEvent.VK_BACK_SPACE && s !=
KeyEvent.VK_SPACE){
            evt.consume();
        }
    }
    public void llencarlbl(){
      try {
            String sql = "select * from evaluacion where rut_alu = " + evalu.getRut_alu();
            ResultSet lista2 = conectar.buscarAlgo(sql);
```

```
            if(!lista2.next()){
                sql = "select * from alumno where rut_alu = "+ evalu.getRut_alu();
                ResultSet lista1 = conectar.buscarAlgo(sql);
                lista1.next();
            }
        } catch (SQLException e) {
        }
    }
    private void llenarCombo() {
        String sql = "select * from Actividad";
        try {
            CmbCodigo.addItem("--");
            ResultSet lista = conectar.buscarAlgo(sql);
            while (lista.next()) {
                CmbCodigo.addItem(lista.getString("cod_actividad") + "-" +
lista.getString("nombre_actividad"));
            }
        } catch (SQLException e) {
        }
    }
```

Modificar Evaluación

```
public class ModEva extends javax.swing.JFrame {
     private void llenarCurso() {
        String sql = "select * from curso";
        try {
            ResultSet lista = conectar.buscarAlgo(sql);
            while (lista.next()) {
                cbx_curso.addItem(lista.getString("nombre_curso"));
            }
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(null, "Problemas de ConexiÛn",
"B⁄SQUEDA", JOptionPane.INFORMATION_MESSAGE);
        }
    }
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        String sql = "UPDATE matricula SET"
                + " rut_alu = " + txtrut.getText()
                + ", semestre = " + cbxSemestre.getSelectedItem()
```

```
            + ", cod_curso = (select cod_curso from curso where nombre_curso='" +
(String) cbx_curso.getSelectedItem()+ "')"
            + " WHERE  folio = '" + mat.getFolio() + "' ;
        conectar.modificar(sql);
    }
```

Mantenedor Actividad

```
public class MantenedorActividad extends javax.swing.JFrame {

    private void llenarTabla() {
        String sql = "select * from actividad";
        DefaultTableModel modelo = (DefaultTableModel) TblActividad.getModel();
        modelo.getDataVector().removeAllElements();
        TblActividad.clearSelection();
        try {
            ResultSet lista = conectar.buscarAlgo(sql);
            while (lista.next()) {
                Vector per = new Vector();
                per.addElement(lista.getString("cod_actividad"));
                per.addElement(lista.getString("nombre_actividad"));
                per.addElement(lista.getString("descripcion"));
                modelo.addRow(per);
            }
            TblActividad.setModel(modelo);
        } catch (IllegalArgumentException e) {
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(null, "Problemas de B˙squeda",
"B⁄SQUEDA", JOptionPane.INFORMATION_MESSAGE);
        }
    }

    private void llenarTabla2() {
        String sql = "select * from curso_actividad";
        DefaultTableModel modelo = (DefaultTableModel) TblActCurso.getModel();
        modelo.getDataVector().removeAllElements();
        TblActCurso.clearSelection();
        try {
```

```
            ResultSet lista = conectar.buscarAlgo(sql);
            while (lista.next()) {
                Vector per = new Vector();
                per.addElement(lista.getString("cod_curso"));
                per.addElement(lista.getString("cod_actividad"));
                per.addElement(lista.getInt("anno"));
                per.addElement(lista.getInt("semestre"));
                per.addElement(lista.getString("fecha"));
                modelo.addRow(per);
            }
            System.out.println(sql);
            TblActCurso.setModel(modelo);
        } catch (IllegalArgumentException e) {
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(null, "Problemas de B˙squeda",
"B⁄SQUEDA", JOptionPane.INFORMATION_MESSAGE);
        }
    }

    private void BTN_VOLVERMouseClicked(java.awt.event.MouseEvent evt) {
        if(permiso.equals("")){
        this.dispose();
        PantallaInicio inicio = new PantallaInicio("2");
        inicio.setVisible(true);
      }else{
        PantallaInicio inicio = new PantallaInicio();
        inicio.setVisible(true);
        this.dispose();
      }
    }

    private void BTN_MODIFICARActionPerformed(java.awt.event.ActionEvent evt) {
        if(TblActividad.getSelectedRow() != -1){
        this.dispose();
        IngModActividad pantalla = new IngModActividad(actividad);
        pantalla.setVisible(true);
        }else{
```

```
        JOptionPane.showMessageDialog(null, "Seleccione una actividad de la tabla", "Modificar", JOptionPane.INFORMATION_MESSAGE);
        }
    }

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
      if (permiso.isEmpty()) {
          this.dispose();
          IngModActividad pantalla = new IngModActividad();
          pantalla.setVisible(true);
       } else {
          this.dispose();
         IngModActividad pantalla = new IngModActividad("2");
         pantalla.setVisible(true);
       }
    }

    private void TblActividadFocusLost(java.awt.event.FocusEvent evt) {
       if (TblActividad.getSelectedRow() != -1) {
          int posicion = TblActividad.getSelectedRow();
actividad.setCod_actividad(String.valueOf(TblActividad.getValueAt(posicion,          0)));
actividad.setNombre_actividad(String.valueOf(TblActividad.getValueAt(posicion, 1)));
          actividad.setDescripcion(String.valueOf(TblActividad.getValueAt(posicion, 2)));
       }
    }

    private void BtnIngresarActionPerformed(java.awt.event.ActionEvent evt) {

       this.dispose();
       IngModCurAct pantalla = new IngModCurAct();
       pantalla.setVisible(true);
    }

    private void TblActCursoFocusLost(java.awt.event.FocusEvent evt) {
       if (TblActCurso.getSelectedRow() != -1) {
          int posicion = TblActCurso.getSelectedRow();
          curAcr.setCod_curso(String.valueOf(TblActCurso.getValueAt(posicion, 0)));
          curAcr.setCod_actividad(String.valueOf(TblActCurso.getValueAt(posicion, 1)));
```

```
            curAcr.setAnno(Year.parse(String.valueOf(TblActCurso.getValueAt(posicion,
2).toString())));

curAcr.setSemestre(Byte.parseByte(String.valueOf(TblActCurso.getValueAt(posicion,
3))));
            curAcr.setFecha(Date.valueOf(TblActCurso.getValueAt(posicion,
4).toString()));
        }
    }

    private void BTN_ELIMINARActionPerformed(java.awt.event.ActionEvent evt) {
        if (TblActividad.getSelectedRow() != -1) {

            String sql = "delete from actividad where cod_actividad = '";
            String rut = TblActividad.getValueAt(TblActividad.getSelectedRow(),
0).toString();
            sql = sql + (String) rut + "'";
            conectar.eliminar(sql);
            TblActividad.clearSelection();
            llenarTabla();
        } else {
            JOptionPane.showMessageDialog(null, "Seleccione una actividad a eliminar",
"ELIMINAR", JOptionPane.INFORMATION_MESSAGE);
        }
    }

    private void BTN_ELIMINAR1ActionPerformed(java.awt.event.ActionEvent evt) {

        if (TblActCurso.getSelectedRow() != -1) {
            String codigocurso = TblActCurso.getValueAt(TblActCurso.getSelectedRow(),
0).toString();
        String codigoact = TblActCurso.getValueAt(TblActCurso.getSelectedRow(),
1).toString();
        String anno =
String.valueOf(TblActCurso.getValueAt(TblActCurso.getSelectedRow(), 2).toString());
        String semestre = TblActCurso.getValueAt(TblActCurso.getSelectedRow(),
3).toString();
```

```
        String fecha = TblActCurso.getValueAt(TblActCurso.getSelectedRow(), 4).toString();
            String sql = "delete from curso_actividad where cod_curso = '"
                    + codigocurso +"' and cod_actividad = '"+ codigoact +"' and anno = '"
                    + anno+"' and semestre = '" + semestre + "' and fecha = '"+ fecha +"'";
            System.out.println(sql);
            conectar.eliminar(sql);
            TblActCurso.clearSelection();
            llenarTabla2();
        }else{
         JOptionPane.showMessageDialog(null, "Seleccione una actividad de curso a eliminar", "ELIMINAR", JOptionPane.INFORMATION_MESSAGE);
        }
    }
```