

Sistema de Matriculas Educacional

(S.M.E.)

Grupo:

❖ Benjamin Fernandez

❖ Jesús Tapia

❖ Yoandri Villarroel

Índice

Introducción.....	3
Contexto del problema.....	3
Descripción de las clases.....	4
[1] Clases.....	4
[2] Controlador.....	7
[3] Vista.....	8
Modelo UML.....	9
Definición y diseño de GUIs.....	10
Mapa de navegación.....	10
Gestión de datos.....	11
Conclusiones.....	12
Bibliografía.....	13

Introducción

En el primer avance del proyecto, fue identificada la problemática en el proceso de matrícula en un colegio en específico, mediante un análisis del dominio del problema, se definió la estructura de este, la cual se dividió en Packages que contienen dentro clases con funcionalidades similares entre sí, con el fin de que la mantención y la actualización del código se facilite.

En este segundo avance del proyecto, se mostrará las actualizaciones y cambios en la implementación del código. Para mostrar una representación visual de las relaciones y lógica del programa, fue empleado el modelo UML con Visual Paradigm. Además, se profundizará en la descripción del código (Clases, atributos, métodos, encapsulamiento). Finalmente se verá la implementación de Java Swing para el diseño de las GUI.

Contexto del problema

El proceso de matrícula es una tarea esencial en el mantenimiento y orden de una institución, no obstante, este proceso puede verse dificultado por lo extenso, agotador y tedioso que puede ser.

Por lo que, la finalidad de este proyecto es desarrollar un programa para una institución específica la cual es el Complejo Educacional “La Granja” de Cajón, donde se creará un software que tenga diferentes funcionalidades que sea más cómodo y rápido en la integración de datos en el proceso de matrícula.

Descripción de las clases

La estructura del programa está dividida en 4 packages, que se usan para almacenar aquellas clases con una misma funcionalidad.

[1] Clases

Función:

Este Package se divide en dos clases que se encargan de gestionar y mantener la información de los **Alumnos** y sus **Apoderados**. Siendo su principal función la creación de objetos de estos tipos, por lo que en su interior se deben encontrar los atributos necesarios para obtener información que facilite el conocimiento y el manejo de los datos de cada objeto.

MÉTODOS GENERALES

Los **métodos** utilizados para ambas clases son los constructores, setters y getters.

Constructor:

Dentro de cada clase se encuentra el método constructor con **parámetros**, el cual inicializa los valores de **cada** objeto de tipo Alumno o Apoderado que creamos. Este método facilita la creación de objetos, pasando la información en el parámetro del objeto que estemos creando.

GETTERS:

Los métodos Getters, se utilizan para obtener el valor **actual** de un atributo en un momento específico del programa. Estos métodos son ocupados en todo lo que tiene que ver con el manejo y creación del archivo CSV (Package Controlador), ya que en este se solicitan los datos de cada **Alumno** y su **Apoderado** para guardarlos dentro del archivo.

SETTERS:

Los métodos Setters, se utilizan en un atributo específico para modificar el valor **actual** que tiene en el programa.



- **Alumno**

Atributos:

```
String rut          , nombres          , apellidos          , edad          ,  
    fechaNacimiento , email          , ciudad          , telefono          ,  
    nacionalidad    , fechaMatricula , direccion          , curso          ,  
    letra          , electivo          , enfermedades          , datosAdicionales ,  
    genero;
```

Otro atributo importante presente en la clase, es el de tipo Apoderado llamado “nuevoApoderado”. Este atributo permite **asociar** un alumno con su apoderado. Tiene un nivel de encapsulamiento privado.

```
private Apoderado nuevoApoderado;
```

Métodos:

```
public Alumno( String rut          , String nombres          , String apellidos          , String edad          ,  
    String fechaNacimiento , String email          , String ciudad          , String telefono          ,  
    String nacionalidad    , String fechaMatricula , String direccion          , String curso          ,  
    String letra          , String electivo          , String enfermedades          , String datosAdicionales ,  
    String genero) {
```

También están los **getters** y **setters** de cada uno de los atributos, los niveles de encapsulamiento de estos métodos son públicos, pero, los getters son de **tipo String** excepto el getter de nuevo Apoderado el cual es de **tipo Apoderado**, mientras que, los Setters son de **tipo Void**.

- **Apoderado**

Atributos:

```
String rut      , nombre      , apellidos      ,  
    parentesco  , telefono    , ciudad        ,  
    direccion   , observaciones , genero;
```

El nivel de encapsulamiento de los atributos es **privado**.

Métodos:

```
public Apoderado( String rut      , String nombre      , String apellidos      ,  
    String parentesco , String telefono    , String ciudad        ,  
    String direccion   , String observaciones , String genero ) {
```

También están los métodos getters y setters donde los getters son de tipo **String** y su nivel de encapsulamiento debe ser **público** y los setters son de tipo **Void** y su encapsulamiento también es **público**.

[2] Controlador

Función:

Este package tiene una única clase, la cual es la **encargada de manejar** los datos del package Clases.

EXPLICACIÓN DE MÉTODOS

agregarAlumnoCSV

Se encarga de agregar los objetos Alumno y SU Apoderado al archivo CSV, en caso de que el archivo no exista, el método es capaz de generar el archivo CSV creando los encabezados en la primera línea, y agregando una nueva línea para empezar a agregar los alumnos y sus apoderados.

En caso de tener el archivo creado, se crean atributos tipo string con un nombre característico, que con ayuda de los getters guardan la información de cada alumno y su apoderado en estos, para finalmente agregarlo y dejar una nueva línea para agregar otro alumno.

listadoAlumnos

Se encarga de obtener una lista de información de cada alumno almacenada en el archivo CSV.

El método abre el archivo CSV y prepara un lector para recorrer cada línea del archivo, comenzando desde la segunda línea para no leer los encabezados.

Luego divide cada línea en campos utilizando la coma como separador, para posteriormente extraer la información relevante del Alumno y su Apoderado, toda esta información se almacena en un array de objetos.

Finalmente devuelve una lista que contiene arrays de objetos, donde cada uno representa la información de un Alumno y su Apoderado.

Adicionalmente este método toma como parámetro un filtro de rut, el cual verifica si el rut del alumno coincide con el filtro. Si este filtro no se considera, todos los alumnos se considerarán.

eliminarAlumnoCSV

Se encarga de eliminar un estudiante del archivo CSV, utilizando como base un archivo temporal para realizar la eliminación y luego reemplazar el archivo original con el archivo temporal.

Se crea un objeto objeto file para representar el archivo de matrículas, un objeto Scanner y un BufferedWriter para escribir en un archivo temporal, y en un bucle while se lee cada línea del archivo original, buscando una que contenga el rut del estudiante a eliminar, si esto pasa, se marca "estudiante encontrado" como verdadero, si no pasa, la línea se escribe en el archivo temporal.

Finalmente se elimina el archivo original, y el archivo temporal se renombra como el archivo original.

- **Controlador**

Atributos :

```
public class Controlador {  
  
    // ===== ATRIBUTOS =====  
    public static final String nombreArchivo = "Matriculas.csv";  
}
```

El nivel de encapsulamiento del atributo es **público**.

Métodos:

1. AGREGAR ALUMNOS AL CSV

```
public static void agregarAlumnoCSV(Alumno nuevoAlumno) {...}
```

2. MOSTRAR ALUMNOS

```
public static List<Object[]> listadoAlumnos(String filtroRut) {...}
```


3. ELIMINAR ALUMNOS AL CSV

```
public static void eliminarAlumnoCSV(String rutEstudiante) {...}
```

[3] Vista

Función:

Este package contiene en su interior las clases que crean y diseñan las interfaces gráficas de usuario (GUI), junto a todo el código detrás de cada ventana y la lógica asociada a cada posible acción del usuario con los elementos de la interfaz.

EXPLICACIÓN DE MÉTODOS

Constructor

Este método se encuentra en cada formulario, ya que es acá definimos las propiedades que tendrá en la ventana, como el título y sus dimensiones. Acá también podemos asignar **eventos** a los componentes dentro de la ventana, como a un botón.

Eventos y ActionListener

Los eventos representan acciones que suceden durante la interacción entre un elemento de la interfaz gráfica de usuario, y el usuario. Por lo tanto, el presionar un botón desencadena una sucesión de acciones con una lógica y orden predefinida dentro de un ActionListener, que es donde se establece la lógica y orden.

Grabar

Estos métodos van vinculados a eventos de botones específicos, ocupan los setters para tomar los datos que rellena el usuario en los campos de texto, y los asignan a un atributo tipo String, para posteriormente usarlos como parámetros para crear un objeto.

Limpiar

Estos métodos con ayuda de los getters, toman todos los campos de texto y grupo de botones, y los ponen en blanco.



Validar

El método define aquellos campos de texto obligatorios, y verifica que todos los campos requeridos estén llenos antes de agregar un nuevo alumno y apoderado.

Cargar

Usa la clase Controlador para obtener los datos y cargar la tabla de alumnos.

Buscar

Obtiene el valor del campo de texto que contiene el rut y mostrar el resultado en la tabla "tblListado"

Eliminar estudiante

Este permite eliminar a un estudiante al confirmar la acción a través de un cuadro de diálogo.

----- fmrMenu -----

Esta clase representa el Menú principal de la aplicación y tiene 3 botones, el btnAgregarAlumno que al hacer clic lleva a la ventana que permite agregar un nuevo Alumno y su Apoderado, además de diversas funciones con el listado de alumnos

El botón institución debería llevar a una ventana que muestre los datos de la institución, pero esta sigue en trabajo por lo que, indica que la ventana se encuentra en desarrollo

Finalmente, al hacer click en el btnSalir termina la ejecución del programa.

- **Clase *fmrAgregarAlumnos***

Atributos:

```
private Alumno nuevoAlumno;
```

El nivel de encapsulamiento del atributo es **privado**.

Métodos:

1. CONSTRUCTOR

```
public fmrAgregarAlumnos() {...}
```

2. GRABAR AL ALUMNO

```
Alumno grabar() {...}
```

3. GRABAR AL APODERADO

```
public void grabarApoderado() {...}
```

4. LIMPIAR LA INTERFAZ DEL ALUMNO

```
public void limpiar() {...}
```

5. LIMPIAR LA INTERFAZ DEL APODERADO

```
public void limpiarApoderado() {...}
```

6. SET NUEVO ALUMNO

```
public void setNuevoAlumno(Alumno nuevoAlumno) { this.nuevoAlumno = nuevoAlumno; }
```

7. VALIDA LOS CAMPOS DEL ALUMNO

```
private void cargarListadoAlumnos(String filtroRut) {...}
```

8. CARGAR LISTADO DE ALUMNOS

```
private void cargarListadoAlumnos(String filtroRut) {...}
```

9. BUSCAR ALUMNO POR RUT

```
private void buscarAlumnoPorRut() {...}
```

10. ELIMINAR ESTUDIANTE

```
private void eliminarEstudiante() {...}
```



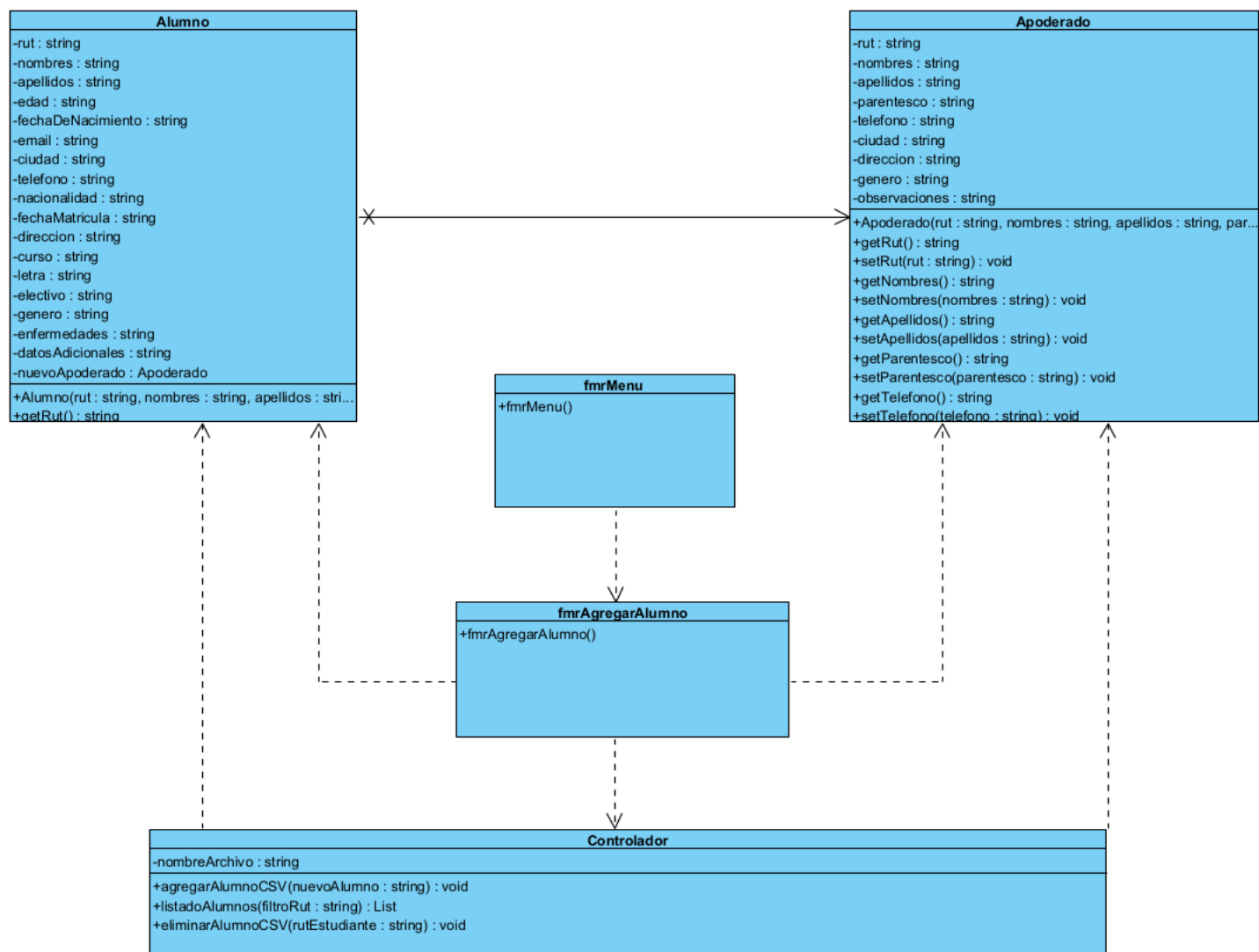
- Clase *fmrMenu*

Métodos

1. CONSTRUCTOR

```
public fmrMenu(){...}
```

Modelo UML



Para la creación de la interfaz de usuario gráfica, se utilizó JFrame, **¿Qué es JFrame?**.

Es una clase que se encuentra en el paquete javax swing lo que permite generar ventanas o marcos en las cuales se pueden ingresar componentes (Contenedores, botones, etc). La mayoría de las ventanas son instancias de la clase de Frame o subclases de esta.



UNIVERSIDAD
DE LA FRONTERA

Mapa de navegación

Si clickeas en agregar alumno te lleva a la ventana correspondiente.

Agregar Alumno

Institución

Sale del programa.

Salir

Al clickear institución te debería llevar a su debida ventana donde muestra los datos de la institución, pero todavía falta crearla.

Al clickear en agregar alumno te lleva a la ventana correspondiente.

Si clickeas institución te debería llevar a su debida ventana donde muestra los datos de la institución, pero todavía falta crearla.

Si clickeas salir te devuelve a la ventana principal

Si clickeas en apoderado, este te lleva al formulario apoderado

Agregar Alumno

Rut

Buscar

Nombres

Apellidos

Edad

Fecha de nacimiento

Email

Ciudad

Teléfono

Nacionalidad

Fecha matrícula

Dirección

Curso

Letra

Electivo

Enfermedades

Datos adicionales

Género

☐ Masculino

☐ Femenino

☐ Otro

Grabar


Limpiar

Borrar

Listar

Modificar

Salir

 Agregar Alumno

Alumno | Apoderado

Agregar Apoderado

Rut	Nombres	Apellidos	Parentesco	Telefono
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
Dirección		Ciudad		
<input type="text"/>		<input type="text"/>		
Observaciones	Género			
<input type="text"/>	<input type="radio"/> Masculino <input type="radio"/> Femenino <input type="radio"/> Otro			
<input type="button" value="Limpiar"/>				

Gestión de datos

En el programa se trabaja con un archivo CSV. Este formato fue escogido gracias a la facilidad que entrega a la hora de programar (creación del archivo, eliminar, registro y listado), como en el manejo y visualización de la información dentro de este.

Los datos almacenados dentro de nuestro archivo CSV, va distribuido en filas, en donde cada fila almacena un objeto Alumno y su Apoderado, y cada columna representa un atributo de cada objeto.

Conclusiones

Se puede apreciar los avances y los cambios implementados en el proyecto, como la utilización de packages para favorecer la estructura y mantenimiento del código, los tipos de relaciones entre las clases (dependencia y asociación específicamente), el tipo de dato de los atributos y el tipo de retorno de los métodos, la encapsulación de los atributos y métodos. Fue utilizada la herramienta Visual Paradigm para crear diagramas de clases permitiendo ver las relaciones y lógica entre las clases de una manera más entendible y simple. Además de usar el tipo de archivo CSV para el manejo y gestión de los datos.

Por otra parte, para la creación de las ventanas para la interfaz gráfica de usuario se utilizó JFrame, por las herramientas que tiene, y la facilidad con la que se agregan los componentes, a la hora de crear ventanas, botones y texto (JLabel, JButton, etc).

Bibliografía

Formularios (JFrame) en Java. (2018, abril 10). PROGRAMACION.

<https://victomanolo.wordpress.com/formularios-en-java/>