



Reporte entrega final

Proyecto POO – Gestión de asistencia de un colegio

Por Sebastián García, Guillermo González y Benjamín Navarrete

Índice

1. Entrega parcial 1	3
1.1. Realizar un análisis de los datos a utilizar y principales funcionalidades a implementar que dan sentido a la realización del proyecto	3
1.2. Diseño conceptual de clases del Dominio y su código en Java	4
1.3. Todos los atributos de todas las clases deben ser privados y poseer sus respectivos métodos de lectura y escritura (getter y setter)	6
1.4. Se deben incluir datos iniciales dentro del código	6
Opcional: Leer datos desde archivo/base de datos	6
2. Entrega parcial 2	7
2.1. Diseño conceptual y codificación de 2 (dos) niveles de anidación de colecciones de objetos	7
2.2. Diseño conceptual y codificación de 2 (dos) clases que utilicen sobrecarga de métodos	7
2.3. Diseño conceptual y codificación de al menos 1 clase mapa del JCF	8
3. Entrega parcial 3	9
3.1. Diseño de diagrama de clases UML de lo considerado hasta la EP3	9
3.2. Menú del sistema	15
3.2.1. Inserción manual/agregar elemento	16
3.2.2. Mostrar por pantalla listado de elementos. Esto para cada una de las 2 colecciones anidadas	18
A. Entrega acumulada A	20
A.2. Menú con funcionalidades de edición y eliminación de elementos	20
A.3. Se debe generar un reporte en archivo txt que considere mostrar datos de las 2 colecciones anidadas (ej: csv)	20
A.4. El código fuente debe estar bien modularizado de acuerdo a lo descrito en el informe además de seguir las buenas prácticas de documentación interna y legibilidad	22
A.5. Todas las funcionalidades pueden ser implementadas mediante consola	22
A.6. Utilización de GitHub (Realización de al menos 3 Commit)	22
4. Entrega parcial 4	23
4.1. Se deben incluir al menos 2 funcionalidades propias que sean de utilidad para el negocio (distintas de la inserción, edición, eliminación y reportes)	23
4.1.1. Seleccionar un objeto por criterio, considera la selección de un objeto basado en un criterio específico, involucrando dos o más colecciones anidadas. Por ejemplo, selección del alumno con la nota final más baja de todos los cursos, o seleccionar el pasajero más joven de todos los buses de una compañía	23
4.1.2. Subconjunto filtrado por criterio: considera la selección de un subconjunto de objetos basado en un criterio específico, involucrando dos o más colecciones anidadas. Por ejemplo, selección de los alumnos con nota final entre 4,0 y 7,0 de entre todos los cursos; o seleccionar a todos los pasajeros que tengan asiento impar de entre todos los buses de la compañía	23

4.2. Diseño y codificación de 2 (dos) clases que utilicen sobreescritura de métodos	23
4.3. Diseño y codificación de 1 (una) clase abstracta que sea padre de al menos 2 (dos) clases. La clase abstracta debe ser utilizada por alguna otra clase (contexto)	24
4.4. Diseño y codificación de 1 (una) interfaz que sea implementada por al menos 2 (dos) clases. La interfaz debe ser utilizada por alguna otra clase (contexto)	25
Opcional: Generar documentación con Javadoc.	25
B. Entrega acumulada B	26
B.2. Se deben implementar al menos 3 ventanas gráficas (GUIs en AWT o SWING): 1 ventana de menú, 1 ventana de agregar elemento y 1 ventana de listar elementos	26
B.3. Se debe aplicar encapsulamiento y principios OO	30
B.4. Continuidad en la utilización de GitHub (Realización de al menos 3 Commit adicionales a los ya hechos en la parte A)	30
F. Entrega final	31
F.2. Implementar el manejo de excepciones capturando errores potenciales específicos mediante Try-catch	31
F.3. Crear 2 clases que extiendan de una Excepción y que se utilicen en el programa	32
F.4. Aplicación del patrón de diseño Strategy (Estrategia)	32
Opcional: Reemplazar los datos iniciales en el código por una conexión con base de datos local (MySQL), archivo de texto CSV o Excel, para la persistencia de datos	33
Opcional: Considerar la implementación del patrón Modelo-Vista-Controlador (MVC) en la arquitectura del sistema	34
8. Consideraciones generales	34
8.1. Desarrollo de la aplicación	34
8.1.1. Codificación y documentación	34
8.1.2. Paquetes de Maven	34
8.2. Utilización de GitHub	34
8.3. Herramientas de utilidad	35

1. Entrega parcial 1

1.1. Realizar un análisis de los datos a utilizar y principales funcionalidades a implementar que dan sentido a la realización del proyecto

La problemática a abordar en este proyecto es la necesidad que tienen los colegios de llevar un registro de asistencia de sus estudiantes para cada año escolar, y para ello, la principal información que deberemos manejar son los **estudiantes** y los **cursos**. Los cursos se dividirán en diferentes niveles y cada nivel subdividirá en paralelos, cada uno de estos contará con la información del profesor a cargo de este curso y la lista de alumnos correspondiente.

Los alumnos contarán con información básica de ellos (nombre, rut, apellido, etc.) y cada uno contará con su registro de asistencia correspondiente al año escolar, con esto buscamos que se generen archivos que muestren la lista de estudiantes de cada curso y la asistencia de estos.

Funcionalidades principales:

- Registrar asistencia de un curso para cada dia de clases.
- Registrar inasistencias.
- Registro de salidas antes de horario.
- Consultar registro de asistencia, inasistencias y salidas antes de horario de cualquier día del año.

Funcionalidades secundarias:

- Calcular porcentaje de asistencia
- Generar ranking de asistencia en distintos niveles del colegio
- Generar reportes en base a la asistencia

Entidades:

- Alumnos del colegio
- Cursos y niveles
- Profesores
- Apoderados
- Registro de asistencia

Manejo de datos:

- El software desarrollado tiene la capacidad de conectarse tanto a una base de datos MySQL (o MariaDB), y también de trabajar con datos almacenados de forma local a través de un archivo CSV, desarrollando con ello todas las funcionalidades necesarias para leer, insertar, actualizar o eliminar datos desde cualquiera sea el origen de datos. Para trabajar con la base de datos, el software incorpora las configuraciones necesarias para ejecutar, a través de **docker**, una base de datos en MySQL con los datos iniciales, y se conecta a ella de forma predeterminada, utilizando las credenciales incorporadas en el archivo **.env**, que por razones de seguridad, no está incluido en el repositorio de GitHub, pero sí en la entrega final.
- Para efecto de trabajar con datos iniciales, se creó una clase Fakedata que incluye una librería externa (Faker). Esta clase permite crear datos iniciales aleatorios, y los guarda en los ficheros CSV respectivos, utilizando la clase Datafile de manejo de archivos.

1.2. Diseño conceptual de clases del Dominio y su código en Java

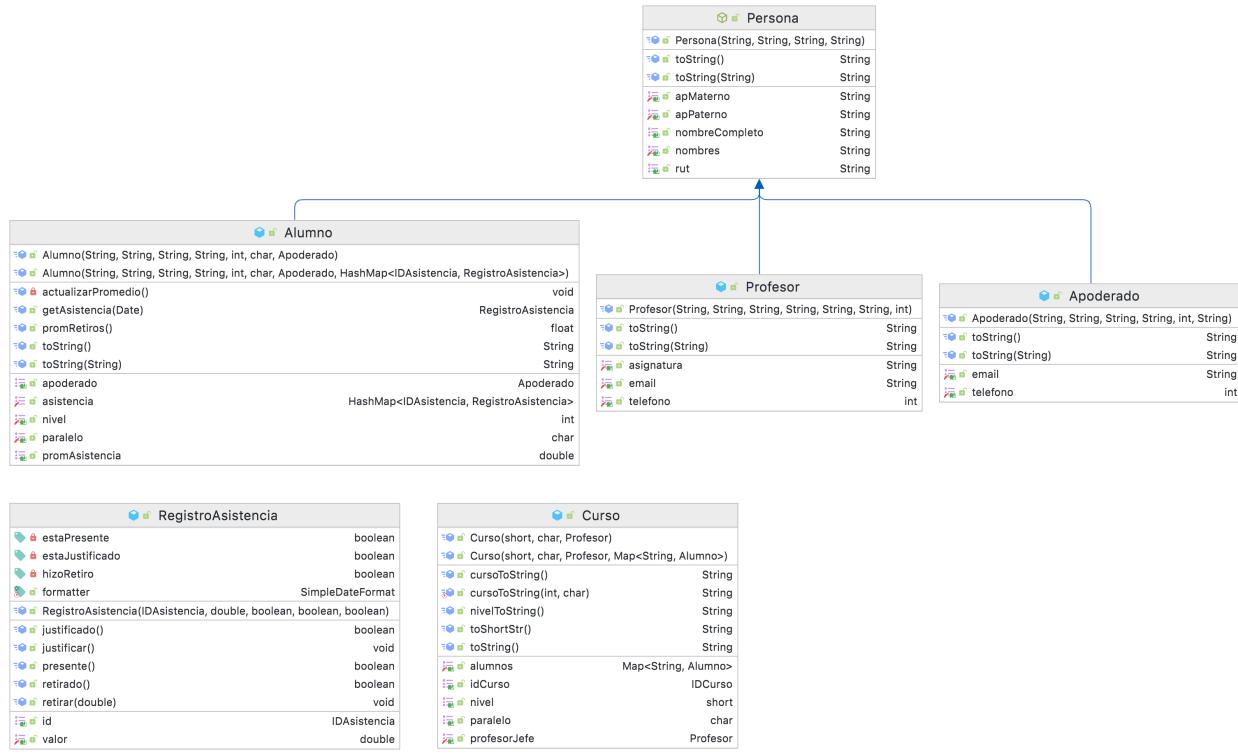


Figura 1: Clases del dominio

La clase **Persona** contendrá los datos genéricos de una persona, y será la clase padre de **Apoderado, Alumno y Profesor**. Además está la clase **Curso** que posee el nivel, valor numérico de 1 a 12, una letra identificadora de paralelo, un profesor jefe, y un Hashmap de alumnos, cuya llave será el RUT del alumno correspondiente.

En el caso de **Apoderado**, este hereda de **Persona** y además tiene los datos de contacto del apoderado de cada alumno. **Alumno** contiene a su apoderado, y lleva su registro de asistencia correspondiente al año escolar.

Para el manejo de los datos, se utilizarán **interfaces** que nos permitan estandarizar la forma de obtener y manejar los datos, para poder incorporar en el futuro, de una forma más simple, manejo de bases de datos.

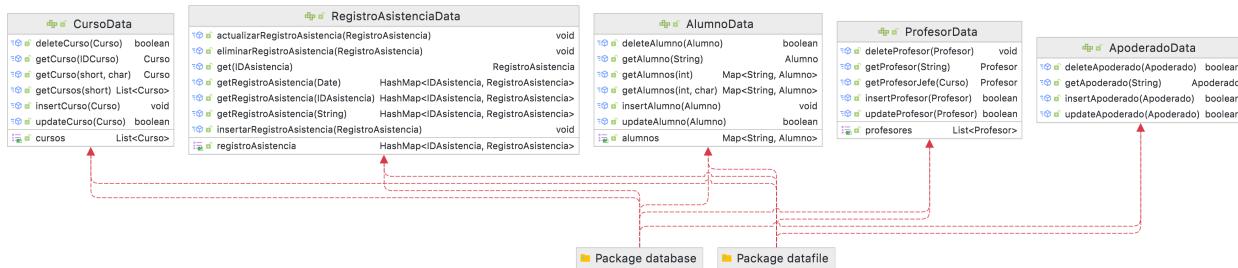


Figura 2: Clases para manejo de datos

Para el manejo de los archivos de texto en donde se almacenan los datos, se utilizará una clase dedicada a trabajar con ellos. Esta clase contendrá los métodos que posibilitan la inserción, listado, eliminación y actualización de los datos, entregados como texto CSV, separado por punto y coma (;). Para llevar los datos al formato CSV, la clase contiene un método estático que permite transformar una lista de String al texto CSV.

RegistroAsistenciaDF	
` ` RegistroAsistenciaDF()	
` ` actualizarRegistroAsistencia(RegistroAsistencia)	void
` ` eliminarRegistroAsistencia(RegistroAsistencia)	void
` ` get(IDAsistencia)	RegistroAsistencia
` ` getRegistroAsistencia(Date)	HashMap<IDAsistencia, RegistroAsistencia>
` ` getRegistroAsistencia(IDAsistencia)	HashMap<IDAsistencia, RegistroAsistencia>
` ` getRegistroAsistencia(String)	HashMap<IDAsistencia, RegistroAsistencia>
` ` insertarRegistroAsistencia(RegistroAsistencia)	void
` ` registroAsistenciaToCSV(RegistroAsistencia)	String
` ` registroAsistenciaFromCSV(String)	RegistroAsistencia
` ` registroAsistencia	HashMap<IDAsistencia, RegistroAsistencia>

AlumnoDF	
` ` AlumnoDF()	
` ` alumnoFromCSV(String)	Alumno
` ` alumnoToCSV(Alumno)	String
` ` deleteAlumno(Alumno)	boolean
` ` getAlumno(String)	Alumno
` ` getAlumnos(int)	Map<String, Alumno>
` ` getAlumnos(int, char)	Map<String, Alumno>
` ` insertAlumno(Alumno)	void
` ` updateAlumno(Alumno)	boolean
` ` alumnos	Map<String, Alumno>

CursoDF	
` ` CursoDF()	
` ` cursoFromCSV(String)	Curso
` ` cursoToCSV(Curso)	String
` ` deleteCurso(Curso)	boolean
` ` getCurso(IDCURSO)	Curso
` ` getCurso(short, char)	Curso
` ` getCursos(short)	List<Curso>
` ` insertCurso(Curso)	void
` ` updateCurso(Curso)	boolean
` ` cursos	List<Curso>

Datafile	
` ` Datafile(String)	
` ` clearFile()	void
` ` deleteLine(String)	boolean
` ` insertLine(String)	boolean
` ` listToCSV(List<String>)	String
` ` updateLine(String, String)	boolean
` ` data	List<String>
` ` fileExists	boolean
` ` filePath	String

ProfesorDF	
` ` ProfesorDF()	
` ` deleteProfesor(Profesor)	void
` ` getProfesor(String)	Profesor
` ` getProfesorJefe(Curso)	Profesor
` ` insertProfesor(Profesor)	boolean
` ` profesorFromCSV(String)	Profesor
` ` profesorToCSV(Profesor)	String
` ` updateProfesor(Profesor)	boolean
` ` profesores	List<Profesor>

ApoderadoDF	
` ` ApoderadoDF()	
` ` apoderadoFromCSV(String)	Apoderado
` ` apoderadoToCSV(Apoderado)	String
` ` deleteApoderado(Apoderado)	boolean
` ` getApoderado(String)	Apoderado
` ` insertApoderado(Apoderado)	boolean
` ` updateApoderado(Apoderado)	boolean

Figura 3: Paquete Datafile

1.3. Todos los atributos de todas las clases deben ser privados y poseer sus respectivos métodos de lectura y escritura (getter y setter)

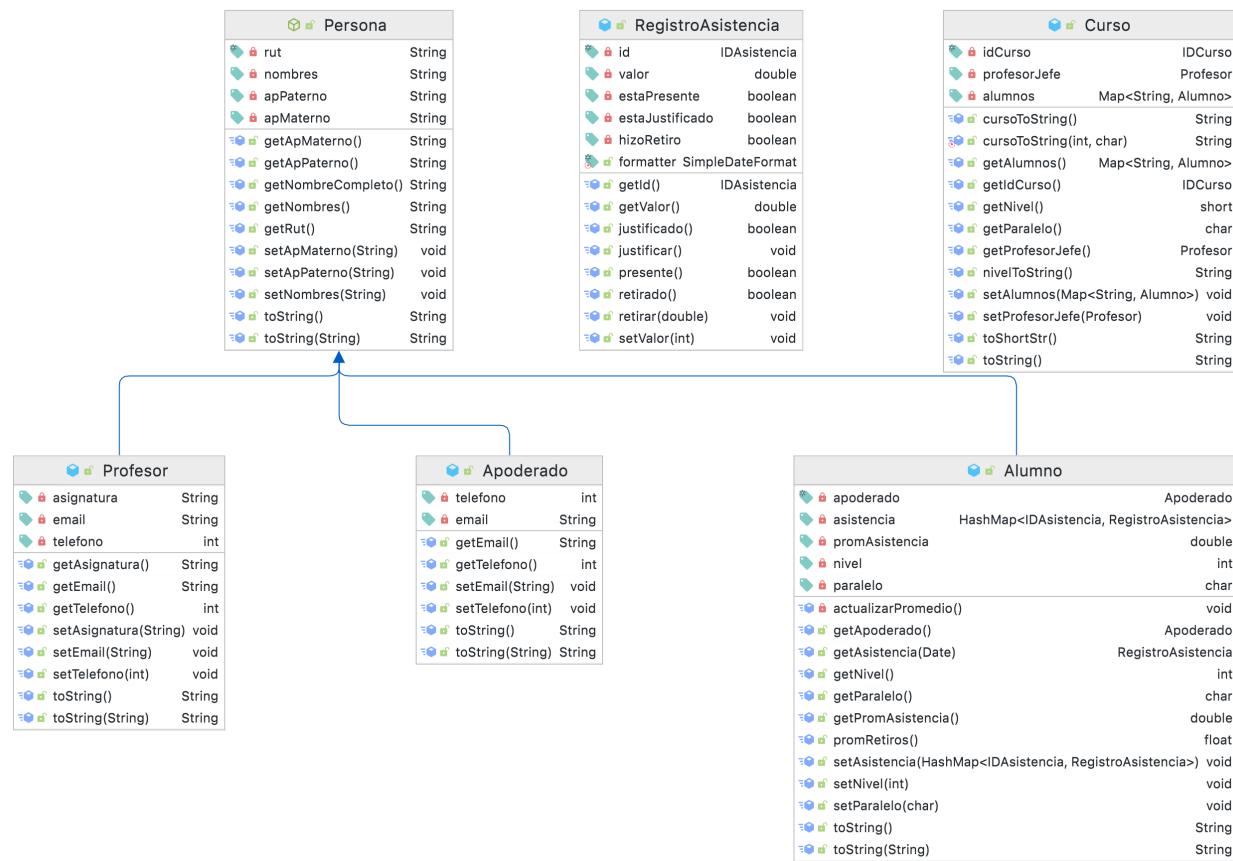


Figura 4: Atributos, setters y getters

Como podemos ver en la **figura indicada**, los atributos de las clases del dominio son todos privados, y tienen definidos sus respectivos **setter** y **getter**.

1.4. Se deben incluir datos iniciales dentro del código

Los datos iniciales fueron creados utilizando la librería **Faker**, disponible asociandola al proyecto **maven**. Los datos iniciales son los mismos, tanto en el archivo plano (CSV) como en la base de datos MySQL. Los datos ya vienen almacenados en sus respectivas carpetas, en el caso de los archivos planos, estos ya están vinculados a la aplicación mediante las clases Datafile (a revisar con más profundidad [en la próxima sección](#)).

Opcional: Leer datos desde archivo/base de datos

Como se verá [más adelante](#), el programa desarrollado tiene la capacidad de utilizar tanto los datos locales, almacenados como CSV, o conectándose a una base de datos MySQL (o MariaDB). La independencia de la solución respecto al origen de los datos es gracias a la utilización del patrón de diseño **Strategy**, que permite generar una interfaz con métodos genéricos al origen de datos, retornando siempre los datos siguiendo los principios de POO y del diseño de clases del software.

2. Entrega parcial 2

2.1. Diseño conceptual y codificación de 2 (dos) niveles de anidación de colecciones de objetos



Figura 5: Anidación de colecciones de objetos

Como podemos ver en la imagen, el proyecto está planificado para contener la colección de cursos, la que a su vez contiene un **HashMap** de *Alumnos* y el *Profesor jefe* a cargo. En cuanto a la asistencia, cada *Alumno* contendrá un **HashMap** con su asistencia personal.

2.2. Diseño conceptual y codificación de 2 (dos) clases que utilicen sobrecarga de métodos

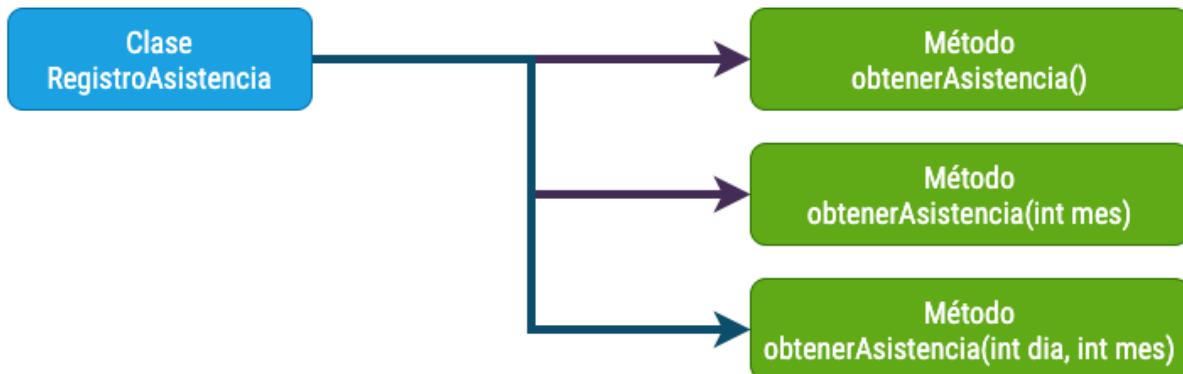


Figura 6: Sobrecarga de métodos: Clase RegistroAsistencia

Los métodos *obtenerAsistencia* de la clase **RegistroAsistencia** obtienen los datos de asistencia del alumno con los siguientes retornos:

- `public float obtenerAsistencia()`: Retorna el promedio de asistencia del alumno, calculado al día de realizada la consulta.
- `public float obtenerAsistencia(int mes)`: Retorna el promedio de asistencia del alumno, del mes especificado como parámetro.
- `public float obtenerAsistencia(int dia, int mes)`: Retorna el valor de asistencia del alumno en el día y mes especificados como parámetros.

Los métodos *getAlumnos* de la interfaz **AlumnoData** obtienen los datos de los alumnos con los siguientes retornos:

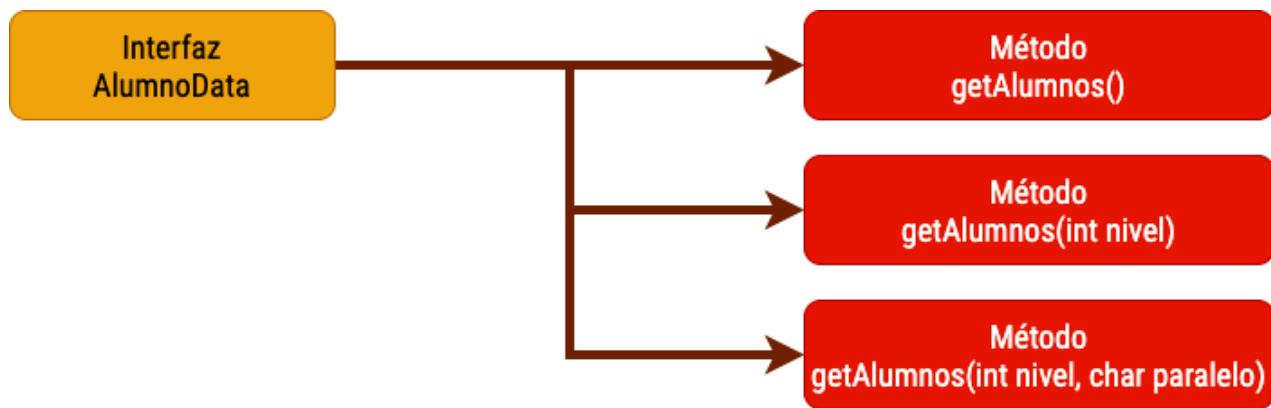


Figura 7: Sobrecarga de métodos: Interfaz AlumnoData

- `public Map<String, Alumno> getAlumnos():` Retorna un HashMap con todos los alumnos almacenados.
- `public Map<String, Alumno> getAlumnos(int nivel):` Retorna un HashMap con todos los alumnos del nivel especificado como parámetro.
- `public Map<String, Alumno> getAlumnos(int nivel, char paralelo):` Retorna un HashMap con todos los alumnos del curso especificado (nivel y paralelo).

2.3. Diseño conceptual y codificación de al menos 1 clase mapa del JCF

Cada **Curso** debe manejar un número indeterminado de **Alumnos**, y su recorrido debe ser lo más eficiente posible a la hora de buscar un alumno específico, dado que el programa debe manejar los registros de asistencia, que están anidados en cada alumno, y a su vez los alumnos están anidados en cada curso. Es por esto que se decidió por almacenar los alumnos en cada curso como **HashMap**, en donde la clave será el RUT del alumno, y el valor contendrá al alumno en sí.

☕ **Clase Curso.java**

```

1  public class Curso {
2      ...
3      private HashMap<String, Alumno> alumnos;
4      ...
5
6      public HashMap<String, Alumno> getAlumnos() {
7          ...
8      }
9  }

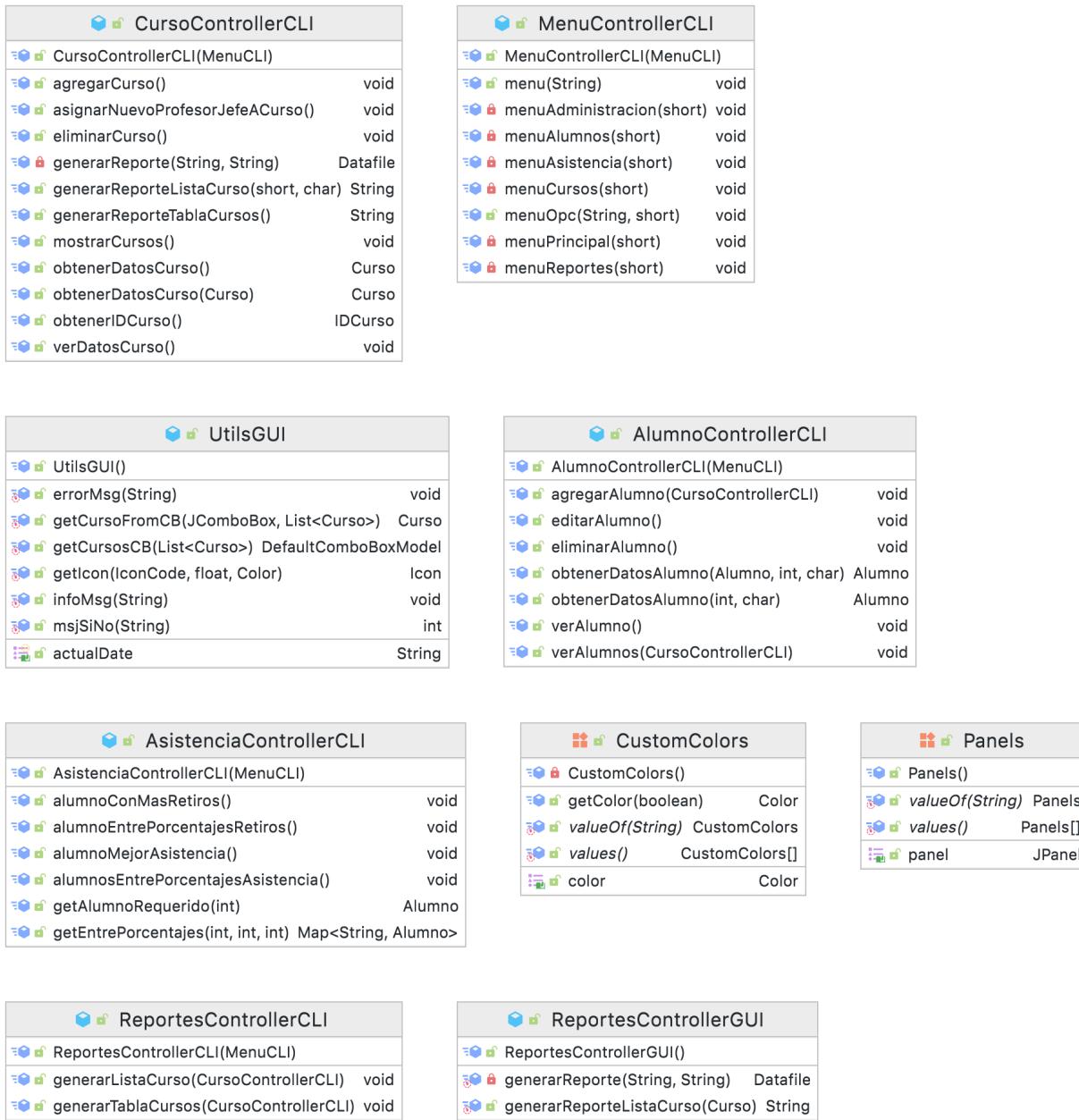
```

3. Entrega parcial 3

3.1. Diseño de diagrama de clases UML de lo considerado hasta la EP3

El diagrama UML completo está disponible en el repositorio de GitHub, y será adjuntado a la entrega final del aula virtual de la asignatura. No obstante, se detallan los paquetes generados para el funcionamiento de la aplicación:

Package controllers



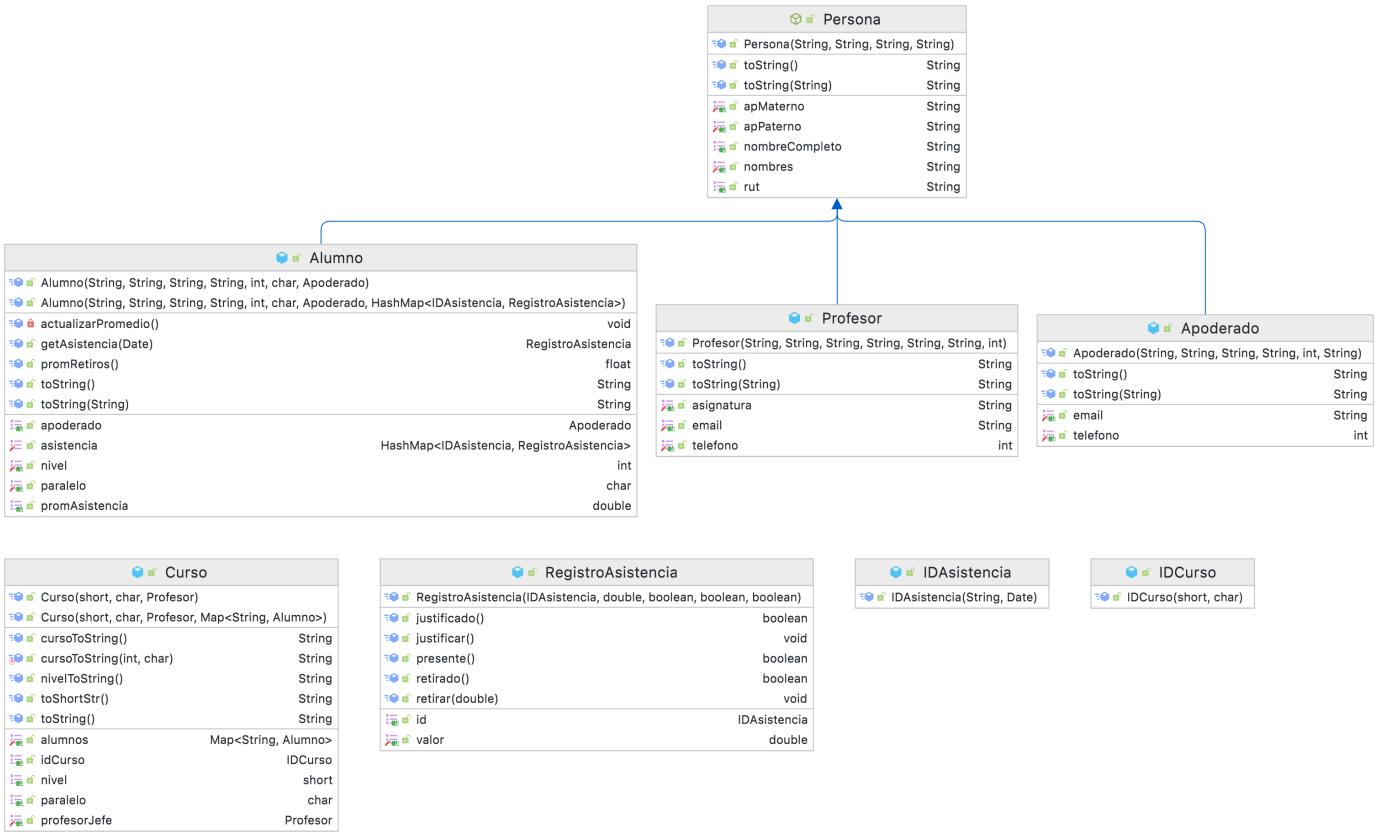
Package data.datafile

<table border="1"> <thead> <tr> <th colspan="2">AlumnoDF</th> </tr> </thead> <tbody> <tr> <td>datafile</td><td>Datafile</td></tr> <tr> <td>apDataFile</td><td>ApoderadoDF</td></tr> <tr> <td>registroAsistenciaDataFile</td><td>RegistroAsistenciaDF</td></tr> <tr> <td>AlumnoDF()</td><td></td></tr> <tr> <td>alumnoFromCSV(String)</td><td>Alumno</td></tr> <tr> <td>alumnoToCSV(Alumno)</td><td>String</td></tr> <tr> <td>deleteAlumno(Alumno)</td><td>boolean</td></tr> <tr> <td>getAlumno(String)</td><td>Alumno</td></tr> <tr> <td>getAlumnos(int)</td><td>Map<String, Alumno></td></tr> <tr> <td>getAlumnos(int, char)</td><td>Map<String, Alumno></td></tr> <tr> <td>insertAlumno(Alumno)</td><td>void</td></tr> <tr> <td>updateAlumno(Alumno)</td><td>boolean</td></tr> <tr> <td>alumnos</td><td>Map<String, Alumno></td></tr> </tbody> </table>	AlumnoDF		datafile	Datafile	apDataFile	ApoderadoDF	registroAsistenciaDataFile	RegistroAsistenciaDF	AlumnoDF()		alumnoFromCSV(String)	Alumno	alumnoToCSV(Alumno)	String	deleteAlumno(Alumno)	boolean	getAlumno(String)	Alumno	getAlumnos(int)	Map<String, Alumno>	getAlumnos(int, char)	Map<String, Alumno>	insertAlumno(Alumno)	void	updateAlumno(Alumno)	boolean	alumnos	Map<String, Alumno>	<table border="1"> <thead> <tr> <th colspan="2">CursoDF</th> </tr> </thead> <tbody> <tr> <td>datafile</td><td>Datafile</td></tr> <tr> <td>prDatafile</td><td>ProfesorDF</td></tr> <tr> <td>alDatafile</td><td>AlumnoDF</td></tr> <tr> <td>CursoDF()</td><td></td></tr> <tr> <td>cursoFromCSV(String)</td><td>Curso</td></tr> <tr> <td>cursoToCSV(Curso)</td><td>String</td></tr> <tr> <td>deleteCurso(Curso)</td><td>boolean</td></tr> <tr> <td>getCurso(IDCurso)</td><td>Curso</td></tr> <tr> <td>getCurso(short, char)</td><td>Curso</td></tr> <tr> <td>getCursos(short)</td><td>List<Curso></td></tr> <tr> <td>insertCurso(Curso)</td><td>void</td></tr> <tr> <td>updateCurso(Curso)</td><td>boolean</td></tr> <tr> <td>cursos</td><td>List<Curso></td></tr> </tbody> </table>	CursoDF		datafile	Datafile	prDatafile	ProfesorDF	alDatafile	AlumnoDF	CursoDF()		cursoFromCSV(String)	Curso	cursoToCSV(Curso)	String	deleteCurso(Curso)	boolean	getCurso(IDCurso)	Curso	getCurso(short, char)	Curso	getCursos(short)	List<Curso>	insertCurso(Curso)	void	updateCurso(Curso)	boolean	cursos	List<Curso>
AlumnoDF																																																									
datafile	Datafile																																																								
apDataFile	ApoderadoDF																																																								
registroAsistenciaDataFile	RegistroAsistenciaDF																																																								
AlumnoDF()																																																									
alumnoFromCSV(String)	Alumno																																																								
alumnoToCSV(Alumno)	String																																																								
deleteAlumno(Alumno)	boolean																																																								
getAlumno(String)	Alumno																																																								
getAlumnos(int)	Map<String, Alumno>																																																								
getAlumnos(int, char)	Map<String, Alumno>																																																								
insertAlumno(Alumno)	void																																																								
updateAlumno(Alumno)	boolean																																																								
alumnos	Map<String, Alumno>																																																								
CursoDF																																																									
datafile	Datafile																																																								
prDatafile	ProfesorDF																																																								
alDatafile	AlumnoDF																																																								
CursoDF()																																																									
cursoFromCSV(String)	Curso																																																								
cursoToCSV(Curso)	String																																																								
deleteCurso(Curso)	boolean																																																								
getCurso(IDCurso)	Curso																																																								
getCurso(short, char)	Curso																																																								
getCursos(short)	List<Curso>																																																								
insertCurso(Curso)	void																																																								
updateCurso(Curso)	boolean																																																								
cursos	List<Curso>																																																								
<table border="1"> <thead> <tr> <th colspan="2">RegistroAsistenciaDF</th> </tr> </thead> <tbody> <tr> <td>datafile</td><td>Datafile</td></tr> <tr> <td>RegistroAsistenciaDF()</td><td></td></tr> <tr> <td>actualizarRegistroAsistencia(RegistroAsistencia)</td><td>void</td></tr> <tr> <td>eliminarRegistroAsistencia(RegistroAsistencia)</td><td>void</td></tr> <tr> <td>get(IDAsistencia)</td><td>RegistroAsistencia</td></tr> <tr> <td>getRegistroAsistencia(Date)</td><td>HashMap<IDAsistencia, RegistroAsistencia></td></tr> <tr> <td>getRegistroAsistencia(IDAsistencia)</td><td>HashMap<IDAsistencia, RegistroAsistencia></td></tr> <tr> <td>getRegistroAsistencia(String)</td><td>HashMap<IDAsistencia, RegistroAsistencia></td></tr> <tr> <td>insertarRegistroAsistencia(RegistroAsistencia)</td><td>void</td></tr> <tr> <td>registroAsistenciaToCSV(RegistroAsistencia)</td><td>String</td></tr> <tr> <td>registroAsistenciaFromCSV(String)</td><td>RegistroAsistencia</td></tr> <tr> <td>registroAsistencia</td><td>HashMap<IDAsistencia, RegistroAsistencia></td></tr> </tbody> </table>	RegistroAsistenciaDF		datafile	Datafile	RegistroAsistenciaDF()		actualizarRegistroAsistencia(RegistroAsistencia)	void	eliminarRegistroAsistencia(RegistroAsistencia)	void	get(IDAsistencia)	RegistroAsistencia	getRegistroAsistencia(Date)	HashMap<IDAsistencia, RegistroAsistencia>	getRegistroAsistencia(IDAsistencia)	HashMap<IDAsistencia, RegistroAsistencia>	getRegistroAsistencia(String)	HashMap<IDAsistencia, RegistroAsistencia>	insertarRegistroAsistencia(RegistroAsistencia)	void	registroAsistenciaToCSV(RegistroAsistencia)	String	registroAsistenciaFromCSV(String)	RegistroAsistencia	registroAsistencia	HashMap<IDAsistencia, RegistroAsistencia>	<table border="1"> <thead> <tr> <th colspan="2">Datafile</th> </tr> </thead> <tbody> <tr> <td>file</td><td>File</td></tr> <tr> <td>type</td><td>String</td></tr> <tr> <td>Datafile(String)</td><td></td></tr> <tr> <td>clearFile()</td><td>void</td></tr> <tr> <td>deleteLine(String)</td><td>boolean</td></tr> <tr> <td>insertLine(String)</td><td>boolean</td></tr> <tr> <td>listToCSV(List<String>)</td><td>String</td></tr> <tr> <td>updateLine(String, String)</td><td>boolean</td></tr> <tr> <td>data</td><td>List<String></td></tr> <tr> <td>fileExists</td><td>boolean</td></tr> <tr> <td>filePath</td><td>String</td></tr> </tbody> </table>	Datafile		file	File	type	String	Datafile(String)		clearFile()	void	deleteLine(String)	boolean	insertLine(String)	boolean	listToCSV(List<String>)	String	updateLine(String, String)	boolean	data	List<String>	fileExists	boolean	filePath	String						
RegistroAsistenciaDF																																																									
datafile	Datafile																																																								
RegistroAsistenciaDF()																																																									
actualizarRegistroAsistencia(RegistroAsistencia)	void																																																								
eliminarRegistroAsistencia(RegistroAsistencia)	void																																																								
get(IDAsistencia)	RegistroAsistencia																																																								
getRegistroAsistencia(Date)	HashMap<IDAsistencia, RegistroAsistencia>																																																								
getRegistroAsistencia(IDAsistencia)	HashMap<IDAsistencia, RegistroAsistencia>																																																								
getRegistroAsistencia(String)	HashMap<IDAsistencia, RegistroAsistencia>																																																								
insertarRegistroAsistencia(RegistroAsistencia)	void																																																								
registroAsistenciaToCSV(RegistroAsistencia)	String																																																								
registroAsistenciaFromCSV(String)	RegistroAsistencia																																																								
registroAsistencia	HashMap<IDAsistencia, RegistroAsistencia>																																																								
Datafile																																																									
file	File																																																								
type	String																																																								
Datafile(String)																																																									
clearFile()	void																																																								
deleteLine(String)	boolean																																																								
insertLine(String)	boolean																																																								
listToCSV(List<String>)	String																																																								
updateLine(String, String)	boolean																																																								
data	List<String>																																																								
fileExists	boolean																																																								
filePath	String																																																								

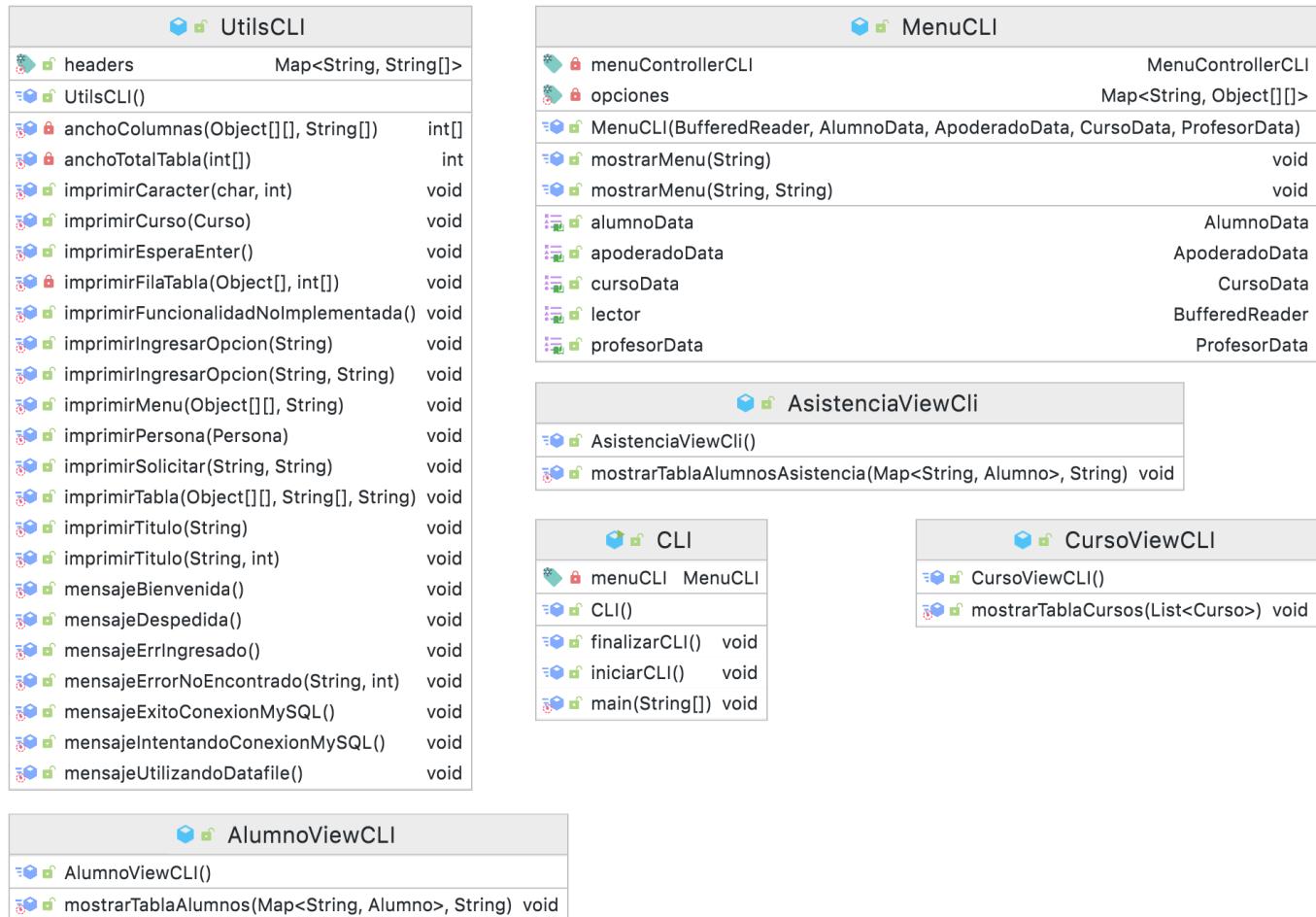
Package data.database

SQLSentences <ul style="list-style-type: none"> GET_TODOS_ALUMNOS GET_ALUMNOS_NIVEL GET_ALUMNOS_NIVEL_PAR GET_ALUMNO_RUT INSERT_ALUMNO UPDATE_ALUMNO DELETE_ALUMNO GET_PROFESORES GET_PROFESOR INSERT_PROFESOR UPDATE_PROFESOR DELETE_PROFESOR GET_TODOS_CURSOS GET_CURSOS_NIVEL GET_CURSO INSERT_CURSO UPDATE_CURSO DELETE_CURSO GET_APODERADO INSERT_APODERADO UPDATE_APODERADO DELETE_APODERADO GET_REG_AS_ID GET_REG_AS_RUT GET_REG_AS_FECHA GET_TODOS_REG_AS INSERT_REG_AS UPDATE_REG_AS DELETE_REG_AS SQLSentences() valueOf(String) SQLSentences values() SQLSentences[] 	AlumnoDB <ul style="list-style-type: none"> apoderadoData ApoderadoDB registroAsistenciaData RegistroAsistenciaDB AlumnoDB() deleteAlumno(Alumno) boolean getAlumno(String) Alumno getAlumnos(int) Map<String, Alumno> getAlumnos(int, char) Map<String, Alumno> insertAlumno(Alumno) void updateAlumno(Alumno) boolean alumnos Map<String, Alumno> 	CursoDB <ul style="list-style-type: none"> alumnoData AlumnoDB profesorData ProfesorDB CursoDB() deleteCurso(Curso) boolean getCurso(IDCurso) Curso getCurso(short, char) Curso getCursos(short) List<Curso> insertCurso(Curso) void updateCurso(Curso) boolean cursos List<Curso> 	
	RegistroAsistenciaDB <ul style="list-style-type: none"> actualizarRegistroAsistencia(RegistroAsistencia) void eliminarRegistroAsistencia(RegistroAsistencia) void get(IDAsistencia) RegistroAsistencia getRegistroAsistencia(Date) HashMap<IDAsistencia, RegistroAsistencia> getRegistroAsistencia(IDAsistencia) HashMap<IDAsistencia, RegistroAsistencia> getRegistroAsistencia(String) HashMap<IDAsistencia, RegistroAsistencia> insertarRegistroAsistencia(RegistroAsistencia) void registroAsistencia HashMap<IDAsistencia, RegistroAsistencia> 	ProfesorDB <ul style="list-style-type: none"> ProfesorDB() deleteProfesor(Profesor) void getProfesor(String) Profesor getProfesorJefe(Curso) Profesor insertProfesor(Profesor) boolean updateProfesor(Profesor) boolean profesores List<Profesor> 	DBConnection <ul style="list-style-type: none"> dotenv Dotenv connection Connection DBConnection() connect() Connection? getQuery(String) ResultSet? updateQuery(String) int
	ApoderadoDB <ul style="list-style-type: none"> ApoderadoDB() deleteApoderado(Apoderado) boolean getApoderado(String) Apoderado insertApoderado(Apoderado) boolean updateApoderado(Apoderado) boolean 		

Package models



Package views.cli



Package views.gui

AlumnoViewGUI	GUI	MostrarViewGUI
alumnoData	AlumnoData	cursoData
apoderadoData	ApoderadoData	bg
cursoData	List<Curso>	JButton1
bg	ShowPanelGUI	JButton
xMouse	int	JComboBox1
yMouse	int	JLabel1
apMatApTextField	JTextField	JLabel14
apMatTextField	JTextField	JLabel15
apPatApTextField	JTextField	JLabel
apPatTextField	JTextField	nombresAlumnoTextField
emailApTextField	JTextField	rutAlumnoTextField
jButton1	JButton	rutApoderadoTextField
jComboBox1	JComboBox<String>	telefonoApTextField
jLabel13	JLabel	AlumnoViewGUI(AlumnoData, ApoderadoData, List<Curso>, ShowPanelGUI)
jLabel14	JLabel	apMatApTextFieldFocusGained(FocusEvent)
jLabel15	JLabel	apMatApTextFieldFocusLost(FocusEvent)
nombresAlumnoTextField	JTextField	apMatTextFieldFocusGained(FocusEvent)
nombresApoderadoTextField	JTextField	apMatTextFieldFocusLost(FocusEvent)
rutAlumnoTextField	JTextField	apPatApTextFieldFocusGained(FocusEvent)
rutApoderadoTextField	JTextField	apPatApTextFieldFocusLost(FocusEvent)
telefonoApTextField	JTextField	apPatTextFieldFocusGained(FocusEvent)
apMatApTextFieldFocusGained(FocusEvent)	void	apPatTextFieldFocusLost(FocusEvent)
apMatApTextFieldFocusLost(FocusEvent)	void	apMatTextFieldFocusGained(FocusEvent)
apMatTextFieldFocusGained(FocusEvent)	void	apMatTextFieldFocusLost(FocusEvent)
apMatTextFieldFocusLost(FocusEvent)	void	apPatApTextFieldFocusGained(FocusEvent)
apPatApTextFieldFocusGained(FocusEvent)	void	apPatApTextFieldFocusLost(FocusEvent)
apPatApTextFieldFocusLost(FocusEvent)	void	apPatTextFieldFocusGained(FocusEvent)
apPatTextFieldFocusGained(FocusEvent)	void	apPatTextFieldFocusLost(FocusEvent)
apPatTextFieldFocusLost(FocusEvent)	void	emailApTextFieldFocusGained(FocusEvent)
emailApTextFieldFocusGained(FocusEvent)	void	emailApTextFieldFocusLost(FocusEvent)
headerMousePressed(MouseEvent)	void	headerMousePressed(MouseEvent)
initComponents()	void	initComponents()
jButton1ActionPerformed(ActionEvent)	void	jButton1ActionPerformed(ActionEvent)
jComboBox1ItemStateChanged(ItemEvent)	void	jComboBox1ItemStateChanged(ItemEvent)
nombresAlumnoTextFieldFocusGained(FocusEvent)	void	nombresAlumnoTextFieldFocusLost(FocusEvent)
nombresAlumnoTextFieldFocusLost(FocusEvent)	void	nombresApoderadoTextFieldFocusGained(FocusEvent)
nombresApoderadoTextFieldFocusLost(FocusEvent)	void	nombresApoderadoTextFieldFocusLost(FocusEvent)
apMatApTextFieldFocusGained(FocusEvent)	void	rutAlumnoTextFieldFocusGained(FocusEvent)
rutAlumnoTextFieldFocusGained(FocusEvent)	void	rutAlumnoTextFieldFocusLost(FocusEvent)
rutAlumnoTextFieldFocusLost(FocusEvent)	void	rutApoderadoTextFieldFocusGained(FocusEvent)
rutApoderadoTextFieldFocusGained(FocusEvent)	void	rutApoderadoTextFieldFocusLost(FocusEvent)
savelicon()	Icon	telefonoApTextFieldFocusGained(FocusEvent)
telefonoApTextFieldFocusGained(FocusEvent)	void	telefonoApTextFieldFocusLost(FocusEvent)
telefonoApTextFieldFocusLost(FocusEvent)	void	cursosCB
cursosCB	DefaultComboBoxModel	MostrarViewGUI(List<Curso>, ShowPanelGUI)
		alumnosATabla(Map<String, Alumno> Object[][])
		initComponents()
		jButton1ActionPerformed(ActionEvent)
		jComboBox1ItemStateChanged(ItemEvent)
		cursosCB
		DefaultComboBoxModel

BuscarViewGUI	PanelModel	ShowPanelGUI	InicioViewGUI
alumnoData	AlumnoData	jPanel1	jLabel1
apoderadoData	ApoderadoData	subtitleLbl	jTextArea1
profesorData	ProfesorData	titleLbl	InicioViewGUI()
bg	ShowPanelGUI	PanelModel(String)	initComponents()
jButton1	JButton	PanelModel(String, String)	initComponents()
jLabel1	JLabel	getTypography(int, int) Font	setActualPanel(JPanel, String)
jScrollPane1	JScrollPane	initComponents()	setActualPanel(JPanel, String, String)
jTable1	JTable	setPanel() JPanel	subtitle
jTextField1	JTextField	subtitle	title
BuscarViewGUI(AlumnoData, ApoderadoData, ProfesorData, ShowPanelGUI)	Object[] []?	title	String
buscarPersona(String)	Icon		
eyelcon()	void		
initComponents()	void		
jButton1ActionPerformed(ActionEvent)	void		

3.2. Menú del sistema

Interfaz de línea de comandos (CLI)

La interfaz de línea de comandos de la aplicación (CLI) se ejecuta a partir del método main contenido en la clase CLI, dentro del package views.cli. Cuenta con un menú principal, que es el primero que se muestra por pantalla, y permite la interacción con el usuario a través del ingreso de opciones numéricas.

>_ Consola: Menú principal del sistema

```
Menú principal
-> ( 1 ) - Administrar colegio
    Permite gestionar los cursos, alumnos, profesores y asistencia
-> ( 2 ) - Generar reportes
    Permite generar reportes con los datos del sistema
-> ( 3 ) - Graficos (próximamente)
    Permite obtener gráficos sobre los datos del sistema
-> ( 0 ) - Salir
    Salir de la aplicación
Ingrese su opción (numérica):
```

La opción 1 nos muestra el menú de administración del colegio, en donde se permite gestionar los datos con los que trabaja la aplicación.

>_ Consola: Menú administración

```
Menú administracion
-> ( 1 ) - Gestionar cursos
    Permite gestionar los datos de los cursos
-> ( 2 ) - Gestionar alumnos
    Permite gestionar los datos de los alumnos
-> ( 3 ) - Gestionar asistencia
    Permite gestionar la asistencia de los alumnos
-> ( 9 ) - Salir
    Salir de la aplicación
-> ( 0 ) - Volver al menú principal
    Salir de la administración del colegio
Ingrese su opción (numérica):
```

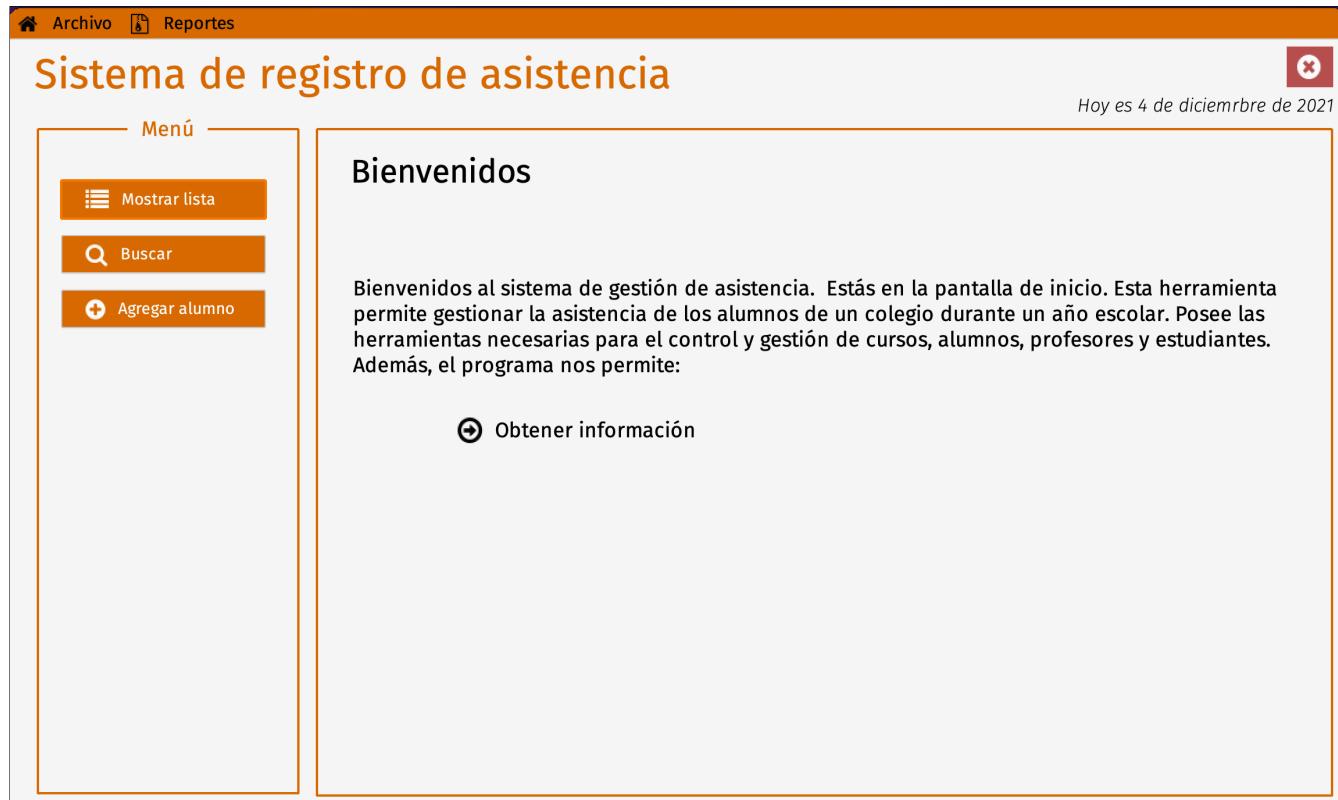
Cada menú permite gestionar los datos de cada uno de los entes involucrados. En este caso, se mostrará el menú de gestión de cursos:

>_ Consola: Menú gestión de cursos

```
Menú cursos
-> ( 1 ) - Agregar curso
    Permite agregar un curso al sistema
-> ( 2 ) - Ver datos de curso
    Permite ver los datos de un curso en específico
-> ( 3 ) - Ver cursos
    Permite ver la lista de cursos disponibles
-> ( 4 ) - Asignar nuevo profesor jefe a curso
    Permite asignar un nuevo profesor jefe al curso
-> ( 5 ) - Eliminar curso
    Permite eliminar un curso en específico
-> ( 9 ) - Salir
    Salir de la aplicación
-> ( 0 ) - Volver al menú de administración
    Salir de la gestión de cursos
Ingrese su opción (numérica):
```

Interfaz gráfica (GUI)

En el caso de la interfaz gráfica, se muestra un menú en la parte izquierda, en donde mediante botones el usuario puede seleccionar entre las funcionalidades proporcionadas.



3.2.1. Inserción manual/agregar elemento

Interfaz de línea de comandos (CLI)

Para agregar elementos al programa, se accede a través del menú en específico del tipo de dato que queremos almacenar. En este caso de ejemplo, para agregar un curso (según el menú mostrado en la parte superior) se realiza con la opción numérica 1. Al presionar el 1, el sistema solicita los datos necesarios para ser agregados:

```
>_ Consola: Funcionalidad: Agregar curso

Ingrese el nivel del curso(número de 1 a 12): 5
Ingrese el paralelo del curso(caracter): C
Ingrese el RUT del profesor jefe(RUT): 12345678-9
Ingrese los nombres del profesor jefe(texto): Profesor Jefe
Ingrese el apellido paterno del profesor jefe(texto): Ejemplo
Ingrese el apellido materno del profesor jefe(texto): Ejemplo
Ingrese la asignatura del profesor jefe(texto): Asignatura de ejemplo
Ingrese el email del profesor jefe(texto): ejemplo@ejemplo.com
Ingrese el teléfono del profesor jefe(texto): 912345678
```

Luego de agregados los datos, se genera un nuevo objeto de tipo Curso y se agrega al archivo de texto plano (a través de los métodos de la clase CursoDatafile, del package data.datafile). Una vez insertados los datos nuevos, el programa muestra la tabla con todos los cursos (y el nuevo curso también presente).

Para el caso de los Alumnos, también se agregan a través de su menú de gestión correspondiente, en donde se solicitarán los datos requeridos:

> Consola: Funcionalidad: Agregar alumno

```
Ingrese el nivel del curso(número de 1 a 12): 12
Ingrese el paralelo del curso(caracter): B
Ingrese el RUT del alumno(RUT): 12345678-9
Ingrese los nombres del alumno(texto): Alumno De
Ingresese el apellido paterno del alumno(texto): Ejemplo
Ingresese el apellido materno del alumno(texto): Ejemplo
Ingresese el RUT del apoderado(RUT): 91234567-8
Ingresese los nombres del apoderado(texto): Apoderado De
Ingresese el apellido paterno del apoderado(texto): Ejemplo
Ingresese el apellido materno del apoderado(texto): Ejemplo
Ingresese el email del apoderado(texto): apoderado@ejemplo.com
Ingresese el teléfono del apoderado(texto): 987654321
```

Al igual que con los Cursos, luego de ser agregado el nuevo alumno, se muestra una tabla con todos los alumnos del curso al que fue incorporado, incluyendo el dato nuevo.

Interfaz gráfica (GUI)

The screenshot shows a window titled "Sistema de registro de asistencia". On the left, there's a sidebar menu with three buttons: "Mostrar lista", "Buscar", and "Agregar alumno" (highlighted in orange). The main area has a title "Agregar alumno" and a subtitle "Agregando alumno al 4 básico A". It contains two sections: "Curso del alumno" (with a dropdown menu showing "4 básico A") and "Datos alumno" (with input fields for "Nombres alumno", "Apellido paterno alumno", and "Apellido materno alumno"). To the right is another section "Datos apoderado" with input fields for "RUT Apoderado", "Nombres apoderado", "Apellido paterno apoderado", "Apellido materno apoderado", "Correo electrónico apoderado", and "Teléfono apoderado". At the bottom is a "Guardar" button.

En el caso de la interfaz gráfica, ésta nos permite agregar un alumno al sistema. Primero se debe seleccionar el curso al que se desea agregar el alumno, y luego completar los datos del alumno y del apoderado respectivamente. El sistema posee una validación al número de teléfono, que sólo puede corresponder a caracteres numéricos. Si esto no se cumple, se genera un mensaje de alerta:

Archivo Reportes

Sistema de registro de asistencia

Hoy es 4 de diciembre de 2021

Menú

- Mostrar lista
- Buscar
- Agregar alumno

Agregar alumno

Agregando alumno al 4 básico A

Curso del alumno

4 básico A

Error

El formato del teléfono debe ser numérico.

RUT alumno Nombres alumno Apellido paterno alumno Apellido materno alumno

Datos apoderado

Nombres apoderado Apellido paterno apoderado Apellido materno apoderado Email apoderado fdaasdf

Cancelar Aceptar Guardar

3.2.2. Mostrar por pantalla listado de elementos. Esto para cada una de las 2 colecciones anidadas

Interfaz de linea de comandos (CLI)

Al igual que con la funcionalidad de agregar elementos, en este caso para listar los elementos, se debe acceder desde el menú que gestiona el tipo de elemento que deseamos recuperar.

Para el caso de los cursos, son listados de la siguiente manera:

> Consola: Funcionalidad: Mostrar cursos

Cursos					
Curso	Nivel	Paralelo	Nombre profesor jefe	Email profesor jefe	Teléfono profesor jefe
1BA	1	A	Shanta Hilll Senger	sue.casper@hotmail.com	987659326
1BB	1	B	Elden Kautzer Abshire	maxima.tremblay@hotmail.com	978042546
2BA	2	A	Grant Cartwright Ferry	jaymie.anderson@gmail.com	947869804
2BB	2	B	Claudette Baumbach Roob	lasonya.swaniawski@yahoo.com	954534919
3BA	3	A	Alex Parisian Hintz	darell.lueilwitz@hotmail.com	952017013
3BB	3	B	Darrel Nicolas Cole	milagros.emmerich@hotmail.com	948755395
4BA	4	A	Tamisha Volkman Fadel	joaquin.stiedemann@gmail.com	985662322
4BB	4	B	Jacques Wolff Bednar	donella.lockman@gmail.com	984890419
5BA	5	A	Marylee Donnelly Bernier	evelin.abshire@gmail.com	990485383
5BB	5	B	Inge Fay McCullough	sharda.fahrey@yahoo.com	991602692
6BA	6	A	Bruce Kulas Goodwin	george.champlin@gmail.com	975853419
6BB	6	B	Marcelle Emmerich Williamson	nickolas.poures@gmail.com	961721076
7BA	7	A	Kory O'Kon Brakus	thanh.waters@hotmail.com	974036963
7BB	7	B	Elease White Jerde	elli.mayer@gmail.com	988854022
8BA	8	A	Kandace Hahn Hammes	edmond.corwin@yahoo.com	950970290
8BB	8	B	Loyce Krajcik Jakubowski	jenine.keebler@hotmail.com	943748407
1MA	9	A	Teodoro Pfeffer Ankunding	lottie.weissnat@gmail.com	993288595
1MB	9	B	Denita Kuvalis Heller	shila.morisette@yahoo.com	969915007
2MA	10	A	Raphael Walter Stoltenberg	hyacinth.senger@gmail.com	984581430
2MB	10	B	Elton Beatty McLaughlin	joya.barrows@gmail.com	941208950
3MA	11	A	Lacy Cassin Bauch	harlan.mitchell@hotmail.com	975709937
3MB	11	B	Dede Goodwin Schuppe	ryan.hand@gmail.com	966145453
4MA	12	A	Noelia Hagenes Swift	renate.swaniawski@hotmail.com	941278587
4MB	12	B	Leif Zulauf Boyle	celina.keeling@yahoo.com	973953363

Y en el caso de los alumnos, se lista de la siguiente manera:

> Consola: Funcionalidad: Mostrar alumnos

Alumnos del 1 medio A

RUT	Ap. Paterno	Ap. Materno	Nombres	Nombre completo apoderado	Teléfono apoderado
45720666-8	Greenholt	Becker	Jamison	Dante Greenholt Beier	985287105
46254622-3	Cassin	Beahan	Byron	Solomon Cassin Goldner	971498469
43100387-7	Wilkinson	Fay	Delmer	Elvin Wilkinson Bechtelar	973228091
37002298-3	Murray	Cummings	Antony	Felicia Murray Lang	946952504
47438615-1	White	Mayer	Katharyn	Katherine White Moore	950651632
45933943-8	Emard	Sauer	Deon	Doretta Emard Mertz	982425038
34581069-3	Reilly	Medhurst	Kasey	Melodee Reilly Walter	947910091
34665351-7	Carroll	Jenkins	Jesica	Weldon Carroll Wisoky	949098199
38226629-8	Legros	Beier	Kristofer	Damion Legros Hegmann	985372752
47026929-7	Huels	Schuppe	Raisa	Elvera Huels Reilly	989935616
38273902-5	Quigley	Conn	Delmy	Saul Quigley Pauck	979615506
31365566-2	D'Amore	Haag	Exie	Darwin D' Amore Schumm	950907383
49860824-4	Torp	Morissette	Michael	Prince Torp Schneider	946380232
49642433-6	Kris	Johns	Ehtel	Christin Kris Zemlak	996654940
37953682-2	MacGyver	Leuschke	Candance	Dane MacGyver Hettinger	997309006

Interfaz gráfica (GUI)

La interfaz gráfica nos permite listar los alumnos de un curso determinado, pudiendo accesar a esta funcionalidad mediante el menú izquierdo.

The screenshot shows the main interface of the 'Sistema de registro de asistencia'. At the top, there's a navigation bar with icons for Home, Archivo, and Reportes. To the right of the navigation bar, it says 'Hoy es 4 de diciembre de 2021'. On the left, there's a vertical 'Menú' sidebar with three buttons: 'Mostrar lista', 'Buscar', and 'Agregar alumno'. The main content area has a title 'Mostrar lista de curso' and a subtitle 'Alumnos del 1 básico A'. Below this, there's a dropdown menu labeled 'Seleccione el curso:' with '1 básico A' selected. To the right of the dropdown is a 'Ver' button. The main table displays student information with columns: RUT, Nombres, Ap. Paterno, Ap. Materno, Apoderado, and Teléfono. The table contains 20 rows of student data.

RUT	Nombres	Ap. Paterno	Ap. Materno	Apoderado	Teléfono
39360608-6	Gwendolyn	Prosacco	Gusikowski	Rhonda Prosacco G...	983791542
34794837-5	Suzy	Schoen	Graham	Oren Schoen Cum...	984880372
34666938-5	Alvaro	Cremin	Abbott	Steve Cremin Bau...	984858467
32567619-8	Ethelene	Bayer	Turner	Winfred Bayer Mur...	991969745
35205567-7	Elicia	Ryan	Weimann	Maryanna Ryan De...	980388912
43943896-7	Wilda	Bruen	Halvorson	Antonetta Bruen B...	967039591
47291340-4	Ron	Sporer	Considine	Carita Sporer Fisher	958684390
35553183-8	Arica	VonRueden	Marks	Juliette VonRueden...	954665316
48458775-5	Catarina	Romaguera	Cremin	Florencia Romague...	986294679
32068786-3	Troy	Witting	Pagac	Tameika Witting La...	952448999
37875472-4	Laura	O'Reilly	Sawayn	Opal O'Reilly Blick	941214070
44754068-1	Kristine	West	Grimes	Irwin West Dietrich	964949933
35714393-3	Lore	Wolff	Cole	Dannie Wolff Kuhic	963494561
42704379-8	Terry	Batz	Mohr	Hipolito Batz Lock...	979068726
32521196-7	Tommie	Bergnaum	Cremin	Kathyne Bergnau...	970127187

A. Entrega acumulada A

A.2. Menú con funcionalidades de edición y eliminación de elementos

A través del menú en la interfaz de consola de comandos (CLI), también nos permite editar y eliminar datos relacionados a los Cursos y los Alumnos. Específicamente, se puede:

- Editar o eliminar alumnos: Se pueden modificar los datos del alumno (exceptuando el RUT). No así los datos del apoderado, ya que es una funcionalidad que estará disponible en las entregas siguientes. También se pueden eliminar los alumnos, lo que provoca a su vez que se elimine del sistema al apoderado que tenga asociado.

```
>_ Consola: Menú alumnos

Menú alumnos
-> ( 1 ) - Agregar alumno
    Permite agregar un alumno al curso
-> ( 2 ) - Ver datos de alumno
    Permite ver los datos de un alumno en específico
-> ( 3 ) - Ver alumnos
    Permite ver la lista de alumnos del curso
-> ( 4 ) - Editar alumno
    Permite editar un alumno en específico
-> ( 5 ) - Eliminar alumno
    Permite eliminar un alumno en específico
-> ( 9 ) - Salir
    Salir de la aplicación
-> ( 0 ) - Volver al menú de administración
    Salir de la gestión de alumnos

Ingrese su opción (numérica):
```

- Asignar nuevo profesor jefe a curso: Se puede modificar un curso, asignándole un nuevo profesor jefe y eliminando al anterior.
- Eliminar curso: El programa permite la eliminación completa de un curso, la que provocará también la eliminación de los datos del profesor, alumnos, y por consiguiente, de los apoderados.

```
>_ Consola: Menú cursos

Menú cursos
-> ( 1 ) - Agregar curso
    Permite agregar un curso al sistema
-> ( 2 ) - Ver datos de curso
    Permite ver los datos de un curso en específico
-> ( 3 ) - Ver cursos
    Permite ver la lista de cursos disponibles
-> ( 4 ) - Asignar nuevo profesor jefe a curso
    Permite asignar un nuevo profesor jefe al curso
-> ( 5 ) - Eliminar curso
    Permite eliminar un curso en específico
-> ( 9 ) - Salir
    Salir de la aplicación
-> ( 0 ) - Volver al menú de administración
    Salir de la gestión de cursos
```

A.3. Se debe generar un reporte en archivo txt que considere mostrar datos de las 2 colecciones anidadas (ej: csv)

El programa permite la generación de reportes en formato CSV (separados por comas), tanto en la interfaz de linea de comandos, como en la interfaz gráfica. Los reportes que se pueden generar son los siguientes:

- Lista del curso: Este archivo CSV contendrá los datos de los alumnos de un curso en específico, entregado por el usuario.
- Tabla de cursos: Este archivo CSV contendrá los datos generales de todos los cursos a los que el programa tiene acceso. Esta tabla, además de mostrar los datos del curso mismo, muestra datos generales del profesor jefe a cargo del mismo.

> Consola: Menú reportes

```
Menú reportes
-> ( 1 ) - Generar lista de curso
    Permite generar un archivo CSV con la lista de un curso (alumnos enumerados)
-> ( 2 ) - Generar tabla de cursos
    Permite generar una tabla con todos los cursos del colegio
-> ( 9 ) - Salir
    Salir de la aplicación
-> ( 0 ) - Volver al menú principal
    Salir de la generación de reportes
Ingrese su opción (numérica):
```

The screenshot shows the 'Sistema de registro de asistencia' application. At the top, there's a navigation bar with 'Archivo' and 'Reportes' tabs, and a sub-menu 'Lista de curso'. The main title 'Sistema de registro de asistencia' is centered above a search form. On the left, a sidebar titled 'Menú' contains three buttons: 'Mostrar lista', 'Buscar', and 'Agregar alumno'. The main area is titled 'Buscar persona' and has a placeholder 'Ingrese rut de persona'. Below it is a search bar with a magnifying glass icon and the word 'Buscar'. A table below the search bar has columns for 'rut', 'Nombres', 'Apellido Paterno', 'Apellido materno', and 'Tipo'. The date 'Hoy es 4 de diciembre de 2021' is displayed at the top right.

This screenshot shows the same application interface as the previous one, but with a modal dialog box overlaid. The dialog is titled 'Seleccione el curso a generar reporte' and contains a dropdown menu with the option '1º básico A'. There are 'Aceptar' and 'Cancelar' buttons at the bottom. The rest of the interface remains visible in the background.

This screenshot shows the application after a report has been generated. A modal dialog box is displayed with the message: 'El reporte se ha generado con éxito en /Users/leivajacinto/Proyectos/proyectoro/estaticos/reportes/lista-curso-6-B.csv. Deseas abrir el archivo?'. It includes 'No' and 'Si' buttons. The rest of the application interface is visible in the background.

A.4. El código fuente debe estar bien modularizado de acuerdo a lo descrito en el informe además de seguir las buenas prácticas de documentación interna y legibilidad

En el desarrollo de este proyecto, se siguieron las buenas prácticas tanto en los estándares de codificación, como en el desarrollo de la documentación. Más detalles respecto al desarrollo de este proyecto, se puede encontrar más adelante en este mismo informe, al que se puede acceder [siguiendo este link](#)

A.5. Todas las funcionalidades pueden ser implementadas mediante consola

Todas las funcionalidades planteadas para el sistema fueron implementadas en la interfaz de consola de comandos (CLI). Solo algunas funcionalidades, indicadas más adelante en este reporte, fueron implementadas a la interfaz gráfica (GUI).

A.6. Utilizacion de GitHub (Realización de al menos 3 Commit)

Respecto a la utilización de la herramienta **GitHub** para control de versiones, se detalla su uso en el apartado Consideraciones, al que se puede acceder [siguiendo este link](#).

4. Entrega parcial 4

4.1. Se deben incluir al menos 2 funcionalidades propias que sean de utilidad para el negocio (distintas de la inserción, edición, eliminación y reportes)

Se ha mejorado el menú principal que se encuentra en la clase CLI, dentro del package **views.cli**, permitiendo funciones específicas de nuestro proyecto como la muestra de información de los alumnos con mejor asistencia, alumnos entre cierto porcentaje de asistencia o retiro.

```
>_ Consola: Menú asistencia

Menú asistencia
-> ( 1 ) - Alumno con mejor asistencia
    Permite ver el alumno con mejor asistencia del año
-> ( 2 ) - Registrar asistencia
    Permite registrar la asistencia de un curso un día determinado
-> ( 3 ) - Alumnos entre % y % de asistencia
    Permite ver los alumnos con los porcentajes de asistencia especificados
-> ( 4 ) - Alumno con mas retiros
    Permite ver el alumno con mas retiros en el año
-> ( 5 ) - Alumnos entre % y % de retiros
    Permite ver los alumnos con los porcentajes de retiros especificados
-> ( 9 ) - Salir
    Salir de la aplicación
-> ( 0 ) - Volver al menú de administración
    Salir de la gestión de cursos

Ingrese su opción (numérica):
```

4.1.1. Seleccionar un objeto por criterio, considera la selección de un objeto basado en un criterio específico, involucrando dos o más colecciones anidadas. Por ejemplo, selección del alumno con la nota final más baja de todos los cursos, o seleccionar el pasajero más joven de todos los buses de una compañía

El sistema permite obtener los datos del alumno que registra más retiros durante un año escolar, mediante el menú asistencia, opción 4.

4.1.2. Subconjunto filtrado por criterio: considera la selección de un subconjunto de objetos basado en un criterio específico, involucrando dos o más colecciones anidadas. Por ejemplo, selección de los alumnos con nota final entre 4,0 y 7,0 de entre todos los cursos; o seleccionar a todos los pasajeros que tengan asiento impar de entre todos los buses de la compañía

El sistema permite obtener los datos de aquellos alumnos que estén entre un rango (porcentaje) de asistencia o retiros definido por el usuario, mediante el menú asistencia, opciones 3 y 5.

4.2. Diseño y codificación de 2 (dos) clases que utilicen sobreescritura de métodos

Las clases que hemos diseñado y codificado con métodos de sobre escritura son aquellas que poseen colección en el package **database** y **datafile**.

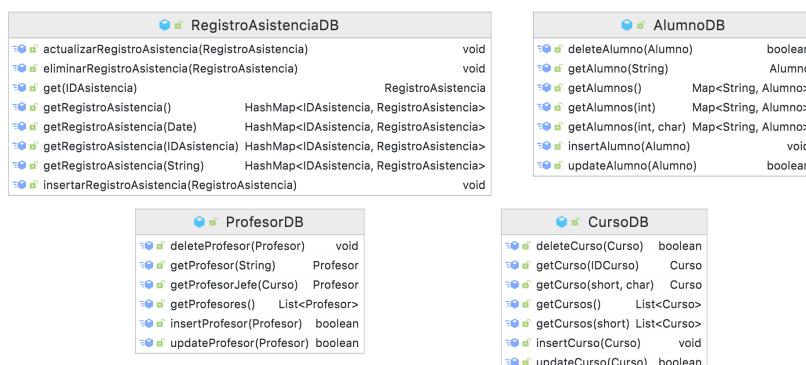


Figura 9: Métodos paquete Database

AlumnoDF		CursoDF	
datafile	Datafile	datafile	Datafile
apDataFile	ApoderadoDF	prDatafile	ProfesorDF
registroAsistenciaDataFile	RegistroAsistenciaDF	alDatafile	AlumnoDF
AlumnoDF()		CursoDF()	
alumnoFromCSV(String)	Alumno	cursoFromCSV(String)	Curso
alumnoToCSV(Alumno)	String	cursoToCSV(Curso)	String
deleteAlumno(Alumno)	boolean	deleteCurso(Curso)	boolean
getAlumno(String)	Alumno	getCurso(IDCurso)	Curso
getAlumnos(int)	Map<String, Alumno>	getCurso(short, char)	Curso
getAlumnos(int, char)	Map<String, Alumno>	getCursos(short)	List<Curso>
insertAlumno(Alumno)	void	insertCurso(Curso)	void
updateAlumno(Alumno)	boolean	updateCurso(Curso)	boolean
alumnos	Map<String, Alumno>	cursos	List<Curso>

RegistroAsistenciaDF		ProfesorDF	
datafile	Datafile	datafile	Datafile
RegistroAsistenciaDF()		ProfesorDF()	
actualizarRegistroAsistencia(RegistroAsistencia)	void	deleteProfesor(Profesor)	void
eliminarRegistroAsistencia(RegistroAsistencia)	void	getProfesor(String)	Profesor
getIDAsistencia()	RegistroAsistencia	getProfesorJefe(Curso)	Profesor
getRegistroAsistencia(Date)	HashMap<IDAsistencia, RegistroAsistencia>	insertProfesor(Profesor)	boolean
getRegistroAsistencia(IDAsistencia)	HashMap<IDAsistencia, RegistroAsistencia>	profesorFromCSV(String)	Profesor
getRegistroAsistencia(String)	HashMap<IDAsistencia, RegistroAsistencia>	profesorToCSV(Profesor)	String
insertarRegistroAsistencia(RegistroAsistencia)	void	updateProfesor(Profesor)	boolean
registroAsistenciaToCSV(RegistroAsistencia)	String	profesores	List<Profesor>
registroAsistenciaFromCSV(String)	RegistroAsistencia		
registroAsistencia	HashMap<IDAsistencia, RegistroAsistencia>		

Figura 10: Métodos paquete Datafile

4.3. Diseño y codificación de 1 (una) clase abstracta que sea padre de al menos 2 (dos) clases. La clase abstracta debe ser utilizada por alguna otra clase (contexto)

La clase abstracta codificada es la **Persona** que posee las características: rut, nombres, apellido paterno y apellido materno, las cuales son pasadas a sus hijos.

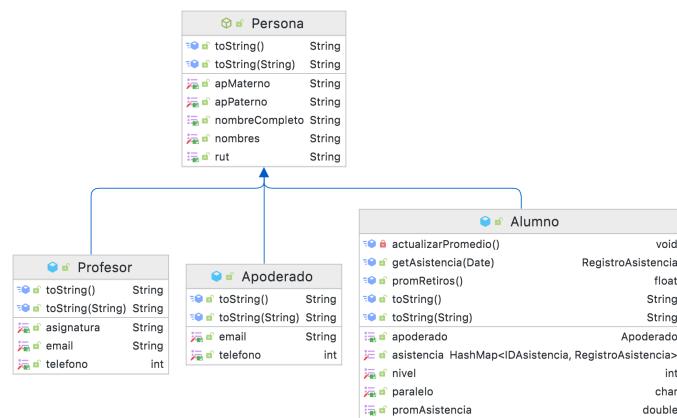


Figura 11: Clase abstracta Persona y sus clases hijas

4.4. Diseño y codificación de 1 (una) interfaz que sea implementada por al menos 2 (dos) clases. La interfaz debe ser utilizada por alguna otra clase (contexto)

Las interfaz diseñadas son al igual que en el punto 2 aquellas que poseen colección el package **database** y **datafile**. El diseño de la interfaz, en conjunto con el patrón de diseño *Strategy*, permite la independencia del sistema al origen de datos, pudiendo cambiar este último sin inconvenientes, mientras implemente los métodos descritos en las interfaces presentes en el package **Data**.

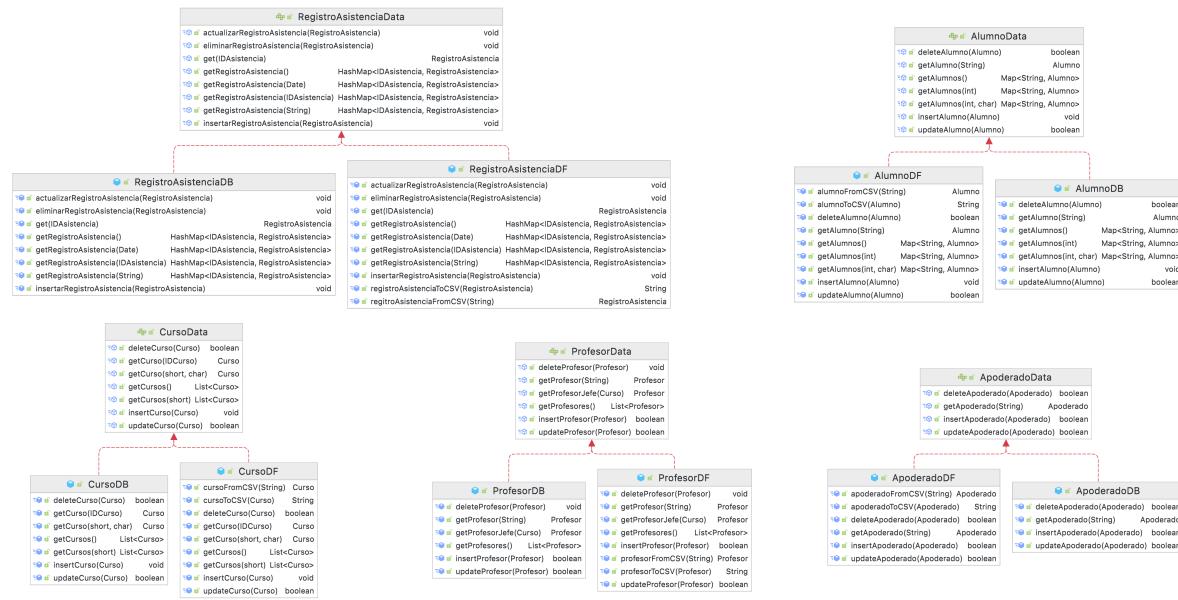


Figura 12: Interfaces Data y sus implementaciones

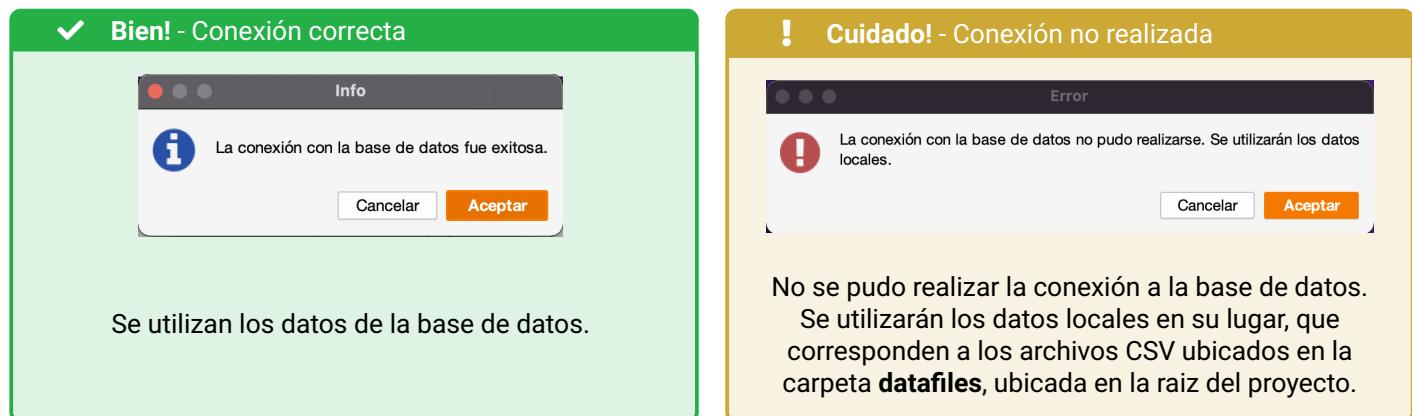
Opcional: Generar documentación con Javadoc.

El desarrollo de la documentación para el proyecto, se puede encontrar en el [siguiente este link](#)

B. Entrega acumulada B

B.2. Se deben implementar al menos 3 ventanas gráficas (GUIs en AWT o SWING): 1 ventana de menú, 1 ventana de agregar elemento y 1 ventana de listar elementos

Cuando inicializa el programa puede emerger una ventana en caso de una correcta conexión con la base de datos o de caso contrario un error y la opción de trabajar con datos locales.



A continuación se muestra la ventana de bienvenida, al lado izquierdo se ve el menú que será visible en todo momento para navegar fácilmente cuando sea requerido dentro de las funcionalidades del programa.

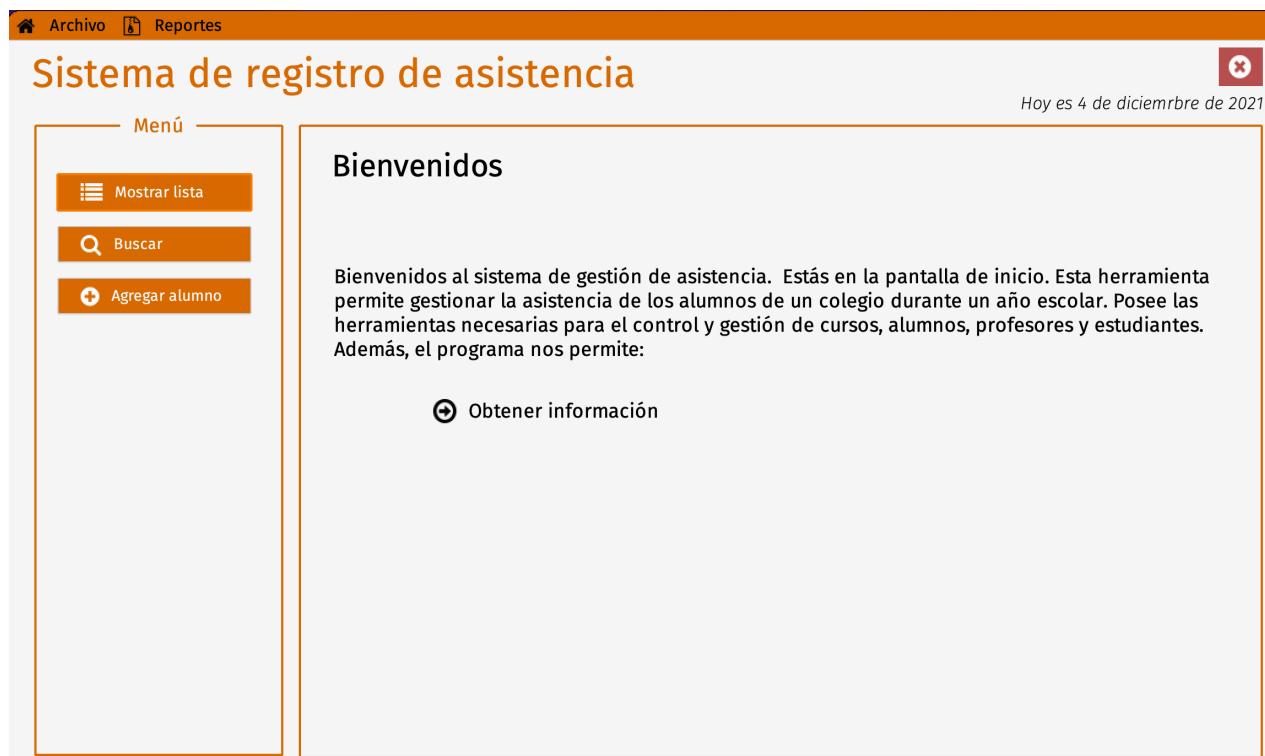


Figura 13: Ventana de bienvenida al programa

Extra: Modo oscuro

Gracias a la inclusión de la librería externa **FlatLaf**, y su inclusión de diferentes estilos y colores gráficos para **Java Swing**, el programa puede cambiar su apariencia en tiempo de ejecución, lo que permite incorporar la opción de ver el programa en **modo oscuro**, opción de la que se puede acceder mediante el menú superior de la ventana.

Info - Modo oscuro

Sistema de registro de asistencia

Hoy es 4 de diciembre de 2021

Menú

- Mostrar lista
- Buscar
- Agregar alumno

Agregar alumno

Agregando alumno al 4 básico A

Curso del alumno

4 básico A

Datos alumno

RUT alumno
Nombres alumno
Apellido paterno alumno
Apellido materno alumno

Datos apoderado

RUT Apoderado
Nombres apoderado
Apellido paterno apoderado
Apellido materno apoderado
Email apoderado
Teléfono apoderado

Guardar

Mostrar lista de curso

Seleccione curso

Seleccione el curso: 1 básico A Ver

RUT	Nombres	Ap. Paterno	Ap. Materno	Apoderado	Teléfono
-----	---------	-------------	-------------	-----------	----------

La interfaz gráfica también nos permite las siguientes operaciones:
Agregar un nuevo alumno

Archivo Reportes

Sistema de registro de asistencia

Hoy es 4 de diciembre de 2021

Menú

- Mostrar lista
- Buscar
- Agregar alumno

Agregar alumno

Seleccione el curso del alumno a agregar.

Curso del alumno

1 básico A

Datos alumno

RUT alumno
Nombres alumno
Apellido paterno alumno
Apellido materno alumno

Datos apoderado

RUT Apoderado
Nombres apoderado
Apellido paterno apoderado
Apellido materno apoderado
Correo electrónico apoderado
Teléfono apoderado

Guardar

Figura 14: Ventana para agregar un nuevo alumno

OJO: La ventana posee un validador para el número de teléfono, aceptando sólo caracteres numéricos, y arrojando una excepción en caso de recibir cualquier otro carácter, mostrando una alerta al usuario.

Error! - Error al ingresar número de teléfono

Archivo Reportes

Sistema de registro de asistencia

Hoy es 4 de diciembre de 2021

Menú

- Mostrar lista
- Buscar
- Agregar alumno

Agregar alumno

Agregando alumno al 4 básico A

Curso del alumno

4 básico A

El formato del teléfono debe ser numérico.

Cancelar Aceptar

Datos alumno

RUT alumno
Nombres alumno
Apellido paterno alumno
Apellido materno alumno

Datos apoderado

RUT Apoderado
Nombres apoderado
Apellido paterno apoderado
Apellido materno apoderado
Email apoderado
fdasdf

Guardar

Listar cursos

The screenshot shows a software interface titled "Sistema de registro de asistencia". The main title bar includes "Archivo", "Reportes", and the date "Hoy es 4 de diciembre de 2021". On the left, a sidebar titled "Menú" contains three buttons: "Mostrar lista", "Buscar", and "Agregar alumno". The central panel is titled "Mostrar lista de curso" and displays a table header "Alumnos del 1 básico A". Below the header, there is a dropdown menu labeled "Seleccione el curso:" with the option "1 básico A" selected, followed by a "Ver" button. The table lists student information with columns: RUT, Nombres, Ap. Paterno, Ap. Materno, Apoderado, and Teléfono. The data is as follows:

RUT	Nombres	Ap. Paterno	Ap. Materno	Apoderado	Teléfono
39360608-6	Gwendolyn	Prosacco	Gusikowski	Rhonda Prosacco G...	983791542
34794837-5	Suzy	Schoen	Graham	Oren Schoen Cum...	984880372
34666938-5	Alvaro	Cremin	Abbott	Steve Cremin Bau...	984858467
32567619-8	Ethelene	Bayer	Turner	Winfred Bayer Mur...	991969745
35205567-7	Elicia	Ryan	Weimann	Maryanna Ryan De...	980388912
43943896-7	Wilda	Bruen	Halvorson	Antonetta Bruen B...	967039591
47291340-4	Ron	Sporer	Considine	Carita Sporer Fisher	958684390
35553183-8	Arica	VonRueden	Marks	Juliette VonKueden...	954665316
48458775-5	Catarina	Romaguera	Cremin	Florencia Romague...	986294679
32068786-3	Troy	Witting	Pagac	Tameika Witting La...	952448999
37875472-4	Laura	O'Reilly	Sawayn	Opal O'Reilly Blick	941214070
44754068-1	Kristine	West	Grimes	Irwin West Dietrich	964949933
35774393-3	Lore	Wolff	Cole	Dannie Wolff Kuhic	963494561
42704379-8	Terry	Batz	Mohr	Hipolito Batz Lock...	979068726
32521196-7	Tommie	Bergnaum	Cremin	Kathyne Bergnau...	970127187

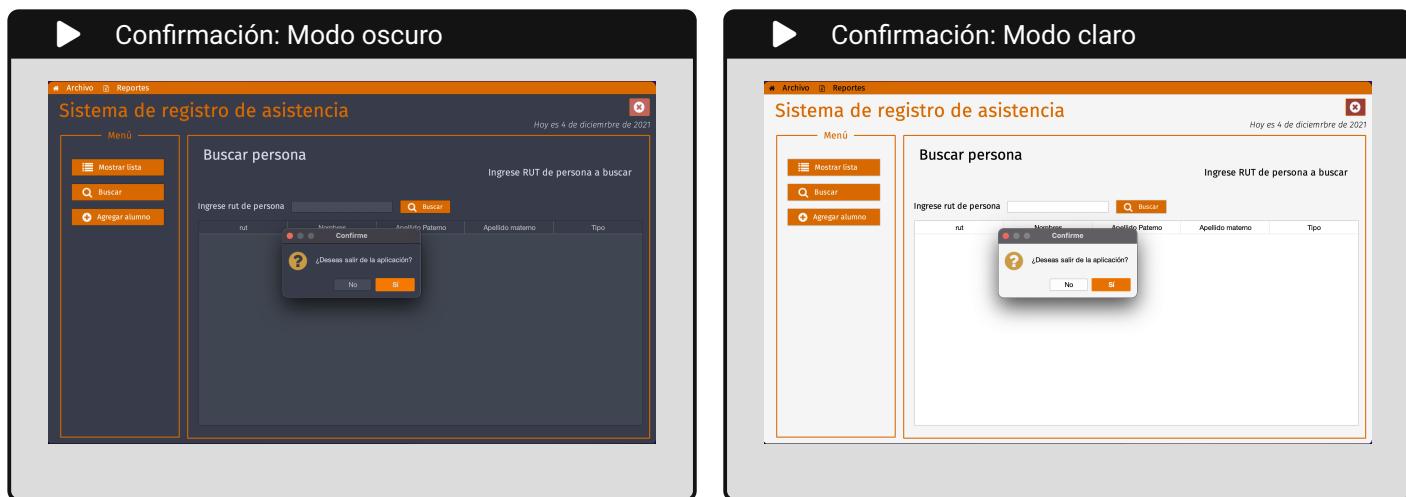
Figura 15: Ventana para mostrar lista de curso específico

Buscar persona (alumno, profesor o apoderado)

The screenshot shows a software interface titled "Sistema de registro de asistencia". The main title bar includes "Archivo", "Reportes", and the date "Hoy es 4 de diciembre de 2021". On the left, a sidebar titled "Menú" contains three buttons: "Mostrar lista", "Buscar", and "Agregar alumno". The central panel is titled "Buscar persona" and displays a search form with a placeholder "Ingrese RUT de persona a buscar". Below the form is a table with columns: rut, Nombres, Apellido Paterno, Apellido materno, and Tipo. The table is currently empty.

Figura 16: Ventana para buscar persona

Confirmación para salir de la aplicación



B.3. Se debe aplicar encapsulamiento y principios OO

Se aplicaron todos los conceptos de encapsulamiento, principios de POO y conceptos aprendidos en clases en el desarrollo de esta aplicación. Más detalles sobre el desarrollo, en términos de código, respecto a este proyecto, se puede encontrar más adelante en este mismo informe, al que se puede acceder [siguiendo este link](#)

B.4. Continuidad en la utilización de GitHub (Realización de al menos 3 Commit adicionales a los ya hechos en la parte A)

Respecto a la utilización de la herramienta **GitHub** para control de versiones, se detalla su uso en el apartado Consideraciones, al que se puede acceder [siguiendo este link](#)

F. Entrega final

F.2. Implementar el manejo de excepciones capturando errores potenciales específicos mediante Try-catch

Se utiliza en reiteradas ocasiones la captura de excepciones que puedan surgir en el programa mediante los bloques try-catch que se detallan a continuación:

- Excepciones de conexión con base de datos SQL (**SQLException**)

- `public Map<String, Alumno> getAlumnos()`
- `public Map<String, Alumno> getAlumnos(int nivel)`
- `public Map<String, Alumno> getAlumnos(int nivel, char paralelo)`
- `public Alumno getAlumno(String rut)`
- `public RegistroAsistencia get(IDAsistencia id)`
- `public RegistroAsistencia getRegistroAsistencia(IDAsistencia id)`
- `public RegistroAsistencia getRegistroAsistencia(Date fecha)`
- `public RegistroAsistencia getRegistroAsistencia(String rutAlumno)`
- `public ResultSet getQuery(String sql)`
- `public ResultSet updateQuery(String sql)`

Se utilizan los bloques try-catch ya que en cada uno de ellos existe la posibilidad de una excepción con la conexión o las consultas a la base de datos generando un **SQLException** o un **SQLTimeoutException**, además los métodos getter utilizados dentro de estos métodos también podrían generar un **SQLException** o un **SQLFeatureNotSupportedException**.

En cuanto a los siguientes métodos, se utilizan bloques try-catch ya que al imprimir los distintos tipos de menú se pueden generar excepciones de tipo **IOException** en operaciones I/O fallidas o interrumpidas.

- Excepciones de entrada/salida de datos por consola (**IOException**)

- `public void menuAdministracion()`
- `public void menu(String menu)`
- `public void menuOpt(int opt)`

Los siguientes son métodos que manejan ficheros y se pueden generar excepciones de **IOException** en la lectura o escritura de un fichero en operaciones I/O fallidas o interrumpidas.

- Excepciones de entrada/salida de datos por ficheros (**IOException**)

- `public Datafile(String type)`
- `public String getData()`
- `public void insertLine(String line)`
- `public void updateLine(String oldLine, String newLine)`
- `public void deleteLine(String line)`
- `public void clearFile()`
- `public String getPath()`
- `public void repListaCursoMenuItemActionPerformed(java.awt.event.ActionEvent evt)`

Luego se pueden producir una excepción de **IOException** o **FontFormatException** ya que se maneja una fuente personalizada, además al cargar el look and feel del Java Swing Application se pueden generar excepciones de **ClassNotFoundException** si la clase **LookAndFeel** no ha podido ser encontrada, **InstantiationException** si no se ha podido crear una nueva instancia de la clase, **IllegalAccessException** si la clase o el inicializador no son accesibles y **javax.swing.UnsupportedLookAndFeelException** si **Info.isSupportedLookAndFeel()** es false (Una excepción que indica que las clases de gestión de apariencia solicitadas no están presentes en el sistema del usuario).

- Excepciones del método main

- `public static void main(String args[])`

Por último, las siguientes excepciones fueron desarrolladas para la solución en específico, y cumplen el propósito de: Generar un error en caso que la conexión con la base de datos no se pueda realizar, y generar un error si el usuario ingresa un dato no numérico en un campo que así lo exige.

- Excepciones de formato numérico (**InvalidNumberException**)

- `public void telefonoApTextFieldFocusLost(java.awt.event.FocusEvent evt)`

- Excepciones de conexión con BD (**DatabaseException**)

- `public Connection connect()`

F.3. Crear 2 clases que extiendan de una Excepción y que se utilicen en el programa



Figura 17: Clases que extienden a Exception

Según las funcionalidades incorporadas en el desarrollo de la solución, se hizo pertinente generar 2 clases que extienden a **Exception**:

- **public class InvalidNumberException extends NumberFormatException**: Esta excepción permite capturar errores al intentar transformar una cadena de texto (ingresada por el usuario), a un tipo de dato numérico (utilizando `Integer.parseInt()`)
- **public class DatabaseException extends SQLException**: Esta excepción permite capturar errores al intentar conectarse a la base de datos MySQL

Ambas clases contienen el método `mostrarMensajeError()`, que mostrará al usuario un mensaje de error, indicandolo.

F.4. Aplicación del patrón de diseño Strategy (Estrategia)

En el desarrollo de la aplicación, se utilizó el patrón de diseño **Strategy** para gestionar el origen de datos del software. El uso de las interfaces presentes en el paquete **Data** contiene los métodos necesarios para obtener los datos a partir de cualquiera sea el origen de estos, lo que es aprovechado por las clases **Datafile** y **Database**, las que obtienen los datos desde archivo CSV y base de datos MySQL, respectivamente.

Opcional: Reemplazar los datos iniciales en el código por una conexión con base de datos local (MySQL), archivo de texto CSV o Excel, para la persistencia de datos

Para la preservación de los datos poseemos tanto una carpeta en el proyecto (**Datafile**) con archivos csv, que nos permite el manejo de datos en formato local. Y también tenemos una conexión a una base de datos (**MySQL**)



Figura 18: Gestión de archivos CSV

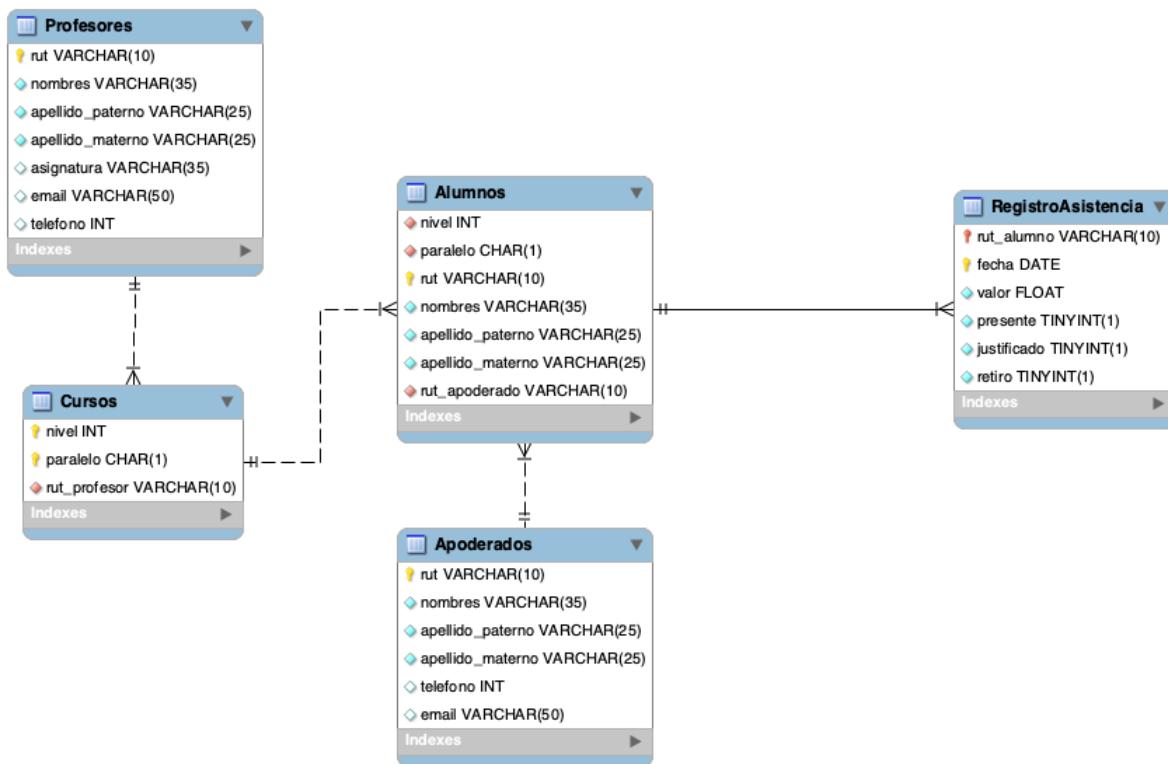


Figura 19: Modelo de datos BD

Opcional: Considerar la implementación del patrón Modelo-Vista-Controlador (MVC) en la arquitectura del sistema

Nuestro programa como se puede apreciar en la siguiente imagen esta estructurado en un formato de modularización, en el cual separamos las clases y sus paquetes correspondientes en un patrón MVC (Modelo-Vista-Controlador).

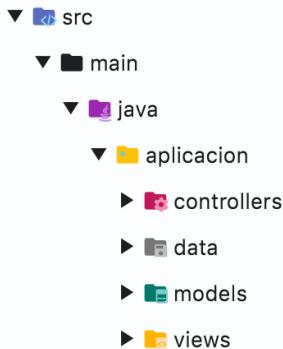


Figura 20: Gestión de archivos para modelo MVC

8. Consideraciones generales

8.1. Desarrollo de la aplicación

Para el desarrollo de esta aplicación, se utilizaron los IDE de desarrollo:

- **IntelliJ Idea:** IDE de desarrollo general de la aplicación.
- **Apache Netbeans:** IDE de desarrollo para diseño de ventanas gráficas (Swing).

La versión del JDK de java utilizado es la 1.8 (Java 8), y también se utilizó Docker para la generación de una instancia para correr la base de datos MySQL, junto con PhpMyAdmin para la administración de la misma.

8.1.1. Codificación y documentación

En la codificación de la solución se siguieron las buenas prácticas de programación, modularización, principios de OO y encapsulamiento. Además, a medida que se generaban nuevas líneas de código, se fue documentando de forma inmediata y generados los **Javadocs** correspondientes para cada una de las entregas. Aprovechando el hosting gratuito otorgado por Github Pages, los Javadocs generados son accesibles a través [de este link](#).

8.1.2. Paquetes de Maven

En el desarrollo del proyecto se utilizó **Maven**, lo que permite incorporar librerías de terceros de forma rápida. Las librerías utilizadas para el desarrollo de esta aplicación son las siguientes:

- **Faker:** Librería que permite la generación de datos de prueba falsos.
- **MySQLConnector/J:** Librería oficial de MySQL que permite la conexión con la base de datos.
- **dotenv-java:** Librería para utilizar variables de entorno, para colocar las credenciales de conexión para la base de datos.
- **jIconFont:** Librería que permite obtener íconos predefinidos, para utilizar en el diseño de la interfaz gráfica.
- **FlatLaf:** Librería que permite cambiar los estilos visuales (**LookAndFeel**) predefinidos para Java Swing.

8.2. Utilización de GitHub

Se utilizó Github como sistema de control de versiones, lo que facilitó el trabajo en equipo a la hora de codificar la solución. Considerando la entrega final del proyecto, se realizaron 65 commits, repartidos entre las diferentes entregas del proyecto, y subidos por los 3 integrantes del grupo del proyecto.

8.3. Herramientas de utilidad

Para facilitar el trabajo en equipo, para este proyecto se utilizaron las siguientes herramientas tecnológicas:

-  **GitHub**: Herramienta para control de versiones. El uso de ramas, en conjunto con el IDE IntelliJ Idea, permitió trabajar en conjunto con el código.
-  **Trello**: Herramienta para gestión de tiempo y plazos de entrega.
-  **Discord**: Herramienta de comunicación para reuniones y juntas de avance.
-  **Suite de Google**: Herramientas online para edición de documentos de forma colaborativa.