Reporte Programación avanzada

Integrantes: Maximiliano Bustamante
Sebastían Cruz
Joaquín fuenzalida

Tema 5: Gestión de Asistencia de alumnos de un colegio

El código es una aplicación de consola Java destinada a gestionar cursos escolares. Permite al usuario seleccionar un curso y realizar varias acciones sobre este, como marcar asistencia, ver alumnos, ver el historial de asistencia, agregar alumnos y eliminar alumnos.

- **SIA1.1 Realizar un análisis de los datos a utilizar y principales funcionalidades a implementar que dan sentido a la realización del proyecto.**

Datos Utilizados:

Calendario General (CSV):

Contiene las fechas en las que se llevarán a cabo las actividades escolares. Estas fechas se cargan al inicio del programa y se utilizan para mostrar el calendario y avanzar al siguiente día.

Archivos CSV por Curso:

Cada curso tiene su propio archivo CSV que almacena la información de los alumnos inscritos en ese curso. Estos archivos se utilizan para cargar y guardar información sobre los alumnos, como nombres, apellidos y detalles de asistencia.

Principales Funcionalidades Implementadas:

Selección de Curso:

El usuario puede seleccionar uno de los cursos disponibles (Primero Básico, Segundo Básico, etc.) desde el menú principal. Esta funcionalidad permite dirigir las acciones específicas al curso elegido.

Gestión de Alumnos:

Ver Alumnos: Permite al usuario ver la lista de alumnos inscritos en el curso seleccionado.

Agregar Alumno: Permite al usuario agregar un nuevo alumno al curso seleccionado.

Eliminar Alumno: Permite al usuario eliminar un alumno existente del curso seleccionado a través del rut o el nombre.

Marcar asistencia: Permite al usuario marcar

Navegación por el Calendario: El usuario puede avanzar al siguiente día en el calendario general. Esto se implementa con la opción "Siguiente día" en el menú principal, lo que permite al usuario avanzar a la siguiente fecha en el calendario.

Ver Asistencia Histórica:

Cargar archivos al CSV: Al finalizar los cambios en el curso seleccionado se guardan los cambios en los archivos csv.

- **SIA1.2 Diseño conceptual de clases del Dominio y su código en Java**

  - **Clase Alumno**
    Alumno.java
    Descripción: Representa a un alumno con atributos básicos como RUT, nombre y apellido.

  - **Clase Curso**
    Curso.java
    Descripción: Gestiona un curso, conteniendo información como el nombre del curso, una lista de alumnos (List<Alumno>), y un registro de asistencias por fecha.

  - **Clase GestorCSV**
    GestorCSV.java
    Descripción: Proporciona funcionalidades para interactuar con archivos CSV, como cargar fechas, cargar alumnos, añadir y eliminar alumnos.

  - **Clase RegistroAsistencia**
    RegistroAsistencia.java
    Descripción: Maneja el registro de asistencia de los alumnos, posiblemente conteniendo la fecha, una lista o conteo de alumnos presentes, etc.

- **SIA1.3 Todos los atributos de todas las clases deben ser privados y poseer sus respectivos métodos de lectura y escritura (getter y setter).**

    - La implementación de estos principios en las clases del dominio asegura que el proyecto cumple con los estándares de la programación orientada a objetos, facilitando el mantenimiento, la escalabilidad y la seguridad del código, donde se asegura cumplir con los atributos privados y poseer setters y getters.

    - **Clase Alumno:**

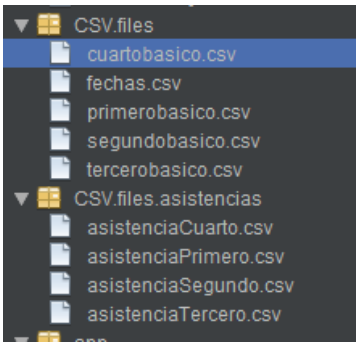
      Atributos privados: rut, nombre, apellido.
      Implementación: Cada atributo tiene su correspondiente método getter y setter, permitiendo un acceso controlado a estos. Por ejemplo, getRut(), setRut(String rut), getNombre(), setNombre(String nombre), etc.

    - **Clase Curso:**

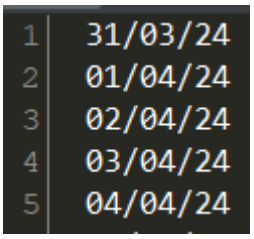
      Atributos privados: nombre, alumnos, totalAlumnos, asistenciasPorFecha.
      Implementación: La clase Curso incluye métodos getters y setters para cada uno de sus atributos, tales como getNombre(), setNombre(String nombre), getAlumnos(), setAlumnos(List<Alumno> alumnos), entre otros. Esto garantiza la encapsulación y proporciona una interfaz clara para manipular la información del curso.

- **SIA1.4 Se deben incluir datos iniciales dentro del código.**

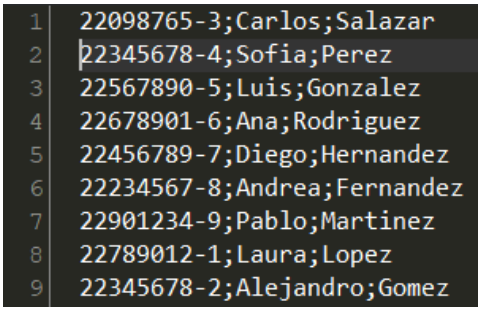
Se crearon 3 tipos de csv donde incluimos datos iniciales los cuales son conformados de la siguiente manera:



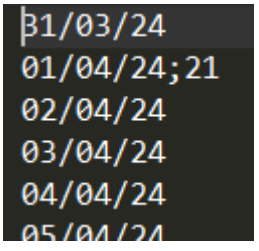
CSV fecha: Este csv se utiliza principalmente como calendario general que tiene la función de hacernos saber en que dia estamos de forma simulada y la cual su formado seria dd/mm/aa.



CSV cursos: Existe 4 csv de este tipo los cuales son de primero, segundo, tercero y cuarto básico los cuales contiene los datos de cada alumno y su formato es rut/nombre/apellido.



CSV asistencia: Existe 4 csv de este tipo los cuales son de asistencia primero , asistencia segundo, asistencia tercero y asistencia cuarto básico, el cual tendrá la fecha al cuales estará sincronizada con el **csv fechas** y cumplirá el rol de guardar la cantidad de alumnos que fueron el presente día. Y su formato es dd/mm/aa/asistencia.



- **SIA1.5 Diseño conceptual y codificación de 2 colecciones de objetos, con la 2ª colección anidada como muestra la figura. Las colecciones pueden ser implementadas mediante arreglos o clases del Java Collections Framework (JCF).**

Primera Colección: Lista de Alumnos:

En el método **cargarAlumnosDesdeCSV(String rutaArchivo)**, utilizas una **List<Alumno>** para almacenar los datos de los alumnos cargados desde un archivo CSV. Esta lista representa la primera colección en tu diseño.

```
public List<Alumno> cargarAlumnosDesdeCSV(String rutaArchivo) {
    List<Alumno> alumnos = new ArrayList<>();

    // Código para cargar datos desde el archivo CSV y añadirlos a la lista

    return alumnos;
}
```

Ubicación en el código: Dentro del método **cargarAlumnosDesdeCSV** en **GestorCSV.java.**

Segunda Colección: Registros de Asistencia

La segunda colección, que es anidada, está conceptualizada en la clase **Curso**. La colección anidada estaría en el manejo de los registros de asistencia, donde asocias una **fecha (LocalDate**) con un registro de asistencia para esa **fecha (RegistroAsistencia)**, probablemente a través de un **Map<LocalDate, RegistroAsistencia>.**

```
public class Curso {

    private Map<LocalDate, RegistroAsistencia> asistenciasPorFecha;
```

Ubicación en el código: Esta implementación estaría en la clase **Curso,** específicamente en el atributo **asistenciasPorFecha.**

- **SIA1.6 Diseño conceptual y codificación de 2 clases que utilicen sobrecarga de métodos (no de constructores)**

1. La clase GestorCSV utiliza la sobrecarga de métodos para proporcionar diferentes maneras de cargar y manipular datos de alumnos desde archivos CSV.

- **Sobrecarga Implementada:**

- **Método para cargar todos los alumnos:** public List<Alumno> cargarAlumnosDesdeCSV(String rutaArchivo)
- **Método sobrecargado para cargar un alumno específico por RUT:** public Alumno cargarAlumnoPorRUT(String rutaArchivo, String rutBuscado)
- **Método sobrecargado para cargar alumnos por nombre:** public List<Alumno> cargarAlumnosPorNombre(String rutaArchivo, String nombreBuscado)

- Estos métodos permiten cargar información de alumnos de manera general o más específica según el RUT o nombre, demostrando la flexibilidad que ofrece la sobrecarga de métodos.

2.La clase GestorCSV

Sobrecarga Implementada:

Cargar todos los alumnos: public List<Alumno> cargarAlumnosDesdeCSV(String rutaArchivo)

Carga una lista completa de alumnos desde un archivo CSV.
Cargar un alumno por RUT: public Alumno cargarAlumnoPorRUT(String rutaArchivo, String rutBuscado)

Busca y carga la información de un alumno específico utilizando su RUT.
Cargar un alumno por Nombre y Apellido: public Alumno cargarAlumnoPorNombreApellido(String rutaArchivo, String nombreBuscado, String apellidoBuscado)

Permite buscar y cargar la información de un alumno basándose en su nombre y apellido.

- **SIA1.7 Diseño conceptual y codificación de al menos 1 clase mapa del Java Collections Framework**

Clase curso

Uso de Map: El atributo **“asistenciasPorFecha”** en la clase “**Curso”** refleja directamente el uso de un mapa del JCF.

Clave-Valor: El mapa utiliza LocalDate como clave, lo que permite una organización eficiente y accesible de los registros por fecha. El valor asociado a cada clave es una instancia de **“RegistroAsistencia”**, que puede contener detalles como los alumnos presentes o ausentes ese día.

```
    private int totalAlumnos;
    private Map<LocalDate, RegistroAsistencia> asistenciasPorFecha;
```

- **SIA1.8 Se debe hacer un menú para el Sistema donde ofrezca las funcionalidades de: 1) Inserción Manual / agregar elemento y 2) Mostrar por pantalla listado de elementos. Esto para la 2ª colección de objetos (colección anidada) del SIA1.5**

Al iniciar el código vemos en un principió el día actual con su fecha al lado y nos pide elegir un curso para gestionar o cambiar de día para cambiar datos de ser necesarios.

```
run:
Día: domingo 31/03/24

Elija un curso:
1. Primero Basico
2. Segundo Basico
3. Tercero Basico
4. Cuarto Basico
8. Dia anterior
9. Dia siguiente
0. Salir
Ingrese una opción:
```

- Luego de **elegir un curso** (Primero Básico en este caso) vemos el siguiente menú de gestión.

```
Menu Curso - Primero Básico:
1. Marcar asistencia
2. Ver alumnos
3. Ver asistencia histórico
4. Agregar alumno
5. Eliminar alumno
0. Volver al menú principal
Ingrese una opción:
```

- **Al agregar un alumno** nos pedirá su rut, nombre y apellido:

```
Ingrese una opción: 4
Ingresa el RUT del alumno: 21031596-9
Ingresa el nombre del alumno: Maximiliano
Ingresa el apellido del alumno: Bustamante
Alumno agregado exitosamente.
```

- Al **ver alumnos** se nos mostrará una lista de los alumnos con los ruts, nombres y apellidos del curso:

```
Alumnos en Primero Básico:
0. 22123456-2 - Maria Gonzalez
1. 22098765-3 - Carlos Rodr□guez
2. 22345678-4 - Sof□a Hern□ndez
3. 22567890-5 - Luis Fern□ndez
4. 22678901-6 - Ana Mart□nez
5. 22456789-7 - Diego L□pez
6. 22234567-8 - Andrea G□mez
7. 22901234-9 - Pablo D□az
8. 22789012-1 - Laura Ruiz
9. 22345678-2 - Alejandro S□nchez
10. 22567890-3 - Carmen Ortiz
11. 22123456-4 - Javier Rojas
```

- **SIA1.9 Todas las funcionalidades deben ser implementadas mediante consola (Sin ventanas)**

Luego de **elegir un curso** (Primero Básico en este caso) vemos el siguiente **menú de gestión.**

```
Menu Curso - Primero Básico:
1. Marcar asistencia
2. Ver alumnos
3. Ver asistencia histórico
4. Agregar alumno
5. Eliminar alumno
0. Volver al menú principal
Ingrese una opción:
```

- **Al agregar un alumno nos pedirá su rut, nombre y apellido:**

```
Ingrese una opción: 4
Ingresa el RUT del alumno: 21031596-9
Ingresa el nombre del alumno: Maximiliano
Ingresa el apellido del alumno: Bustamante
Alumno agregado exitosamente.
```

- **Al eliminar un alumno nos preguntará si queremos buscar el alumno a través de su nombre y apellido o rut:**

```
Ingrese una opción: 5
¿Deseas eliminar por RUT o por nombre y apellido? (R/N)
R
Ingresa el RUT del alumno a eliminar:
21031596-9
Alumno con identificador '21031596-9' eliminado exitosamente.
```

- **Al marcar asistencia el código nos pedirá ingresar la cantidad de alumnos presentes en el día:**

```
Ingrese una opción: 1
Ingrese la cantidad de alumnos presentes:
21
```

- **SIA1.10 Utilización de herramientas de gestión:**

Recursos Usados:

- ❖ JDK 17.
- ❖ Netbeans 12.
- ❖ ChatGPT.
- ❖ Trello
- ❖ Git y Github.

1.10.1 GitHub: Utilización de GitHub (Realización de al menos 3 Commit).
https://github.com/sebacruz1/ProyectoPA.git

1.10.2 Trello: Definir y administrar tareas (Completar al menos 5 tareas)
https://trello.com/invite/b/yZG2JbEb/ATTIb1fe695e6e197f317a9dbb5e7edd7475944031AA/gestion-de-asistencia