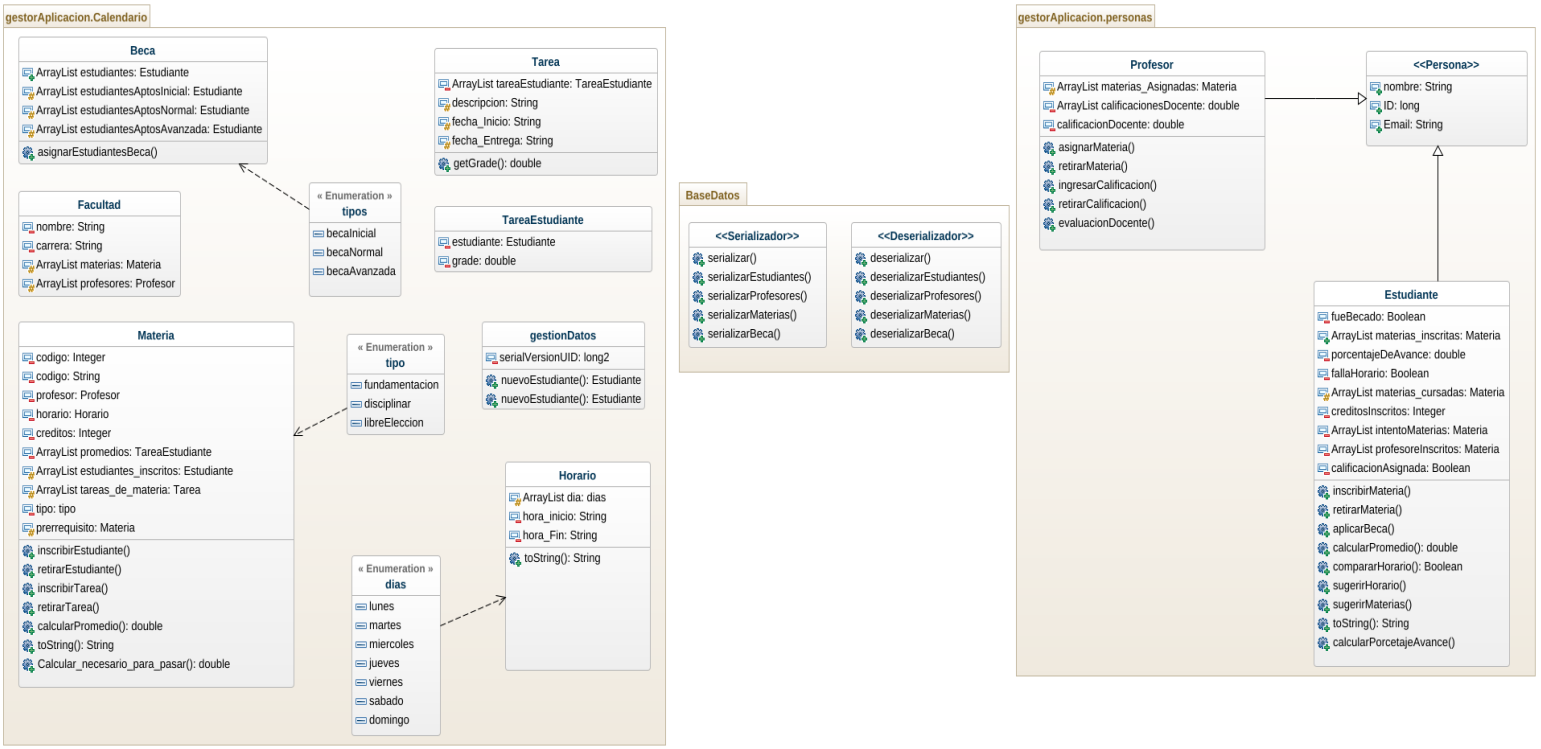
Sistema de gestión universitaria

- **Descripción general**

El sistema de gestión universitaria es un proyecto java, basado en el SIA (sistema de información académica) de la Universidad Nacional de Colombia. Que surge de los inconvenientes que suelen experimentar los estudiantes con el exceso de materias, profesores, trabajos, sistemas e información en general. Este posee un diseño cómodo para que el usuario acceda a la información depositada con facilidad, además ofrece al usuario varias opciones para mejorar su experiencia universitaria, dónde podrá crear su propio horario académico, ver si este tiene fallas; y en caso de que si la aplicación le dará un horario recomendado, mirar su acceso a becas, calificar a sus docentes y realizar cálculos pertinentes para de la nota necesaria para pasar una materia.



- **Implementación POO**

<u>Clase abstracta</u> – > *“Persona.java”*
<u>Herencia</u> – > *“Estudiante.java” y “Profesor.java” heredan de “Persona.java”*

<u>Atributos de clase</u>

En “Beca.java”

public static ArrayList<Estudiante> estudiantes;

<u>Métodos de clase</u>

En *“Estudiante.java”*

public static void aplicarBeca (Estudiante estudiante) {

```
            Beca.estudiantes.add(estudiante);
        }
```

Sobrecarga de constructores

En "Materia.java"

```
public Materia(int codigo, String nombre, Profesor profesor, Horario horario, int creditos,Materia prerrequisito, tipo tipo) {
    this.codigo = codigo;
    this.nombre = nombre;
    this.profesor = profesor;
    this.horario = horario;
    this.creditos = creditos;
    this.aprobado = false;
    tareas_de_materia = new ArrayList<Tarea>();
    estudiantes_inscritos = new ArrayList<Estudiante>();
    this.prerrequisito=prerrequisito;
    this.tipo = tipo;
        }

public Materia(int codigo, String nombre, Profesor profesor, Horario horario, int creditos, tipo tipo) {
    this.codigo = codigo;
    this.nombre = nombre;
    this.profesor = profesor;
    this.horario = horario;
    this.creditos = creditos;
    this.aprobado = false;
    tareas_de_materia = new ArrayList<Tarea>();
    estudiantes_inscritos = new ArrayList<Estudiante>();
    this.tipo = tipo;
        }
```

En *"Tarea.java"*

```
public Tarea(String descripcion, Materia materia, String fecha_Entrega, String fecha_Inicio) {
            this.descripcion =descripcion;
            this.fecha_Entrega = fecha_Entrega;
            this.fecha_Inicio = fecha_Inicio;
            tareaEstudiantes = new ArrayList<>();

        }

public Tarea(Materia materia, String fecha_Entrega) {
            fecha_Inicio = "fecha";
            this.fecha_Entrega = fecha_Entrega;
            tareaEstudiantes = new ArrayList<>();
        }
```

This

En *"Tarea.java"*

```
 public void setGrade(Estudiante estudiante, double grade) {
        tareaEstudiantes.add(new TareaEstudiante(this, estudiante, grade));
   }

public void setDescripcion(String descripcion) {
        this.descripcion=descripcion;
}
public void setFecha_Entrega(String fecha_Entrega) {
        this.fecha_Entrega=fecha_Entrega;
}
```

En "Estudiante.java"

```
public double calcularPromedio() {
            double finalScore = 0.0;
            int numMaterias = materias_inscritas.size();
            for (Materia materia : materias_inscritas) {
                double materiaScore = 0.0;
                int numTareas = materia.getTareasDeMateria().size();
                for (Tarea tarea : materia.getTareasDeMateria()) {
                    materiaScore += tarea.getGrade(this);
                }
                if (numTareas > 0) {
                    materiaScore /= numTareas;
                }
                finalScore += materiaScore;
            }
            if (numMaterias > 0) {
                finalScore /= numMaterias;
            }
            return Math.round(finalScore * 100.0) / 100.0;
        }
```

<u>Enum</u>:

Enum Tipos**:** ***"Beca.java"***
Enum *Tipo:* ***"Materia.java"***
Enum *dias:* ***"Horario.java"***

Esta aplicación trabajará inicialmente con el paquete ***gestorAplicacion.personas*** que contendrá las clases de toda la aplicación (***"Estudiante.java", "Persona.java", "Profesor.java")*** y con el paquete **gestorAplicacion.Calendario** que contiene las clases ***("Tarea.java", "Beca.java", "TareaEstudiante.java", "Facultad.java" y " Materia.java"***)

El paquete ***gestorAplicacion.personas*** contendrá las clases ***"Estudiante.java"***y ***"Profesor.java"*** que implementan la herencia de la clase **"*Persona.java"***

<u>Encapsulamiento</u>

La clase ***"Persona.java"*** posee los atributos:

- nombre : String
- ID : long

- Email : String

Todos estos datos serán registrados para usarse en las clases ***"Estudiante.java" y "Persona.java"***

La clase ***"Estudiante.java"*** posee los atributos:

- ***Private*** fueBecado:Boolean
  Este atributo será usado en la clase ***Beca*** en el método asignarEstudiantesBeca, para comprobar que el objeto Estudiante puede aplicar o no a una de los 3 tipos de beca.

- **ArrayList<Materia>** materias_inscritas: Materia
  Este atributo será usado en la primera funcionalidad detectar problemas con franja horaria, puesto a que cada materia inscrita por el estudiante tendrá su propio horario y dos materias no pueden chocar en la misma franja horaria. también se usará en el método de estudiante retirarMateria().

- ***Private*** porcentajeDeAvance: double
  Se utilizará en el método de estudiante calcularPorcetajeAvance().

- ***Private*** fallaHorario:Boolean
  Este atributo también será usado en la primera funcionalidad detectar problemas con franja horaria, el cual tomará un valor inicial de false, y si la funcionalidad detecta un problema con el horario, el atributo tomará un valor de true, y dará a lugar a la segunda funcionalidad, Sugerir Horario y materia.

- ***protected ArrayList<Materia>*** materias_cursadas: Boolean
  Esta lista será usada en la segunda funcionalidad Sugerir Horario y materia, en el método sugerirMaterias(), en donde si el estudiante ya ha tomado esa materia antes, no será recomendado en el nuevo horario que da a lugar si el horario creado por el estudiante anteriormente presentó fallas. También esta lista se usará en el cálculo del metodo de estudiante calcularPorcetajeAvance, en donde se toma el tamaño de la lista.

- **private** créditos inscritos: int
  Trabaja con la variable intentoCreditos, en donde se revisará en el método de estudiante inscribirMateria(), si los créditos inscritos son al menos 10.

- **private** Lista intentoMaterias (de carácter y de tipo "Materia")
  Esta lista trabajara dentro del método estudiante inscribir Materia(), y tendrá la función de comprobar que la materia a añadir al horario por el estudiante si cumpla con lo requerido, es decir el prerrequisito y como mínimo una materia de tipo fundamentación.

- ***private ArrayList<Materia>*** profesores inscritos
  Esta lista inscribe a los profesores según la materia y el horario elegido por el estudiante.

La clase ***"Estudiante.java"***posee los siguientes métodos:

- inscribirMateria(String, ArrayList<Materia>)

Este método tendrá como parámetros el nombre de la materia a inscribir y una lista de materias disponibles, tendrá un if, el cual comprobará los prerrequisitos de una materia y luego lo añadirá a intentoMateria, posteriormente comprobará si intentoMateria cumple con un requisito adicional, el cual es, que como mínimo se cumpla tener una materia de tipo fundamentación. Por último se comprueba que el número de créditos sea al menos 10 y se cumpla con una materia de tipo fundamentación.

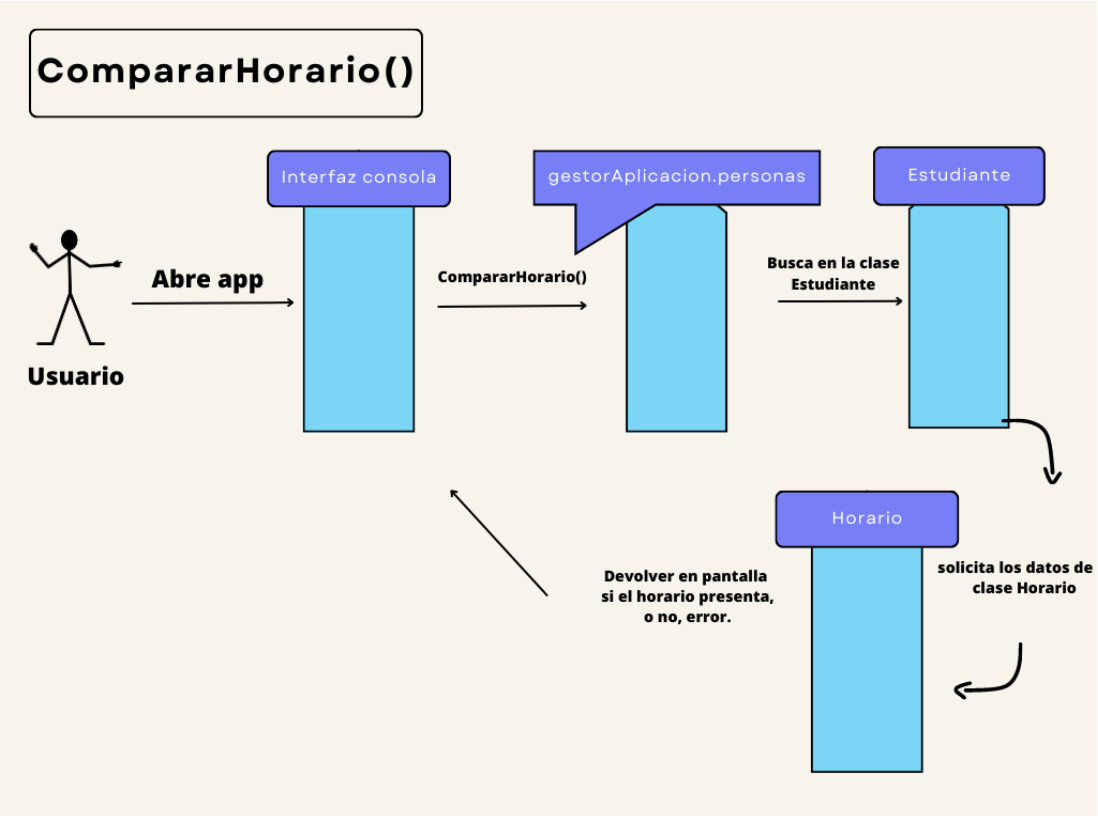
- retirarMateria(Materia)
  Se retira el objeto materia que se añadió a materias_inscritas.

- aplicarBeca(Estudiante)
  Se añaden todos los estudiantes de manera automática a la lista estudiantes de la clase ***Beca***, donde luego serán catalogados.

- calcularPromedio()
  Se toma el tamaño de la lista de materias_inscritas y la lista TareasDeMateria de la clase materia, se obtienen los valores de las notas mediante la variable grade de la clase TareaEstudiante y finalmente se calcula el promedio del estudiante.

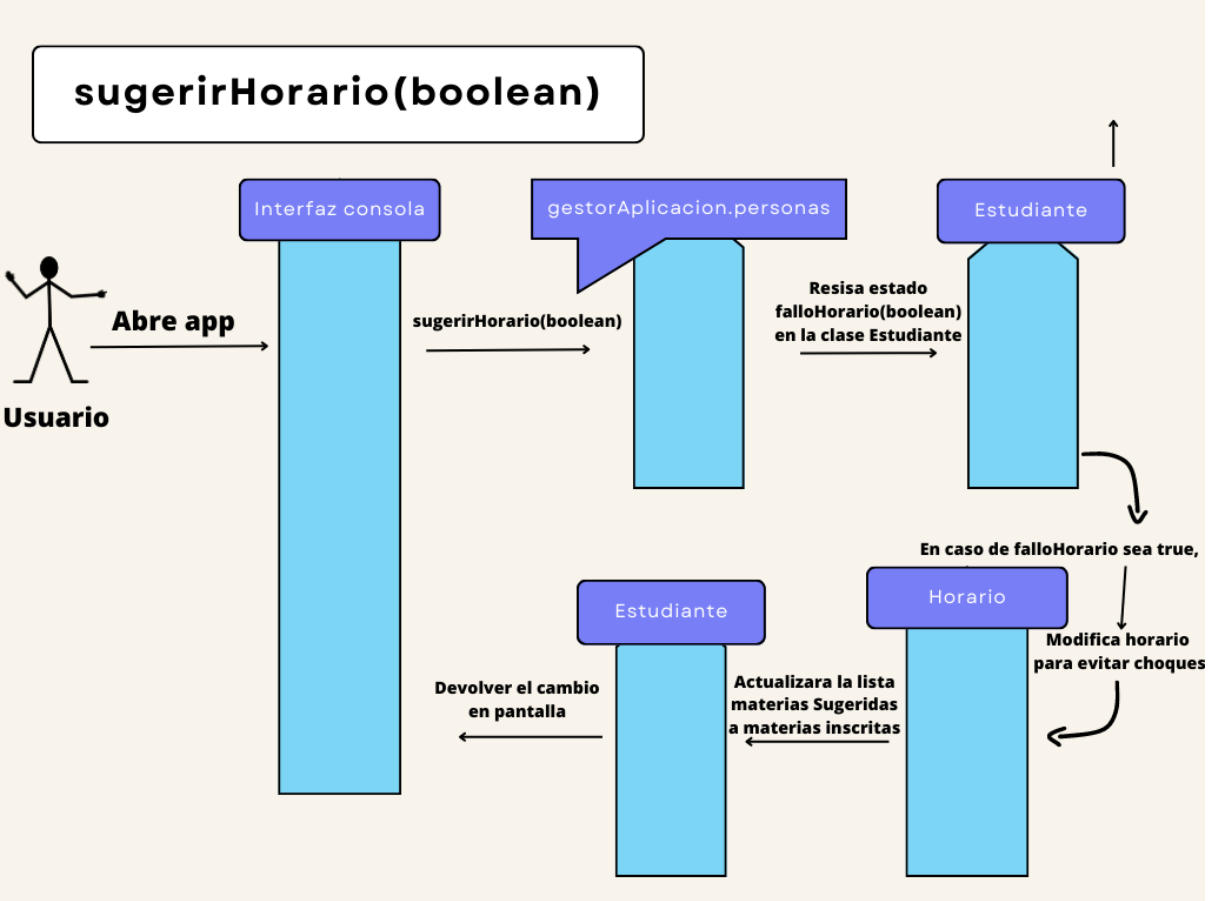
Funcionalidad 1: Detectar problemas con franja horaria

- **compararHorario(ArrayList<Materia>): boolean**
  Esta primera funcionalidad relaciona la clase ***"Estudiante.java"*** con "***Horario.java"*** y la clase ***"Materia.java"***, por medio de la lista donde el usuario guardó las materias inscritas (**materias_inscritas)** se realiza un recorrido mediante ciclos for para encontrar posibles fallas en el horario, por medio del **enum: dias** y las variables **hora_inicio** y **hora_Fin** se busca que las materias no se interpongan entre la hora y el dia. Al final por medio de la variable **fallaHorario**, en caso de que cambie su valor a True, sabremos que se presentaron intersecciones en **materias_inscritas**.

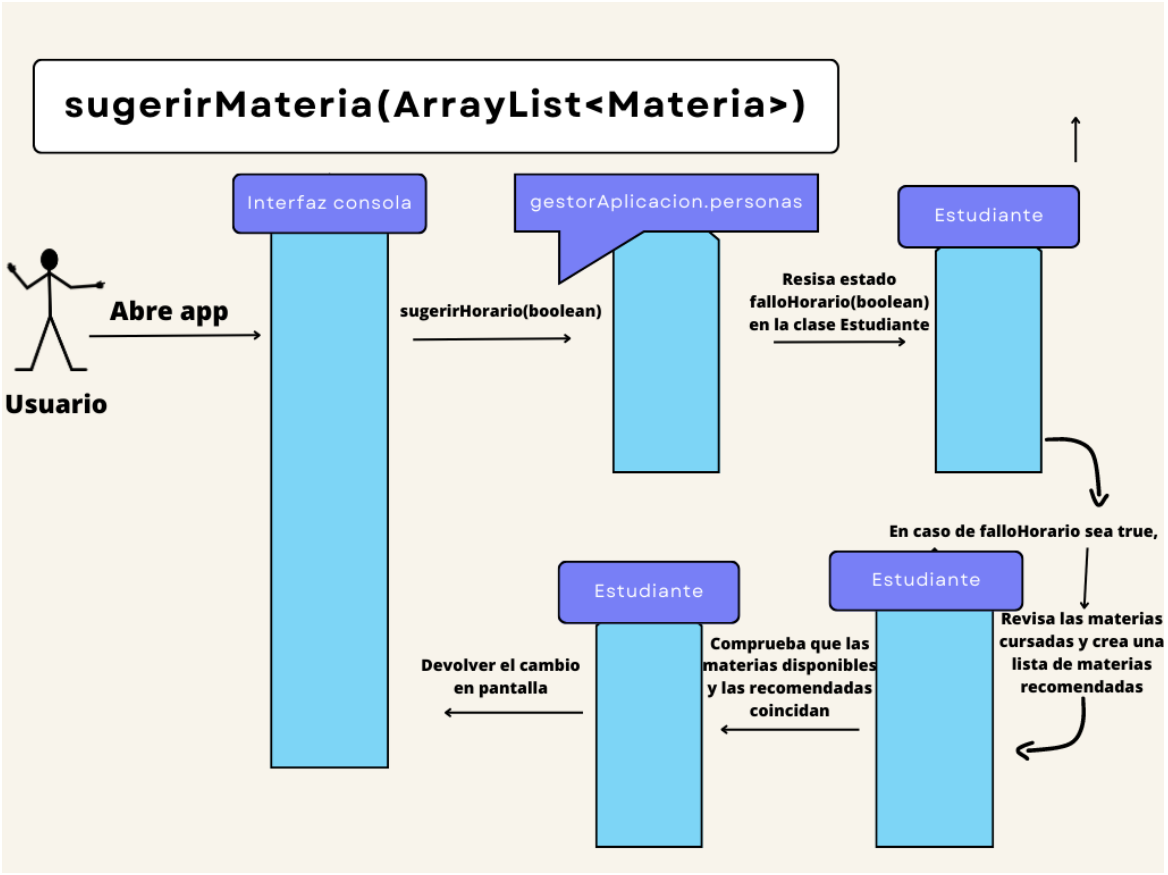


Funcionalidad 2: Sugerir Horario y materia

- **sugerirHorario(boolean)**
  Esta segunda funcionalidad relaciona la clase ***"Estudiante.java"*** con "***Horario.java"*** y la clase ***"Materia.java"***, es posible ejecutarla cuando **fallaHorario** es True, lo que hará es modificar el horario para evitar los choques con las franjas horarias manipulados por **enum: dias, hora_inicio** y **hora_Fin** y si no hay conflictos finalmente la lista materias_insctritas se actualizará con la lista de **materiasSugeridas** que sobreescribirá a **materias_inscritas**.



- **sugerirMaterias(ArrayList<Materia>)**
  Este método dentro de la funcionalidad relaciona "**Estudiante.java"** con la clase ***"Materia.java"***, el método establecerá las sugerencias basándose en las listas **materias_cursadas** y **materiasDisponibles**. Primero se creará una nueva lista llamada **materiasRecomendadas**, posteriormente se comprobará que las materias en **materias_cursadas** es igual a una materia en **materiasDisponibles**, cuando esto se comprueba, la siguiente materia disponible se agregara a la lista **materiasRecomendadas**.
  Si el método anterior **compararHorario** devuelve un valor de True para la lista **materiasRecomendadas**, la materia se eliminará de la lista. finalmente, la lista de **materias_inscritas** se actualizará con la lista de **materiasRecomendadas**.



- toString()
  Devuelve el nombre del estudiante.

- calcularPorcetajeAvance()
  Este método lo que hará es calcular el porcentaje de avance, dividiendo el tamaño de la lista materias_cursadas por 9 y multiplicando el resultado por 1.0, luego redondea el resultado y lo multiplica por 100, finalmente actualiza este valor en la variable de PorcentajeDeAvance.

la clase ***"Profesor.java"***posee los siguientes atributos:

- ***protected ArrayList<Materia>*** materias_Asignadas
  Es la materia que se le asigna a cada profesor, así como su horario.

- ***private ArrayList<double>*** calificaciones Docentes
  Es la lista en donde se almacenan las calificaciones de los docentes realizadas por los estudiantes.

- ***private*** calificacionDocente: double
  Se almacena el valor arrojado después de la cuarta funcionalidad Evaluacion docente.

la clase ***"Profesor.java"*** posee los siguientes métodos:

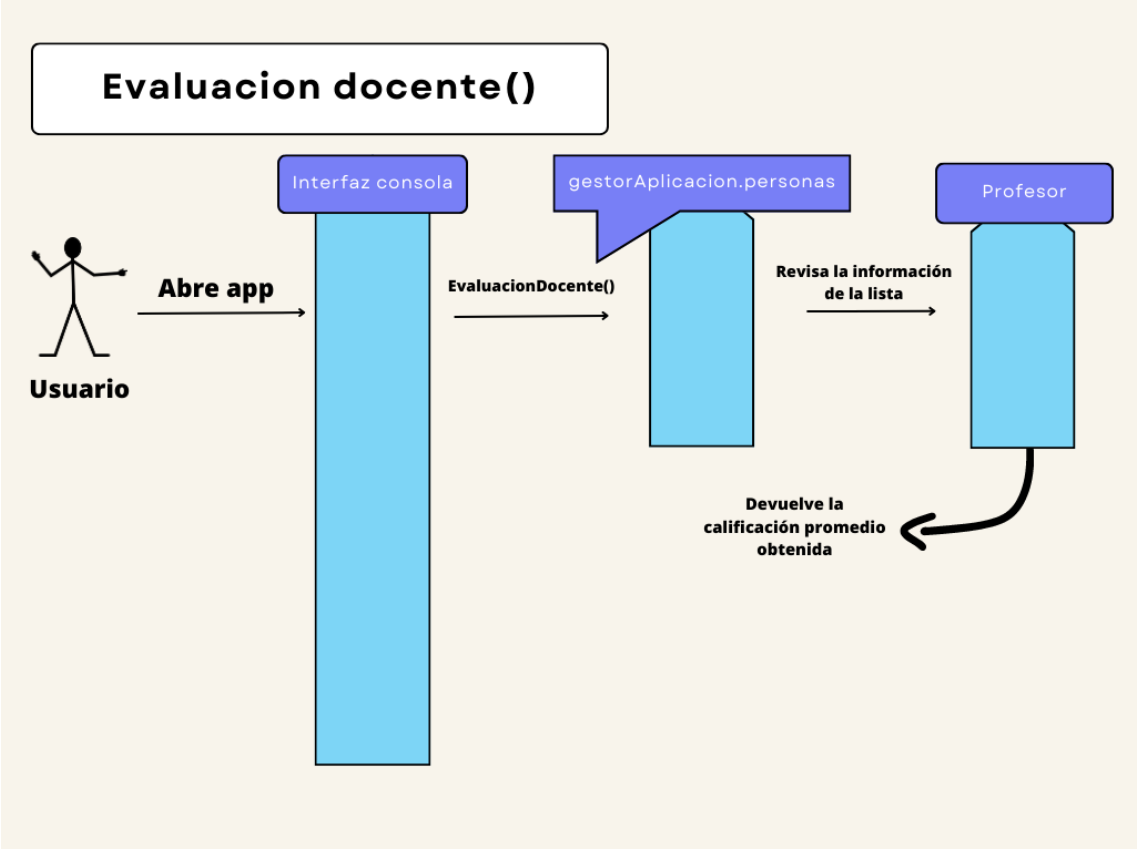
- asignarMateria()
  Se le asigna una materia a un profesor y se almacena en materias_Asignadas.
- retirar Materia()
  Se le retira una materia asignada a profesor.

- ingresarCalificacion()
  Se asigna la calificación del docente realizada en la cuarta funcionalidad Evaluación docente.

- retirarCalificacion()
  Se le retira una calificación ya asignada a profesor, de la cuarta funcionalidad Evaluación docente.

Funcionalidad 4: Evaluación docente

- **evaluacionDocente()**
  En esta funcionalidad interactúan los componentes de la clase ***"Profesor.java"*** haciendo uso de ***"Object.java"***, primero se inicializa la variable llamada **totalCalificaciones** en 0. luego se utiliza el ciclo for para iterar las calificaciones en **calificaciónDocentes** y suma cada una en la variable **totalCalificacion**. después se divide el total de las calificaciones por el tamaño de **calificacionesDocente**, redondeando **math.round** el valor y asignando este **calificacionDocente**.



El segundo paquete, el cual es ***gestorAplicacion.Calendario***. Este paquete es el encargado de guardar y almacenar los datos, así como gestionarlos para la correcta implementación de las funcionalidades.

La clase "***Beca.java"*** posee los siguientes atributos :

- **private** tipos
  Dentro de la aplicación existen 3 tipos de becas almacenadas en un enum, cada una de ellas posee diferentes requisitos para que un estudiante pueda aplicar a una de ellas.

- ***Public ArrayList<Estudiante>*** estudiantes
  Todos los estudiantes por automático ya quedan registrados en el sistema de becas, donde posteriormente se irán evaluando según los requisitos si aplica para alguna o para ninguna beca.

- ***protected ArrayList<Estudiante>*** estudianesAptosInicial
  Para el primer tipo de beca, los estudiantes aptos se guardarán en esta lista, los requisitos para aplicar en esta beca son: que el estudiante tenga un porcentaje de avance mayor o igual a 20 y menor a 40, el promedio sea mayor o igual a 4.5 y que no haya sido becado anteriormente.

- ***protected ArrayList<Estudiante>*** estudiantesAptosNormal
  Para el segundo tipo de beca, los estudiantes aptos se guardarán en esta lista, los requisitos para aplicar a esta beca son: que el estudiante tenga un porcentaje de avance mayor o igual a 40 y menor a 60, el promedio mayor o igual a 4.0 y que no haya sido becado anteriormente.

- ***protected ArrayList<Estudiante>*** estudiantesAptosAvanzada
  Para el tercer tipo de beca, los estudiantes aptos se guardarán en esta lista, los requisitos para aplicar a esta beca son: que el estudiante tenga un porcentaje de avance mayor o igual a 60, un promedio mayor o igual a 3.5, y que no haya sido becado anteriormente.

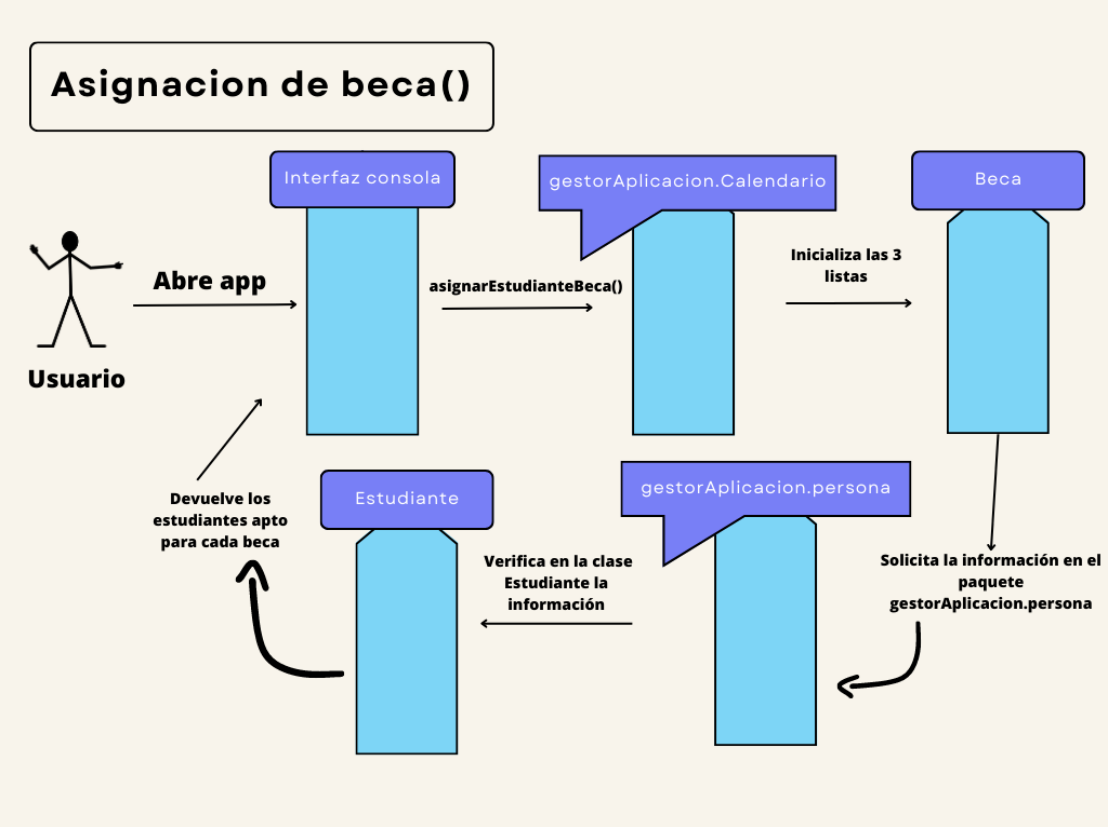
La clase **"*Beca.java"*** posee el siguiente método:
Funcionalidad 3: Asignación de becas:

- **asignarEstudiantesBeca**()
  En esta funcionalidad interactuan las clases ***"Beca.java"*** y la clase ***"Estudiante.java"*** Primero se inicializan las 3 listas de estudiantes aptos **estudiantesAptosInicial2**,**estudiantesAptosNormal2**, **estudiantesAptosAvanzada2** para cada una de las becas, se utiliza el ciclo for para iterar sobre la lista estudiantes, la cual contiene a todos los estudiantes y así poder catalogarlos según corresponda la beca a la que puede aplicar, verificando si cumple con los requisitos mencionados anteriormente para ser elegible, si el estudiante cumple con los requisitos, se agrega a la lista correspondiente de estudiantes aptos para beca.
  Después del ciclo for, el método ordena cada una de las listas de estudiantes aptos para becas en orden descendente según su promedio calculado en el método de ***"Estudiante.java"*** **calcularPromedio()**, después se verifica si cada lista tiene al menos dos estudiantes, y en caso de ser un solo estudiante se agrega a la lista correspondiente.

La clase "***Facultad.java"*** posee los siguientes atributos:

- **private** nombre: String
  Se asigna como: Minas.

- **private** carrera: String
  Se asigna como: IngenieriaSistemas



- ***protected ArrayList<Materia>*** materias
  Esta lista contiene todas las materias que se usarán dentro de la aplicación.

- ***protected ArrayList<Profesor>*** profesores
  Esta lista contiene todos los profesores que se usarán dentro de la aplicación.

La clase "***Facultad.java"*** posee el siguiente constructor:

- Dentro del constructor de la clase, se almacenarán las materias, con su nombre, profesor, horario, créditos y tipo, los profesores, con sus nombres, ID y sus correos, y los horarios, con los días establecidos, su hora de inicio y su hora de finalización.
  las materias que se usarán en la aplicación son: Calculo Diferencial, Calculo Integral y Calculo Varias Variables que son de tipo fundamentación, Fundamentos Programacion, Programacion Orientada Objetos y Estructura Datos que son de tipo disciplinar, y Cátedra Antioquia, Cátedra Apun y Cátedra Felicidad que son de tipo libre elección.

La clase ***"Horario.java"*** posee los siguientes atributos:

- ***protected ArrayList<dia>*** días
  Se almacenan los días que posteriormente tomarán y se le asignaran a las diferentes materias. Este atributo se usa en la clase "***Facultad.java"***en su constructor donde está almacenada la información de las materias, y en el método **compararHorario()** de ***"Estudiante.java"***.

- **private** hora_inicio: String)
  Indica mediante un String la hora de inicio de una materia. Este atributo se usa en la clase "***Facultad.java"***en su constructor donde está almacenada la información de las materias, y en el método **compararHorario()** de ***"Estudiante.java"***.

- **private** hora_Fin: String
  Indica mediante un String la hora de finalización de una materia. Este atributo se usa en la clase "***Facultad.java"***en su constructor donde está almacenada la información de las materias, y en el método **compararHorario()** de ***"Estudiante.java"***.

- **public enum dias** {lunes, martes, miércoles, jueves y viernes, sábado, domingo}

La clase ***"Horario.java"*** posee el siguiente método:

- toString()

La clase ***"Materia.java"*** posee los siguientes atributos:

- **private** código: int
  Almacena el código de la materia el cual se establece en el constructor de la clase "***Facultad.java".***

- **private** nombre: String
  Almacena el nombre de la materia el cual se establece en el constructor de la clase "***Facultad.java".***

- **private** profesor: profesor
  Almacena el profesor de la materia el cual se establece en el constructor de la clase "***Facultad.java".***

- **private** horario: horario
  Almacena el horario de la materia el cual se establece en el constructor de la clase "***Facultad.java".***

- **private** créditos: int
  Almacena los créditos de la materia el cual se establece en el constructor de la clase "***Facultad.java"***y en el método de estudiante inscribirMateria().

- ***private promedios : ArrayList<TareaEstudiante>***
- ***protected estudiantes_inscritos : ArrayList<Estudiante>***
- ***protected tareas_de_materia : ArrayList<Tarea>***
- **public** tipo
  Almacena los 3 tipos de materia los cuales se establecen en el constructor de la clase "***Facultad.java"***y el tipo de fundamentacion se usa en el método de estudiante inscribirMateria() como requerimiento.

- **private** tipo: tipo
- **protected** prerrequisito : Materia
  Almacena los prerrequisitos de ciertas materias, se establecen en el constructor de la clase "***Facultad.java"***. Cálculo Diferencial es prerrequisito de Cálculo Integral, y Cálculo Integral es prerrequisito de Cálculo Varias Variables. Fundamentos Programación es prerrequisito de Programación Orientada Objetos, y Programación Orientada Objetos es prerrequisito de Estructura Datos.

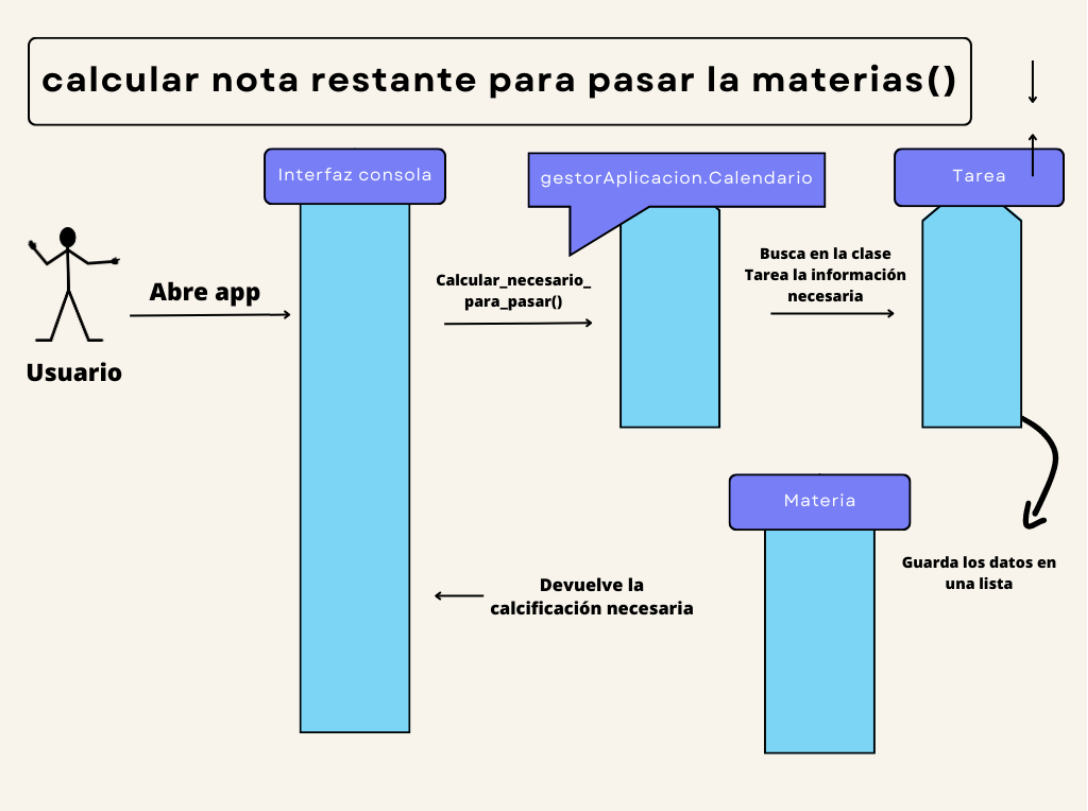
La clase ***"Materia.java"*** posee los siguientes métodos:

- **Public** inscribirEstudiante(Estudiante)
- **Public** retirarEstudiante(Estudiante)
- **Public** inscribirTarea(Tarea)
- **Public** retirarTarea(Tarea)
- **Public** calcularPromedio(Estudiante): double
  El método toma como parámetro un objeto estudiante y empieza a iterar con un ciclo for para empezar a iterar sobre tareas_de_materia. Dentro del ciclo, se utiliza el método getGrade de tarea para obtener la calificación del estudiante en esa tarea y la suma a una variable llamada totalScore. después del ciclo for, el método calcula el promedio dividiendo el total de calificaciones por el número de tareas y redondea el resultado

- toString():String

Funcionalidad 5: calcular nota restante para pasar la materias:

- **Calcular_necesario_para_pasar(Estudiante): double**
  Esta funcionalidad relaciona la clase ***'"Materia.java"*** con la clase ***"Estudiante.java",*** este método toma como parámetro un objeto estudiante y usa un ciclo for para iterar sobre **tareas_de_materia** que es de tipo tareas. Dentro del ciclo el método utiliza **getGrade()** de tarea para obtener la calificación del estudiante en esa tarea y la suma a la variable **totalScore**. También se lleva un contador de cuántas tareas hay. posterior al ciclo for, el método calcula el promedio dividiendo el total de calificaciones por el número de tareas y redondea el resultado.
  Luego se verifica si el promedio es menor a 3, en caso de serlo, calcula la nota necesaria para pasar y este resultado se lo asigna a la variable llamada **Nota_necesaria** y devuelve este valor como resultado. Si el promedio es mayor o igual a 3, el método devuelve 0 como resultado.



La clase **"*Tarea.java"*** posee los siguientes atributos:

- **private *ArrayList<tareaEstudiante>*** tareaEstudiante
- **protected** descripción: String
- **protected** fecha_Entrega: String

La clase **"*Tarea.java"*** posee el siguiente método:

- getGrade()
  Devuelve la calificación de una tarea de un estudiante. el método toma como parámetro un objeto estudiante y utiliza un ciclo for para iterar sobre tareaEstudiante. dentro del ciclo el método verifica si estudiante está asociado con el objeto TareaEstudiante es igual al objeto estudiante del parámetro. si es así, retornara la calificación del estudiante utilizando el método getGrade() del objeto TareaEstudiante. si no se encuentra una coincidencia, el método retorna 0 como resultado.

La clase ***"TareaEstudiante.java"*** posee los siguientes atributos:

- **private** estudiante: Estudiante
- **private** grade: double