**PRACTICA 1**

**SISTEMA GESTION ACADEMICO Z.I.A**

**PROGRAMACION ORIENTADA A OBJETOS**

**GRUPO 1, EQUIPO 8**

**TOMAS VELASQUEZ EUSSE**

**LINA MARCELA SANCHEZ MORA**

**SERGIO MARIO MORALES MARTINES**

**STIVEN SANTIAGO ROSERO QUEMAG**

**JHOAN ALEXIS RUA GARCIA**

**PROFESOR**

**JAIME ALBERTO GUZMAN LUNA**

**UNIVERSIDAD NACIONAL**

**COLOMBIA**

**MEDELLIN**

**2025**

**DESCRIPCION GENERAL**

Este proyecto es un sistema de gestión académico enfocado a mejorar las actividades de una universidad, este sistema está dirigido únicamente a los coordinadores académicos. Este sistema cuenta con unas funcionalidades especificas para administras de manera eficiente materias, grupos, horarios y estudiantes.

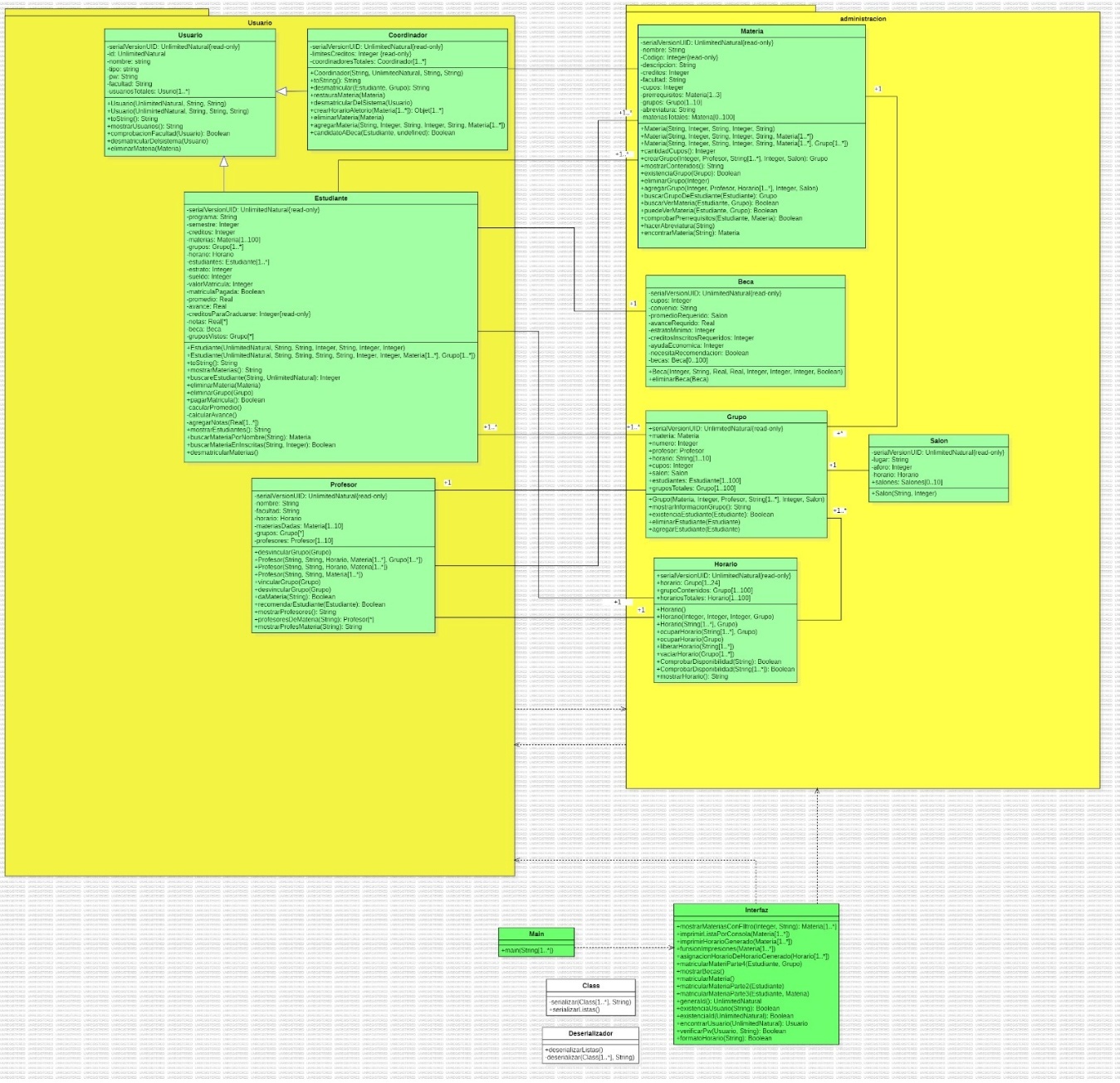
Para llevar acabo este proyecto se realizó una abstracción de los procesos involucrados en la matriculación y administración de diferentes procesos académicos. El perfil del usuario es de coordinador, el cual tiene el poder de agregar o eliminar tanto materias como grupos, desmatricular alumnos y buscar becas para los estudiantes.

**Estructura**

El proyecto cuenta con 3 paquetes básicos distribuidos de la siguiente manera:

* baseDAtos
  + Deserilizador
  + Serializador
* gestorAplicacion
  + administración
    - Beca
    - Grupo
    - Horario
    - Materia
    - Salon
  + Usuario
    - Coordinador
    - Estudiante
    - Profesor
    - Usuario
* uiMain
  + Interfaz
  + Main

**UML**



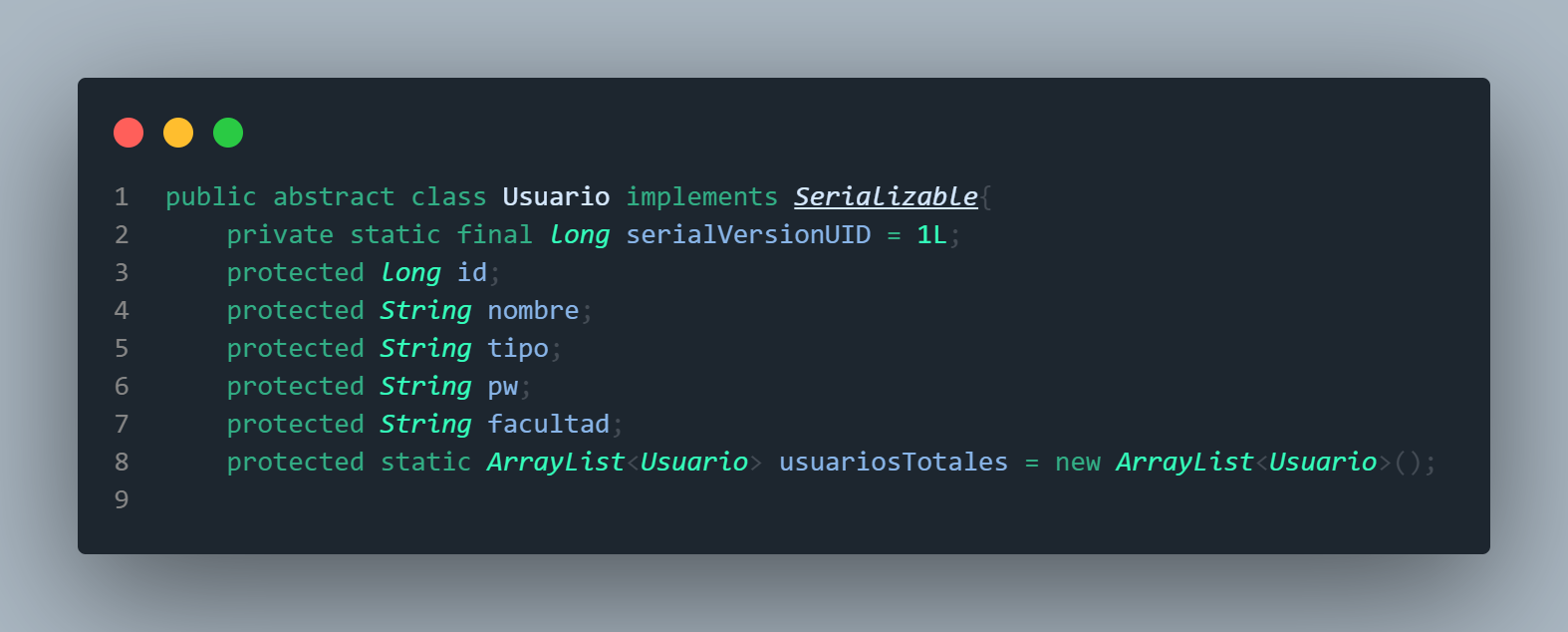
Para visualizar mejor el uml:

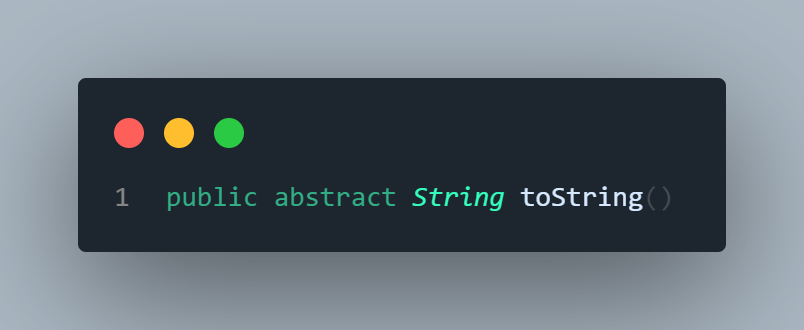
[**https://miro.com/app/board/uXjVLn3JtfU=/?share\_link\_id=279082591092**](https://miro.com/app/board/uXjVLn3JtfU=/?share_link_id=279082591092)

**Implementación de las características de la POO**

**Clase y método abstracto.**

La clase Usuario es de abstracta y tiene el método abstracto toString, el cual se verá implementado en sus clases hijas Coordinador y Estudiante.

****

****

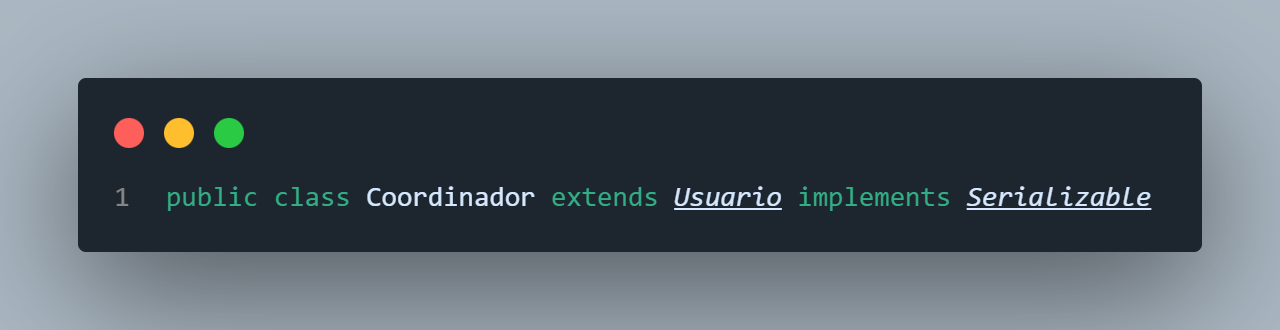
**Interface.**

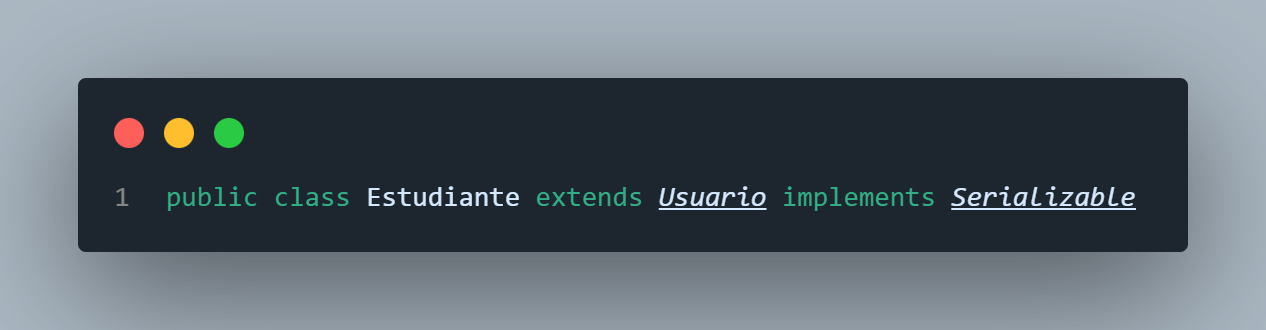
La interface del proyecto contiene todos los métodos que nos servirán al momento de ejecutar nuestro main. (Aquí una pequeña parte del código)

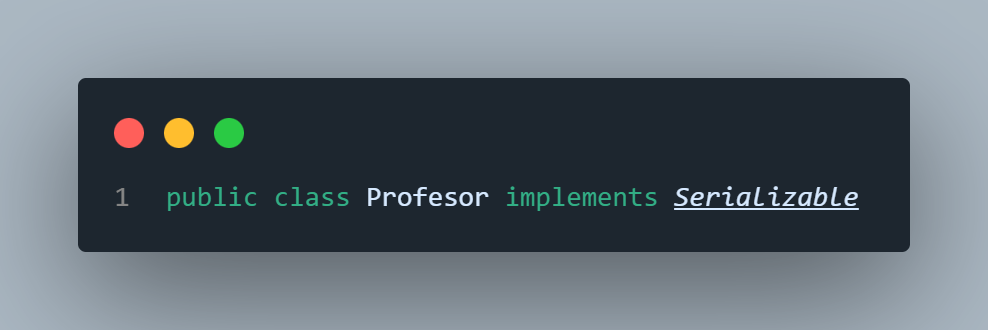


**Herencia**

La herencia es un concepto clave en la POO, en nuestro sistema, Usuario, la clase abstracta, es la clase padre de Coordinador, Estudiante, Profesor



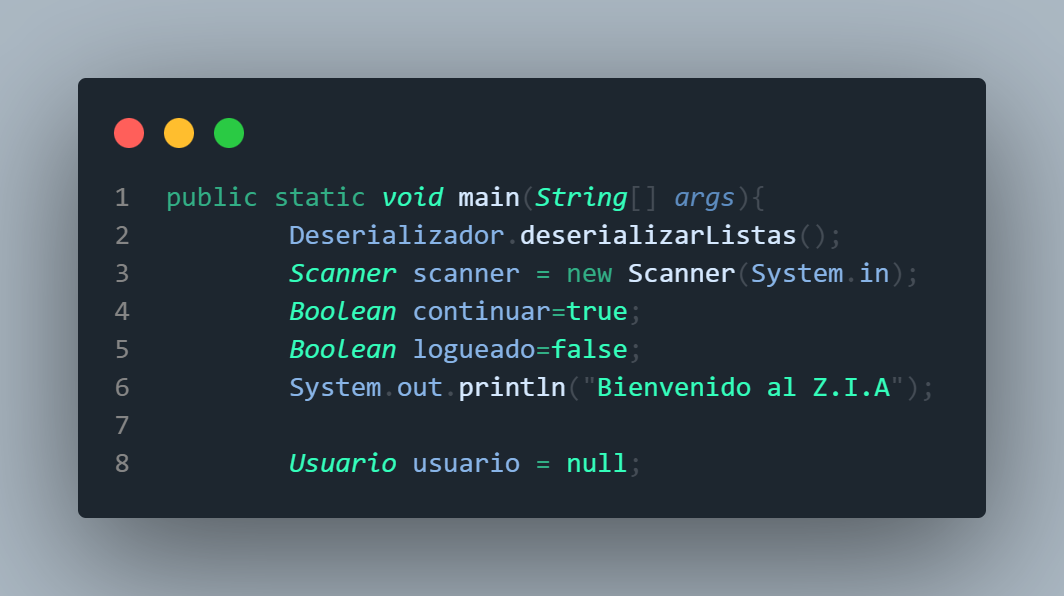


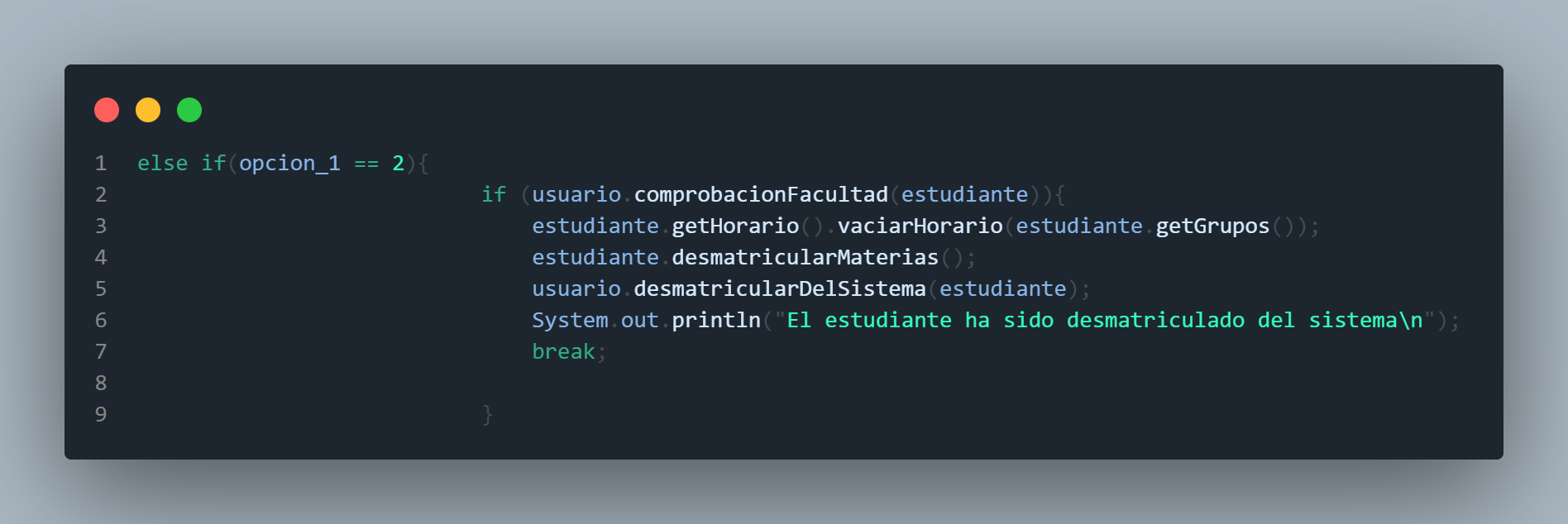


**Ligadura dinámica.**

Durante la ejecución del programa en la funcionalidad Desmatricular Alumno, cuando se invocan los métodos comprobacionFacultad y desmatricularDelSistema en el apuntador “usuario” de tipo Usuario (creado en el inicio de sesión donde apunta a un objeto de tipo Coordinador), la ligadura dinámica se encarga de enlazar esas llamadas a los métodos correspondientes de la clase más específica. Esto significa que se ejecutará la implementación específica de comprobacionFacultad y desmatricularDelSistema para la clase Coordinador.

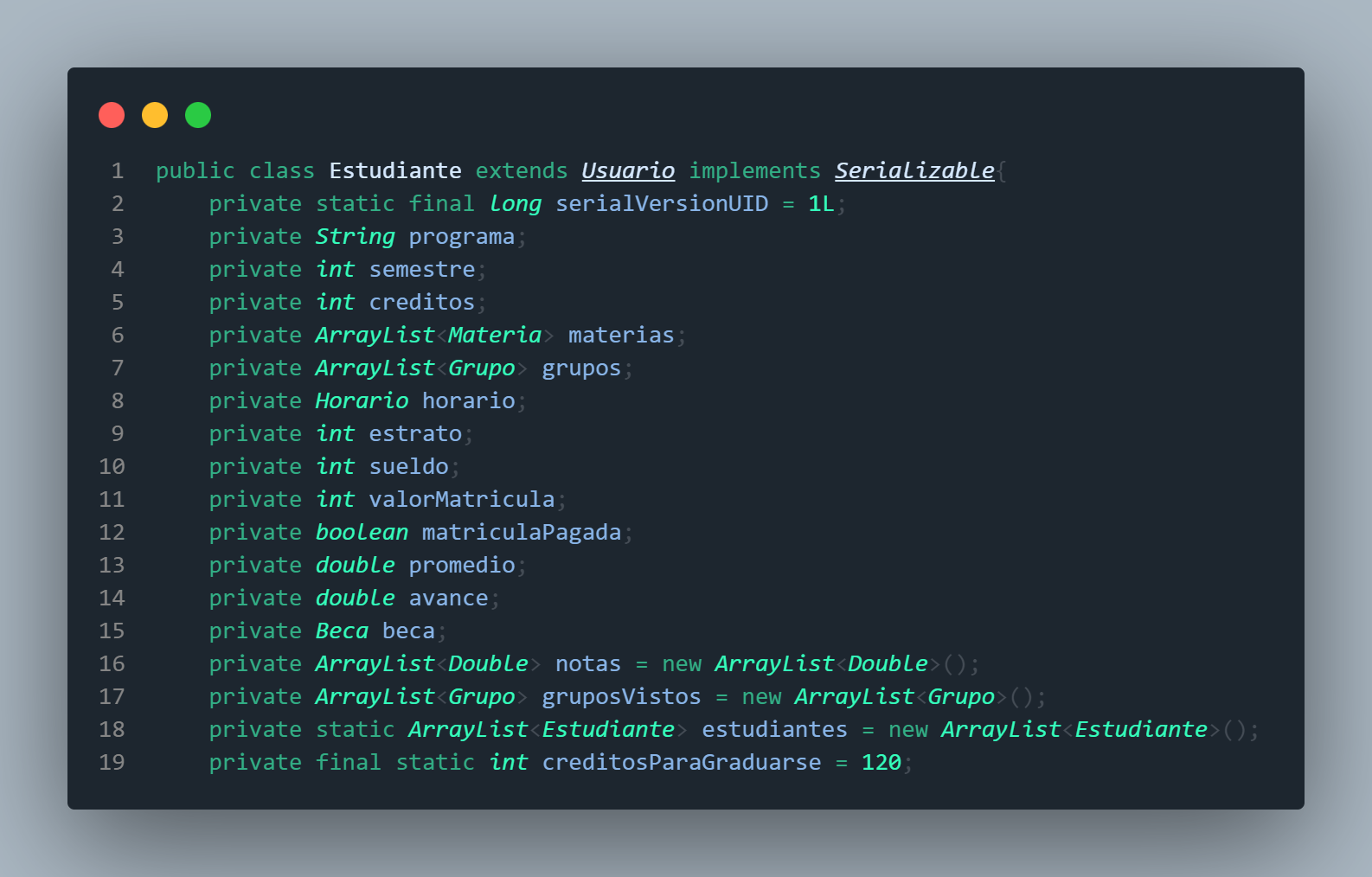




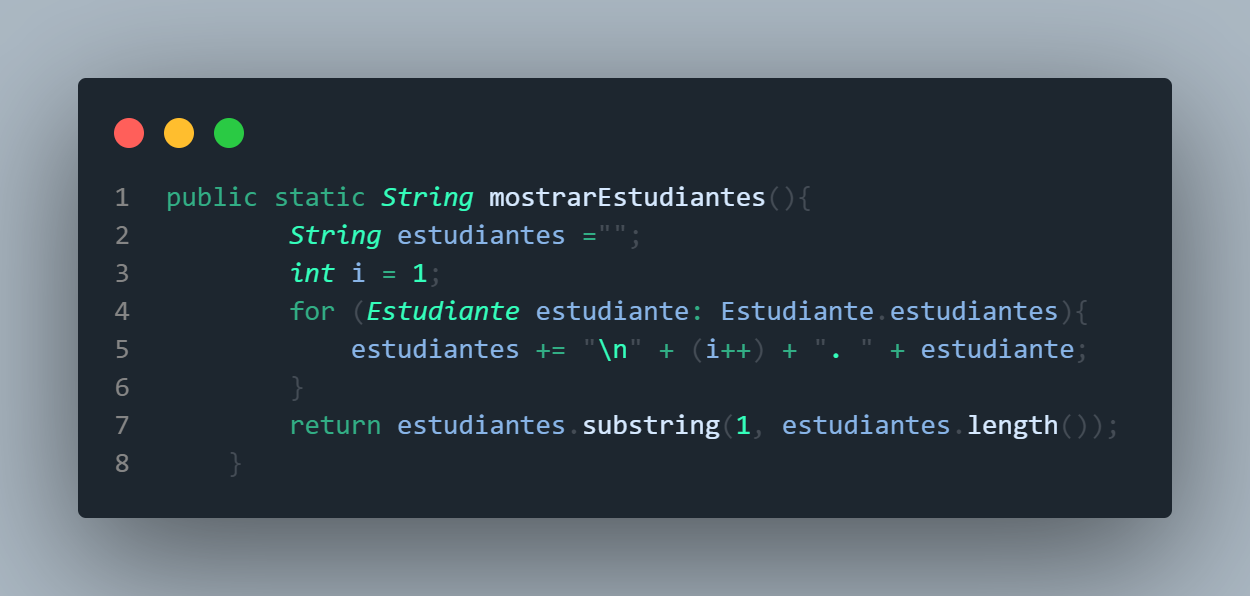


**Atributos y métodos static (de clase)**

Esta característica se uso principalmente como atributos para alamacenar las instancias de la clase, por ejemplo de Estudiante

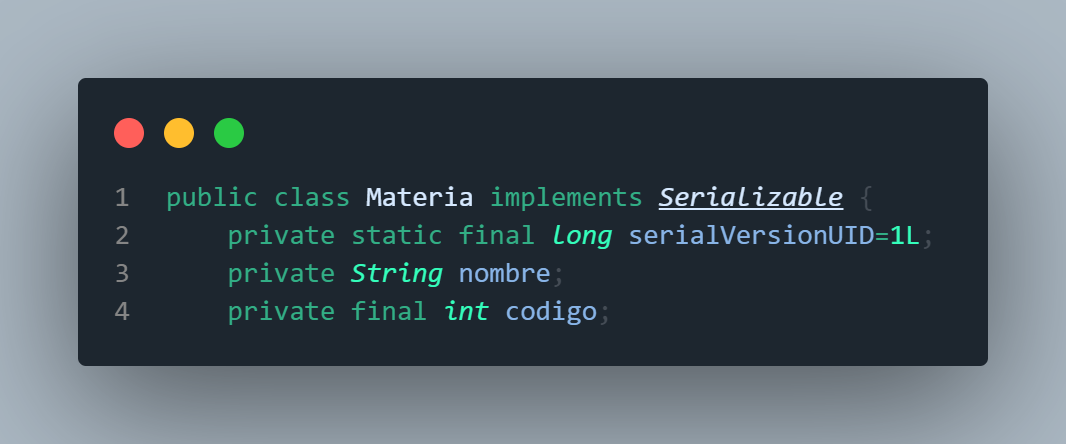


Por parte de los métodos, también en Estudiante tenemos mostrarEstudiantes:



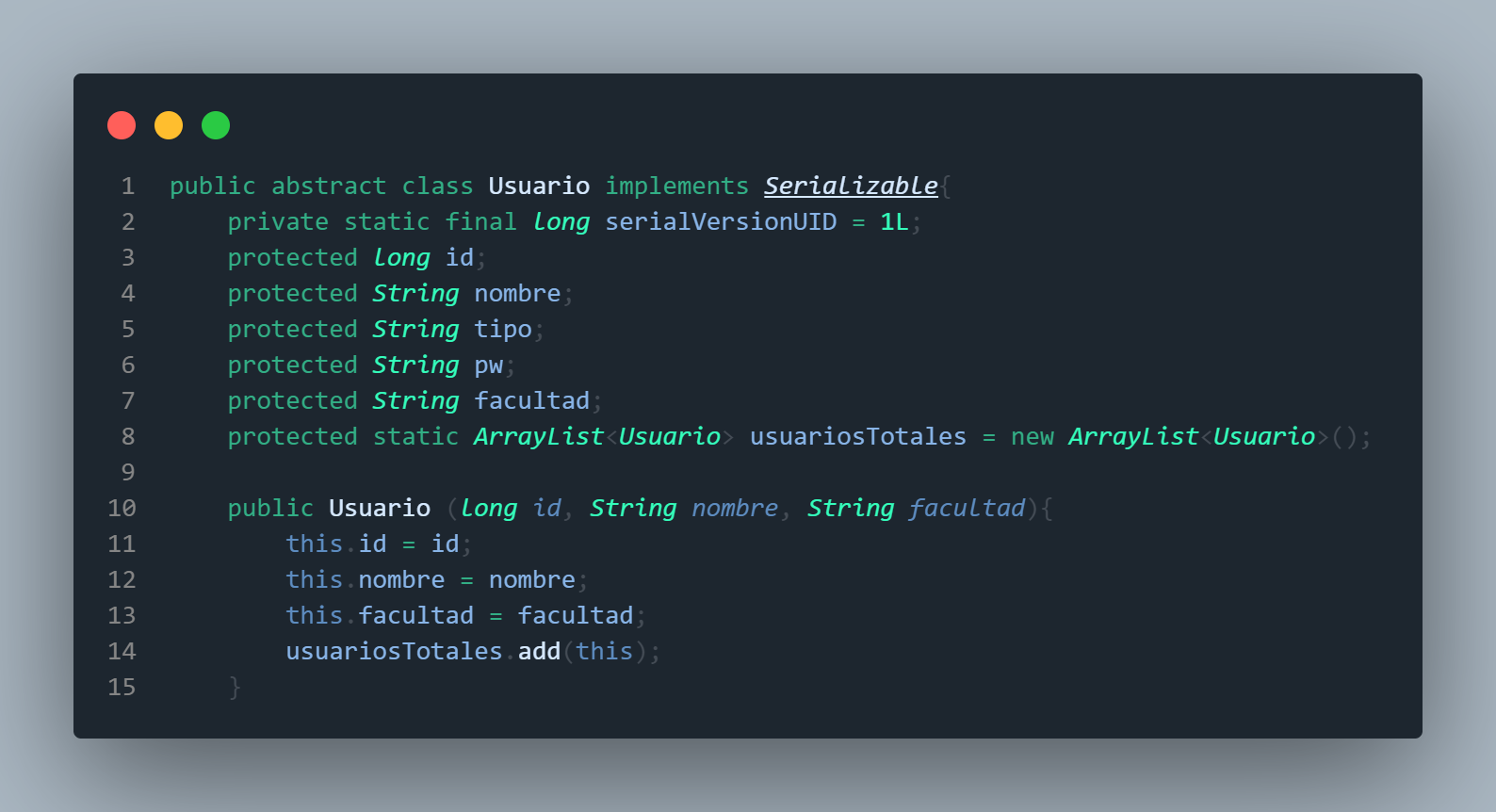
**Constantes**.

La clase Materia tiene un atributo código, el cual es constante y único para cada materia creada.



**Encapsulamiento.**

El encapsulamiento se usa en todo el programa, un ejemplo puede ser Usuario, donde se hace uso de los modificadores private, protected y public:

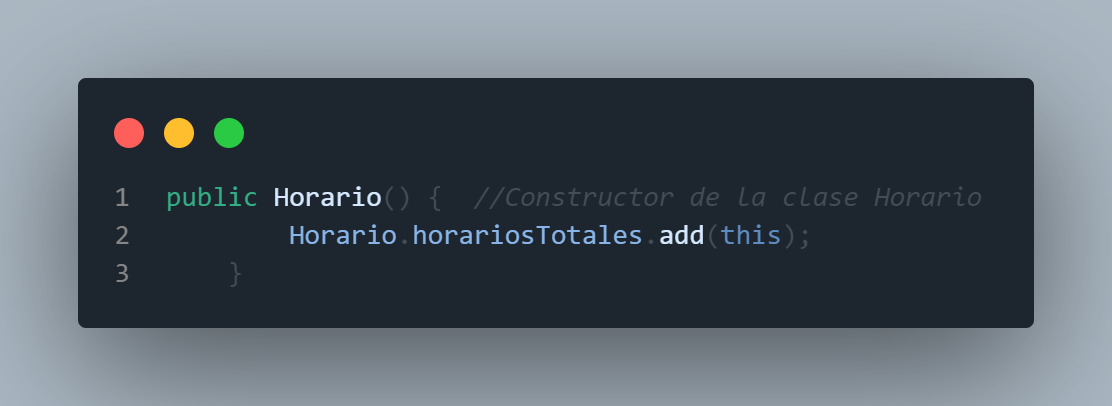


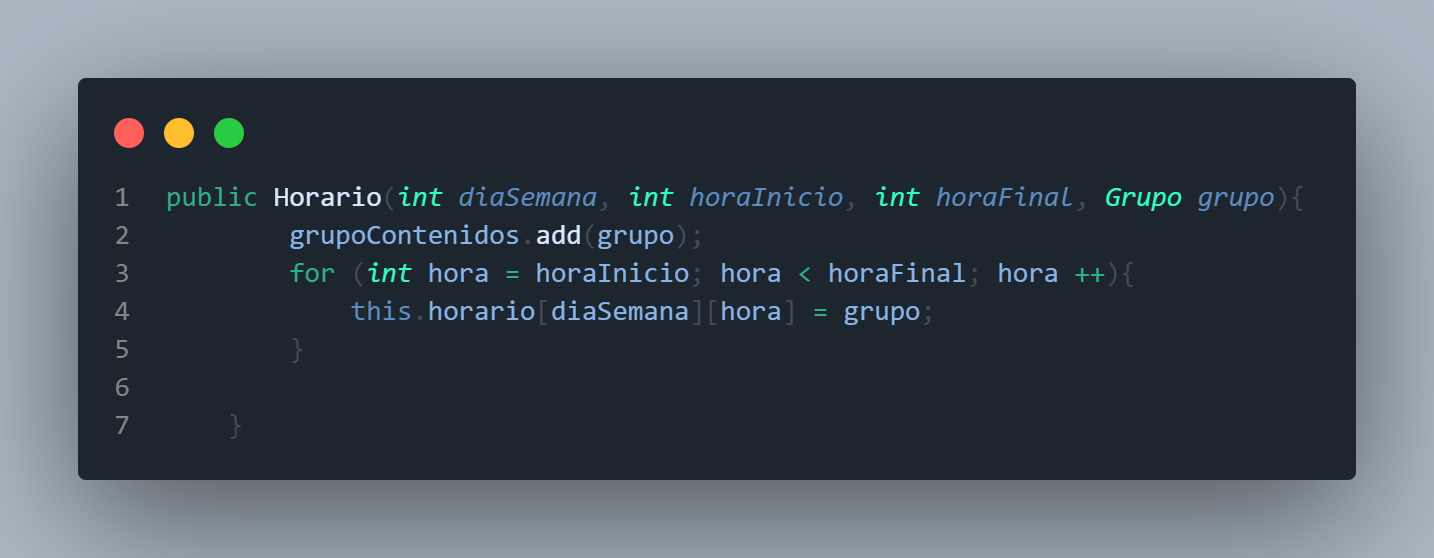
**Sobrecarga.**

Un ejemplo de sobrecarga de métodos lo vemos en Horario, donde el metodo comprobarDisponibilidad() se sobrecarga al definir uno con una entrada de tipo String y otra ArrayList<String>)



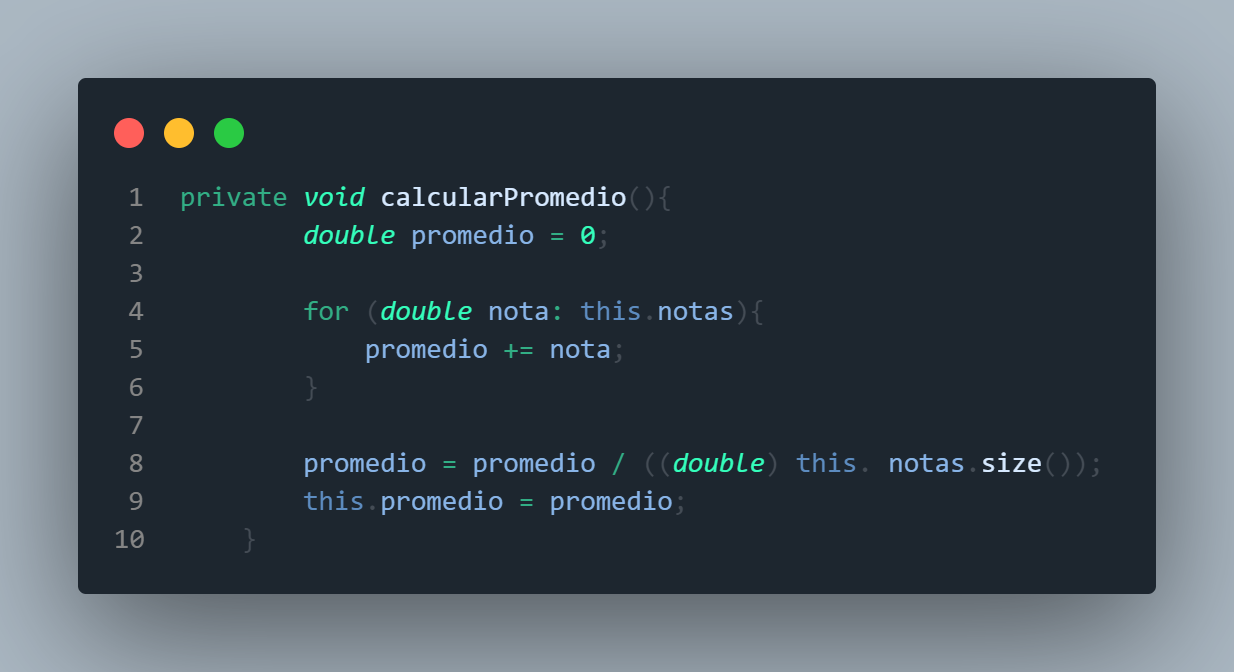
De manera similar, en la clase Horario también contamos con un ejemplo de sobrecarga de constructores



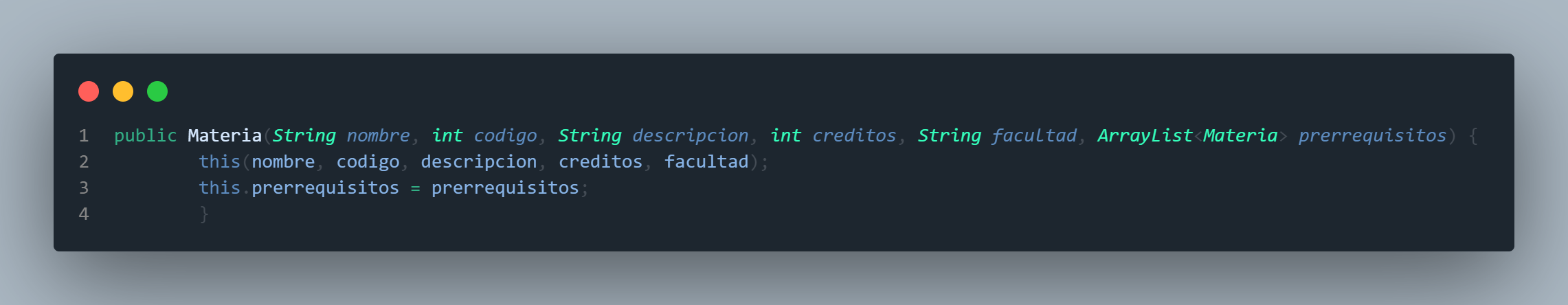


**This**

El this lo utilizamos para varias cosas, principalmente para evitar ambigüedades, un ejemplo dentro de la clase Estudiante, específicamente en el metodo calcularPromedio, para diferenciar entre las variables de nombre Promedio



Tambien el this() lo usamos para llamar un constructor sobrecargado dentro de otro de la misma clase, ejemplo en Materia

****

**Enumeración**

La enumeración se usa en la clase Horario dentro del método mostrarHorario. La enumeración DiaSemana consiste en los 7 días de la semana a los cuales se les asigna dos valores de tipo Int que son el índice y length. También se declara un método dentro de esta para obtener uno de los días de la semana según un índice que se ingrese como parámetro.

Esta enumeración se utiliza para, a medida que se hace un ciclo for para formar el String que retornará mostrarHorario, obtener de manera sencilla la longitud que pueden tener los nombres de las materias matriculadas que se muestren según el día de la semana al que corresponda.

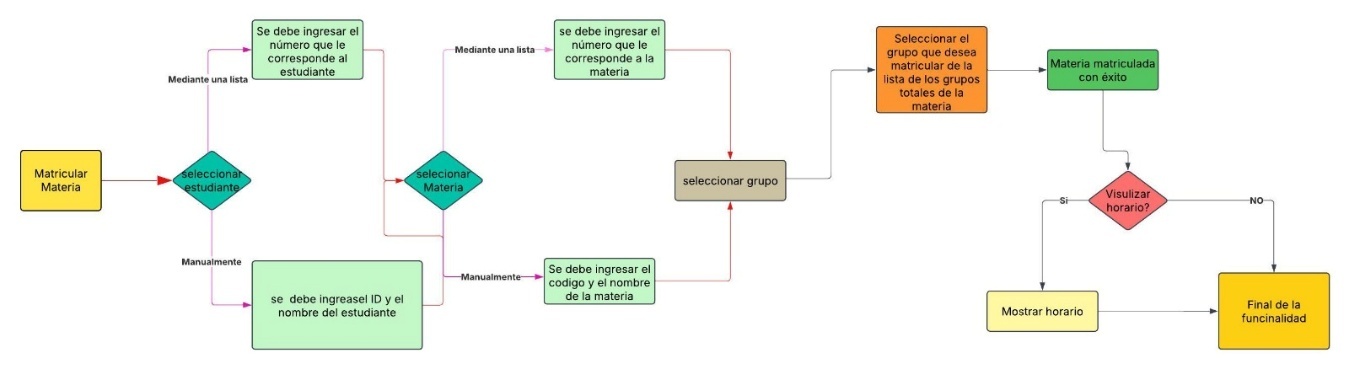


**Funcionalidades.**

1. **Matricular materia**: los coordinadores pueden matricular a los estudiantes en las materias, seleccionando una de las diferentes materias disponibles,
2. **Generar Horario**: Los coordinadores puede generar un horario, lo que le permite visualizar la distribución de las clases
3. **Agregar/eliminar materia/grupo**: Los coordinadores pueden agregar o eliminar una materia o un grupo de alguna materia
4. **Des matricular alumno**: Los coordinadores pueden des matricular a los estudiantes de las materias en las que ya no participaran, o también eliminarlos del sistema académica en general.
5. **Búsqueda y postulación de becas**: Los coordinadores pueden buscar y postular estudiantes a becas que les pueden ayudar a financiar sus estudios.

**Matricular materia.**

Flujo



Descripción

La funcionalidad de matricular materias permite a los coordinadores académicos inscribir a los estudiantes en las materias disponibles en el sistema de manera eficiente. Para garantizar una matriculación precisa, el proceso de inscripción tiene en cuenta diversos factores, como los créditos ya inscritos por el estudiante, la disponibilidad de grupos en las materias seleccionadas, la agenda disponible del estudiante y los prerrequisitos de cada asignatura.

El coordinador puede elegir ver a los estudiantes en forma de lista o buscarlo de manera manual mediante el nombre o su ID. Luego, de manera análoga, el coordinador puede elegir como visualizar la materia, en lista o buscarla por su nombre o código; luego el sistema verifica la disponibilidad de cupos, prerrequisitos y créditos disponibles del estudiante antes de hacer efectiva la matricula.

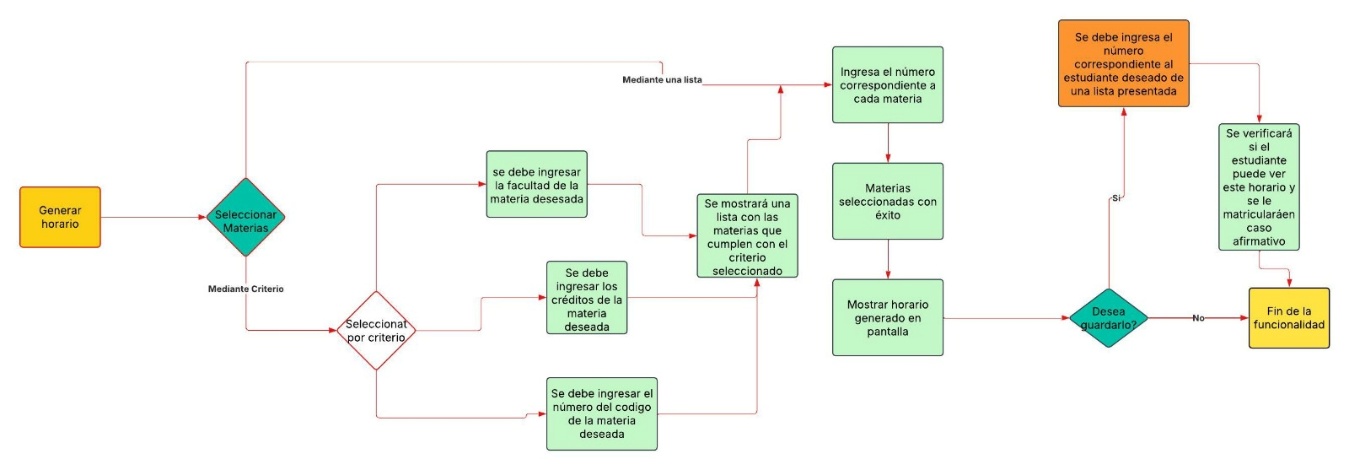
Finalmente, se presentará una lista de grupos disponibles para la materia seleccionada y se verificará la disponibilidad del estudiante en relación con el horario de los diferentes grupos. El coordinador podrá seleccionar el grupo deseado y la matriculación de la materia estará lista.

Secuencia

1. El usuario accede al sistema como coordinador y selecciona la opción de matrícula materia (1), luego seleccionar como desea seleccionar a el estudiante(1-lista, 2-manual)
   1. Si se selecciona esta opción, la funcionalidad hará un filtrado al Arraylist que se obtiene de **getEstudiantes()**, método estático de la clase Estudiante. El filtrado es para verificar que **isMatriculaPagada()** sea true y también se verifica que cada estudiante al invocar el método **getCreditos()** no sea igual al límite de créditos impuestos por la clase Coordinador que verificamos con el método estático **getLimitesCreditos()**. Por último, se debe ingresar el número que le corresponde al estudiante deseado.
   2. Si se selecciona esta opción se le pedirá al usuario ingresar el ID y el nombre del estudiante que desea seleccionar, esto para buscarlo más fácilmente con el método **buscarEstudiante(nombre, id)** de la clase Estudiante, la cual verificará que exista el estudiante, si no existe se le notificará al usuario. Antes de pasar al siguiente paso se verificará que **isMatriculaPagada()** sea true y también se verifica que cada estudiante al invocar el método **getCreditos()** no sea igual al límite de créditos impuestos por la clase.
2. El usuario selecciona la materia de manera análoga (1-lista, 2-manual)
   1. Si se selecciona esta opción, se hará un filtrado de las materias totales con respecto al estudiante seleccionado en el paso anterior y se deberá ingresar luego el número que le corresponde a la materia que desea matricular. Entonces primer se verifica que el estudiante cumpla con todos los prerrequisitos de la materia con el método estático **comprobarPrerrequisitos(estudiante, materia)** de la clase Materia que hará un recorrido por todos los prerrequisitos y nos devolverá un booleano, luego comprobamos que la materia tenga cupos con el método **getCupos()** de la clase Materia, luego comprobamos que al ser matriculada al estudiante, esta materia no le haga sobrepasar el límite de créditos impuesto por la clase Coordinador, lo comprobamos con el método estático **getLimitesCreditos()**, con el método de la clase Estudiante **getCreditos()** y con el método de la clase Materia **getCreditos().** Por último, se verifica que el estudiante no la esté viendo este semestre revisando el **ArrayList** que devuelve el método de la clase Estudiante de **getMaterias()**
   2. Si se selecciona esta opción, el usuario deberá ingresar el código y nombre de la materia deseada, con lo cual se podrá invocar el método estático de la clase Materia que comprueba la existencia de la materia, el cual es **buscarMateria(nombre, código)**. Si no existe la materia se le notifica al usuario y si existe, entonces procede a comprobar todo lo siguiente. Primer se verifica que el estudiante cumpla con todos los prerrequisitos de la materia con el método estático **comprobarPrerrequisitos(estudiante, materia)** de la clase Materia que hará un recorrido por todos los prerrequisitos y nos devolverá un booleano, luego comprobamos que la materia tenga cupos con el método **getCupos()** de la clase Materia, luego comprobamos que al ser matriculada al estudiante, esta materia no le haga sobrepasar el límite de créditos impuesto por la clase Coordinador, lo comprobamos con el método estático **getLimitesCreditos(),** con el método de la clase Estudiante **getCreditos()** y con el método de la clase Materia **getCreditos().**Por último, se verifica que el estudiante no la esté viendo este semestre revisando el ArrayList que devuelve el método de la clase Estudiante de **getMaterias().**
3. Luego se debe seleccionar el grupo de la materia que desea matricular. En este paso se le mostrará una lista con todos los grupos disponibles para que estudiante pueda ver. Primero se verifica la disponibilidad de horario del estudiante, con lo cual invocamos el método de la clase Estudiante getHorario() y luego el método de la clase Horario comprobarDisponibilidad(horario), con el anterior proceso obtendremos un booleano que nos dirá si hay disponibilidad en el horario o no. Si no, entonces ese grupo no se mostrará y si sí, entonces procedemos a comprobar los cupos que tiene ese grupo con el método de la clase Grupo getCupos(). Mostrada la lista, se debe ingresar el número del grupo que desea matricular.
4. De la clase Estudiante obtendremos dos ArrayList, el primero para añadir los grupos que está viendo el estudiante **getGrupos()** y el otro para ver las materias que está viendo el estudiante **getMaterias(),** al primero añadimos el grupo seleccionado y al segundo la materia seleccionada, para hacer válido el anterior proceso invocamos el método de la clase estudiante **setMaterias()** y **setGrupos().** Luego se añadirá el estudiante seleccionado a los estudiantes que están en el grupo seleccionado con el método de la clase Grupo **agregarEstudiante(estudiante).** A la materia y al grupo se le restará un cupo de los totales de la materia y del grupo, esto se hace con los métodos de cada clase, la clase Materia con los métodos **getCupos()** y luego con el método **setCupos().** Lo mismo con la clase Grupo. Por último, se modifica los créditos que está viendo el estudiante con el método de la clase Estudiante **getCreditos(),** sumándole los créditos de la materia seleccionada con el método de la clase Materia **getCreditos()** y luego se le pondrán estos créditos al estudiante con el método **setCreditos().** Antes de pasar al siguiente paso se debe modificar el horario del estudiante, con lo cual invocamos el método de la clase Estudiante **getHorario()** y luego con el método de la clase Horario cambiamos este horario **ocuparHorario(grupoSeleccionado).**
5. Por último, se pregunta al usuario si desea visualizar el horario del estudiante, si la respuesta es afirmativa, entonces se invoca el método de la clase Estudiante **getHorario()** y luego se invoca el método de la clase Horario **mostrarHorario()**. Con esto finaliza la funcionalidad.

**Generar horario**

Flujo



Descripción

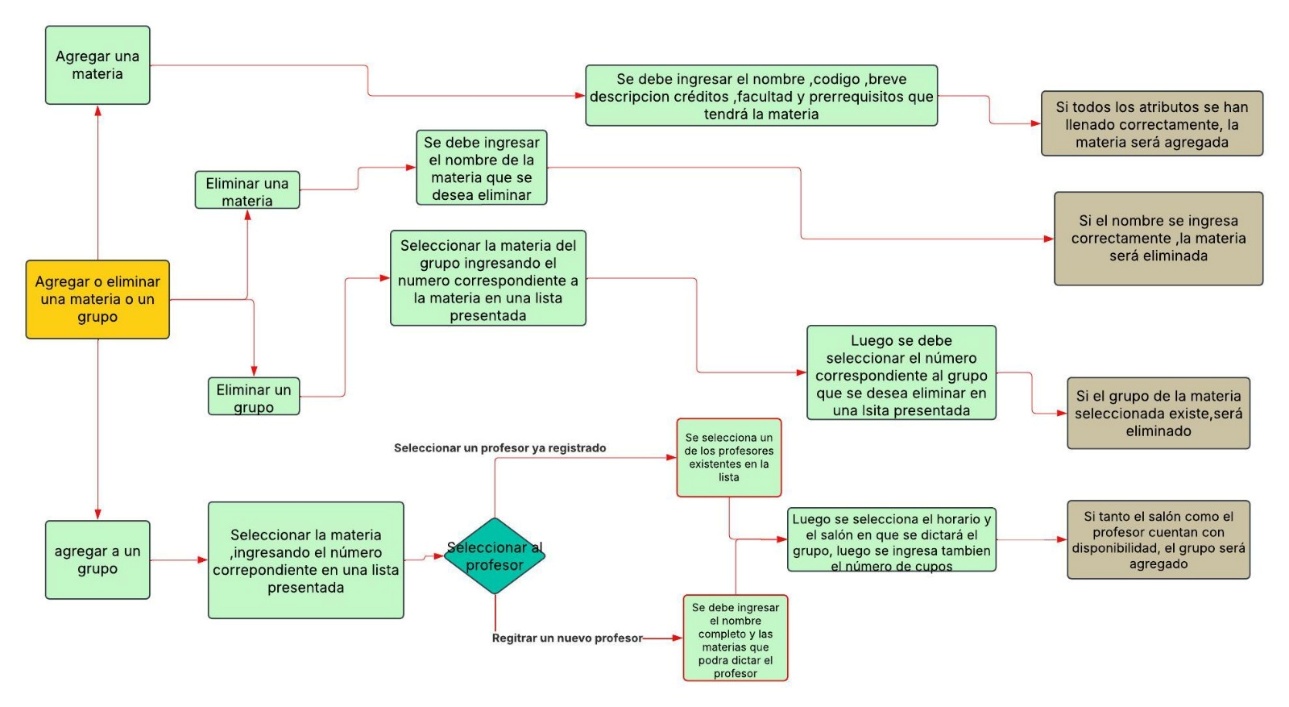
La funcionalidad tiene por objetivo crear un horario según unas materias que la consola le pedirá al usuario, si es posible crear el horario este te lo mostrará por consola, de lo contrario se notificará cuál es la materia que le impide poder generarlo. Una vez se tenga el horario generado, la consola preguntará si se desea desechar el horario o conservarlo, al conservarlo se tiene que elegir a qué estudiante asignárselo, de forma que el horario que tenía el estudiante es reemplazado por el nuevo horario, en este punto pueden pasar dos cosas: el estudiante si puede ver el horario y se le asigna o el estudiante no puede ver el horario porque no cumple los prerrequisitos, por lo tanto, no se le puede asignar.

Secuencia

1. Se le pasa una lista que contiene objetos de tipo Materia al método crearHorarioAleatorio de la clase Coordinador que genera un horario, el cual es una instancia de Horario y también:
   1. Consulta la disponibilidad en el horario para cada grupo, haciendo uso del método **comprobarDisponibilidad()** de Horario
   2. Luego de comprobar la disponibilidad, usa el método **ocuparHorario()** para mostrar el horario generado
2. Mostrar el horario generado en pantalla
   1. Usamos el método **mostrarHorario()** de Horario.
3. En caso de elegir la opción de asignar el horario a un estudiante, comprobar si este puede verlo.
   1. Mediante el método **asignacionDehorarioGenerado()** en el Main, podemos realizar diferentes comprobaciones gracias a los atributos e instancias de Grupo y al método **puedeVerMateria()** de Materia.
4. Realizar asignación del horario al estudiante
   1. Con el método **matricularMateriaParte4** asignamos un grupo a un estudiante, este método interactúa con Grupo y Materia

**Agregar/eliminar materia/grupo**

Flujo



Descripción

Esta funcionalidad permite a un coordinador agregar o eliminar una materia a la base de datos o un grupo a una materia ya definida. En esta se muestra un menú con cuatro opciones: agregar materia, eliminar materia, agregar grupo y eliminar grupo.

En la opción 1 (Agregar materia), se solicitan los datos necesarios para crear una nueva materia. Una vez que se ingresan los datos, se agrega la materia a la base de datos.

En la opción 2 (Eliminar materia), se solicita el nombre de la materia que desea eliminar. Si la materia existe, esta se elimina del registro de materias.

En la opción 3 (Agregar grupo), se solicita el nombre de la materia a la cual desea agregar un nuevo grupo. Luego se solicitan los datos, elementos necesarios para su creación: profesor, salón, cupos y horario.

En la opción 4 (Eliminar grupo), se solicita el nombre de la materia a la cual se le desea eliminar el grupo y el número al que este corresponde. Si el grupo existe, este se elimina del arreglo de grupos de la materia.

Secuencia

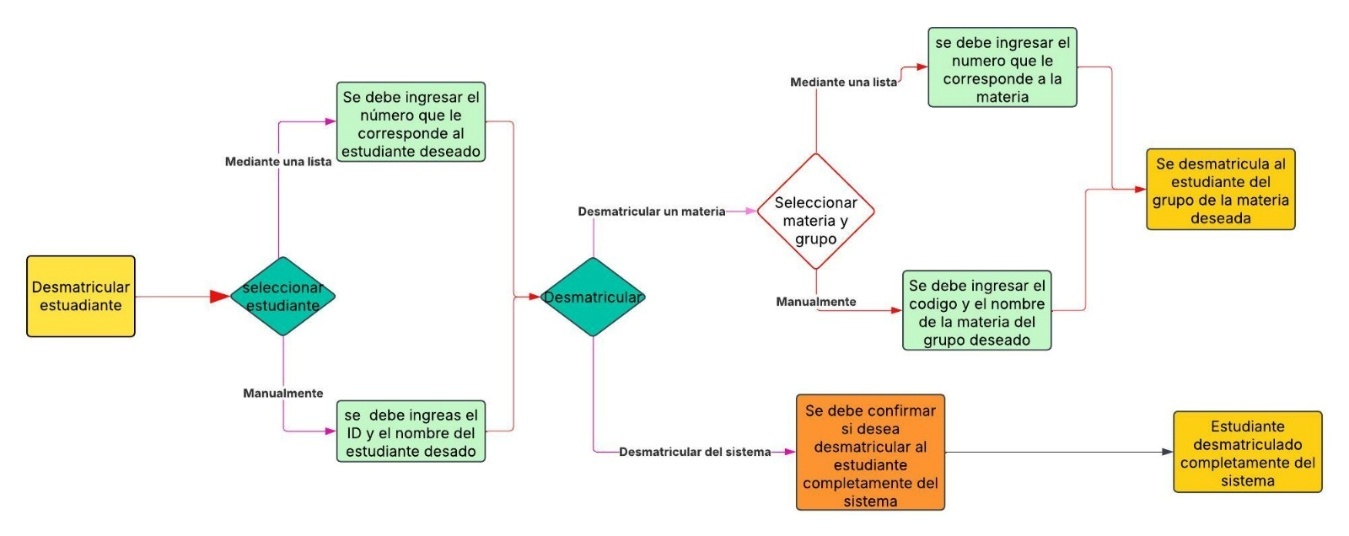
1. Agregar materia
   1. Ingresar el nombre. Se comprueba que no haya una materia ya existente con el mismo nombre, recorriendo **materiasTotales**.
   2. Ingresar código. Se comprueba que no haya una materia ya existente con el mismo código, recorriendo **materiasTotales**.
   3. Ingresar descripción, créditos, y facultad.
   4. Ingresar el nombre de las materias prerrequisto, se hace un Split, para buscar cada materia en **materiasTotales.**
   5. Se llama **agregarMateria()** de Coordinador y se le pasan los datos anteriormente proporcionados.
2. Eliminar materia
   1. Se ingresa el nombre de la materia que se desea eliminar de **materiasTotales**. Si la materia está registrada, se llama al método **eliminarMateria**, el cual verifica que la materia exista y llama al método **restaurarMateria.** Este método recorre grupos haciendo:
      1. Llama al método **desvincularGrupo** del profesor de la materia, el cual libera los espacios ocupados por ese grupo en su horario y lo elimina de su ArrayList grupos.
      2. Llama al método **liberarHorario** en el horario del salón para liberar también sus espacios.

Luego, se recorre el ArrayList de los estudiantes matriculados en el grupo. A cada estudiante se le aplica el método **desmatricular** para eliminar el grupo de su ArrayList grupos.

1. Agregar grupo
   1. Se selecciona el nombre de la materia a la cual se le va a agregar un grupo. Se muestra una lista con todas las materias dentro materiasTotales llamando al método mostrarMaterias de la clase Materia. El usuario debe ingresar el número correspondiente a la materia que desee seleccionar.
   2. Se pregunta cómo se quiere asignar el profesor a la nueva materia. Si seleccionar un profesor ya existente en el ArrayList profesores de la clase Profesor o crear un nuevo profesor.
      1. Si es la primera opcion, se llama **mostrarProfesMateria()**, luego se llama **profesoresDeMateria()** para que devuelva la lista de profesores y el usuario elija uno
      2. Si es la segunda opción, se debe ingresar el nombre del nuevo profesor, luego se muestran las materias registradas y se le pide al usuario seleccionar una a una las materias las cuales el profesor puede dictar, obligatoriamente dentro de estas debe estar la seleccionada en el paso a)
   3. Preguntar horario para el grupo
      1. Los horarios se guardan como Strings de un formato d-hi-hf (día, hora inicio, hora final).
      2. Ingresar la cantidad de sesiones a la semana.
      3. Preguntar el dia de la semana (valor de d).
      4. Preguntar hora inicial.
      5. Preguntar hora final.
      6. Con los datos anteriores se agregará el horario en el formato dado en un ArrayList.
   4. Preguntar numero de cupos del grupo
   5. Preguntar Salon donde se dará la clase
      1. Se muestra la lista de salones con el método **mostrarSalones()** de Salon
      2. Ingresar el numero correspondiente al salón a elegir.
   6. Con todos estos datos, llamamos al método **agregarGrupo()** de Materia para intentar asignar este nuevo grupo a la materia seleccionada
      1. Este método llama a **comprobarDisponibilidad() para** verificar la disponibilidad del profesor y del salón en el día y el horario asignado. Para esto, se descompone el String en formato d-hi-hf con substrings.
      2. El método también verifica que la materia a la cual corresponde el grupo se encuentre en el ArrayList **materiasDadas** del profesor por medio del método **daMateria**.
      3. Si se comprueba la información anterior, el grupo será asignado a los grupos de la materia definida. Con esto, se llamará al método **ocuparHorario** de la clase horario para rellenar los espacios ocupados por el horario del grupo en el horario del salón seleccionado. También se llama al método **vincularGrupo** de la clase profesor para agregarle el grupo a su ArrayList grupos y también invocar al **ocuparHorario** para rellenar su atributo horario.
      4. El atributo número del grupo se obtiene al obtener el tamaño del ArrayList grupos de la materia y sumarle 1 a este valor.
   7. Si el grupo fue agregado exitosamente, se mostrará un mensaje en pantalla indicándolo. De lo contrario, se mostrará un mensaje explicando por qué no se pudo agregar el grupo.
2. Eliminar Grupo
   1. Seleccionar el nombre de la materia.
      1. Se muestran todas las materias.
      2. Ingresar el numero correspondiente
   2. Ingresar el numero del grupo a eliminar.
      1. Si el número del grupo corresponde a un grupo existente dentro de los grupos de la materia se procederá con la eliminación.
   3. Se llama a **eliminarGrupo()**  y se mostrará un mensaje de confirmación.
      1. Llama **liberHorario()** de Salón.
      2. Llama a **desvincularGrupo** de Profesor y **liberarHorario** de su respectivo horario.
      3. Luego de eliminar el grupo del ArrayList grupos de la materia, se recorrerán los grupos que se encontraban delante del grupo eliminado y se le restará 1 a su atributo numero.

**Des matricular alumno**

Flujo



Descripción

La funcionalidad en cuestión es un proceso de desmatriculación de un estudiante en un sistema de matrícula de materias. El usuario selecciona la opción de desmatricular a un estudiante y se le presentan varias opciones para seleccionar al estudiante en cuestión. Una vez que el estudiante ha sido seleccionado, se le presentan varias opciones para desmatricularlo de una materia específica o eliminarlo completamente del sistema.

Si el usuario elige desmatricular al estudiante de una materia específica, se le presentan opciones para seleccionar la materia y el grupo. Si se selecciona la opción de buscar la materia y el grupo, se le pedirá al usuario que ingrese el nombre de la materia y el sistema buscará la materia y el grupo en los que está matriculado el estudiante. Si se selecciona la opción de ver la lista de materias y grupos, se presentará una lista de las materias y los grupos en los que está matriculado el estudiante y el usuario deberá seleccionar el número de la materia y el número del grupo correspondiente para desmatricular al estudiante.

Si el usuario elige desmatricular al estudiante del sistema, se le pedirá al usuario que confirme su acción. Si se confirma, el estudiante será eliminado completamente del sistema y todas las materias y grupos en los que estaba matriculado serán eliminados de su horario.

Secuencia

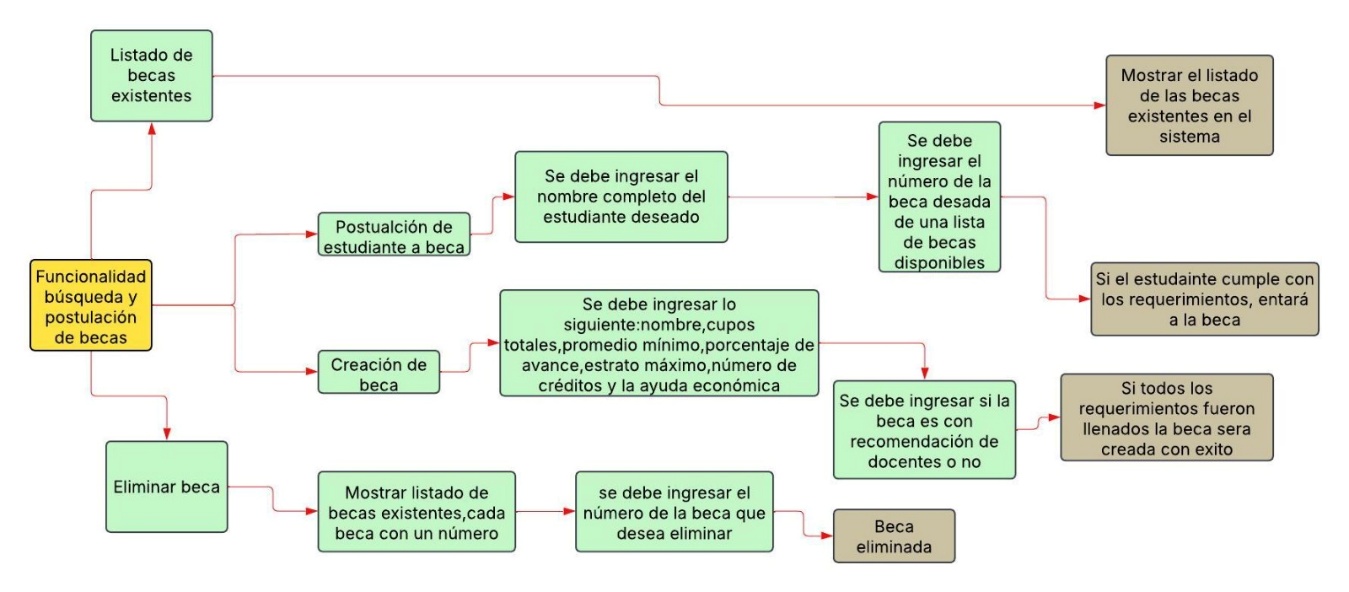
Se selecciona el estudiante a desmatricular, mediante una lista o de manera manual mediante su Id y nombre.

Después de esto se tienen las dos ramificaciones de la funcionalidad.

1. Desmatricular de una materia
   1. Elegir materia y grupo de la lista de materias que tiene inscritas el estudiante seleccionado
   2. Al ser seleccionada la materia y su respectivo grupo se usa el método estático **buscarGrupo** de Grupo, el cual recibe como parámetros las instancias materia y grupo y retorna una referencia del grupo respectivo del atributo **gruposTotales** de Grupo para hacer los cambios necesarios en la referencia del grupo de la base de datos.
   3. Luego de seleccionados estos objetos, se usa el método **existenciaEstudiante** de grupo para comprobar si el estudiante está matriculado en el grupo.
   4. Si el método retorna true se continúa llamando el método **eliminarEstudiante** del objeto grupo que elimina al estudiante de la lista de estudiantes del grupo, suma el cupo liberado al atributo cupos del grupo, elimina al grupo y la materia de las respectivas listas del estudiante y le resta los créditos de la materia al estudiante.
   5. Desde el atributo horario del estudiante se llama el método **liberarHorario** de horario que se encarga de excluir el horario del grupo que se pasa como parámetro del horario del estudiante.
2. Desmatricular del sistema.
   1. Se llama el método comprobacionFacultad de objeto coordinador desde el apuntador tipo usuario del usuario que inició sesión y se le pasa como parámetro el estudiante que se quiere desmatricular para comprobar si el coordinador es de la facultad del estudiante y tiene el permiso de desmatricularlo. Si el método retorna true se sigue con el proceso, si retorna false se termina.
   2. Desde el atributo horario del estudiante se llama el método **vaciarHorario**
   3. Se usa el método **desmatricularMaterias** de estudiante donde se desvincula de todos los grupos y materias en los que estaba matriculado el estudiante
   4. Desde el usuario se llama el método **desmatricularDelSistema** de Coordinador encargado de eliminar al estudiante de la lista de estudiantes y usuarios del programa.

**Búsqueda y postulación de becas.**

Flujo



Descripción

Esta funcionalidad le permite a un coordinador tener control sobre las becas existentes (eliminarlas o crearlas), además permite al estudiante, siempre y cuando cumpla con los requisitos individuales de la beca a la cual este quiera aplicar, el beneficiarse de la ayuda económica brindada por las entidades responsables de cada una de estas.

En esta funcionalidad tenemos 4 opciones:

1. Ver becas existentes
   1. En esta opción se despliega la lista de materias, esto através del método estático **mostrarBecas()**, el cual recorre una a una las becas que existen y las muestra en una lista numerada, además de información adicional que le facilita al usuario saber el estado de la beca a través de los métodos **getCupos()**, **getEstratoMinimo()** **y getCreditosInscritosRequeridos().**
2. Aplicar beca a estudiante
   1. Se le pide al usuario que ingrese el nombre completo del estudiante al que le quiere aplicar la beca, para que sea mas fácil, se muestra la lista de estudiantes existente.
   2. Se le muestra al usuario la lista de becas actuales con el método **mostrarBecas()** de la clase Interfaz y se le pide que ingrese el número que corresponde a la beca a la cual quiere aplicar. Se verifica que la beca tenga cupos, si hay disponibles, se verifican los requisitos con el método **candidatoABeca()** de Coordinador; si la beca requiere recomendación de un docente, se llamara al método **recomendarEstudiante()**, el cual mediante la librería Math, realiza cierta lógica para establecer si el Estudiante recibe la recomendación del profesor o no.
   3. Si lo anterior se cumple se le sumara la ayuda económica de beca al sueldo del estudiante en cuestión con **setSueldo()**, y asi pagar la matricula.
3. Crear beca
   1. Ingresar el número de cupos totales.
   2. Ingresar el nombre de la beca. Se verifica que no exista.
   3. Ingresar promedio mínimo para aplicar (con coma)
   4. Ingresar el porcentaje de avance mínimo (número entero)
   5. Ingresar estrato máximo
   6. Ingresar número de créditos mínimos inscritos.
   7. Ingresar ayuda económica (sin puntos ni coma)
   8. Seleccionar si la beca requiere recomendación de un profesor o no.
   9. En caso de que todo se ingrese correctamente, estos datos serán los parámetros del constructor de Beca.
4. Elimina beca.
   1. Se le muestra al usuario la lista de becas actuales con mostrarBecas() y se le pide que ingrese el número que corresponde a la que quiera eliminar. Se le informa al usuario que la beca fue eliminada con éxito.

**Manual de usuario.**

**Matricular materia**

Paso 1: Seleccionar la funcionalidad matricular materia.

Una vez que haya ingresado al sistema, busque la opción "Matricular materia" en el menú principal y selecciónela. Se le presentarán dos opciones:

1. Ver la lista de estudiantes disponibles para matricular materias.

2. Buscar estudiante manualmente.

Paso 2: Selección del estudiante

Seleccione al estudiante que desea matricular en una materia. Si selecciona la opción 1, verá una lista de los estudiantes disponibles para matricular materias. Si selecciona la opción 2, debe ingresar el nombre y el ID del estudiante manualmente.

El sistema verificará que el estudiante pueda matricular más materias en función de los créditos ya inscritos y la matrícula pagada.

Paso 3: Selección de la materia

Después de seleccionar al estudiante, se le presentará un menú para seleccionar la materia que desea matricularle al estudiante. Se le ofrecerán dos opciones:

1. Ver la lista de materias disponibles para matricularle al estudiante seleccionado.

2. Buscar manualmente la materia que le desea matricular al estudiante.

Si selecciona la opción 1, se le presentará una lista de materias disponibles para matricular. Si selecciona la opción 2, debe ingresar manualmente el nombre y el código de la materia. El sistema verificará los cupos disponibles, los prerrequisitos de la materia y el límite de créditos permitido para el estudiante antes de permitir la selección de la materia deseada.

Paso 4: Selección del grupo

Después de seleccionar la materia, se le presentará una lista de los grupos disponibles con cupos para matricular. El sistema también verificará la disponibilidad del estudiante seleccionado en función del horario de los diferentes grupos.

Seleccione el grupo que desea matricular al estudiante seleccionado y la matriculación de la materia estará completa. Luego de haber seguido los pasos anteriores habrá matriculado exitosamente a un estudiante en una materia.

Como recordatorio, se puede repetir este proceso para matricular a más estudiantes a las diferentes materias disponibles.

**Desmatricular alumno.**

Paso 1. Ejecuta el programa y selecciona la opción 4 para des matricular a un estudiante.

Paso 2. Se te pedirá que elijas cómo deseas seleccionar al estudiante que deseas desmatricular. Puedes seleccionar entre tres opciones:

Paso 3. Ver la lista de estudiantes: muestra una lista de estudiantes disponibles para su des matriculación. Selecciona el número correspondiente al estudiante que deseas desmatricular. (pasar al paso 6)

Paso 4. Buscar estudiante por ID y nombre: busca al estudiante ingresando su nombre e ID. Si se encuentra al estudiante, se mostrará un mensaje de confirmación y se procederá a la desmatriculación. (pasar al paso 6). Si el estudiante no se encuentra, se mostrará un mensaje de error.

Paso 5. Salir: salir de la selección del estudiante y volver al menú principal.

Paso 6. Si has seleccionado un estudiante, se te pedirá que selecciones de qué deseas desmatricular al estudiante. Puedes seleccionar entre tres opciones:

Paso 7. Desmatricular de una materia: te permite seleccionar una materia y un grupo de la lista de materias a las que el estudiante está matriculado. Una vez seleccionado, el estudiante será desmatriculado de ese grupo. (pasar al paso 10)

Paso 8. Desmatricular del sistema: elimina al estudiante del sistema de matrícula de materias por completo. (pasar al paso 14)

Paso 9. Retroceder: volver a la selección de estudiante.(pasar al paso 2)

Paso 10. Si has seleccionado "Desmatricular de una materia", se te pedirá que selecciones cómo deseas seleccionar la materia y el grupo. Puedes seleccionar entre dos opciones:

Paso 11. Ver lista de materias y grupos: muestra una lista de materias y grupos a los que el estudiante está matriculado. Selecciona el número correspondiente a la materia y al grupo que deseas desmatricular al estudiante. (pasar al paso 13)

Paso 12. Buscar materia y grupo: te permite buscar una materia por nombre e ingresar el número de grupo correspondiente para desmatricular al estudiante. (pasar al paso 13)

Paso 13. Una vez que hayas seleccionado la materia y el grupo de tu elección, el estudiante será desmatriculado y se mostrará un mensaje de confirmación. (pasar al paso 15)

Paso 14. Si has seleccionado "Desmatricular del sistema", se te pedirá que confirmes tu acción. Si el estudiante pertenece a una facultad diferente, se mostrará un mensaje de error y se te pedirá que intentes nuevamente.

Paso 15. Si has completado la desmatriculación, se te pedirá que salgas o vuelvas a intentarlo con otro estudiante.

**Generar horario.**

Paso 1. Ejecuta el programa, acceder como coordinador y seleccionar la opción generar horario.

Paso 2. El programa te solicitará que selecciones cómo quieres seleccionar las materias. A continuación, se muestran las opciones que tendrás en el menú de selección de materias: (1) Ver la lista de materias. (2) Buscar por criterio (Facultad - Créditos - Código) y (3) Salir.

Paso 3. Si eliges la opción "Ver la lista de materias", se mostrará una lista de todas las materias disponibles y posteriormente se deben seleccionar las materias deseadas. (pasar al paso 7).

Paso 4. Si eliges la opción "Buscar por criterio”. Se te presentará otro menú que te permitirá buscar materias según tres criterios: por facultad, por créditos o por código.

Paso 5. Una vez que selecciones el criterio de búsqueda, se te pedirá que ingreses el valor correspondiente al criterio elegido.

Paso 6. El programa mostrará una lista de materias que cumplen con el criterio seleccionado.

Paso 7. Luego de seleccionar las materias, el programa generará un horario aleatorio a partir de las materias seleccionadas.

Paso 8. Puedes generar tantos horarios como desees.

**Eliminar o agregar materia o grupo de una materia.**

Paso 1. Ingrese a la aplicación, inicie sesión como coordinador, seleccione la opción de eliminar o agregar materia o grupo de una materia y luego seleccione la opción correspondiente a agregar o eliminar una materia o grupo.

Paso 2: Si desea agregar una materia:

* Ingrese el nombre de la materia que desea agregar.
* Ingrese el código de la materia que desea agregar.
* Ingrese una breve descripción de la materia.
* Ingrese los créditos que le asigna a la materia.
* Ingrese la facultad a la que pertenece la materia.
* Ingrese los prerrequisitos que tiene la materia para poder ser inscritas por el estudiante (separadas por comas con su respectivo espacio).

Si todos los pasos anteriores se han hecho correctamente, la materia será agregada.

Paso 3: Si desea eliminar una materia:

* Ingrese el nombre de la materia que desea eliminar.
* Si el nombre se ingresa correctamente, la materia será eliminada.

Paso 4: Si desea agregar un grupo

* Seleccione la materia
* Seleccione la como quiere agregar al profesor
  + Seleccionar profesor existente
    - Seleccione el numero de la lista
  + Crear nuevo profesor
    - Ingrese el nombre completo
    - Seleccione las materias que puede dictar (números uno por uno, 0 para terminar)
* Seleccione el horario
  + Ingrese el numero de sesiones a la semana, por cada sesión haga lo siguiente;
    - Seleccionar dia de la semana con el numero de la lista
    - Seleccionar hora de inicio con el numero de la lista
    - Seleccionar hora de finalización con el numero de la lista
* Indique el número de cupos
* Seleccione el salón.

Paso 5. Si desea eliminar un grupo.

* Seleccione el nombre de la materia del grupo que quiere eliminar.
* Seleccione el número del grupo

**Búsqueda y postulación de becas.**

Paso 1: Ingrese a la aplicación, inicie sesión como Coordinador, seleccione la opción de Búsqueda y Postulación de Becas.

Paso 2: Seleccione la opción que mejor se ajuste a lo que desea hacer:

Paso 3: Si desea ver el menú de becas existentes actualmente:

Se desplegará un menú numerado, el cual mostrará una a una las becas existentes juntos, los cupos actualizados de las mismas, el estrato máximo para acceder a esta y el mínimo de créditos inscritos al semestre requeridos.

Se le redireccionará nuevamente al menú.

Paso 4: Si desea aplicar beca a estudiante:

Ingrese el nombre completo (con mayúsculas iniciales en cada palabra) del estudiante que desea postular, para mayor comodidad, una lista con los estudiantes actualizados será mostrada en consola.

Se mostrará en pantalla el listado de las becas actuales, seleccione el número que corresponde a la cual quiere aplicar.

Si los datos ingresados exitosamente, recibirá una notificación de lo que pasó con la postulación, si el estudiante pudo aplicar con éxito o si no.

Será redireccionado al menú principal, recuerde seleccionar la opción 6 para que los datos queden guardados en la base de datos.

Paso 5: Si desea crear una nueva beca:

Ingrese el número de cupos totales que tendrá la nueva beca.

Ingrese el nombre que tendrá la nueva beca, recuerde que en caso de ya existir una con ese nombre, se le redireccionará al menú principal nuevamente.

Ingrese el promedio mínimo que debe tener el estudiante para poder aplicar a la beca.

Ingrese el número que representa el porcentaje de avance con el que debe contar el estudiante para poder aplicar a la beca.

Ingrese el estrato máximo que puede tener el estudiante para poder aplicar a la beca.

Ingrese el número de créditos inscritos en el semestre que debe tener el estudiante para aplicar a la beca.

Ingrese la ayuda económica a la que puede acceder el estudiante una vez tenga la beca, está escrita sin puntos ni comas.

Se le preguntará si dicha beca necesitará de recomendación de algún profesor para poder aplicar a la misma.

Una vez todos los datos hayan sido ingresados con éxito, se le notificará.

Será redireccionado al menú principal, recuerde seleccionar la opción 6 para que los datos queden guardados en la base de datos.

Paso 6: Si desea borrar una beca existente:

Se le mostrará la lista de becas existente y debe ingresar el número que corresponde a la cual quiere eliminar.

Se le informará que la beca ha sido eliminada con éxito si los pasos se siguieron correctamente.

Será redireccionado al menú principal, recuerde seleccionar la opción 6 para que los datos queden guardados en la base de datos.

**¿Como ingresar al sistema?**

Para esto tenemos 6 coordinadores básicos, uno por cada facultad (id, contraseña, facultad)

* 11111 – 11111 – Sede
* 22222 – 22222 – Facultad de minas
* 33333 – 33333 – Facultad de ciencias humanas y económicas
* 44444 – 44444 – Facultad de ciencias agrarias
* 55555 – 55555 – Facultad de ciencias
* 66666 – 66666 – Facultad de arquitectura

**Estudiantes base**

1. Nombre: Libardo Jose Navarro Pedrozo Documento: 1234

2. Nombre: Carolina Alvarez Murillo Documento: 2218

3. Nombre: Catalina Restrepo Documento: 1806

4. Nombre: Jennifer Chica Acero Documento: 9564

5. Nombre: Thomas Rodríguez Documento: 2398

6. Nombre: Alejandro Orozco Documento: 2384

7. Nombre: Juan Jose Silva Gomez Documento: 7651

8. Nombre: Efrain Gomez Ramirez Documento: 7652

9. Nombre: Daniela Giraldo Zapata Documento: 7631

10. Nombre: Juliana Gonzales Giraldo Documento: 7731

11. Nombre: Alfredo Zapata Mani Documento: 7001

12. Nombre: Camilo Andres Jaramillo Garzon Documento: 87654

13. Nombre: Alejandro Sanchez Vanegas Documento: 51757

14. Nombre: Oscar Rodríguez Meza Documento: 39473

15. Nombre: Maria Arango Benavides Documento: 51757

16. Nombre: Julio Diaz Salas Documento: 39473

17. Nombre: Davinson Sanchez Mina Documento: 53694

18. Nombre: James Usuga Vanegas Documento: 14511

19. Nombre: Jorge Carrascal Garcia Documento: 30191

20. Nombre: Jefferson Duque Sanchez Documento: 28684

21. Nombre: Islan Barrera Marquez Documento: 61575

22. Nombre: Ana Sofia Gomez Zapata Documento: 4305

23. Nombre: Camilo Zapata Ortiz Documento: 4218

24. Nombre: Tomas Hoyos Perez Documento: 4310

25. Nombre: Luciana Piedrahita Lopera Documento: 4260

26. Nombre: Javier Gomez Gomez Documento: 4114

**Materias base**

1. Nombre: Introduccion a sistemas Código: 3010438

2. Nombre: Fundamentos de programacion Código: 3010435

3. Nombre: Programacion orientada objetos Código: 3007744

4. Nombre: Ingenieria de software Código: 3007853

5. Nombre: Ingenieria de requisitos Código: 3007852

6. Nombre: Calidad de software Código: 3010440

7. Nombre: Gestion proyectos software Código: 3007851

8. Nombre: Productos de software Código: 3007850

9. Nombre: Calculo diferencial Código: 1000004

10. Nombre: Calculo integral Código: 1000005

11. Nombre: Calculo en varias variables Código: 1000006

12. Nombre: Geometria vectorial y analitica Código: 1000008

13. Nombre: Algebra lineal Código: 1000003

14. Nombre: Fundamentos de matematicas Código: 3010334

15. Nombre: Fundamentos de matematicas discretas Código: 3010390

16. Nombre: Probabilidad Código: 3010394

17. Nombre: Catedra Ingenierias Código: 3009511

18. Nombre: Catedra Vida Universitaria Código: 1000089

19. Nombre: Catedra del mar Código: 3008108

20. Nombre: Catedra Antioquia Código: 3007373

21. Nombre: Fisica mecanica Código: 1000019

22. Nombre: Mecanica basica Código: 3010448

23. Nombre: Ecuaciones diferenciales Código: 1000007

24. Nombre: Comercio mundial Código: 3007182

25. Nombre: Biologia celular Código: 3006802

26. Nombre: Analisis de circuitos electricos Código: 3010481

27. Nombre: Dinamica de sistemas Código: 3007311

28. Nombre: Bioquimica Código: 3006828