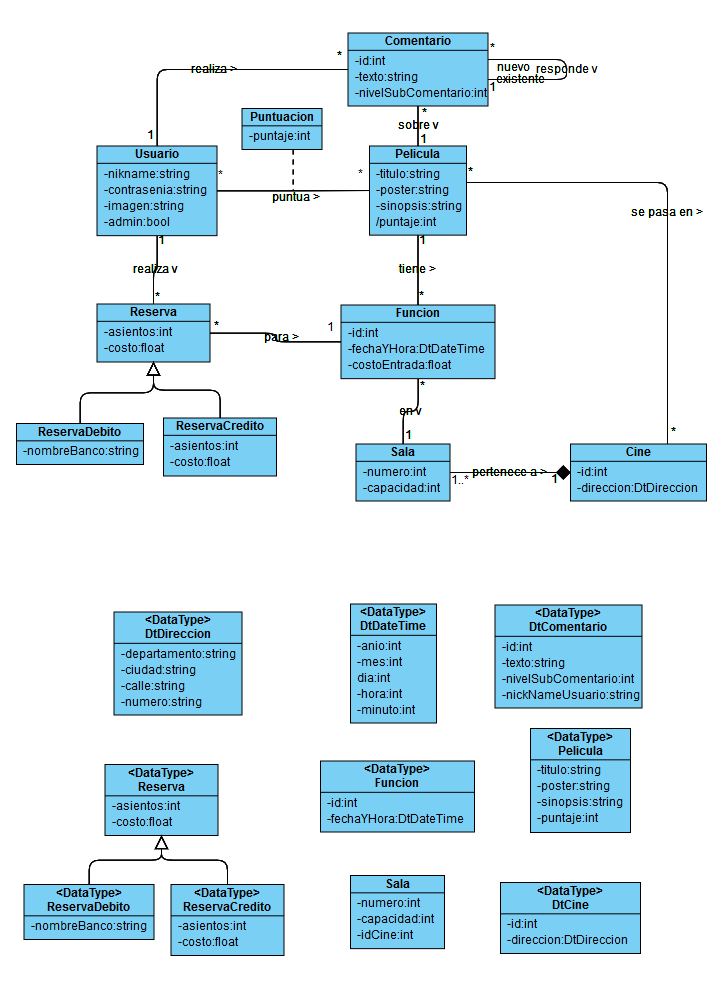
Programación Avanzada

Laboratorio 2

|  |  |
| --- | --- |
| **Grupo 2** | |
|  | |
| **Integrantes** | |
| Juan Álvarez | CI: 4.710.147-5 |
| Facundo Camilo | CI: 5.354.793-4 |
| Lucas Garrido | CI: 4.866.163-4 |
| Julio Arrieta | CI: 3.968.069-5 |

|  |  |
| --- | --- |
| **Docente:** | Nicolás Escobar |

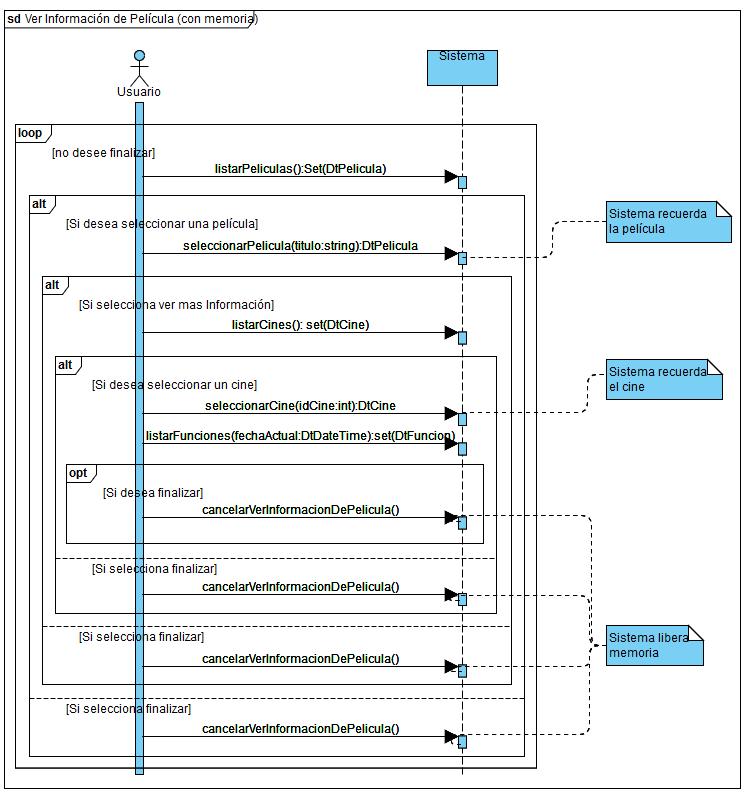
**Modelo de dominio**



Restricciones del modelo de dominio:

* El nicknames identifica a los usuarios
* El id identifica a los comentarios
* El titulo identifica a las peliculas
* El id identifica a las funciones
* El id identifica a los cines
* El número identifica a la sala dentro de un mismo cine.
* La cantidad de asientos de una reserva debe ser menor o igual a la cantidad de asientos disponibles en la sala donde se realiza la función.
* La suma de la cantidad de asientos reservados para una función no puede ser mayor que la cantidad de asientos de la sala.
* No puede haber dos funciones al mismo tiempo en la misma sala. Pero si en el mismo cine.
* Al definir una función, la fecha de ésta no puede ser anterior a la actual.
* El puntaje de una pelicula se calcula como el promedio de los puntajes que el usuario le ha dado a la pelicula

**Diagramas de secuencias de sistema**



**Contratos**

En la siguiente tabla se listan las firmas de operaciones para las cuales fue necesario realizar su contratos, indicando para cada una en que DSS de caso de uso se utiliza.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Firma de funcion** | **Iniciar sesion** | **Crear reserva** | **Puntuar pelicula** | **Eliminar película** |
| cancelarEliminarPelicula() |  |  |  | x |
| cancelarNuevaReserva() |  | x |  |  |
| confirmarEliminarPelicula() |  |  |  | x |
| confirmarNuevaReserva() |  | x |  |  |
| iniciarSesion(nickName:string, contraseña:string):bool | x |  |  |  |
| listarCines():Set(DtCine) |  | x |  |  |
| listarFunciones(fechaActual:DtDateTime):set(DtFuncion) |  | x |  |  |
| listarPeliculas():Set(DtPelicula) |  | x | x | x |
| nuevaReserva(cantAsientos:int) |  | x |  |  |
| obtenerPuntajeDadoPorUsuario():int |  |  | x |  |
| pagoDebito(nombreBanco:string) |  | x |  |  |
| pagoCredito(nombreFinanciera:string):float |  | x |  |  |
| puntuarPelicula(puntaje:int) |  |  | x |  |
| seleccionarCine(idCine:int):DtCine |  | x |  |  |
| seleccionarFuncion(id:int):DtFuncion |  | x |  |  |
| seleccionarPelicula(titulo:string):DtPelicula |  | x | x | x |
| vistaPreviaDeReserva():DtReserva |  | x |  |  |

cancelarEliminarPelicula()

Pre: existe un link pelicula actual

Post: se elimina el link

cancelarNuevaReserva()

Pre: (ninguna)

Post: si existe un link pelicula actual se eliminará

Post: si existe un link cine actual se eliminará

Post: si existe un link funcion actual se eliminará

confirmarEliminarPelicula()

Pre: existe un link pelicula actual

Pre: el usuario actual debe ser administrador

Post: se elimina la instancia de comentario vinculada con pelicula actual, y los comentarios que se vinculaban a este

Post: se elimina el link entre los cines que se vinculaban a pelicula actual

Post: se eliminan las instancias de funcion y las reservas vinculadas a ella

Post: se eliminan las instancias de puntuacion vinculadas a pelicula actual

Post: se elimina la instancia pelicula de pelicula actual

confirmarNuevaReserva()

Pre: existe un link reserva actual

Pre: existe un link usuario actual

Pre: existe un link funcion actual

Post: se crea un link que vincula reserva actual con usuario actual

Post: se crea un link que vincula reserva actual con funcion actual

Post: se elimina el link de pelicula actual

Post: se elimina el link de cine actual

Post: se elimina el link de funcion actual

Post: se elimina el link de reserva actual

iniciarSesion(nickName:string, contraseña:string): bool

Pre: no existe en el sistema un link vinculado a un usuario actual

Pre: existe una instancia de usuario con nickname = nickname y contraseña = contraseña

Post: si existe en el sistema una instancia de usuario con el nickname = nickname y la contraseña = contraseña se creará un link de usuario actual vinculado a dicha instancia, de lo contrario no se creará ningun link y retornará false

listarCines(): set(DtCine)

Pre: existe un link pelicula actual

Post: retorna un conjunto de datatypes de tipo DtCine con los datos de los cines que pasan la película actual

listarFunciones(fechaHoraActual:DtDateTime): set(DtFuncion)

Pre: existe un link pelicula actual

Pre: existe un link cine actual

Pre: fechaHoraActual debe ser igual la fecha actual del sistema

Post: retorna un conjunto de datatypes de tipo dtFuncion con los datos de las funciones para la pelicula actual en el cine actual que se pasarán con fecha posterior a fechaHoraActual

listarPeliculas(): set(DtPelicula)

Pre: (ninguna)

Post: Retorna un conjunto de datatypes de tipo DtPelicula con los datos de cada instancia de película en el sistema

nuevaReserva(cantAsientos:int)

Pre: existe un link funcion actual

Pre: cantidad de asientos mayor a 0

Post: el sistema mantiene en memoria la cantidad de asientos ingresada

ObtenerPuntajeDadoPorUsuario(): int

Pre: existe un link usuario actual

Pre: existe un link pelicula actual

Post: si el usuario ha puntuado la pelicula actual retorna el puntaje dado, de lo contrario retorna -1

pagoCredito(nombreFinanciera:string): float

Pre: el nombre de la financiera no debe ser vacio

Pre: existe en memoria la cantidad de asientos para la reserva mayor a 0

Post: se crea una instancia de tipo reserva credito llamada reserva actual y se le asigna la cantidad de asientos y el nombre de la financiera

Post: se calcula el costo en base a la cantidad de asientos, el precio de la funcion actual y el descuento correspondiente

Post: se crea un link reserva actual

Post: retorna el porcentaje de descuento que tendrá la reserva

pagoDebito(nombreBanco:string)

Pre: el nombre del banco no debe ser vacio

Pre: existe en memoria la cantidad de asientos para la reserva mayor a 0

Post: se crea una instancia de tipo reserva debito llamada reserva actual y se le asigna la cantidad de asientos y el nombre del banco

Post: se calcula el costo en base a la cantidad de asientos y el precio de la funcion actual

Post: se crea un link reserva actual

puntuarPelicula(puntaje:int)

Pre: existe un link usuario actual

Pre: existe un link pelicula actual

Pre: puntaje debe estar comprendido entre 1 y 5

Post: si existe una instancia de la clase puntuacion vinculada al usuario usuario actual y a la pelicula pelicula actual, se sobreescribe el valor puntaje por puntaje, de lo contrario se creará una nueva instancia puntuacion vinculada a usuario usuario actual y a pelicula pelicula actual con puntaje = puntaje

seleccionarCine(id:int): DtCine

Pre: existe una instancia de cine con id=id

Pre: existe un link pelicula actual

Post: se creará un link cine actual vinculado a dicha instancia

Post: retorna un datatype de tipo DtCine con los datos del cine seleccionado

seleccionarFuncion(id:int): DtFuncion

Pre: existe una instancia de funcion con id=id

Pre: existe un link pelicula actual

Pre: existe un link cine actual

Post: se creará un link función actual vinculado a dicha instancia

Post: retorna un datatype de tipo DtFuncion con los datos de la funcion seleccionada

seleccionarPelicula(titulo:string): DtPelicula

Pre: el parámetro titulo no debe ser vacío

Pre: existe una instancia de pelicula con titulo = titulo

Post: se creará un link película actual vinculado a dicha instancia

Post: retorna un datatype de tipo DtPelicula con los datos de la pelicula seleccionada

vistaPreviaDeReserva(): DtReserva

Pre: existe un link reserva actual

Post: retorna un datatype de tipo DtReserva con los datos de la reserva actual