Universidad Nacional de General Sarmiento

PROGRAMACIÓN II

Trabajo Práctico: Sistema Ticketek

Comisión-4

1° cuatrimestre 2025

Profesores:

Adriana Gaudiani

Daniel Carrillo

Alumnos:

Juan Vernon Petre

D.N.I:43.447.330

Email:vernonpetre@gmail.com

Andrea Galvan

D.N.I:42.315.390

Email:andreagalvan0130@gmail.com

Mariana Brussa

D.N.I: 35.121.900

Email:marianabrussaa@gmail.com

Introducción:
En este informe se plantea una posible solución para el sistema de venta de entradas (de espectáculos) para la empresa Ticketek. De tal forma se implementará todo lo visto en clases anteriormente aplicando cuando sea necesario polimorfismo, herencia, sobrecarga y más, en base a los TADs creados.

Posibles TADs

- SistemaTicketek
- Usuario
- Entrada
- Funcion
- Espectaculo
- Sede
- EstadioDeFutbo
- MiniEstadio
- Teatro
- Sector
- Campo
- Platea

Especificación de TADs:

A continuación, daremos la especificación de cada uno de los TADs que se observa en el diagrama:
1. TAD SistemaTicketek
Atributos

- +map<nombreUsuario,Usuario> usuarios
- +list<Espectaculo> espectaculos
- +list<Sede> sedes

Operaciones

- +void registrarSede(String nombreSede, String direccion,int capacidad)
- — Añade una nueva sede al sistema.
- +void registrarUsuario(String nombreUsuario, String nombre, String apellido, String contraseña)
- — Crea y almacena un usuario.
- +void registrarEspectaculo(String nombreEspectaculo)

- — Crea un espectáculo .
- +void venderEntrada(String nombreUsuario, String nombreEspectaculo, String nombreSede, int numAsiento, String contraseña)
- — Vende una entrada para la función seleccionada, validando credenciales y disponibilidad.
- +List<Sede> listarSedesDeEspectaculo(String nombreEspectaculo) — Devuelve las sedes donde se representa un determinado espectáculo.
- +List<Entrada> listarEntradasDeUsuarioNuevas(String nombreUsuario) — Muestra las entradas adquiridas por el usuario y aún no consumidas. +List<Entrada> listarTodasLasEntradasDelUsuario(String nombreUsuario) — Muestra todo el historial de entradas de un usuario.
- +void anularEntrada(int codigoEntrada, String contraseña) — Cancela una entrada, validando contraseña y reglas de cancelación.
- +void cambiarEntradaDeSede(int codigo, String nombreNuevaSede) — Reubica una entrada a otra sede disponible para la misma función.
- +double costoDeEntrada(String nombreUsuario, int codigoEspectaculo) — Calcula el costo que pagará el usuario por una entrada (incluye recargos).
- +double consultarValorDeEntrada(String nombreEspectaculo, String nombreSede,String nombreSector) — Devuelve el precio de base más los incrementos de un sector en una sede.
- +double calcularTotalRecaudado(int codigoEspectaculo) — Suma lo facturado de todas las entradas vendidas de un espectáculo.
- +void agregarFuncion(String nombreEspectaculo, String fecha, String sede, double precioBase)

2. TAD Usuario

Atributos

- +nombreUsuario : String
- +nombre : String
- +apellido : String
- +contraseña : String
- +set<Entrada> entradas

Operaciones

- +void autenticar() — Verifica que la contraseña coincida
- con la almacenada.
- +void comprarEntrada(String nombreEspectaculo) —
- Añade una entrada a su listado personal.
-
-

Relaciones

- Un usuario “tiene” muchas Entrada.
- El sistema mantiene un mapa de usuarios por nombreUsuario.

3. TAD Espectaculo

Atributos

- +codigoEspectaculo : int
- +nombreEspectaculo : String
- +list<Funcion> funciones

Operaciones

-
- +boolean:agregarFuncionAlEspectaculo(localDate fecha,idouble precio,String sede)
- agrega una nueva funcion al espectaculo.

Relaciones

- Un espectáculo “tiene” muchas Funcion.
- El sistema mantiene una lista de espectáculos.

4. TAD Funcion

Atributos

- +Sede sede
- +fecha : Date
- +precioBase: double

Operaciones

- (ninguna específica: se utiliza al vender y calcular precios)

Relaciones

- Cada función pertenece a un Espectaculo.
- Cuando se crea una entrada, se referencia una función.

5. TAD Entrada

Atributos

- +codigo : int
- +String:nombreUsuario
- +String:
- nombreEspectaculo
- +Date:fechaFuncion
- +precio : double
- +idSector : Sector
  +ubicacion

Operaciones

- (ninguna adicional: sus datos son consultados/cancelados)

Relaciones

- Una entrada referencia a un Usuario, un Espectaculo, una Funcion y un Sector.

6. Clase abstracta Sede

Atributos

- +nombre : String
- +capacidad: int
- +direccion: String
- +list<Sector>sect ores

Operaciones

- +tipoSede(): String
- — Devuelve el tipo concreto de sede (Estadio, Teatro…).

Relaciones

- Es base de las clases EstadioDeFutbol, MiniEstadio y Teatro.
- Cada sede contiene múltiples Sector.

6.1. Subclase EstadioDeFutbol

Atributos

- +precioFijo : double

Operaciones

- (hereda tipoSede())

6.2. Subclase MiniEstadio

Atributos

- +puestosComida : int
- +puestosMerchandising: int
- +cantidadDePuestos : int

Operaciones

- (hereda tipoSede())

6.3. Subclase Teatro

Atributos

-

Operaciones

- (hereda tipoSede())

7. Clase abstracta Sector

Atributos

- +nombre : String

Operaciones

- +darNombre(): String

- — Retorna el nombre del sector. .
- +calcularPrecio(precioBase: double): double — Aplica el
- recargo y devuelve el precio final para el sector.
-
-

Relaciones

- Es base de Campo y Platea.
- Una Sede agrupa múltiples instancias de Sector.

7.1. Subclase Campo

Atributos

- +numeroDeAsiento : int

Operaciones

- +calcularPrecio(precioBase: double): double

7.2. Subclase Platea

Atributos

- +tipoPlatea : enum
- +asientosPorFila : int
- +porcentajeAdicional : double
- +filas : int

Operaciones

- +calcularPrecio(precioBase: double): double —
- Aplica recargo adicional específico de platea.
- #darPorcentajeIncremento(): double — devuelve
- el porcentaje de recargo concreto.

Interfaz de Ticketek:

datos:
usuarios
HashMap<String nombreUsuario, Usuario>
//Mapa que asocia nombre de usuario(email) con su objeto Usuario. Permite búsquedas rápidas por nombre de usuario.
espectaculos
linkedList<Espectaculo>espectaculos
//Lista de todos los espectáculos registrados en el sistema.
sedes
linkedList<Sede>sedes
//Lista de todas las sedes donde se pueden realizar funciones.

métodos: 1. void registrarSede(String nombreSede, String direccion,int capacidad) // Añade una nueva sede (como teatro, estadio, etc.) al sistema. //sedes → se agrega un nuevo objeto Sede.
2. void registrarUsuario(String nombreUsuario, String nombre, String apellido, String contraseña) // Crea un nuevo usuario y lo almacena. usuarios → se guarda en el mapa con nombreUsuario como clave y el Usuario como valor.
3. void registrarEspectaculo(String nombreEspectaculo) //registra un nuevo espectáculo y lo agrega al sistema. //espectaculos → se añade un nuevo objeto Espectaculo.
4. void venderEntrada(String nombreUsuario, String nombreEspectaculo, String nombreSede, int numAsiento, String contraseña) //Vende una entrada para un espectáculo en una sede y función específica. Valida: que //el usuario exista y su contraseña sea correcta, que haya disponibilidad de asiento.
5. set<Sede> listarSedesDeEspectaculo(String nombreEspectaculo) // Devuelve una lista de sedes donde se representa el espectáculo dado. Devuelve //set<Sede> con todas las sedes en las que aparece alguna función del espectáculo. // utilizada: espectaculos y sedes (para cruzar datos).
6. List<Entrada> listarEntradasDeUsuarioNuevas(String nombreUsuario) //Devuelve todas las entradas no usadas (aún válidas) del usuario.

// List<Entrada> con las entradas activas. //utilizada: usuarios.get(nombreUsuario).entradas.
7. List<Entrada> listarTodasLasEntradasDelUsuario(String nombreUsuario) //Devuelve todas las entradas del historial del usuario. List<Entrada> con todas las //entradas del usuario (usadas o no). //utilizada: usuarios.
8. void anularEntrada(int codigoEntrada, String contraseña) //Cancela una entrada si se cumplen las condiciones: contraseña válida del usuario //se elimina o marca como anulada en la lista de entradas del Usuario.
9. void cambiarEntradaDeSede(int codigoEntrada, String nuevaSede) //Permite cambiar la sede de una entrada a otra sede para la misma función. //Modifica la entrada correspondiente dentro del usuario.
10. double costoDeEntrada(String nombreUsuario, int codigoEspectaculo) //Calcula el costo que debe pagar un usuario por una entrada a un espectáculo. // recargos especiales devuelve: el valor final en double. utilizada: usuarios, //espectaculos.
11. double consultarValorDeEntrada(String nombreEspectaculo, String nombreSede,String nombreSector) // Devuelve el valor base + recargos de una entrada para un sector determinado de //una sede.el precio calculado (double). utilizada: Sector.calcularPrecio(precioBase).
12. double calcularTotalRecaudado(int codigoEspectaculo) //Suma el total de dinero recaudado por todas las entradas vendidas del espectáculo. //devuelve: double con el total. Recorre entradas asociadas al espectáculo.
13.void agregarFuncion(String nombreEspectaculo, String fecha, String sede, double precioBase)// agrega una nueva funcion al espectaculo

Aclaracion!! Todos los TADs tienen sus respectivos constructores y toString() ,geters y seters de ser necesarios y sus metodos abstractos.

Diagrama de Clase: