**PRÁCTICA 1**

**UDManager**

**INTEGRANTES**

Fabián Andrés Hurtado Arango

Christian Bustos Betancur

José Mauricio Toscano Aguas

Jesús Daniel Pérez Petro

Grupo 02 - Equipo 01

**ASIGNATURA**

Programación Orientada a Objetos

Jaime Alberto Guzmán Luna

Universidad Nacional de Colombia

Sede Medellín

2024

**Índice**

**Descripción general de la solución.**

Se aclara previamente que el programa es de uso de una persona. Esta persona será un administrador, no un cliente.

1. **Problemas y Necesidades**

* Problemas Identificados:

**Gestión ineficiente de las instalaciones deportivas:**

Conflictos en los horarios de reserva de canchas, piscinas y otros espacios deportivos.

Dificultad para coordinar eventos deportivos y musicales, lo que genera solapamientos y problemas logísticos.

**Pérdida de ingresos potenciales:**

Espacios subutilizados debido a la falta de un sistema eficiente para gestionar las reservas.

Sistema de suscripciones poco eficiente y beneficios difíciles de aplicar manualmente.

Procesos manuales que ralentizan la inscripción en actividades formativas, torneos, reservas, compra de boletos para eventos y compra de suscripciones.

**Falta de organización en la gestión financiera:**

Dificultades para controlar pagos, suscripciones y ventas, lo que impacta en la transparencia y el flujo de caja.

* Necesidades Identificadas:

Centralizar la gestión de reservas, asegurando la no repetición de estas mismas, así evitando conflictos logísticos y administrativos.

Automatizar la inscripción a deportes formativos, automatizando los procesos de asignación de categorías.

Permitir la compra de implementos deportivos, ya sean relacionados al formativo o no, en la tienda de la unidad deportiva.

Tener una herramienta para organizar torneos y eventos, que permita la asignación eficiente de recursos y espacios.

Gestionar la venta de boletos y pagos en un sistema integrado que soporte flujos seguros y claros para administradores y clientes.

1. **Requisitos Funcionales y No Funcionales:**

* Requisitos Funcionales:

**Registrar y Cancelar Reservas de instalaciones:**

Permitir la consulta de la disponibilidad de las instalaciones deportivas.

Facilitar el proceso de reserva mediante un “formulario” que incluya datos como la fecha, hora, tipo de instalación y propósito de uso.

Permitir la gestión administrativa de reservas, incluyendo la cancelación de estas.

**Registrar Inscripciones a Deportes Formativos:**

Disponer de un catálogo de actividades deportivas formativas con información detallada (tipo de deporte, horarios, entrenadores, costos, etc.).

Facilitar la inscripción de jóvenes, mediante formularios con datos personales, preferencias y selección de planes de suscripción.

Permitir a los administradores gestionar las inscripciones, verificando cupos disponibles.

Incluir opciones de venta de equipamiento deportivo como parte del proceso de inscripción.

**Registrar la Creación de Torneos:**

Proporcionar herramientas para que los administradores configuren torneos deportivos.

Permitir la asignación de equipos participantes, horarios y personal de apoyo.

Integrar la posibilidad de registrar reglas específicas del torneo según el tipo de deporte.

**Registrar la Creación de Eventos no deportivos:**

Facilitar la creación y configuración de eventos no deportivos, como conciertos, ferias o exposiciones, que utilicen las instalaciones deportivas.

Permitir la configuración de espacios para el evento, definiendo áreas específicas (escenarios, áreas VIP, zonas de comida, etc.) según las necesidades del organizador.

Gestionar la venta de boletos para los eventos.

Incluir herramientas para coordinar al personal de apoyo necesario para la logística del evento.

**Registrar los “pagos”**

Registrar y almacenar información sobre los pagos realizados, asociándolos a cada cliente, reserva o inscripción.

Permitir diferentes medios de pago para los clientes como efectivo, transferencias, tarjetas de crédito, débito y los procesos que estas conllevan.

Generar comprobantes de pago.

* Requisitos No Funcionales:

El sistema debe ser capaz de manejar múltiples instalaciones y actividades simultáneamente.

Las reservas y eventos creados deben ser almacenados y tenidos en cuenta en futuras acciones.

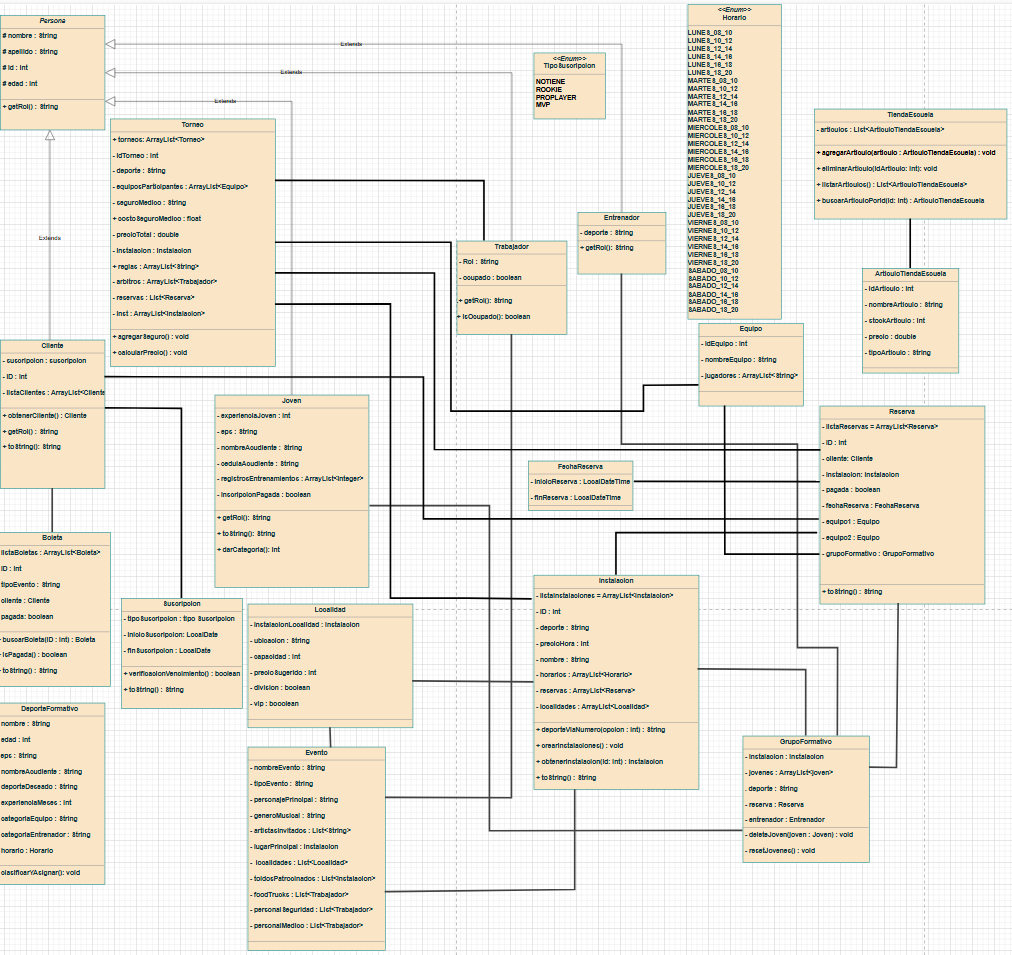
La interfaz debe ser intuitiva y fácil de usar.

La aplicación debe estar preparada para crecer a medida que la unidad deportiva amplíe su oferta de instalaciones o actividades. Por ejemplo, si se agregan nuevas canchas, piscinas o espacios para eventos, el sistema debe poder gestionarlas sin necesidad de grandes modificaciones.

**Descripción del diseño estático del sistema en la especificación UML.**

UML

[**https://app.diagrams.net/#G1dAHE1gTlgRmYuLyE7LoVl1xUdC21Mvh9#%7B%22pageId%22%3A%22c4acf3e9-155e-7222-9cf6-157b1a14988f%22%7D**](https://app.diagrams.net/#G1dAHE1gTlgRmYuLyE7LoVl1xUdC21Mvh9#%7B%22pageId%22%3A%22c4acf3e9-155e-7222-9cf6-157b1a14988f%22%7D)



**Descripción de las clases**

**1. Persona**

* **Propósito:** Es una clase base que representa a cualquier persona en el sistema. Contiene atributos comunes como nombre, apellido e id.
* **Relaciones:**
  + Es la clase padre de Cliente, Trabajador, Entrenador y Joven, lo que significa que todas estas clases heredan sus atributos y métodos.

**2. Cliente**

* **Propósito:** Representa a una persona que usa el sistema para realizar reservas, inscribirse en torneos, comprar boletas y gestionar su suscripción.
* **Relaciones:**
  + Hereda de Persona.
  + Se relaciona con Suscripción (1:1), ya que cada cliente puede tener una única suscripción activa.
  + Se relaciona con Boleta (1:N), porque un cliente puede comprar varias boletas.
  + Se relaciona con Reserva (1:N), porque un cliente puede realizar múltiples reservas.

**3. Suscripción**

* **Propósito:** Maneja el sistema de membresías del cliente, definiendo diferentes niveles de suscripción que afectan beneficios como descuentos y acceso a torneos.
* **Relaciones:**
  + Relación con Cliente (1:1), ya que un cliente solo puede tener una suscripción activa a la vez.
  + Relación con TipoSuscripción (1:1), ya que cada suscripción debe estar clasificada en un tipo específico (ej. ROOKIE, MVP).

**4. Boleta**

* **Propósito:** Representa una entrada que un cliente puede comprar para asistir a eventos o torneos.
* **Relaciones:**
  + Relación con Cliente (N:1), porque un cliente puede comprar varias boletas.

**5. TipoSuscripción (Enum)**

* **Propósito:** Enumera los diferentes tipos de suscripción disponibles (NOTIENE, ROOKIE, PROPLAYER, MVP). Cada tipo define sus beneficios, descuentos y restricciones.
* **Relaciones:**
  + Asociado a Suscripción (1:1), ya que cada suscripción pertenece a un tipo específico.

**6. Torneo**

* **Propósito:** Representa la organización de un torneo deportivo, incluyendo reglas, equipos participantes, instalación asignada y árbitros.
* **Relaciones:**
  + Relación con Equipo (1:N), ya que un torneo involucra varios equipos.
  + Relación con Instalación (1:1), porque un torneo se juega en una única instalación.
  + Relación con Reserva (1:N), ya que el torneo puede generar múltiples reservas.
  + Relación con Trabajador (1:N), ya que se asignan árbitros al torneo.

**7. Equipo**

* **Propósito:** Representa un equipo participante en un torneo. Contiene su nombre y lista de jugadores.
* **Relaciones:**
  + Relación con Torneo (N:1), ya que un equipo pertenece a un torneo específico.
  + Relación con Reserva (N:1), si participa en una reserva programada para un partido.

**8. Evento**

* **Propósito:** Representa un evento no deportivo (ej. un concierto o festival). Administra información como nombre, artista principal, localidades y seguridad.
* **Relaciones:**
  + Relación con Localidad (1:N), porque un evento tiene varias localidades (tribunas, gramilla, VIP).
  + Relación con Instalación (1:1), ya que cada evento se lleva a cabo en una única instalación.
  + Relación con Trabajador (1:N), para asignar personal de seguridad y paramédicos.

**9. Localidad**

* **Propósito:** Representa diferentes secciones dentro de una instalación para eventos, como tribunas (norte, sur, oriental, occidental) y gramilla.
* **Relaciones:**
  + Relación con Instalación (N:1), porque una instalación puede contener varias localidades.
  + Relación con Evento (N:1), ya que un evento tiene varias localidades habilitadas.

**10. Trabajador**

* **Propósito:** Representa a cualquier trabajador en el sistema, como árbitros, personal de seguridad o médicos.
* **Relaciones:**
  + Hereda de Persona.
  + Relación con Evento (N:N), ya que se asignan trabajadores como seguridad o paramédicos a los eventos.
  + Relación con Torneo (N:N), para la asignación de árbitros.

**11. Entrenador**

* **Propósito:** Representa a los entrenadores de los grupos formativos.
* **Relaciones:**
  + Hereda de Trabajador.
  + Relación con GrupoFormativo (1:N), ya que un entrenador puede dirigir varios grupos.

**12. Joven**

* **Propósito:** Representa a los jóvenes inscritos en los programas formativos. Contiene datos como edad, EPS y registros de entrenamiento.
* **Relaciones:**
  + Hereda de Persona.
  + Relación con GrupoFormativo (N:1), ya que cada joven pertenece a un único grupo formativo.

**13. GrupoFormativo**

* **Propósito:** Organiza a los jóvenes en grupos de entrenamiento según el deporte y su nivel de experiencia.
* **Relaciones:**
  + Relación con Joven (1:N), ya que cada grupo tiene varios jóvenes.
  + Relación con Entrenador (1:1), ya que cada grupo tiene un entrenador asignado.
  + Relación con Instalación (1:1), ya que cada grupo entrena en una instalación específica.

**14. Instalación**

* **Propósito:** Representa los espacios físicos donde se realizan torneos, entrenamientos y eventos. Puede ser una cancha, piscina o coliseo.
* **Relaciones:**
  + Relación con Reserva (1:N), ya que una instalación puede estar reservada múltiples veces.
  + Relación con Torneo (N:1), porque cada torneo usa una única instalación.
  + Relación con Evento (N:1), porque cada evento se desarrolla en una instalación.
  + Relación con Localidad (1:N), si la instalación tiene diferentes secciones.

**15. Reserva**

* **Propósito:** Maneja la reserva de una instalación, ya sea por un cliente, torneo o evento.
* **Relaciones:**
  + Relación con Cliente (1:1), si es una reserva particular.
  + Relación con Instalación (1:1), porque cada reserva se hace sobre una instalación.
  + Relación con Equipo (1:N), si se reserva para un torneo con varios equipos.

**16. TiendaEscuela**

* **Propósito:** Representa la tienda de la escuela deportiva, donde se pueden comprar artículos deportivos.
* **Relaciones:**
  + Relación con ArticuloTiendaEscuela (1:N), porque la tienda tiene múltiples artículos a la venta.

**17. ArticuloTiendaEscuela**

* **Propósito:** Representa un artículo deportivo en la tienda, como uniformes o balones.
* **Relaciones:**
  + Relación con TiendaEscuela (N:1), ya que cada artículo pertenece a la tienda.

**18. FechaReserva**

* **Propósito:** Maneja la fecha de inicio y fin de una reserva.
* **Relaciones:**
  + Relación con Reserva (1:1), ya que cada reserva tiene una fecha asociada.

**Descripción de la implementación de características de programación orientada a objetos en el proyecto.**

**Implementación de Clase Abstracta y Método Abstracto:**

Texto

Descripción generada automáticamente

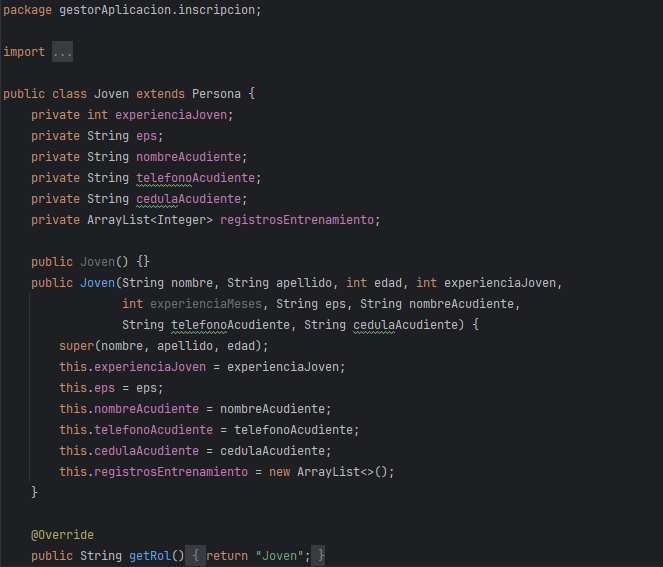
Como se visualiza en la captura, esta clase y método abstracto se encuentran en la carpeta entidades, que a su vez se encuentra en la carpeta gestorAplicacion.

El uso que se le da a esta clase se ve reflejado en la herencia de la clase Trabajador, Cliente y Joven. El método abstracto getRol() sirve para consultar el rol que ocupa dicha persona dentro de la Unidad Deportiva (Árbitro, Seguridad, Médico, Paramédico).

**Interfaces:**

No se agregaron interfaces debido a que no se vio la necesidad de emular la herencia múltiple.

**Herencia:**



Como se visualiza en la captura, esta clase, que implementa la herencia se encuentran en la carpeta entidades, que a su vez se encuentra en la carpeta gestorAplicacion.

Su uso es para reutilizar código como los atributos de nombre y apellido (ejemplos).

**Ligadura Dinámica:** Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

La primera captura se ubica en el main, método asignarPersonalSeguridad, la segunda también se encuentra en el main, en el método crearEvento(). Ambas ayudan a ejemplificar el uso de la ligadura dinámica en el proyecto, ya que en realidad ambas capturas ejemplifican el mismo caso.

En el método crearEvento, se asignan objetos de tipo Trabajador (que heredan de la clase Persona) al evento como personal de seguridad. Aunque la lista contiene objetos de tipo Trabajador, el método getRol() es resuelto dinámicamente en tiempo de ejecución, según la implementación de getRol() en la clase Trabajador.

El método getRol() en Trabajador sobrescribe la definición en la clase base Persona. Cuando el programa ejecuta set.getRol() Se utiliza la implementación específica del método getRol() en la clase Trabajador, a pesar de que la referencia es del tipo Persona. Esto es un caso claro de **ligadura dinámica** porque el método invocado depende del tipo real del objeto en tiempo de ejecución y no del tipo estático de la variable.

**Ligadura estática:**

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Ambos fragmentos de código se encuentran en el Main, el primero está ubicado en el switch principal del programa, y hace un llamada a crearTorneos() que se encuentran en esta misma clase pero fuera del método main.

La ligadura estática asegura que el método crearTorneos reciba exactamente lo que necesita para funcionar:

Una lista de médicos (ArrayList<Trabajador>), una lista de árbitros (ArrayList<Trabajador>) y una lista de instalaciones (ArrayList<Instalacion>).

Esto permite mantener un flujo claro y organizado de datos entre las distintas partes del programa. No necesitas resolver dinámicamente qué datos enviar ni cómo recibirlos, ya que el método espera parámetros de tipos bien definidos.

**Atributos y Métodos de clase:**

Texto

Descripción generada automáticamente

Como se visualiza en la captura, este atributo y método de clase se encuentran en la carpeta pagos, que a su vez se encuentra en la carpeta gestorAplicacion.

El uso de ArrayList de Boletas estático es para guardar las boletas creadas de manera global, indiferente de la instancia de esta clase que se cree.

El uso del método estático buscarBoleta se ve en la funcionalidad de Taquilla – Pagos, a la hora de seleccionar la opción 2 de “Pago Boleta”. Es estático para poder importarlo importarlo al Main de manera fácil.

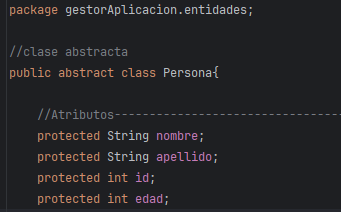
**Uso de Constante y de Enumeraciones:**



Ambos se encuentran en el Enum ubicado en la carpeta pagos, que a su vez se encuentra en la carpeta gestorAplicacion.

En la enumeración TipoSuscripcion, se definen cuatro constantes que representan diferentes tipos de suscripción dentro del programa. Cada una de estas constantes tiene atributos específicos que determinan sus beneficios y restricciones.

**Encapsulamiento:**



Texto

Descripción generada automáticamente

La primera captura se ubica en la carpeta entidades – carpeta gestorAplicacion. La segunda en la carpeta torneo – carpeta gestorAplicacion.  
  
Este encapsulamiento permite controlar el acceso a los atributos y métodos de una clase, asegurando la integridad y seguridad de los datos. **public** permite el acceso desde cualquier clase, lo que facilita la interacción global pero reduce el control. **private** restringe el acceso solo dentro de la misma clase, protegiendo los datos y obligando a usar métodos *getters* y *setters* para manipularlos. **protected** permite el acceso dentro del mismo paquete y por sus subclases (herencia), proporcionando un equilibrio entre restricción y reutilización de código.

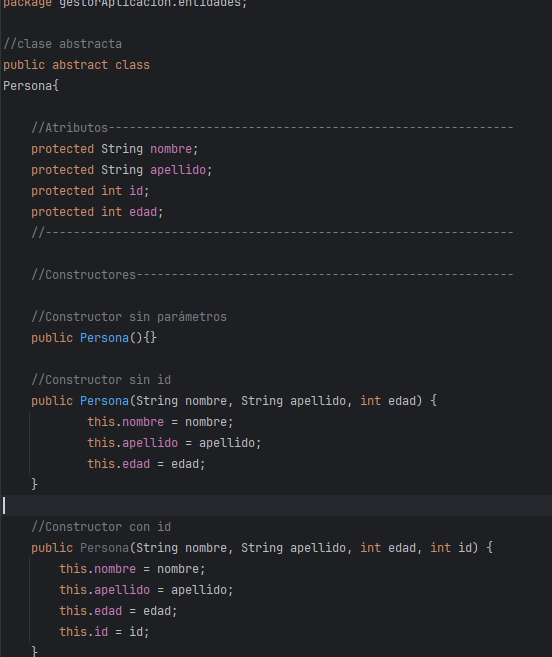
**Sobrecarga de Métodos, Constructores y uso del this:**

Texto

Descripción generada automáticamente

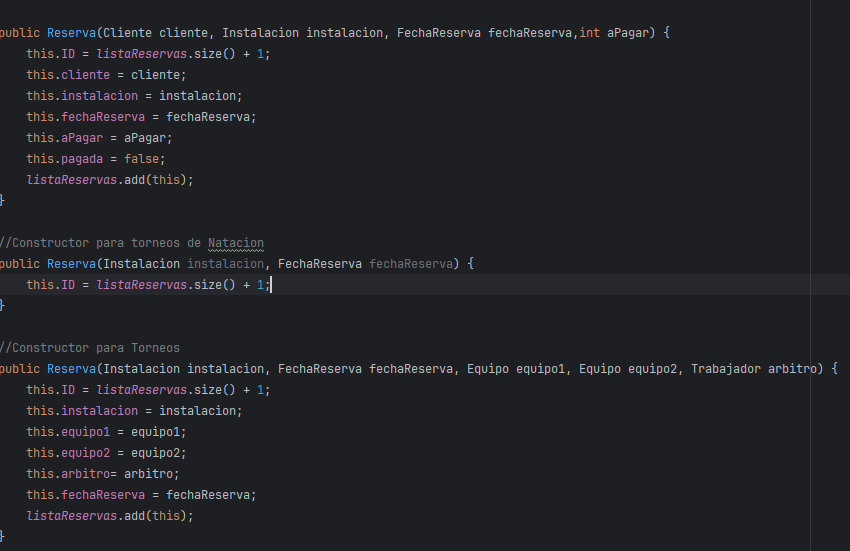
Aquí se muestra una sobrecarga del método toString() en la clase reserva ubicada en la carpeta reservas – carpeta gestorAplicacion.

Esta permite imprimir datos de diferentes tipos de reservas solicitados en el Main.



En la página anterior se muestra una sobrecarga de constructores y el uso del this ubicado en la clase abstracta Persona, carpeta entidades – carpeta gestorReservas.

Esta sirve para no tener errores a la hora de llamar al super() desde las clases que heredan de Persona.



En la anterior captura se evidencia la sobrecarga de constructores y el uso del this ubicado en la clase Reserva en la carpeta reservas – carpeta gestorAplicacion.

Su uso es permitir la creación de reservas desde distintas funcionalidades, como para torneos de Natación, Torneos en general y demás.

**Descripción de cada una de las 5 funcionalidades implementadas.**

**1. reservasUI**

Descripción General  
Esta funcionalidad gestiona la creación y la visualización de reservas de las diferentes instalaciones deportivas (o áreas) disponibles.

Permite:

* Crear una nueva reserva pidiendo al usuario datos como el ID del cliente, el deporte y la instalación, así como la fecha y horario de la reserva.
* Visualizar una reserva existente a partir de su ID.

Flujo de 3 Interacciones

1. Interacción 1: Búsqueda y validación del Cliente
   * El usuario elige la opción “Crear Reserva”.
   * El sistema solicita el ID del Cliente.
   * Se llama al método estático Cliente.obtenerCliente(idCliente).
   * Si el cliente existe, se obtiene un objeto de la clase Cliente; de lo contrario, se informa que no se encontró.
   * Objetos involucrados: Scanner (para leer datos), Cliente (búsqueda en la lista estática).
2. Interacción 2: Selección de Instalación y cálculo de costo
   * Una vez validado el cliente, el sistema solicita el deporte (Fútbol, Baloncesto, Natación o Voleibol).
   * Utiliza el método estático Instalacion.deporteViaNumero(opcion) para convertir la elección en el String correspondiente.
   * Muestra las instalaciones disponibles de ese deporte llamando a Instalacion.getListaInstalaciones() y filtrando por instalacion.getDeporte().
   * El usuario elige la instalación por ID. Luego el sistema pide la fecha y horas de inicio y fin de la reserva.
   * Se calcula el costo total multiplicando el número de horas (Duration.between(inicioHora,finHora).toHours()) por el precio por hora de la instalación (instalacionEscogida.getPrecioHora()).
   * Objetos involucrados:
     + Instalacion para obtener/filtrar instalaciones,
     + Scanner para leer la selección y las fechas/horas,
     + Duration y LocalDateTime para calcular la duración de la reserva y así el monto a pagar.
3. Interacción 3: Creación y/o Visualización de la Reserva
   * Creación: Se instancia un nuevo objeto Reserva, pasando el Cliente, la Instalacion, la FechaReserva (que contiene LocalDateTime de inicio y fin) y el costo calculado. Se agrega esta reserva a la lista de reservas de la instalación.
   * El sistema muestra el resumen de la reserva y el monto a pagar.
   * Visualización: Si el usuario elige “Visualizar Reserva” en el submenú, se pide el ID de la reserva y se llama Reserva.buscarReserva(idReserva). Se imprime la información de la reserva si existe.
   * Objetos involucrados:
     + Reserva (creación de la instancia y asignación a la lista estática),
     + Cliente (referencia guardada dentro de Reserva),
     + FechaReserva (encapsula las horas de inicio/fin).

**2. gestionarInscripcion**

Descripción General  
Permite inscribir a un joven en un deporte formativo, definiendo su experiencia, la categoría a la que pertenece, el horario de entrenamiento y la posibilidad de comprar equipamiento (uniformes, balones, etc.) en la TiendaEscuela.

Flujo de 3 Interacciones

1. Interacción 1: Captura de datos de inscripción
   * El sistema solicita información sobre el joven: nombre, apellido, documento, edad, EPS, nombre y teléfono del acudiente, etc.
   * Con esos datos, se instancia un objeto de la clase auxiliar DeporteFormativo (que solo encapsula la lógica de clasificación), asignando todos los atributos.
   * Objetos involucrados: Scanner (lectura de datos), DeporteFormativo (almacena temporalmente los datos de inscripción).
2. Interacción 2: Clasificación y creación de objetos del dominio
   * Se ejecuta df.clasificarYAsignar(), que determina la categoría del joven (Categoría 1, 2 o 3) y el horario (mañana o tarde) según la experiencia y la edad.
   * Se instancian los objetos relevantes:
     + Joven: con todos sus atributos (nombre, apellido, edad, EPS, acudiente…).
     + Entrenador: se crea un entrenador “simulado” según la categoría.
     + GrupoFormativo: que indica el deporte, la instalación seleccionada y el entrenador asignado.
   * Objetos involucrados:
     + DeporteFormativo para determinar la categoría,
     + Joven, Entrenador, GrupoFormativo (se crea y asocia al joven).
3. Interacción 3: Interacción con la Tienda para equipamiento
   * El sistema pregunta si se desea comprar equipamiento relacionado con el deporte.
   * Se listan artículos de TiendaEscuela filtrados por articulo.getTipoArticulo() (por ejemplo, todos los artículos cuyo tipo sea “Futbol” si el joven se inscribió en fútbol).
   * El usuario selecciona un ID de artículo. El sistema busca el artículo (tienda.buscarArticuloPorId(id)) y verifica disponibilidad de stockArticulo.
   * De realizarse la compra, se reduce el stock y se muestra un mensaje de confirmación.
   * Objetos involucrados:
     + TiendaEscuela (almacena la lista de ArticuloTiendaEscuela),
     + ArticuloTiendaEscuela (representa el ítem comprado),
     + Scanner para la selección del artículo.

**3. crearTorneos**

Descripción General  
Facilita la creación de un nuevo torneo de un deporte específico (Fútbol, Baloncesto, Natación o Voleibol). Involucra la selección de la instalación, la configuración de reglas personalizadas (por ejemplo, la duración del partido o la altura de la malla) y la inscripción de equipos.

Flujo de 3 Interacciones

1. Interacción 1: Configuración inicial del Torneo y selección de Deporte
   * El usuario ingresa sus datos básicos (nombre, apellido, edad, ID) y elige el deporte.
   * Se crea un objeto Torneo y se asigna la disciplina llamando a torneo.setDeporte(deporteTorneo).
   * Se buscan las instalaciones posibles para ese deporte con torneo.getInstalaciones(deporteTorneo, inst).
   * El usuario elige la instalación que albergará el torneo.
   * Objetos involucrados:
     + Cliente (representa el organizador del torneo, aunque se crea ad-hoc),
     + Torneo (se instancia y configura).
2. Interacción 2: Definición de Reglas y Registro de Equipos
   * Dependiendo del deporte, el sistema muestra opciones de configuración (p.ej. número de sustituciones, duración en fútbol, sets en voleibol, relevos en natación, etc.).
   * El usuario define las reglas y el sistema las guarda en torneo.setReglas(reglasX).
   * Se procede a inscribir equipos: se solicitan nombres de equipos y jugadores, y se crean objetos Equipo.
   * Cada Equipo se registra en torneo.setEquiposParticipantes(...).
   * Objetos involucrados:
     + Torneo (guarda reglas y la lista de Equipo),
     + Equipo (almacena nombre y jugadores),
     + Scanner para la recolección de los datos de reglas y jugadores.
3. Interacción 3: Asignación de recursos (médicos, árbitros) y creación de reservas
   * El sistema asigna uno o más médicos (objetos Trabajador con rol “Medico”) a cada equipo para valoración.
   * De forma aleatoria, se simula un jugador que no pasa la valoración y se solicita un nuevo jugador para reemplazarlo.
   * Se ofrece un seguro médico opcional. Si se elige, el torneo agrega ese costo (torneo.agregarSeguro(...)).
   * Se seleccionan 2 árbitros aleatorios de la lista arbitrosTodos y se marcan como ocupados.
   * Finalmente, se crea la(s) Reserva(s) correspondientes a los partidos. Cada Reserva asocia la Instalacion, un rango de horario y los equipos que se enfrentarán.
   * Se generan Boleta(s) para el torneo, que pueden luego pagarse en la funcionalidad de “Taquilla”.
   * Objetos involucrados:
     + Trabajador (médicos y árbitros con diferentes roles),
     + Reserva (para los horarios de los partidos),
     + Boleta (emisión de tiquetes del torneo).

**4. crearEvento**

Descripción General  
Crea un evento no deportivo (Concierto o Festival), configurando nombre, artista principal, invitados, localidades (tribunas), servicios de comida (FoodTrucks), toldos patrocinados, personal de seguridad/paramédico y la reserva base donde se realizará el evento.

Flujo de 3 Interacciones

1. Interacción 1: Captura de datos del cliente y tipo de Evento
   * El sistema solicita datos personales (nombre, apellido, edad, ID) del organizador.
   * Se pide el nombre del evento, el tipo de evento (1. Festival, 2. Concierto) y el artista principal.
   * Se crea un objeto Evento y se ajustan sus atributos (tipoEvento, nombreEvento, personajePrincipal, etc.).
   * Si es Concierto, se pide adicionalmente el género musical; además, se puede registrar artistas invitados.
   * Objetos involucrados:
     + Cliente (organizador en ese momento),
     + Evento (creación y set de atributos fundamentales).
2. Interacción 2: Configuración de localidades y servicios adicionales
   * Se van configurando Localidad(es): se pide capacidad, se activa o no la zona VIP, se divide o no para menores, etc.
   * El sistema muestra la posibilidad de seleccionar toldos patrocinados (1 a 3) y FoodTrucks (hasta 5).
   * Se asigna personal de seguridad (seguridadDisponible) y personal médico (paramedicosDisponibles) en número suficiente para el evento (por ejemplo, 10 guardias y 3 paramédicos).
   * Objetos involucrados:
     + Localidad (objeto que modela cada tribuna/zona),
     + Instalacion (si los toldos o food trucks son también instalaciones),
     + Trabajador (seguridad y paramédicos).
3. Interacción 3: Creación de la Reserva y emisión de Boletas
   * El usuario define la fecha y rango horario del evento (mínimo 6 horas).
   * Se crea un objeto Reserva asociando la Instalacion principal (por ejemplo, un estadio), las fechas y la referencia al evento.
   * Se generan las boletas (Boleta) para ese evento, que luego pueden pagarse.
   * Se muestra un resumen de todo lo configurado: localidades, seguridad asignada, paramédicos, toldos, food trucks, etc.
   * Objetos involucrados:
     + Reserva (encapsula la instalación, la fecha/hora),
     + Boleta (refleja el costo base que se deberá pagar),
     + Evento (guarda la lista de Localidad, PersonalSeguridad, PersonalMedico, etc.).

**5. pagosUI**

Descripción General  
Gestiona toda la lógica de pagos en el sistema:

* Permite pagar reservas (ID de reserva),
* Pagar boletas para eventos,
* Administrar suscripciones (crear, renovar, cambiar de tipo).

Flujo de 3 Interacciones

1. Interacción 1: Pago de una Reserva por ID
   * El usuario elige “1: Pago via ID” y digita el ID de la reserva.
   * Se busca la reserva con Reserva.buscarReserva(idIngresado).
   * Si existe y no está pagada, se muestra la información junto al costo.
   * Se pregunta si se desea confirmar el pago:
     + Si se paga, se invoca reserva.setPagada() y se indica el descuento según la suscripción del cliente (reserva.getCliente().getSuscripcion().getTipoSuscripcion().getDescuento()).
   * Objetos involucrados:
     + Reserva (estado de pago, ID),
     + Cliente (aplicación de descuento de su Suscripción).
2. Interacción 2: Pago de Boleta de Evento
   * El usuario elige “2: Pago boleta de evento” y digita el ID de la boleta.
   * Se llama a Boleta.buscarBoleta(idBoleta) para encontrarla en la lista estática.
   * Si la boleta no ha sido pagada, el sistema muestra el precio y solicita la confirmación de pago.
   * De pagarla, se marca con boleta.setPagada(true) y se informa el resultado (costo total, vuelto, etc.).
   * Objetos involucrados:
     + Boleta (almacena el precio, ID, estado pagado/no pagado),
     + Cliente (quien aparece como titular en la boleta, aunque el pago puede ser por un tercero, la boleta está asociada al cliente dueño).
3. Interacción 3: Administración de Suscripciones
   * El usuario ingresa al submenú “Administrar Suscripción”.
   * Puede renovar su suscripción si aún la tiene activa o crear una nueva suscripción si no posee.
   * Se solicita el ID del cliente y se obtiene el objeto Cliente.
   * Elige un tipo de suscripción (ROOKIE, PROPLAYER, MVP), con su respectivo precio y beneficios (descuentos, reservas gratis, etc.).
   * Se actualizan los datos de la suscripción (suscripcion.setTipoSuscripcion(...)) y, si es renovación, se reajusta la fecha finSuscripcion.
   * Objetos involucrados:
     + Suscripcion (tipoSuscripcion, fechas de inicio/fin),
     + TipoSuscripcion (enum con precios y descuentos),
     + Cliente (vincula su suscripción).

**Diagrama de interacción:**

**->Probar códigos en la siguiente página:**

<https://mermaid.live/edit#pako:eNqdVb1u2zAQfpUDJ7tQUXQV0CxOgnoICsRwh0LLhTrbRGlSJam0bpCH6QN06tbVL9aTRNmy5d9qku7n-767I08vQtqcRCo8fSvJSLpVOHe4zAzwU6ALSqoCTYCpL9Ep23c8oDLTMaCv32DgyJN7Rj8dD_vBI63IBOo7xsYH1CiVNX3nYwOZRVeU8vbmpuFO4VMh1fqPgfcwGDlCBxY-K47S6id_REVRDmqWUQdtYCE-DRzjRoYUJlaziIAwvoWc9K7-w1ruzPP69_GEDUf0pPBUeslqyIBW3AQgH9a_uHaEgX0KZMjF0OEWpS6isQL94LQOw5lKciqs24_vlzEhTbIax5GMDUNncincKx0cgtrayIOn-fqvOQOzEfpQcgN6GLnyhTXqSZM_p_xOqzk1_d-C8Om4RH9seMc2UPlwN7Pq_biDe3AAZ4YwI7lAWMHCcqUDZdhq382UGfZRjp6wHIP1VZFdsBMq2vQRallqBGl9sPChTvPwBgpHrOLjaZB4Z1Iw9L29QAPZnMSk2-3kvlIVI5Kaa3gJ7oSANxDNERBYZHMluMh2rVzS5vYISWtmyi3jmFYQLMtj3ALn6HaBSHvanaqxVwy2ZSTn7D6yybeGmqa9uEcYzqBDJroAhotkX24z0aFpOWu-ziq8euVVM3D7SaeWnjvKsBlxs_Daw6Pyx5393F6xaL2yRZtLcUh33Y0W9_-6f6jHJheJWBIfNJXzr_SlMmciLGhJmUj5NUf3tRrPK8dhGexkZaRIgyspEc6W84VIZ8jaElEWXED7E25D-B_4xdplDHr9B0VMdks>

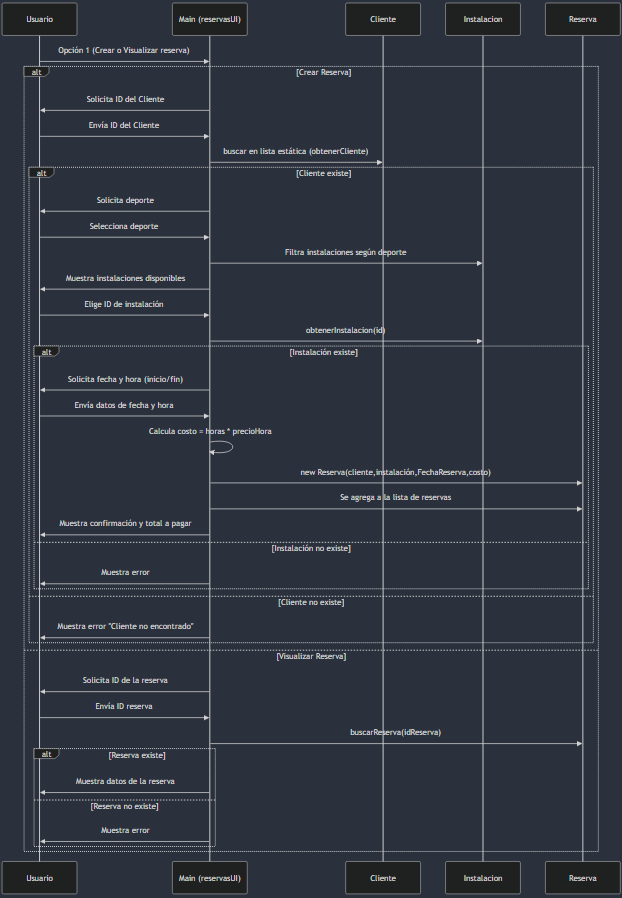
**Códigos con cada diagrama a continuación:**

**1.reservasUI :**

**Código:**

sequenceDiagram participant Usuario participant MainUI as Main (reservasUI) participant Cliente participant Instalacion participant Reserva Usuario->>MainUI: Opción 1 (Crear o Visualizar reserva) alt Crear Reserva MainUI->>Usuario: Solicita ID del Cliente Usuario->>MainUI: Envía ID del Cliente MainUI->>Cliente: buscar en lista estática (obtenerCliente) alt Cliente existe MainUI->>Usuario: Solicita deporte Usuario->>MainUI: Selecciona deporte MainUI->>Instalacion: Filtra instalaciones según deporte MainUI->>Usuario: Muestra instalaciones disponibles Usuario->>MainUI: Elige ID de instalación MainUI->>Instalacion: obtenerInstalacion(id) alt Instalación existe MainUI->>Usuario: Solicita fecha y hora (inicio/fin) Usuario->>MainUI: Envía datos de fecha y hora MainUI->>MainUI: Calcula costo = horas \* precioHora MainUI->>Reserva: new Reserva(cliente,instalación,FechaReserva,costo) MainUI->>Reserva: Se agrega a la lista de reservas MainUI->>Usuario: Muestra confirmación y total a pagar else Instalación no existe MainUI->>Usuario: Muestra error end else Cliente no existe MainUI->>Usuario: Muestra error "Cliente no encontrado" end else Visualizar Reserva MainUI->>Usuario: Solicita ID de la reserva Usuario->>MainUI: Envía ID reserva MainUI->>Reserva: buscarReserva(idReserva) alt Reserva existe MainUI->>Usuario: Muestra datos de la reserva else Reserva no existe MainUI->>Usuario: Muestra error end end

**Diagrama de Interacción reservasUI**

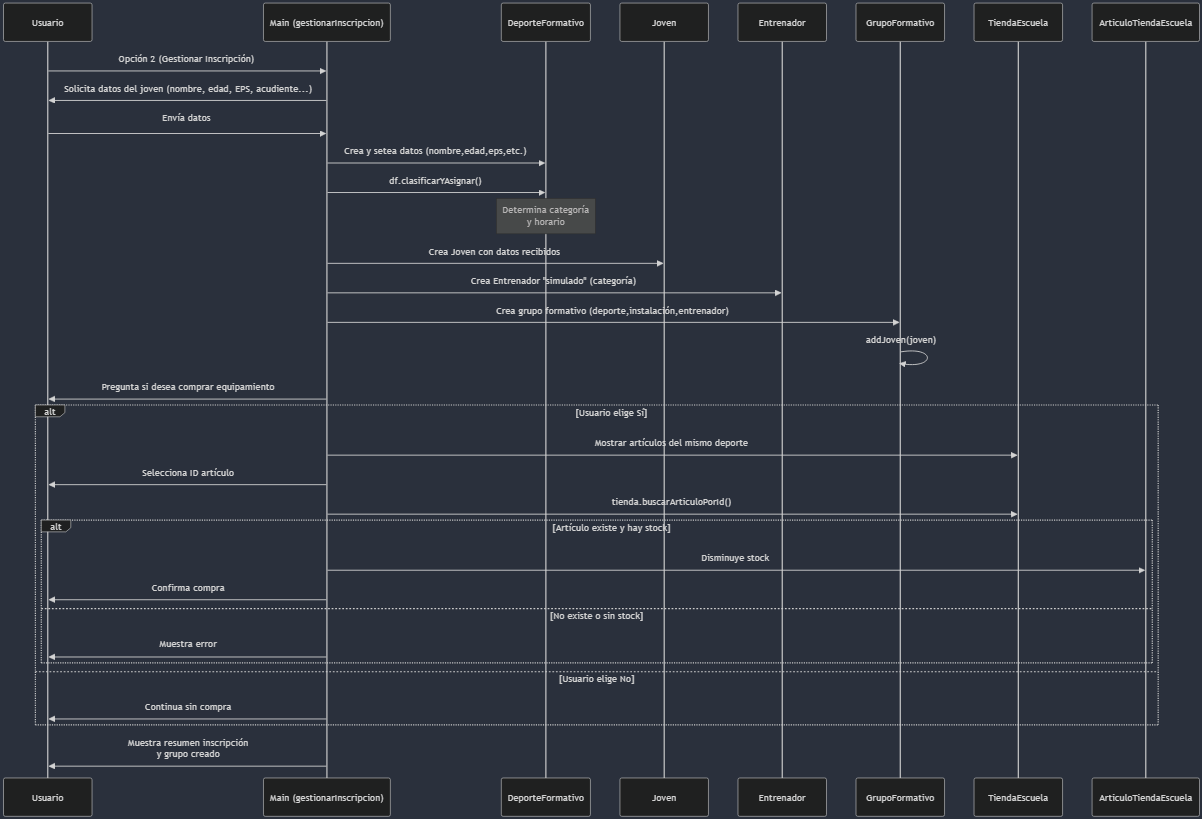


**2. gestionarInscripcion**

**->Código:**

sequenceDiagram participant Usuario participant MainUI as Main (gestionarInscripcion) participant DeporteFormativo participant Joven participant Entrenador participant GrupoFormativo participant TiendaEscuela participant ArticuloTiendaEscuela Usuario->>MainUI: Opción 2 (Gestionar Inscripción) MainUI->>Usuario: Solicita datos del joven (nombre, edad, EPS, acudiente...) Usuario->>MainUI: Envía datos MainUI->>DeporteFormativo: Crea y setea datos (nombre,edad,eps,etc.) MainUI->>DeporteFormativo: df.clasificarYAsignar() Note over DeporteFormativo: Determina categoría<br/>y horario MainUI->>Joven: Crea Joven con datos recibidos MainUI->>Entrenador: Crea Entrenador "simulado" (categoría) MainUI->>GrupoFormativo: Crea grupo formativo (deporte,instalación,entrenador) GrupoFormativo->>GrupoFormativo: addJoven(joven) MainUI->>Usuario: Pregunta si desea comprar equipamiento alt Usuario elige Sí MainUI->>TiendaEscuela: Mostrar artículos del mismo deporte MainUI->>Usuario: Selecciona ID artículo MainUI->>TiendaEscuela: tienda.buscarArticuloPorId() alt Artículo existe y hay stock MainUI->>ArticuloTiendaEscuela: Disminuye stock MainUI->>Usuario: Confirma compra else No existe o sin stock MainUI->>Usuario: Muestra error end else Usuario elige No MainUI->>Usuario: Continua sin compra end MainUI->>Usuario: Muestra resumen inscripción<br/>y grupo creado

**Diagrama de Interacción**



**3.crearTorneo**

**Código:**

sequenceDiagram

participant Usuario

participant MainUI as Main (crearTorneos)

participant Cliente

participant Torneo

participant Instalacion

participant Equipo

participant Trabajador as Medico/Arbitro

participant Reserva

participant Boleta

Usuario->>MainUI: Opción 3 (Crear Torneos)

MainUI->>Usuario: Solicita datos del Cliente organizador

Usuario->>MainUI: Envía datos

MainUI->>Cliente: new Cliente(...)

MainUI->>Usuario: Selecciona deporte

Usuario->>MainUI: Deporte (ej. Fútbol)

MainUI->>Torneo: new Torneo()

MainUI->>Torneo: setDeporte(deporte)

MainUI->>Torneo: instalacionesDisponibles = getInstalaciones(deporte, inst)

MainUI->>Usuario: Muestra instalaciones para ese deporte

Usuario->>MainUI: Selecciona instalación

MainUI->>Torneo: setInstalacion(opcion)

MainUI->>Usuario: Configura reglas específicas (ej. Sustituciones, Duración)

User->>MainUI: Envía valores

MainUI->>Torneo: setReglas(listaReglas)

Note over MainUI,Torneo: Registro de Equipos

loop para cada equipo

MainUI->>Usuario: Solicita nombre de equipo y jugadores

User->>MainUI: Envía datos

MainUI->>Equipo: new Equipo(...)

MainUI->>Torneo: agregar a lista de equiposParticipantes

end

Note over MainUI,Trabajador: Asignar médicos y revisión

MainUI->>Trabajador: Asigna un médico a cada equipo<br/>Simula revisión

alt Jugador no pasa revisión

MainUI->>Usuario: Solicita reemplazo

end

MainUI->>Usuario: Ofrece seguro médico adicional

alt Usuario acepta

MainUI->>Torneo: agregarSeguro("Seguro", costo)

end

Note over MainUI,Trabajador: Asigna dos árbitros a torneo

Note over MainUI,Reserva: Genera reservas para cada partido

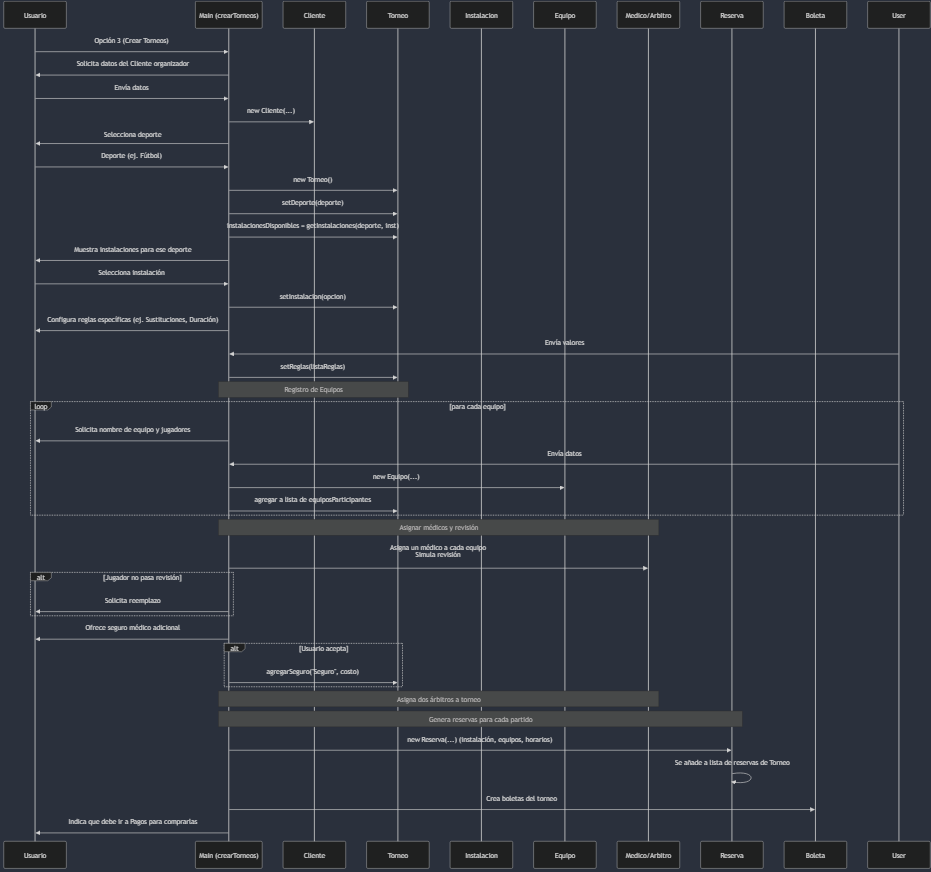
MainUI->>Reserva: new Reserva(...) (instalación, equipos, horarios)

Reserva->>Reserva: Se añade a lista de reservas de Torneo

MainUI->>Boleta: Crea boletas del torneo

MainUI->>Usuario: Indica que debe ir a Pagos para comprarlas

**Diagrama de Interacción:**



**4.crearEvento**

**Código:**

sequenceDiagram

participant Usuario

participant MainUI as Main (crearEvento)

participant Cliente

participant Evento

participant Localidad

participant Instalacion

participant Trabajador as Seguridad/Paramédico

participant Reserva

participant Boleta

Usuario->>MainUI: Opción 4 (Crear Evento)

MainUI->>Usuario: Pide datos organizador

Usuario->>MainUI: Envía datos

MainUI->>Cliente: new Cliente(...)

MainUI->>Usuario: Solicita nombre, tipo (Festival/Concierto), artista

Usuario->>MainUI: Responde

MainUI->>Evento: new Evento(...)

alt Tipo = Concierto

MainUI->>Usuario: Solicita género musical

Usuario->>MainUI: Envía

MainUI->>Evento: setGeneroMusical(...)

end

MainUI->>Usuario: ¿Cuántos artistas invitados?

Usuario->>MainUI: Cantidad e invitados

MainUI->>Evento: setArtistasInvitados(...)

MainUI->>Usuario: Configurar localidades (capacidad, VIP, etc.)

loop 5 localidades

MainUI->>Localidad: new Localidad(...) con parámetros

MainUI->>Evento: agrega a lista de Localidad

end

MainUI->>Usuario: Seleccionar toldos patrocinados

Usuario->>MainUI: Selecciona toldos

MainUI->>Evento: setToldosPatrocinados(...)

MainUI->>Usuario: Seleccionar foodtrucks

Usuario->>MainUI: Selecciona

MainUI->>Evento: setFoodTrucks(...)

MainUI->>Trabajador: Asigna personal de Seguridad/Paramédicos

Evento->>Evento: setPersonalSeguridad(...)<br/>setPersonalMedico(...)

MainUI->>Usuario: Fecha y horas del evento

Usuario->>MainUI: Envía horario

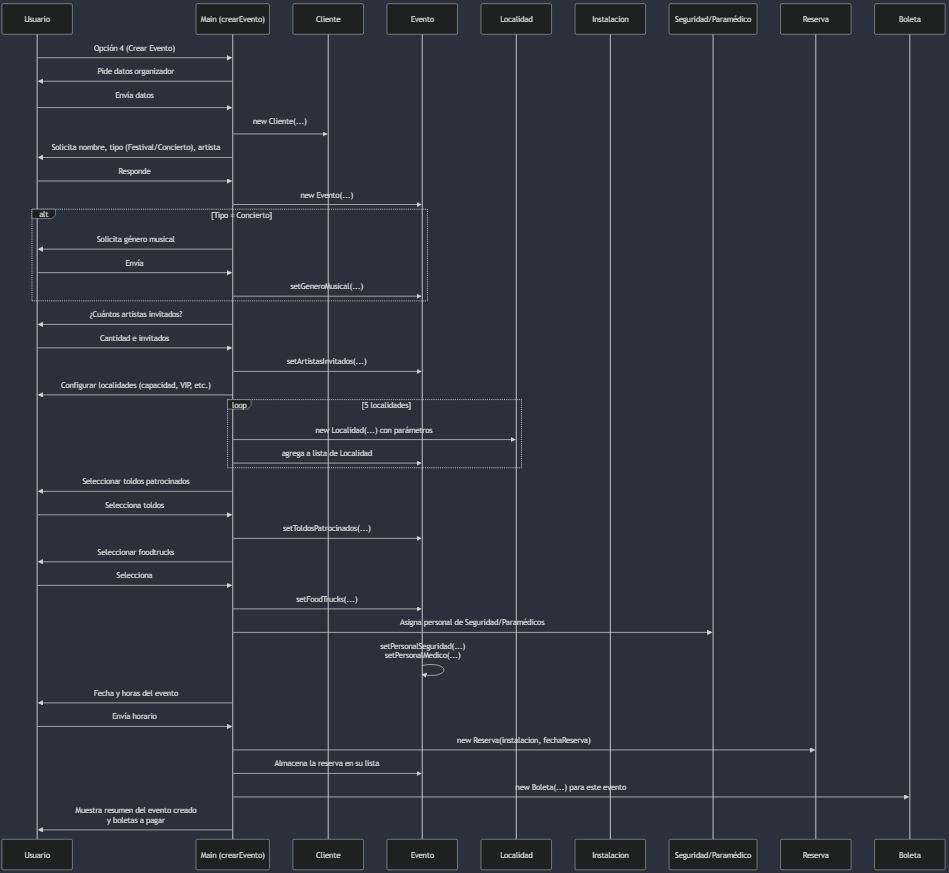
MainUI->>Reserva: new Reserva(instalacion, fechaReserva)

MainUI->>Evento: Almacena la reserva en su lista

MainUI->>Boleta: new Boleta(...) para este evento

MainUI->>Usuario: Muestra resumen del evento creado<br/>y boletas a pagar

**Diagrama de Interacción:**



**5.pagosUI**

**Código:**

sequenceDiagram

participant Usuario

participant MainUI as Main (pagosUI)

participant Reserva

participant Boleta

participant Suscripcion

participant Cliente

Usuario->>MainUI: Opción 5 (Taquilla - Pagos)

MainUI->>Usuario: Muestra submenú (1. Pagar Reserva, 2. Pagar Boleta, 3. Suscripción)

alt Pagar Reserva

MainUI->>Usuario: Solicita ID reserva

Usuario->>MainUI: Envía ID

MainUI->>Reserva: buscarReserva(ID)

alt Reserva encontrada

MainUI->>Usuario: Muestra detalles y total (aplicando descuento por suscripción)

Usuario->>MainUI: Confirma pago

alt Confirma

MainUI->>Reserva: setPagada(true)

MainUI->>Usuario: Muestra confirmación

else Cancela

MainUI->>Usuario: Muestra mensaje de cancelación

end

else Reserva no existe

MainUI->>Usuario: Error

end

else Pagar Boleta

MainUI->>Usuario: Solicita ID boleta

Usuario->>MainUI: Envía

MainUI->>Boleta: buscarBoleta(ID)

alt Boleta existe y no pagada

MainUI->>Usuario: Muestra precio y solicita confirmación

Usuario->>MainUI: Elige

alt Paga

MainUI->>Boleta: setPagada(true)

MainUI->>Usuario: Muestra confirmación

else Cancela

MainUI->>Usuario: Mensaje de cancelación

end

else No existe o ya pagada

MainUI->>Usuario: Mensaje de error

end

else Administrar Suscripción

MainUI->>Usuario: 1. Renovar / 2. Crear suscripción

Usuario->>MainUI: Opción suscripción

MainUI->>Usuario: Solicita ID Cliente

Usuario->>MainUI: Envía

MainUI->>Cliente: obtenerCliente(ID)

alt Cliente existe

MainUI->>Suscripcion: Verifica estado y tipo

alt Renovar

MainUI->>Usuario: Informa fecha fin y costo

Usuario->>MainUI: Confirma

alt Si

Suscripcion->>Suscripcion: Ajusta finSuscripcion

MainUI->>Usuario: Confirma renovada

else No

MainUI->>Usuario: Cancela

end

else Crear Suscripción

MainUI->>Usuario: Selecciona tipo (Rookie, ProPlayer, MVP)

Usuario->>MainUI: Opción

MainUI->>Suscripcion: setTipoSuscripcion(...)

MainUI->>Usuario: Muestra confirmación

end

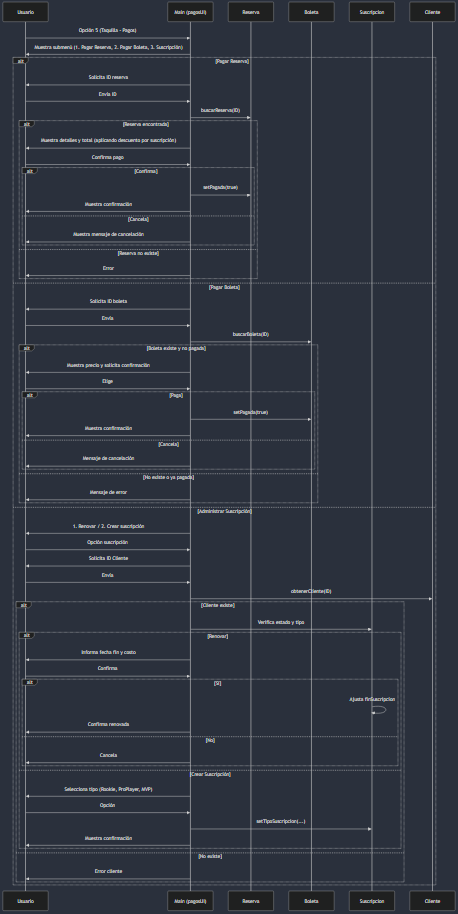
else No existe

MainUI->>Usuario: Error cliente

end

end

**Diagrama de Interacción:**



**MANUAL DE USUARIO**

**1. DESCRIPCIÓN GENERAL DE LA APLICACIÓN**

La aplicación ofrece diversas funcionalidades orientadas a la administración de:

* Reservas de instalaciones deportivas (por horas).
* Inscripciones de jóvenes en deportes formativos.
* Creación de torneos (fútbol, baloncesto, natación y voleibol).
* Creación de eventos (conciertos o festivales).
* Pagos de reservas, boletas y suscripciones.
* Gestión de clientes (creación y visualización de datos).

La aplicación funciona en consola (línea de comandos), mostrando un menú con opciones numeradas. El usuario elige la opción que desea, escribiendo el número correspondiente y presionando la tecla Enter.

**2. REQUISITOS PREVIOS**

1. Java instalado: Requiere una versión reciente del JDK (Java Development Kit) u OpenJDK.
2. Compilación: Si se cuenta con archivos .java:
   * Asegurarse de tener todos los archivos en las carpetas adecuadas.
   * Compilarlos con el comando javac o mediante un IDE (por ejemplo, Eclipse, IntelliJ IDEA o NetBeans).
3. Ejecución:
   * Una vez compilado, ejecutar la clase principal uiMain.Main que contiene el método public static void main(String[] args).
   * En la consola, se escribiría algo como:
     + java uiMain.Main

**3. INICIAR LA APLICACIÓN**

Al ejecutar el programa, aparecerá el menú principal en la consola:

===== Menú Principal =====

1. Realizar y/o Visualizar Reservas

2. Realizar Inscripciones a Deportes Formativos

3. Crear Torneos

4. Crear Eventos (Concierto)

5. Taquilla (Pagos)

6. Administar Clientes

7. Salir del sistema

Seleccione una opción:

Para moverse entre opciones:

1. Escribir el número de la funcionalidad (del 1 al 7).
2. Presionar Enter.
3. Seguir las instrucciones que aparecerán en pantalla.

**4. FUNCIONALIDADES PRINCIPALES**

**4.1. REALIZAR Y/O VISUALIZAR RESERVAS (OPCIÓN 1)**

1. Teclear 1 y presionar Enter.
2. Se mostrará:

-----Menu Reservas-----

1: Crear Reserva

2: Visualizar Reserva

1. Crear Reserva (opción 1):
   * Se solicita el ID del cliente. Si no existe el cliente, primero deberá crearse en la opción 6 (Administrar Clientes).
   * Seleccionar el deporte (Fútbol, Baloncesto, Natación o Voleibol).
   * Se mostrarán las instalaciones disponibles para ese deporte.
   * Elegir la instalación mediante su ID.
   * Ingresar la fecha y horario (hora de inicio y hora de fin).
   * El sistema calcula el costo total (precioHora × cantidad de horas).
   * Se muestra la reserva creada y el monto a pagar (aún no pagado).
2. Visualizar Reserva (opción 2):
   * Se ingresa el ID de la reserva.
   * Se muestra la información si la reserva existe.

¿Para qué sirve?  
Para que usuarios o clientes reserven canchas o piscinas por horas.

**4.2. REALIZAR INSCRIPCIONES A DEPORTES FORMATIVOS (OPCIÓN 2)**

1. **Teclear 2 y presionar Enter.**
2. **Se pedirán datos del joven: nombre, apellido, documento, edad, EPS, acudiente, etc.**
3. **Se solicitará el deporte y la experiencia en meses.**
4. **Según la edad y experiencia, el sistema asigna una categoría (1, 2 o 3) y un horario (mañana o tarde).**
5. **Se crea un objeto “Joven” y se asocia a un “GrupoFormativo” con un “Entrenador”.**
6. **Compra de equipamiento (opcional):**
   * **Ofrece la Tienda Escuela con artículos (uniformes, balones, etc.).**
   * **Se elige un artículo por ID y, si hay stock, se descuenta el inventario.**

**¿Para qué sirve?**  
**Para inscribir niños/jóvenes en clases de formación deportiva y adquirir indumentaria.**

**4.3. CREAR TORNEOS (OPCIÓN 3)**

1. Teclear 3 y presionar Enter.
2. Se solicitan datos del cliente organizador (nombre, apellido, edad, ID).
3. Escoger el deporte del torneo (Fútbol, Baloncesto, Natación o Voleibol).
4. El sistema muestra las instalaciones disponibles y pide elegir una.
5. Configurar reglas específicas (por ejemplo, sustituciones máximas, duración de partidos, etc.).
6. Inscribir equipos y sus jugadores (para Fútbol, Baloncesto, Voleibol o Natación).
7. Asignar médicos para valoración; simular un caso de jugador que no pasa la revisión.
8. Opción de seguro médico.
9. Asignar árbitros.
10. Crear las reservas automáticamente para cada partido.
11. Generar boletas para asistir a los partidos.

¿Para qué sirve?  
Para organizar un torneo con múltiples equipos y partidos, facilitando la logística de reservas y árbitros.

**4.4. CREAR EVENTOS (OPCIÓN 4)**

1. Teclear 4 y presionar Enter.
2. Ingresar datos del cliente organizador.
3. Ingresar nombre del evento (Concierto o Festival), artista principal y/o género musical.
4. Registrar artistas invitados (opcional).
5. Configurar localidades (ej.: tribunas Norte, Sur, VIP, etc.), capacidad, precio sugerido.
6. Elegir toldos patrocinados y food trucks (opcionales).
7. Se asigna personal de seguridad y paramédicos.
8. Reservar la instalación principal indicando la fecha (mínimo 6 horas).
9. Generar boletas que luego se pueden pagar en la taquilla.
10. Se muestra un resumen del evento.

¿Para qué sirve?  
Para crear un evento musical o cultural, configurando localidades y servicios.

**4.5. TAQUILLA (PAGOS) (OPCIÓN 5)**

1. Teclear 5 y presionar Enter.
2. Aparecerá un submenú de pagos:

1: Pago via ID

2: Pago boleta de evento

3: Administrar Suscripción

4: Salir de pagos

1. Pago de Reserva (1):
   * Solicita ID de la reserva.
   * Verifica si existe y muestra el costo (incluyendo posible descuento por suscripción).
   * Puede confirmar el pago o cancelar.
2. Pago Boleta (2):
   * Solicita ID de la boleta.
   * Muestra el costo y permite pagar.
3. Administrar Suscripción (3):
   * Se ingresa ID de cliente.
   * Permite crear una nueva suscripción (Rookie, Pro Player, MVP) o renovar si ya existe.
   * Ajusta precio y fecha de vencimiento.
4. Salir de pagos (4) para regresar al menú principal.

¿Para qué sirve?  
Para gestionar cobros de reservas, boletas y suscripciones.

**4.6. ADMINISTRAR CLIENTES (OPCIÓN 6)**

1. Teclear 6 y presionar Enter.
2. Submenú:

1: Crear cliente

2: Visualizar cliente

1. Crear cliente:
   * Solicita nombre, apellido y edad.
   * Muestra un ID asignado. Úselo en reservas o pagos.
2. Visualizar cliente:
   * Solicita ID de cliente.
   * Muestra la información si existe.

¿Para qué sirve?  
Para registrar nuevos clientes o consultar datos de clientes existentes.

**5. CONSEJOS Y SOLUCIÓN DE PROBLEMAS (FAQ)**

1. "No se encontró cliente con ese ID"

* Asegúrate de crear el cliente antes.
* Utiliza el ID correcto que te dio el sistema.

2. "Reserva o Boleta no encontrada"

* Verifica que el ID digitado sea válido.
* Los ID se generan secuencialmente en la aplicación.

3. Descuentos y Suscripciones

* Revisa si la suscripción está vigente (no vencida).
* Si no tienes suscripción, crea o renueva en la opción 5 → 3.

4. Instalaciones incorrectas

* Asegúrate de seleccionar el deporte correcto.

5. Error al compilar

* Revisa que tu versión de Java sea compatible.
* Confirma que todos los archivos .java estén en sus paquetes adecuados.

**6. NAVEGACIÓN RESUMIDA DEL MENÚ**

1. Reservas
   * Crear Reserva
   * Visualizar Reserva
2. Inscripción a Deportes Formativos
3. Crear Torneos
4. Crear Eventos (Concierto/Festival)
5. Taquilla (Pagos)
   * Pagar Reserva
   * Pagar Boleta
   * Administrar Suscripción
   * Salir
6. Administrar Clientes
   * Crear Cliente
   * Visualizar Cliente
7. Salir del Sistema

Al final, para salir por completo, elegir la opción 7.

**7. RECOMENDACIONES PARA USUARIOS SEGÚN SU NIVEL DE CONOCIMIENTO**

* Usuarios Principiantes  
  Sigan el menú en orden. Primero creen un cliente en la opción 6, luego reserven (opción 1) y paguen (opción 5).  
  Si cometen un error, pueden volver a iniciar el programa.
* Usuarios Intermedios  
  Combinen funcionalidades según necesiten: crear varios clientes, torneos, etc.  
  Pueden pagar reservas (opción 5) más adelante.
* Usuarios Avanzados  
  Pueden personalizar reglas de deportes, organizar torneos con muchas reservas y jugadores, asignar personal, etc.

**8. CIERRE Y CONTACTO**

¡Listo! Con este manual, cualquier usuario está preparado para:

* Realizar reservas.
* Inscribir a jóvenes en deportes formativos.
* Organizar torneos y eventos.
* Pagar reservas y boletas.
* Administrar clientes y suscripciones.

Si todavía tiene dudas o problemas no cubiertos aquí, consulte con el equipo de soporte o revise la documentación técnica de la aplicación.

¡Disfrute de las funcionalidades de la plataforma!