Informe de Diseño e Implementación – Práctica 1

Introducción

Este programa ha sido desarrollado con el objetivo de gestionar usuarios, pistas y reservas para un sistema de reservas deportivas. El sistema ofrece un menú sencillo que permite al usuario interactuar con las opciones de gestión de usuarios, pistas y reservas. Para su funcionamiento, el programa cuenta con tres gestores principales: el **GestorUsuarios**, el **GestorPistas**, y el **GestorReservas**.

Toda la información de los usuarios, pistas y reservas se almacena temporalmente en la memoria durante la ejecución del programa. Actualmente, el sistema no guarda la información en archivos una vez que el programa se cierra, pero está diseñado para permitir una actualización futura que agregue esta funcionalidad sin realizar grandes cambios.

Estructura del Menú

Al iniciar el programa, el usuario ve un menú principal que le ofrece cuatro opciones: gestión de usuarios, gestión de pistas, gestión de reservas y salir del programa. Cada sección de gestión cuenta con su propio submenú, lo cual hace que navegar entre las distintas funciones sea sencillo y accesible.

- **Gestión de Usuarios:** Permite agregar, modificar y listar usuarios. Para modificar un usuario, el programa verifica que el correo electrónico esté registrado y permite cambiar datos como nombre, fecha de nacimiento y si el usuario cuenta con un bono.
- **Gestión de Pistas:** Se pueden crear nuevas pistas, ver las pistas ocupadas y buscar pistas disponibles según el número de jugadores y el tipo de pista (interior o exterior).
- **Gestión de Reservas:** Ofrece la opción de hacer reservas normales o usar un bono, así como consultar reservas futuras o buscar reservas para un día y pista específica. Al crear una reserva, se verifica que el usuario existe y que la fecha esté correctamente ingresada.

Descripción de los Gestores

Cada gestor se encarga de diferentes funcionalidades de acuerdo con la entidad que administra:

1. **GestorUsuarios** : Esta clase usa una lista para almacenar los datos de los usuarios registrados, y asegura que el correo electrónico de cada usuario sea único para evitar duplicados. Ofrece métodos para agregar nuevos usuarios, modificar la información de los existentes y listar todos los usuarios.

2. **GestorPistas** : Esta clase administra una lista de pistas disponibles. Permite crear y buscar pistas en función de la disponibilidad, el tipo de pista, y la cantidad de jugadores para la que están diseñados.
3. **GestorReservas** : Encargado de gestionar las reservas de pistas. Ofrece opciones para hacer una reserva individual o utilizar un bono y permite consultar las reservas futuras o por fecha. Antes de realizar una reserva, el sistema verifica que el usuario esté registrado y que tenga un bono activo si desea usarlo para la reserva.

Control de Errores y Validaciones

El programa incluye validaciones para asegurar la calidad de los datos ingresados y mejorar la experiencia del usuario. Algunos ejemplos de validaciones son:

- **Verificación de Correo** : Para evitar duplicados, al agregar o modificar un usuario, el programa comprueba que el correo electrónico no esté registrado anteriormente.
- **Formato de Fecha** : Tanto para la fecha de nacimiento como para la fecha de reservas, el sistema sigue el formato "AAAA-MM-DD". Si el usuario no introduce la fecha en este formato, el programa muestra un mensaje de error.
- **Comprobación de Bono** : Al reservar con un bono, se verifica que el usuario posee uno activo. Si no es así, se muestra un mensaje de error que indica la necesidad de un bono válido.

Estructura del Proyecto en Eclipse

El proyecto está organizado en carpetas para facilitar la navegación:

- **Interfaz** : Contiene el archivo Main, donde se administra el menú principal y se gestionan las llamadas a los métodos de cada gestor.
- **ej1** : Incluye las clases base como **Jugador** y **Pista** , que representan las entidades de usuario y pista, respectivamente.
- **ej2** : Incluye la clase **Reserva** , que almacena la información de cada reserva.
- **ej3** : Agrupa las clases **GestorUsuarios** , **GestorPistas** y **GestorReservas** , donde reside la lógica para gestionar cada entidad.

Conclusión

Cada gestor tiene sus propias responsabilidades y organiza sus datos de forma independiente, lo cual hace que el programa sea fácil de mantener y de mejorar en el futuro. Aunque actualmente los datos solo se almacenan temporalmente, el diseño modular permite agregar métodos de persistencia para guardar la información en archivos o bases de datos sin modificar su estructura base. El programa permite futuras mejoras de manera sencilla.