Sistemas distribuidos (EI1021)
Práctica 1. Desarrollo local de la aplicación ejemplo

En esta primera práctica vamos a desarrollar de modo local (no distribuido) la aplicación ejemplo sobre la que vamos a trabajar durante todo el curso. Se trata de un servicio para reservar sesiones en un gimnasio. Los usuarios/clientes podrán reservar sesiones, modificar las reservas, cancelarlas y consultar sus reservas y las sesiones con plazas disponibles.

La aplicación permitirá gestionar las reservas de la próxima semana. Esto es, si un usuario se conecta al servicio un martes, podrá gestionar sus reservas desde el miércoles hasta el siguiente martes, inclusive, pero no podrá acceder a las reservas del día actual ni de los anteriores. A continuación mostramos un ejemplo del calendario de sesiones disponibles para una semana cualquiera. Este es el calendario generado por defecto por la aplicación, pero podría cambiarse por cualquier otro y la aplicación debería seguir funcionando.

Hora	Lunes	Martes	Miércoles	Jueves	Viernes
09:00–10:00	Taichí	—	Taichí	Taichí	—
10:00–11:00	Ironfit	Ironfit	—	Ironfit	Ironfit
18:00–19:00	—	Yoga	Yoga	—	Yoga

Cuadro 1: Calendario semanal de sesiones disponibles por día y hora.

El objetivo de esta práctica es desarrollar y probar en modo local la funcionalidad básica de la aplicación. A lo largo de las siguientes prácticas aprovecharemos el código desarrollado en ésta para implementar la misma aplicación de modo distribuido utilizando las principales tecnologías que se abordan en la asignatura: sockets, objetos remotos, servicios web y servlets.

Durante la ejecución de la aplicación, los datos de las reservas y las sesiones se guardarán y modificarán en dos diccionarios en la memoria principal. Sin embargo, para preservar los datos y las modificaciones de las reservas cuando la aplicación no esté en marcha, la información se leerá de un fichero de texto al arrancar la aplicación y se guardará cada vez que un usuario salga. A lo largo de todas las prácticas utilizaremos JSON como tecnología de codificación, almacenamiento e intercambio de información estructurada. Así, las reservas se guardarán como texto en formato JSON en el fichero y los distintos métodos devolverán los datos utilizando esa misma codificación.

En las prácticas usaremos una implementación sencilla de JSON incluida en el paquete `json-simple`. En la siguiente página web puedes encontrar un tutorial para conocer la API y el uso de esta implementación (`https://www.tutorialspoint.com/json_simple/json_simple_quick_guide.htm`). Puedes encontrar información adicional sobre cómo escribir y leer objetos JSON en ficheros de texto: (`https://mkyong.com/java/json-simple-example-read-and-write-json/`).

A continuación se describen las clases y métodos a implementar en esta práctica, parte de las cuales se ponen a vuestra disposición. Antes de lanzarse a implementar los métodos, conviene revisar los ya existentes y sus implementaciones para aprovechar el código proporcionado y no reinventar la rueda. No debe modificarse la cabecera de ningún método ni añadirse ningún nuevo método público, aunque pueden añadirse todos los métodos privados que se desee. Es importante documentar cada nuevo método siguiendo el ejemplo de los métodos proporcionados. También es importante comentar adecuadamente el código para ayudar a entenderlo, tanto al profesor, como a ti mismo dentro de unas semanas.

De cada sesión se almacenan los siguientes datos:

- `actividad`: deporte a practicar durante la sesión.
- `hora`: hora de la sesión guardada como un valor entre 0 y 23.
- `plazas`: número de plazas disponibles.

De cada reserva se almacenan los siguientes datos:

- `codReserva`: código único de la reserva generado cuando la hace un usuario.
- `codUsuario`: código único del usuario que ha hecho la reserva.
- `actividad`: actividad para la que se ha hecho la reserva.
- `dia`: día de la semana (lunes a domingo) guardado con el enumerado `DiaSemana`, que os proporcionamos.
- `hora`: hora de la sesión en que se ha hecho la reserva, guardada como un valor entre 0 y 23.

El resultado a entregar de esta primera práctica consistirá en un proyecto Java con el enumerado `DiaSemana` y cuatro clases: `Reserva`, `Sesion`, `GestorReservas` y `UsuarioLocal`.

A continuación describiremos brevemente las distintas clases. Revisa el código proporcionado y la documentación de los distintos métodos para conocer más detalles sobre los mismos.

La clase `Reserva` contendrá los datos de una reserva, un constructor al que se le pasan los datos como parámetros, un constructor al que se le pasan todos los datos en formato JSON y lo siguientes métodos públicos:

- Los getters y setters necesarios para acceder y modificar los atributos de la clase.
- `toString`: sobrecarga el método para devolver la información de la reserva como una cadena.
- `toJSON`: construye y devuelve un objeto JSON con los datos de la reserva.

La clase `Sesion` contendrá los datos de una sesión, un constructor al que se le pasan los datos como parámetros y lo siguientes métodos públicos:

- Los getters y setters necesarios para acceder y modificar los atributos de la clase.
- `toJSON`: construye y devuelve un objeto JSON con los datos de la sesión.

La clase `GestorReservas` implementará las funciones básicas del servicio, consultando y modificando los datos en memoria y actualizando el fichero cuando cada usuario sale del programa. Al implementar los métodos nos basaremos en que cada usuario solo podrá consultar, modificar y cancelar sus propias reservas. Las reservas se guardarán en memoria en un diccionario indexado por el código del usuario. Las sesiones disponibles se guardarán en otro diccionario indexado por el día de la semana. Además, la clase también tendrá como atributo un *stream* de escritura del fichero de datos. Esta clase tendrá un constructor y los siguientes métodos:

- `generaSesiones`: genera y guarda en el diccionario de sesiones las de la próxima semana.
- `generaReservas`: genera y hace unas reservas por defecto.
- `guardaDatos`: guarda los datos de las reservas del diccionario en un fichero de texto apoyándose en el siguiente método.
- `escribeFichero`: copia los datos del diccionario a un array JSON y lo escribe en un fichero de texto. El array contendrá objetos JSON con los datos de cada reserva.
- `leeFichero`: lee los datos del fichero en un array JSON y usa el método siguiente para modificar los diccionarios.
- `rellenaDiccionarios`: hace las reservas almacenadas en un array JSON.
- `buscaSesion`: busca y devuelve una sesión dada su actividad, día y hora.
- `listaReservasUsuario`: devuelve un array JSON con los datos de las reservas de un usuario dado.
- `listaPlazasDisponibles`: devuelve un array JSON con los días, horas y plazas disponibles en las sesiones de una actividad dada.
- `hazReserva`: hace una reserva de un actividad dada, si hay una sesión con plazas disponibles el día y hora solicitados. Devuelve un objeto JSON con los datos de la reserva hecha.
- `buscaReserva`: busca en un vector de reservas una reserva con un código dado y la devuelve.
- `modificaReserva`: permite a un usuario cambiar de día y hora una de sus reservas, pero manteniendo la actividad. Devuelve un objeto JSON con los datos de la reserva modificada.
- `cancelaReserva`: permite a un usuario cancelar una de sus reservas. Devuelve un objeto JSON con los datos de la reserva cancelada.

La cuarta de las clases a implementar, `UsuarioLocal`, contendrá el método principal (`main`) que permitirá a los usuarios que quieran gestionar sus reservas acceder al servicio utilizando un objeto de la clase `GestorReservas`. El modo en que se almacenan las reservas y sesiones es transparente a esta clase. Esta clase NO maneja objetos de tipo `Reserva` o `Sesion`, ni depende de cómo estos se almacenan. Tan solo maneja objetos o arrays JSON conteniendo información sobre las reservas devuelta por los métodos del gestor.

El método principal empezará por pedir el código del usuario y, a continuación, presentará repetidamente un menú de opciones que le permitirá realizar las siguientes operaciones del servicio de reservas:

- Salir del programa guardando los datos en el fichero.
- Mostrar todas sus reservas.
- Mostrar las plazas disponibles en sesiones de una actividad dada.
- Hacer una reserva.

- Modificar una de sus reservas.
- Cancelar una de sus reservas.

Cada operación mostrará por pantalla la información adecuada para que el usuario conozca el resultado de la misma en todos los casos. Por ejemplo, si no se ha podido modificar una reserva, deberá mostrarse el mensaje adecuado.