Práctica y Trabajos Tutelados de Internet y Sistemas Distribuidos

3º Curso – Grado en Informática
Curso académico 2020-2021

1 Introducción

La práctica y los trabajos tutelados de la asignatura “Internet y Sistemas Distribuidos” consistirán en la aplicación de los conceptos y tecnologías aprendidos en la asignatura para el desarrollo de un servicio simplificado de gestión de carreras populares de running.

Se desarrollará una aplicación que siga una arquitectura en capas como la estudiada en la asignatura, incluyendo la capa acceso a datos, la capa lógica de negocio, la capa servicios, la capa acceso a servicios y la capa interfaz de usuario. La aplicación podrá ser invocada remotamente usando REST y, opcionalmente, Apache Thrift. La capa acceso a datos utilizará una base de datos relacional para guardar la información pertinente.

El apartado 2 especifica la funcionalidad de la práctica. El apartado 3 especifica la funcionalidad de los trabajos tutelados.

2 Práctica

2.1 Visión global

RunFic se dedica a ofrecer información de carreras populares de atletismo y permite a sus usuarios inscribirse en ellas pagando una cuota de inscripción.

Cuando a un usuario le interesa participar en una determinada carrera, debe inscribirse en ella y efectuar el pago de la cuota de inscripción con tarjeta de crédito. En el momento de inscribirse, se le asignará un dorsal para esa carrera (para cada carrera los dorsales se empezarán a asignar empezando por el número 1 y se asignarán correlativamente en orden de inscripción) y un código para recoger el dorsal. Las inscripciones pueden realizarse hasta 24 horas antes de comenzar la carrera, y se guardará el día y hora a la que se hizo cada inscripción.

Para poder participar en la carrera, el usuario debe recoger el dorsal en el local de *RunFic*, para lo cual debe presentar la tarjeta con la que hizo la inscripción junto con el código que se le proporcionó en el momento de inscribirse.

El objetivo de la práctica es desarrollar una aplicación que dé soporte a algunos aspectos de la operación de *RunFic*. Más concretamente, debe permitir dar de alta carreras, buscarlas por identificador, buscarlas por fecha y localidad, inscribirse en una carrera, buscar todas las inscripciones de un usuario y marcar que el dorsal correspondiente a una inscripción ha sido recogido.

2.2 Funcionalidad de la Capa Modelo

En esta sección se proporciona más información sobre la funcionalidad que debe soportar la capa modelo. Nótese que deben tratarse las condiciones de error que se desprenden de la descripción realizada en esta y en la sección anterior.

1. Para dar de alta una carrera, se indicará la ciudad donde se celebra, una descripción, la fecha y hora, el precio de la inscripción y el número máximo de participantes. Además, se guardará la fecha y hora en la que se ha dado de alta la carrera.
2. Será posible buscar carreras por su identificador. La información devuelta de la carrera incluirá, además de la información proporcionada al darla de alta, el número de inscritos en ella.
3. Será posible buscar carreras que se celebren antes de una fecha (debe ser una fecha futura y el resultado contendrá únicamente las carreras que aún no se han celebrado). Opcionalmente, se podrá indicar el nombre de una ciudad, en cuyo caso se devolverán solamente las carreras que se celebren en esa ciudad. Al igual que en el punto anterior, la información devuelta de cada carrera incluirá, además de la información proporcionada al darla de alta, el número de inscritos en ella.
4. Será posible que un usuario se inscriba en una carrera hasta 24 horas antes de su celebración. Además de otros parámetros que puedan ser necesarios, recibe como entrada un e-mail para identificar al usuario, y un número de tarjeta de crédito. En caso de ejecutarse con éxito, devuelve un código que será necesario para recoger el dorsal, y se almacena la inscripción, quedando registrado el número de dorsal asignado al participante y la fecha y hora a la que se hizo la inscripción.
5. Será posible obtener todas las inscripciones que un usuario ha realizado a lo largo del tiempo. Deben devolverse todos los datos almacenados para cada inscripción.
6. Será posible indicar que un usuario ha recogido el dorsal correspondiente a una inscripción. Un usuario recoge el dorsal correspondiente a una inscripción en el local de *RunFic*, presentando el código o identificador obtenido al realizar la inscripción y el número de tarjeta de crédito utilizada para pagarla. A partir de esos datos, un empleado de *RunFic* podrá indicar que el dorsal correspondiente a esa inscripción se ha entregado. Es necesario contemplar todos los posibles casos de error, como que el código de inscripción y el número de tarjeta no se correspondan con ninguna inscripción, o que el dorsal correspondiente a esa inscripción ya ha sido entregado previamente.

IMPORTANTE: Para simplificar la implementación de la práctica, NO es necesario guardar datos de los usuarios del servicio. Sólo se guardará su e-mail al hacer una inscripción.

2.3 Detalles de la Capa Servicios

La capa servicios expondrá la funcionalidad de la capa modelo a las aplicaciones remotas usando un servicio web REST que trabajará con datos en formato JSON. Sin embargo:

- En la búsqueda de carreras será obligatorio especificar, además de la fecha, la ciudad en la que se desean buscar las carreras.
- Los datos de las carreras devueltos como resultado de una consulta no incluirán la fecha de alta de la carrera.

2.4 Detalles de la Aplicación Cliente

Para simplificar, en la práctica se desarrollará un único cliente que permitirá invocar todas las operaciones ofrecidas por la capa servicios (si bien, en un caso real habría diferentes clientes para los usuarios finales y para los administradores o empleados de *RunFic*).

En las operaciones que muestran datos de las carreras (buscar carreras por fecha y ciudad, y buscar

carrera por identificador), la aplicación mostrará el número de plazas disponibles en lugar del número de inscritos.

Debe implementarse la capa acceso al servicio usando REST.

Aunque no se realice el trabajo tutelado de Apache Thrift (ver siguiente apartado), la arquitectura de la práctica deberá contemplar la posibilidad de que se desarrollasen en un futuro varias implementaciones de las capas servicios y acceso al servicio, de forma que el cliente pudiese utilizar una capa de acceso al servicio u otra modificando simplemente un parámetro de configuración.

3 Trabajos tutelados

Opcionalmente se podrá realizar un trabajo tutelado que se explica en el apartado 3.1.

3.1 Apache Thrift

En este trabajo tutelado se propone implementar la capa servicios usando un servicio Thrift.

Además, los alumnos que realicen este trabajo tutelado, deberán implementar la capa acceso al servicio del cliente usando dicha tecnología. El cliente podrá cambiar de la versión REST a la versión Thrift del servicio modificando simplemente un parámetro de configuración.

4 Normativa y evaluación

4.1 Composición de los grupos y trabajo a realizar por cada alumno

La práctica se realizará en grupos de 3 personas. El diseño de la práctica se hará en común, pero **es obligatorio que cada alumno se encargue de la implementación de dos funcionalidades (a nivel de capa modelo, capa servicios, capa acceso al servicio y capa interfaz de usuario).** Las funcionalidades se repartirán de la siguiente forma (los componentes de cada grupo se pondrán de acuerdo para repartirlas):

- Alumno 1: Funcionalidades 1 y 3
- Alumno 2: Funcionalidades 4 y 5
- Alumno 3: Funcionalidades 2 y 6

En el caso de los grupos de dos personas, cada alumno tendrá que implementar, al menos, dos funcionalidades:

- Alumno 1: Funcionalidades 4 y 5.
- Alumno 2: Funcionalidades 2 y 6.
- Las funcionalidades 1 y 3 se podrán repartir de cualquier forma entre el alumno 1 y el alumno 2.

Este reparto no quiere decir que un alumno no pueda tocar nada relacionado con las funcionalidades de sus compañeros (por ejemplo, si detecta un bug, puede arreglarlo), pero **el grueso del código relacionado con esas funcionalidades (en todas las capas) tiene que estar implementado por el alumno al que le correspondan**.

Los alumnos que deseen realizar el trabajo tutelado tendrán que implementar la parte de la capa servicios y de la capa acceso al servicio (usando Apache Thrift) correspondiente a las mismas funcionalidades que hayan implementado en la práctica. **No es necesario que todos los miembros del grupo hagan el trabajo tutelado** (por ejemplo, si de un grupo de 3 alumnos, solamente uno de ellos quiere presentar el trabajo tutelado, entonces solamente implementará la

parte correspondiente a las dos funcionalidades que haya implementado en la práctica).

4.2 Estándar de codificación

Con objeto de escribir código de calidad y fácilmente legible, se seguirá un sistema de codificación común, que define reglas para nombrar clases, atributos y métodos, normas de indentación, etc. Esto permite que en un equipo de desarrollo el aspecto del código sea el mismo, independientemente de qué programador lo haya escrito, lo que facilita el mantenimiento. Para la práctica se utilizará el estándar de codificación [JAVACON]. Los ejemplos de la asignatura siguen estas sencillas convenciones de nombrado. Para no alargar la práctica, no será necesario realizar documentación de las clases (e.g. JavaDoc).

4.3 Iteraciones y entregas

Para la realización de la aplicación se seguirá un enfoque basado en iteraciones, de manera que cada iteración incorpora más funcionalidad sobre la anterior, hasta que en la última iteración se termina con un software que implementa toda la funcionalidad.

En particular, la aplicación se hará en tres iteraciones. Es **obligatorio** entregar las dos primeras en plazo y con los contenidos definidos para cada una de ellas como obligatorios totalmente implementados. En otro caso, la práctica será calificada con un cero para todos los miembros del grupo. La entrega de la tercera iteración es opcional.

- A pesar de ser obligatoria su entrega, la corrección de la **primera iteración** no llevará una nota asociada. El objetivo de la corrección de esta iteración es detectar errores importantes, y en ese caso, orientar al alumno hacia su resolución.
- En la **segunda iteración** se completará la práctica y, opcionalmente, el trabajo tutelado. En la corrección de esta iteración se pondrá la nota de la práctica, y también del trabajo tutelado a los alumnos que lo hayan entregado. La nota puede ser diferente para cada miembro del grupo en función de la calidad de la implementación de la parte que haya hecho cada uno.
- La **tercera iteración es opcional**. En esta iteración se permite entregar el trabajo tutelado en caso de no haberlo entregado en la segunda iteración. En la corrección se pondrá la nota del trabajo tutelado. La nota puede ser diferente para cada miembro del grupo en función de la calidad de la implementación de la parte que haya hecho cada uno.

Para la entrega de cada iteración de la práctica, se utilizará el repositorio Git [GIT]. Se deben subir al repositorio **sólo los ficheros fuente** (e.g. .java, pom.xml, ficheros de configuración, etc.), **y no los ficheros objeto** (e.g. .class, .war, etc.). Antes de la entrega de cada iteración se proporcionarán instrucciones detalladas sobre cómo realizar la entrega.

A continuación se describe cada una de las tres iteraciones:

- **Primera iteración**. Se implementará toda la capa modelo (incluidas las pruebas de integración). **Plazo de entrega: domingo 22 de Noviembre.** La corrección de esta iteración se realizará durante la semana del 23 de Noviembre y será no presencial. **Esa semana no habrá clases de laboratorio**. El profesor enviará un correo a los miembros de cada grupo con los problemas que haya identificado al analizar el código de su práctica.
- **Segunda iteración**. Se implementará el servicio web REST que permite el acceso remoto a la capa modelo, así como el cliente de línea de comandos necesario para acceder al mismo. Los alumnos que lo deseen pueden presentar también el trabajo tutelado. **Plazo de entrega: domingo 3 de Enero.** La corrección de esta iteración se realizará durante la semana del 11 de Enero y será no presencial. **Esa semana no habrá clases de laboratorio**.

- **Tercera iteración**. Se presentará el trabajo tutelado. **Plazo de entrega: 7 de Febrero**. La corrección de esta iteración se realizará una vez finalizado el plazo de entrega y será no presencial.

4.4 Evaluación

La práctica y los trabajos tutelados se evaluarán de la siguiente manera:

- **Práctica**. Puntuación: **de 0 a 10 puntos**. Tal y como especifican las normas de evaluación de la asignatura, se recuerda que en la evaluación de la práctica es preciso obtener un **mínimo de 5 puntos**.
- **Trabajos tutelados**. Puntuación: **de 0 a 10 puntos**. En este caso no es preciso obtener ninguna nota mínima.

Para la **corrección de cada iteración**, el profesor descargará el código del repositorio Git y evaluará su correcto funcionamiento, la calidad del diseño y la calidad del código. **El profesor comprobará a través del historial de Git que cada alumno haya implementado una parte de la práctica, tal y como se indica en el apartado 4.1. En caso de que en alguna de las dos primeras iteraciones algún alumno no haya implementado su parte de la práctica, su puntuación será 0, y no penalizará a sus compañeros siempre y cuando estos se hayan encargado de que estén implementadas todas las funcionalidades de la práctica (da igual cómo se repartan las funcionalidades del alumno que no ha implementado las suyas). Como se dijo anteriormente, si en alguna de las dos primeras iteraciones no está implementada toda la funcionalidad especificada como obligatoria, la nota será un cero para todos los miembros del grupo.**

Como se comentó en el apartado 4.1, **no es necesario que todos los miembros del grupo presenten el trabajo tutelado. Los alumnos que decidan presentarlo únicamente tienen que implementar su parte (si sus compañeros deciden no presentarlo, su parte del trabajo tutelado quedaría sin implementar).**

Una práctica copiada significará un suspenso para el grupo que ha dejado copiar y para el que ha copiado; a todos los efectos, no se hará ninguna distinción. Los suspensos por práctica copiada tendrán que realizar una práctica distinta, que además deberán proponer (y ser aceptada).

Si alguno de los miembros de un grupo no supera la segunda iteración pero el resto sí lo hace, el alumno suspenso (o no presentado) deberá desarrollar en solitario en la segunda oportunidad una versión extendida de la práctica con funcionalidades adicionales.

Para la convocatoria extraordinaria de Julio se hará la misma práctica (excepto los suspensos en alguno de los dos casos anteriores), sin posibilidad de entregar la primera iteración: se presentará directamente la versión final de la práctica y del trabajo tutelado.

5 Referencias

[JAVACON] Sun Microsystems, “Java Code Conventions”, http://www.oracle.com/technetwork/java/codeconventions-150003.pdf.

[GIT] Git: https://git-scm.com/.