B) Plantear un ejercicio similar al que se ha visto en las prácticas de mantenimiento (practicaTestingYMantto-2014-11-23) vinculado al producto de su empresa teniendo en cuenta los siguientes aspectos:

a. Debe incluir la implementación de un caso de prueba no reflejado en el apartado anterior.
b. Debe incluir alguna actividad de mantenimiento correctivo, una de mantenimiento perfectivo y una de mantenimiento preventivo.

Enunciado propuesto del problema

Actualmente nuestra empresa se encuentra involucrada en el proceso de desarrollo de un programa de gestión. La aplicación en cuestión es CRySport y se encarga de gestionar las actividades deportivas de una determinada región. Esta aplicación esta enfocada principalmente a organismos públicos como pueden ser ayuntamientos, diputaciones o incluso comunidades autónomas. Cada una de estas regiones dispone únicamente de un único organizador, por lo que no se establecen restricciones en función de los roles ya que el organizador es el único que puede manejar todos los datos.

Todos los datos que se manejan en la aplicación se encuentran almacenados en un servicio de hosting externo. Por lo que la empresa encargada de usar el software no se preocupara de el mantenimiento de la base de datos, ya que la información se almacenará en servidores externos.

Los datos que almacenamos de la persona se encuentran en dos tablas de la base de datos. Una tabla es *usuarios*, en esta tabla se almacenan los siguientes datos:

- Nombre de usuario y contraseña: estos datos son los que nos permiten identificar a la persona que accede. En nuestro caso serán siempre del organizador ya que es el único que tiene acceso a la aplicación.

La segunda tabla donde se almacenan datos de la persona es conexión, y se guardan los datos siguientes:

- Nombre de usuario y fecha de último acceso: se mostrará al organizador cuando inicie la aplicación para saber cuando fue el último acceso.

Estos campos se almacenan en la base de datos y no son más que la implementación de los requisitos solicitados por el usuario.
Por otro lado también se almacena en la base de datos datos sobre cada una de las competiciones deportivas. Los datos que se encuentran almacenados en otra tabla (competiciones) son:

- Modalidad: este campo sirve para almacenar de que modalidad es la competición que estamos gestionando.
- Fecha: almacena la fecha de celebración del acontecimiento deportivo.
- Organizador: persona que ha añadido la aplicación al sistema.
- id: este campo nos permite identificar a una actividad de manera unívoca.

La aplicación desarrollada funciona relativamente bien pero sabemos que tenemos algunos fallos y ademas la aplicación va lenta en lo que concierne a los procesos de interacción que involucran una comunicación con la base de datos. También sabemos que el programa no muestra de manera correcta los datos de los organizadores que acceden a la aplicación.

Por otro lado nuestro cliente nos ha manifestado la siguiente queja:

- **Tras unos días de utilización de la aplicación esta dejo de funcionar. Tras un primer diagnóstico pudimos ver que el problema se encontraba en que el login no se realizaba correctamente. Más tarde localizariamos el error, que no era más que el servidor de hosting contratado dejó de ofrecer el servicio.**

A nuesto equipo de trabajo se le ha entregado lo siguiente:

- **Proyecto completo de Eclipse de *CRySport*. Este proyecto se encuentra organizado en varios paquetes que son:**
    - ***es.uclm.esi.lambdasoft.dominio:* incluye toda la información relativa a la lógica del programa. Se distribuye en 4 clases: *Competicion, Usuario, GestorDeCompeticiones, GestorDeCredenciales.***
    - ***es.uclm.esi.lambdasoft.persistencia:* este paquete incluye unicamente una clase Agente.java que es la encargada de establecer las conexiónes con el servidor remoto.**
    - ***es.uclm.esi.lambdasoft.presentacion:* se incluye en el todas las clases referentes a la interfaz así como los recursos (imagenes) que son usadas en el programa. Contiene un formulario principal (JFrame) en el cual se irán alternando los diferentes paneles (en total 4) según sea necesario.**

- **Así mismo en el paquete del proyecto también se encuentra una carpeta *src/test* en la cual hay 10 tests, que son superados en su totalidad, que cubren todas las clases del paquete de dominio. Las clases de test son:**
    - **CompeticionTest**
    - **LoginTest**
- **Cabe destacar que la estructuración del programa usando Maven facilita mucho la realización de las pruebas así como la gestión de las dependencias.**

A nuestro grupo de trabajo se le pide:

- **Definir e implementar los caso de prueba necesarios para poder detectar los problemas que pueden derivar de un mal desarrollo del programa así como futuros errores.**
- **Todos los test deben ser pasados sin errores.**
- **Definir una tarea de mantenimiento correctivo.**
- **Definir una tarea de mantenimiento preventivo: debido a que la aplicación puede tener mucho éxito se pide que adapte tanto esta, así como la base de datos, para que sea escalable es decir que pueda soportar multitud de competiciones así como competiciones con atributos diferentes (como nombres con caracteres especiales).**
- **Definir una tarea de mantenimiento perfectivo.**

a. Debe incluir la implementación de un caso de prueba no reflejado en el apartado anterior.

Como comentaremos a continuación como parte del mantenimiento perfectivo hemos añadido un caso de prueba que valide que se pueden realizar consultas con el carácter (‘), pues esto generaba errores en la aplicación.

```
public void testSQLInjection(){

        /* Setup */
        Usuario u = new Usuario("'g02.03","'g02.03");

        /* Ejecución del esceneario */
        boolean resultado = u.select();

        /* Oráculo */
        assertTrue(resultado);

}
```

Si nos fijamos al principio de las dos cadenas que mandamos al constructor de usuario el primer carácter es una comilla simple.

b) Debe incluir alguna actividad de mantenimiento correctivo, una de mantenimiento perfectivo y una de mantenimiento preventivo.

Mantenimiento correctivo:

- Hemos detectado un error que provocaba que al poner en un campo de texto el caracter comilla (‘) este provocaba un error en la construcción de la cadena sql, llevando a una ejecución errónea del programa (la base de este error es la misma que la del SQL Injection). El mantenimiento perfectivo ha consistido en prevenir este error limpiando las cadenas antes de construir las consultas

  Una posible implementación que nos arreglaría este problema es limpiar las comillas que traigan las cadenas que recogemos de los campos de texto, como por ejemplo la solución implementada en el constructor de la clase usuario:

```
public Usuario(String clave,String nombre){
        this.nombre=nombre.replace("\'", "");
        this.clave=clave.replace("\'", "");
}
```

(Tras la contrabarra está el carácter ” ‘ “)

Mantenimiento perfectivo:

- En el proceso de diseño y ejecución de las pruebas del apartado a) hemos detectado que habia una sección de código al que no se accedía en ningún caso. Esta se encontraba en la clase usuario concretamente eran los métodos get de los diferentes atributos. Tras analizar las causas del porqué no se recorre esa sección de código hemos llegado a la conclusión de que se cometió un error en el desarrollo de la práctica, ya que no se muestran los datos de la persona que accede a la aplicación (como venía definido en los requisitos de la aplicación). Se ha corregido el error.

- Hemos realizado un cambio en el servicio de hosting para la base de datos. Esto se debe a un fallo que fue detectado tras terminar el desarrollo de la aplicación y ver que tras unos días la aplicación dejaba de conectar con la base de datos almacenada en el hosting. La solución adoptada fue utilizar SQLite para evitar contratar un hosting.

- El mantenimiento correctivo consistirá en corregir este error realizando la implementación de los requisitos pertinentes.

Mantenimiento preventivo:

- En cuanto al mantenimiento preventivo hemos decidido realizar las siguientes tareas:
    - Mejorar la estructuras de los paquetes del proyecto mejorando la compresión y facilitando futuras tareas de mantenibilidad.
    - Mejorar la estructuración de código así como un renombrado de ciertas variables para mejorar la comprensión y facilitar el mantenimiento.
    - Mejorar los nombres de las clases de la aplicación evitando "_" y otros nombres poco profesionales.