Práctica 2. TBD, Branch by abstraction y feature toggles

Enunciado

Se dispone de una aplicación que gestiona una librería online. Esta librería ofrece un interfaz web y una API REST para la gestión de libros, con sus correspondientes pruebas unitarias, de API REST y Selenium. Se proporciona el código de dicha aplicación como un zip en el aula virtual.

Se desea introducir dos funcionalidades adicionales utilizando el modelo de desarrollo trunk based development.

1. Partición de líneas
Concretamente, los dueños de la librería nos han pedido que las descripciones de los libros se guarden en la base de datos separada por líneas con un número de caracteres máximo en cada línea. Es decir, como ocurre con un procesador de textos cuando se llega al final de la línea, que se empieza una nueva. Las palabras no se parten, si no cabe la última palabra de la línea, pasa a la línea siguiente. No obstante, si la línea es tan pequeña que una palabra no cabe, esa palabra se partirá. Para que se sepa que la palabra se ha partido, se deberá poner un guión como último carácter de cada línea donde se parte una palabra larga. Además, nos han pedido que que hagamos un procesamiento de los espacios de la siguiente forma:

- Sustituir varios espacios consecutivos entre dos palabras por un único espacio.
- No permitir espacios al final de una línea.
- No permitir espacios al principio de una línea.

El cliente no tiene claro todavía qué longitud de línea establecer, así que nos han pedido que la aplicación esté preparada para cualquier tamaño de línea, aunque de momento será de 10. Nunca será menor que 2.

Este servicio de partición de líneas y sus correspondientes tests se proporcionan como dos clases separadas que pueden descargarse del aula virtual. Es decir, no hay que implementarlo. Lo único que hay que hacer es usar este servicio desde el BookService de la aplicación para separar en líneas la descripción del libro antes de ser guardado en la base de datos.

2. Notificación de cambios mediante eventos Spring
Actualmente los eventos se escriben en el log de forma síncrona. Se desea añadir la posibilidad de escribir los eventos en el log de forma asíncrona utilizando el mecanismo de eventos de Spring[1].

1 https://docs.spring.io/spring-framework/docs/current/reference/html/core.html#context-functionality-events

La práctica consiste en **introducir ambas funcionalidades en la aplicación utilizando** de forma escrupulosa el modelo de desarrollo **trunk based development**, sin ramas, teniendo en cuenta las siguientes consideraciones:

- Ambas funcionalidades deben estar controladas por un flag. Se recomienda usar ff4j, pero cualquier mecanismo es válido.
- Estos flags deben permitir activar o desactivar estas funcionalidades, concretamente:
  - Cuando la partición de líneas está activada, al crear un nuevo libro, su descripción se parte en líneas si supera la longitud dada (10).
  - Cuando la notificación basada en eventos está activada, las acciones se envían como eventos, y un listener las recibe y las escribe en el log.
- La funcionalidad deberá incluirse como requiere trunk based development, y toda versión debe ser funcional.
- La funcionalidad se implementará en los siguientes commits:
  1. Inclusión del flag correspondiente al line breaker (de momento desactivado). Este commit marca el comienzo del desarrollo de esta funcionalidad.
  2. Inclusión del fichero de tests de linebreaker en el proyecto (estos tests no pasarán, dado que no está el código que implementa la funcionalidad, todos los demás tests siguen pasando)
  3. Inclusión del flag correspondiente al mecanismo de notificación basado en eventos (desactivado de momento). Este commit marca el comienzo del desarrollo de esta funcionalidad.
  4. Inclusión de los ficheros con la funcionalidad de line breaker en el proyecto (los tests de linebreaker pasan)
  5. Inclusión de las clases necesarias para permitir la notificación vía eventos.
  6. Modificación de las clases necesarias para permitir la notificación basada en eventos (uso del flag), pero de momento manteniendo la funcionalidad desactivada.
  7. Modificación de BookService para llamar al servicio de linebreaker, si el flag está activado, antes de guardar el objeto. Se activa este flag. Todos los tests pasan.
  8. Activación del flag de notificaciones basadas en eventos.
- Dado que cada commit es estable y puede ir a producción, no se suelen mantener de manera estricta las versiones maven, sino que las versiones vienen dadas por el commit que se despliega. Se valorará que se añada a la aplicación información sobre el commit del que salió su construcción. Para ello, recomiendo la lectura: https://www.baeldung.com/spring-git-information.
- La aplicación debe funcionar con cualquiera de los dos flags activado o desactivado. Es decir, los 4 posibles estados deben ser funcionales y hacer lo que corresponde.

Formato de entrega

Toda la práctica se debe desarrollar en un repositorio GitHub privado, cuyo nombre será el del identificador del correo del alumno (sin @urjc.es), al que se invitará al profesor (usuario GitHub: gortazar). Concretamente, este repositorio debe tener las propiedades exigidas por el modelo de desarrollo trunk based development, los cambios realizados deben hacerse siguiendo este modelo, y los profesores evaluarán que el modelo se haya seguido

concienzudamente. Por ello, no se deberá borrar ninguna rama.

Adicionalmente, la práctica se entregará por el aula virtual teniendo en cuenta los siguientes aspectos:

- Las prácticas se podrán realizar de forma individual o por parejas. En caso de que la práctica se haga por parejas sólo será entregada por uno de los alumnos.
- La práctica se entregará como un fichero .zip del repositorio GitHub. El nombre del fichero .zip será el usuario URJC del alumno. En caso de dos alumnos, el nombre del zip será el usuario URJC separado por guión (p.perezf-z.gonzalez.zip)
- En el fichero pom.xml se deberá incluir el siguiente nombre del proyecto (donde nombre.alumno corresponde con el identificador del alumno o los alumnos):

```
<artifactId>nombre.alumno</artifactId>
```