Facultad de Informática – Universidad Nacional de La Plata
Cátedra de Bases de Datos II
https://groups.google.com/forum/?hl=es#!forum/bbdd2_alumnos_2016

Proyecto Integrador

Información general

Fecha de publicación: Viernes 18 de Marzo 2016
Carácter: grupal (2 integrantes)
Completar el siguiente formulario para registrar el grupo de trabajo:
https://docs.google.com/forms/d/17m8IGbwNbDDRq--Qt0rE4Dbui7tUX-jijn0MER_6ipQ/viewform?usp=send_form

Características generales
El proyecto integrador, consta de las siguientes etapas:

- ETAPA I: Implementación usando Java de la especificación brindada.
  - Para poder validar la implementación realizada, se provee una serie de tests que deberán pasar.
- ETAPA II: Mapeo usando Hibernate.
- ETAPA III: Implementación de consultas usando HQL.

Software y versiones
Para configurar sus equipos, pueden seguir las instrucciones detalladas en este sitio:
http://catedras.s3.amazonaws.com/bbdd2-2016/instructions.html

ETAPA I: Implementación y testeo del modelo en Java

Fecha de entrega: 11 de Abril

Descripción del modelo

El proyecto consiste en una aplicación colaborativa para el aprendizaje de idiomas. Los usuarios pueden tomar cursos de diversos idiomas. Para aprobar un curso en particular, el usuario debe haber aprobado todas las lecciones correspondientes a dicho curso.

Además, los usuarios a su vez pueden ayudar a la traducción de pequeños extractos de documentos (párrafos) de un idioma a otro. Las traducciones realizadas por los usuarios son revisadas por moderadores que califican dicha traducción con un puntaje determinado.

La cátedra proporciona una especificación compuesta de un diagrama de clases UML y una especificación textual del protocolo de las clases. El diagrama de clases puede ser encontrado en “Modelo.pdf”, dentro del directorio “Práctico Integrador -ETAPAS-/ETAPA I”. La especificación textual complementaria se presenta a continuación.

NOTA: el diagrama de clases debe ser implementado en su totalidad, respetando el protocolo y los tipos de datos.
NOTA II: la especificación incluida a continuación se enfoca en el protocolo mas relevante de algunas clases, el comportamiento de ciertos métodos que aparecen en el diagrama de clases, y no en la especificación, es evidente.

Clase	**Método**	**Comportamiento**
Sitio	public void registrarUsuario(Usuario)	Agrega al usuario recibido como parámetro en la colección de usuarios.
	public void agregarCurso(Curso)	Agrega al curso recibido como parámetro en la colección de cursos.
	public void agregarDocumento(Documento)	Agrega al documento recibido como parámetro en la colección de documentos.
Usuario	public Usuario(String email, String nombre, Date fechaDeCreacion)	Constructor, recibe por parámetro el email, el nombre y la fechaDeCreacion.
	public int nivel(Idioma)	Retorna el nivel máximo entre las cursadas aprobadas por el usuario del Idioma recibido por parámetro.
	public Collection<Cursada> cursadasAprobadas(Idioma)	Retorna todas las cursadas aprobadas cuyos cursos son del idioma recibido por parámetro.

Moderador	public Moderador(String email, String nombre, Date fechaDeCreacion)	Constructor, recibe por parámetro el email, el nombre y la fechaDeCreacion.
	public int reputación()	Retorna la reputación del moderador, que se calcula según la cantidad de evaluaciones realizadas.
	public void evaluar(Traduccion, String, Integer)	Registra una nueva evaluación (terminada) para la Traduccion recibida por parámetro en la colecciones de evaluaciones. El puntaje y la descripción de la evaluación son recibidos por parámetro. En este caso, se toma el día actual como fecha de la evaluación.
	public boolean manejaIdioma(Idioma)	Retorna un booleano indicando si el moderador tiene al Idioma recibido por parámetro entre los idiomas que maneja.
Cursada	public Cursada(Curso curso, Date inicio, Usuario usuario)	Constructor, recibe el curso, la fecha de inicio y el usuario como parámetros.
	public boolean finalizada()	Retorna un booleano indicando si la cursada esta finalizada. Esto significa que existe al menos una prueba (aprobada) para cada lección del curso correspondiente.
	public Collection<Leccion> leccionesAprobadas()	Retorna una colección con todas las lecciones

		del curso correspondiente para las que existe una prueba aprobada.
	public int getNivel()	Retorna el nivel del curso correspondiente.
Prueba	public Prueba(Leccion leccion, Integer puntaje)	Constructor, recibe la lección y el puntaje por parámetro.
	public boolean aprobada()	Retorna un booleano indicando si la prueba está aprobada. Se considera aprobada una prueba cuando el puntaje de la misma es mayor o igual que 60.
Traduccion	public Traduccion(Date fecha, String descripcion, Boolean completa, String texto, Parrafo parrafo, Idioma idioma)	Constructor, recibe como parámetros la fecha de la misma, su descripción, si esta completa, el texto de la traducción, el párrafo traducido y el idioma al que se tradujo.
	public Idioma getIdiomaOriginal()	Retorna el idioma original en el cual el párrafo traducido fue escrito.
Documento	public Documento(String nombre, Idioma idioma)	Constructor, recibe por parámetro el nombre y el idioma en el que está escrito.
	public Parrafo agregarParrafo(String)	Recibe un string por parámetro que debe ser utilizado para agregar un Parrafo en la colección de párrafos. El nuevo párrafo creado debe ser retornado.

Tener en cuenta para la implementación:

- Usar Collection al momento de *tipar* colecciones, luego se instanciará con una clase concreta. Por ejemplo:

```
private Collection<Usuario> usuarios = new HashSet<Usuario>()
```

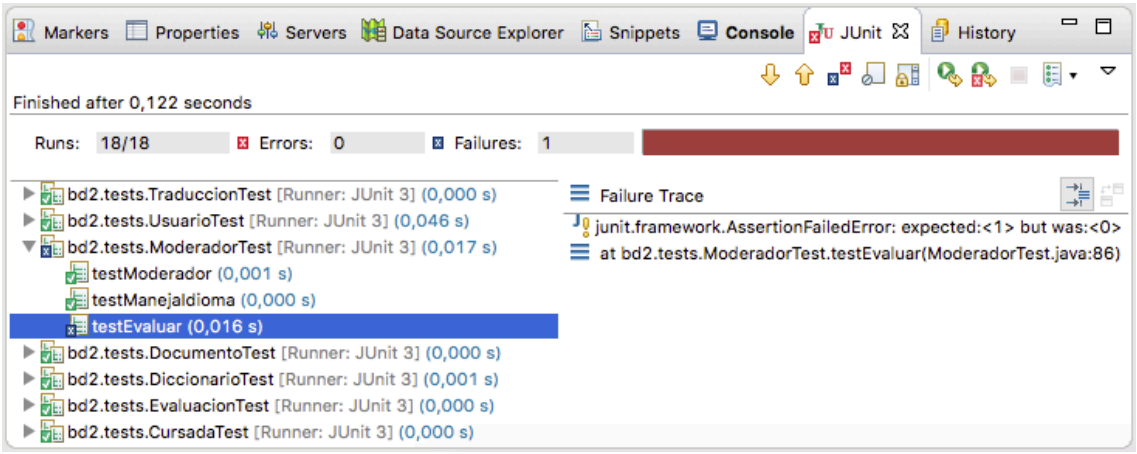
Testing

Una vez implementado el modelo, ejecutar los tests provistos en el proyecto. Para correr los tests basta con presionar el botón derecho sobre un método, una clase o incluso un paquete, y seleccionar:

Run As -> JUnit Test

En la ventana de JUnit se mostrarán los resultados (en caso de no ver la ventana, se puede abrir desde Window -> Show View -> Other... -> JUnit).

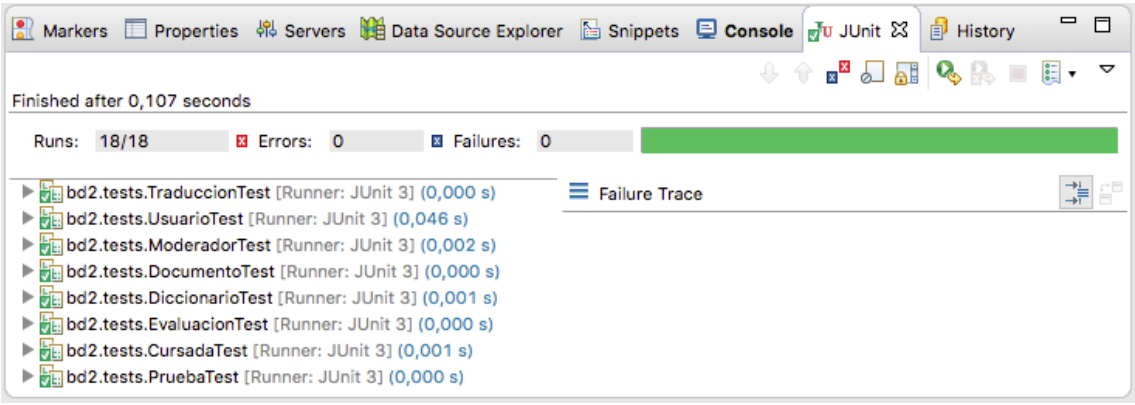
Si algún test falla aparecerá el indicador en rojo y, en el reporte, se destacarán los que fallaron con un ícono azul. En estos casos, se muestra también el Failure Trace, un detalle de las pruebas específicas (assertions) que fallaron, como muestra la siguiente figura.



En el ejemplo que se ve en la figura, el test del método evaluar() de la clase Moderador falló (puede verse en el ícono azul), y en el Failure Trace se indica el resultado esperado en contraste con el obtenido de la siguiente manera:

junit.framework.AssertionFailedError: expected:<1> but was:<0>

Para ir directamente al código que produjo el fallo se puede hacer doble click en el método dentro del listado del reporte. Una vez corregidos los errores, al correr nuevamente los tests el indicador debería quedar en verde.



Consideraciones contempladas para la aprobación de la Etapa I

1. Respetar nombre del proyecto y paquetes
2. Las clases y métodos deberán estar documentados
3. El comportamiento debe ser el solicitado
4. La implementación debe respetar la especificación dada
5. Los test deberán ejecutar y pasar.
    a. *Importante: Los test no deben ser alterados.*
6. La entrega se debe realizar a término
7. Al menos un integrante del grupo debe asistir a la devolución

---

ETAPA II: Mapeo en Hibernate

El documento de esta etapa se publicará oportunamente

ETAPA III: Consultas HQL

El documento de esta etapa se publicará oportunamente