Trabajo Práctico Especial

Segunda iteración

Objetivo

El objetivo de la segunda iteración del trabajo práctico especial es realizar modificaciones en la aplicación ya implementada en base a nuevos requerimientos técnicos y funcionales.

Tecnología a utilizar

- La aplicación debe poseer una arquitectura según **Domain Driven Design**, aplicando el patrón de inyección de dependencias a través de **Spring**.
- Se deben escribir **tests** de los objetos del modelo de dominio (no es necesario testear repositorios).
- La capa web debe estar implementada utilizando los patrones MVC y Front Controller, a través de **Spring MVC**. La vista debe estar compuesta por archivos **JSP** con **JSTL** (no deben contener código Java).
- Para la persistencia de los objetos de dominio se debe utilizar **Hibernate** (no se deben ejecutar consultas mediante la API de **JDBC**).
- Para la administración y construcción del proyecto se debe utilizar **Maven**.
- La aplicación debe almacenar los datos en una base de datos **PostgreSQL**. El nombre de la misma debe ser pawN, donde N es el número de grupo. El usuario para que la aplicación se conecte debe ser **paw**, y la contraseña **paw**. El *encoding* de la base de datos debe ser **UTF-8** y el *locale* **en_US**.
- El versionado de código debe realizarse mediante el repositorio **GIT** provisto por la cátedra.

Requerimientos

A continuación se detallan los requerimientos obligatorios a desarrollar. La implementación correcta de los mismos otorgará una nota máxima de 8 puntos. Además, se cuenta con requerimientos opcionales adicionales que otorgan puntaje extra. De acumular un puntaje total mayor a 10, la diferencia se acumulará para la próxima iteración (luego de haberla aprobado).

Quienes realicen el trabajo en forma individual, no considerar los requerimientos obligatorios del 5 al 9. Es decir, implementando correctamente los requerimientos del 1 al 4 inclusive se obtiene una nota máxima de 8 puntos. Los opcionales son los mismos que para el caso grupal y tienen los mismos puntajes.

Un requerimiento opcional prometido y no implementado al cierre del sprint descontará el 50% de su valor al puntaje total.

Cada grupo debe enviar un mail a la cátedra antes del jueves 16 de mayo a las 22hs, indicando si va a implementar requerimientos opcionales y cuáles son.

En caso de considerar que deben agregar o quitar requerimientos durante la iteración deben consultarlo con la cátedra, que determinará si corresponde o no hacerlo, y si implica alguna penalización

Todos los requerimientos prometidos y/o implementados por el grupo en el sprint anterior deben funcionar correctamente, y deben responder a las observaciones hechas en las devoluciones del trabajo (incluyendo aquellas que no tienen una penalización).

#	Requerimientos obligatorios
1	Se debe permitir que un restaurante esté relacionado con más de un tipo de comida.
2	Los usuarios registrados deben poder solicitar el alta de nuevos restaurantes. Para esto, deben completar un formulario con todos los datos del mismo. La solicitud queda pendiente a ser aprobada (no se puede acceder al detalle del restaurante ni aparece en ningún listado).
3	Los usuarios administradores deben poder consultar el listado de solicitudes pendientes, ordenado por fecha de solicitud descendentemente. Para cada una se debe mostrar el nombre del restaurante, el usuario que la envió, la fecha y un link para ver el detalle. Cuando se ingresa al detalle, se deben poder ver todos los datos del restaurante, y elegir aceptar o rechazar la solicitud. Si se acepta, se crea un nuevo restaurante en el sistema, y la solicitud se archiva en estado “aceptada”. Si se rechaza, no se crea el restaurante y la solicitud se archiva en estado “rechazada”. En ambos casos desaparece del listado de solicitudes pendientes, pero se mantiene almancenada en la base de datos.
4	En el detalle de un restaurante, los usuarios registrados deben poder indicar si un comentario les resultó útil o no. Para esto debe haber dos links en cada comentario. El rating de un comentario es la resta entre la cantidad de usuarios que marcaron ese comentario como útil y la cantidad de usuarios que lo marcaron como no útil. Los comentarios en el perfil de un restaurante deben ser ordenados según ese rating, que debe ser mostrado para cada comentario. Un usuario puede especificar que un comentario no le resultó útil después de marcarlo como útil, y vice versa; pero no puede “desmarcarlo”.
5	Los usuarios registrados deben tener un perfil en donde se muestre su nombre, fecha de registración en el sistema, y las calificaciones realizadas, así como los comentarios asociados a esas calificaciones. El perfil se debe poder acceder haciendo click en el nombre del usuario.
6	Los usuarios administradores deben poder ver un listado de los usuarios registrados en el sistema, con nombre de usuario, email y fecha de registración en el sistema.
7	Los usuarios administradores deben poder marcar o desmarcar a otro usuario del sistema como administrador, desde el perfil del usuario a modificar. Los usuarios administradores no pueden desmarcarse a sí mismos como administradores.
8	Los usuarios administradores deben poder borrar comentarios hechos por cualquier usuario.
9	Cuando un restaurante se crea a través de una solicitud aprobada, debe aparecer el nombre del publicador del restaurante en el perfil de éste. En el perfil de un usuario, además, se debe poder ver un listado con los restaurantes publicados por ese usuario.

#	Requerimientos opcionales	Puntaje
1	En el caso de que ocurra un error fatal en la aplicación, se debe enviar un mail a alguna dirección de correo con el stack trace del mismo.	0.75
2	En el perfil de cada restaurante, se debe mostrar un máximo de 6 comidas ofrecidas en el mismo. Las comidas deben tener una imagen, un nombre y un precio asociado, y deben poder ser dadas de alta o borradas por el o los usuarios administradores.	1.75
3	En el perfil de un restaurante, deben aparecer hasta tres restaurantes recomendados. Los restaurantes recomendados deben ser aquellos preferidos por usuarios que calificaron en forma positiva el restaurante actual.	1.5

Material a entregar

El código fuente se debe entregar a través del repositorio asignado por la cátedra.

El repositorio debe contener como mínimo:

- branch **master**: Se deberá utilizar durante el desarrollo del proyecto. Su contenido debe ser un proyecto Maven, no debe contener código compilado.
- tags para cada entrega: El repositorio debe contener por lo menos un tag por cada entrega. Para la segunda iteración el tag debe llamarse **sprint2**.

Dentro del directorio **src/test/resources** deben existir los archivos **update.sql** e **insert.sql** que actualizan el esquema de base de datos e insertan nuevos datos de prueba respectivamente. Los datos que debe cargar este script deben ser suficientes como para poder mostrar el correcto funcionamiento de los requerimientos implementados.

Para que la entrega se considere válida, se debe poder ejecutar *mvn clean package*, ejecutar los scripts de actualización e inserción de datos y hacer un deploy en Tomcat.
Dado que consideramos que la aplicación del primer sprint ya está en funcionamiento, los scripts de actualización deben estar escritos de modo tal que adapten toda la información existente al nuevo esquema de base de datos.

Fecha de entrega

El proyecto se debe entregar a través del repositorio **antes del jueves 30 de mayo a las 18:00 hs**.

Evaluación

En cada iteración se evaluará el grado de cumplimiento de los requerimientos (en base a la estimación inicial y cambios autorizados) y la calidad del desarrollo (funcionamiento, usabilidad, diseño de objetos).

Consultas

Las consultas se realizarán en el horario de laboratorio, o por correo electrónico a la dirección paw@it.itba.edu.ar.