Capítulo VI.

Pruebas

Como es bien sabido, nada en este mundo puede ser confiable si no se ha probado de alguna manera metodológica su correcto funcionamiento. Lo mismo aplica en el mundo del software y para eso existen varios niveles de pruebas a las que se puede someter una aplicación.

El nivel más básico de pruebas es el famoso "*Unit Testing*" (Pruebas de Unidad), en las cuales se someten las unidades mínimas del software (clases, componentes, módulos, etc.) a una rutina que verifica que los resultados sean los esperados. En el Sistema Asistente para la Generación de Horarios de Cursos se realizaron dichas pruebas de manera manual sin hacer uso de algunos *frameworks* especializados como JUnit. Cada clase de Java fue probada meticulosamente de manera aislada verificando la validez de sus operaciones y cada error encontrado fue solucionado de raíz sin dejar cabida a "parches" que posteriormente podrían representar una debilidad en el sistema.

Las pruebas de unidad realizadas a las entidades consistieron en primer lugar, crear objetos que no tuviesen relaciones con otras entidades, persistirlas a través de Hibernate y verificar que los datos fueran correctamente guardados en MySQL usando el software MySQL Query Browser. Una vez que todas las entidades fueron verificadas individualmente, se procedió a verificar que su recuperación fuese igualmente exitosa. Posteriormente se realizo el mismo procedimiento creando objetos relacionados con otros a través de composiciones, colecciones, etc. Cuando se verificó la información almacenada en MySQL se encontraron algunos casos donde las entidades relacionadas con un objeto no eran persistidas automáticamente al indicar a Hibernate que debía de guardar el objeto

principal. Gracias a este tipo de situaciones se pudo tener una mejor comprensión del mecanismo de persistencia de Hibernate y se solucionaron los problemas en los que se incurría incluso en la etapa de recuperación de árboles de relaciones complejos y situaciones de recursividad.

Los DAOs fueron probados en manera muy similar, creando instancias de cada tipo de DAO en el sistema, alimentándolos con las entidades adecuadas y corroborando manualmente su buen funcionamiento en cuanto a la comunicación con Hibernate para almacenar, eliminar y actualizar entidades. El principal problema que se encontró al probar los DAOs fue la dificultad para depurar el funcionamiento interno de Hibernate en su proceso de recuperación de objetos cuando alguna instrucción de HQL fallaba, pero gracias a la excelente documentación de Hibernate y sus foros de ayuda fue posible solucionar todo sin recurrir a llamados SQL que hubieran roto el esquema objetos persistentes.

Como ya se explicó anteriormente, la estructura de los Validators consiste en pruebas incrementales para verificar la validez de las entidades antes de ser persistidas. Las pruebas siguieron el mismo esquema proporcionando objetos diseñados para romper cada una de reglas hasta tener un objeto completamente válido y listo para ser almacenado en la base de datos. En el caso de los Validators, no sólo era importante que el resultado de la validación fuera correcta, sino que en caso de que resultara negativa, la lista de errores correspondiera perfectamente a cada problema encontrado.

Continuando con las pruebas, se procedió a comprobar el funcionamiento del sistema de generación de sugerencias. Se crearon *datalogs* que contuvieran información de secciones, preferencias, salones y profesores que limitaran las opciones del sistema para entregar resultados. Si el modelo generado era exactamente el modelo esperado se continuaba generando escenarios más complejos que pudieran tener soluciones o incluso

aquellos donde no hubiera. El sistema funcionó tan adecuadamente que incluso encontró soluciones en situaciones que se consideraban imposibles pero resultaron ser completamente válidas.

El siguiente nivel de pruebas es el de integración. En esta fase, se prueban las relaciones entre las interfaces y los componentes del sistema. En otras palabras, se verifica la correcta interacción entre los componentes gráficos, lógicos y de datos del sistema para comprobar que todo funciona como un sistema bien acoplado. He aquí donde entran las pruebas a los Servlets que fungen como Controladores.

Las primeras pruebas a los Servlets consistieron en simplemente ver si eran capaces de redirigir las peticiones a las clases que controlaban la lógica de negocios adecuadamente. Hasta aquí todo parecía ser perfecto ya que la comunicación de la solicitud del cliente era dirigida adecuadamente y la respuesta era regresada al Controlador para ser finalmente entregada a la Vista; sin embargo durante el despliegue de información en la Vista se encontraron muchos problemas como que algunos despliegues estaban fuera de sincronización con la base de datos o que no se podía acceder a las relaciones de las entidades. El último problema desapareció cuando se introdujo el Hibernate Filter que ya fue explicado anteriormente y el primer problema se solucionó cuando se entendió plenamente Hibernate maneja a los objetos que aún no han sido preservados en la base de datos a través de un *commit* pero se encuentran en el espacio de objetos de Hibernate como si lo estuvieran ya que sólo esperan la instrucción para ser bajados a la base de datos.

Una vez que los problemas mencionados quedaron resueltos se procedió con las pruebas de cada interfaz gráfica a la que los usuarios tendrán acceso verificando que el reporte de errores en la generación de alguna Entidad fuera el correcto en caso de existir problemas, verificando que la lista de entidades desplegadas fuera exactamente la misma

que la encontrada en la base de datos y finalmente que las notificaciones de éxito en las creaciones, modificaciones y eliminaciones fuera correcto.

Una vez que se terminaron de verificar todos los puntos anteriores, se decidió que era necesario simular el uso del sistema en el ambiente real de trabajo. Con dicho propósito se procedió a generar en el sistema todas las secciones ofrecidas en el semestre de Otoño 2007 para la carrera de Ingeniería en Sistemas Computacionales. Se decidió limitar las pruebas a una sola carrera porque en ella se pueden comprobar todas las condiciones para las que el sistema fue diseñado (como validación de traslapes de horarios de salones y profesores, preferencias de profesores, planes de estudio, etc.).

Para iniciar esta prueba era necesario que se contara con toda la información necesaria respecto a la carrera en cuestión. Por lo tanto se procedió a ingresar todos los salones en los que se impartieron clases de la carrera durante el semestre de Otoño 2007 y asociarlos con la carrera; todos los profesores que impartieron clases durante el mismo semestre; toda la información de los planes de estudio ISC 2002 y 2007; y finalmente se crearon preferencias de profesores usando conocimiento propio respecto a sus gustos por las diferentes áreas de la carrera de manera semi-aleatoria.

Esta prueba, como se esperaba, fue exitosamente finalizada ya que la entrada manual de horarios, salones y profesores, tal y como fueron impartidas las clases fueron permitidas y las sugerencias, cuando fueron solicitadas, siempre incluían una alternativa de sección que se igualara con la realidad.