Evaluación de septiembre
Procedimiento de evaluación

- Con respecto a la gestión, comprobarán que:
    - Hemos seguido las instrucciones de entrega provistas en el documento “On your deliverables”.
    - El proyecto se gestionó correctamente. Prestarán especial atención a comprobar que las tareas tienen sentido, que no se crearon todas juntas, y que seguimos las guías de los profesores.
- Con respecto a las leyes, comprobarán que:
    - El proyecto sigue la LOPD. La única excepción se hará con respecto a las comunicaciones seguras, a menos que queramos un A+.
    - El proyecto sigue la LSSI. La única excepción se hará con respecto a informar a la Cámara de Comercio sobre nuestro dominio de internet.
    - El proyecto sigue la ley de transposiciones.
- Con respecto a la documentación, comprobarán que:
    - Los costes están documentados de forma que tenga sentido.
    - El código tiene comentarios donde sea apropiado, no son triviales y tienen sentido.
- Con respecto a los proyectos Eclipse/Maven, comprobarán que:
    - Hemos instanciado y customizado la plantilla de proyecto de acuerdo con las guías provistas. Prestarán especial atención a que la base de datos y las URLs concuerden con el nombre del proyecto.
    - Hemos customizado la plantilla de proyecto para adaptarla a las regulaciones españolas.
- Con respecto a los modelos de dominio, comprobarán que:
    - Usamos el vocabulario del cliente.
    - El modelo conceptual representa los requisitos adecuadamente.
    - El modelo conceptual no tiene ningún atributo void.
    - El modelo conceptual no tiene scaffolding.
    - El modelo UML no tiene artefactos no soportados por Java.
    - El modelo UML no tiene relaciones que son ineficientes de implementar.
    - El modelo UML no tiene scaffolding.
    - El modelo Java es limpio y eficiente.
    - El modelo Java incluye todas las anotaciones necesarias para representar restricciones implícitas en el modelo UML.
    - Usamos tipos wrapper y primitivos correctamente en el modelo Java.
    - Usamos @Valid y @NotNull correctamente.
- Con respecto al modelo de persistencia, comprobarán que:

    - Representa al modelo UML adecuadamente.
    - El “PopulateDatabase.xml” especifica los suficientes objetos de cada tipo, y que hay suficiente variedad, p.ej. si la entidad A puede estar relacionada con varias entidades tipo B (0..1, 0..*, 1..*), entonces comprobarán que PopulateDatabase.xml tiene una entidad A relacionada con 0 B, una entidad A relacionada con una B, y así.
    - La utilidad “PopulateDatabase.java” se puede ejecutar desde Eclipse y persiste las entidades correctamente.
    - La utilidad “QueryDatabase.java” puede ejecutar las queries JPQL y devuelven los resultados esperados, que se deben documentar propiamente.
- Con respecto a repositorios y servicios, comprobarán que:
    - Las queries son simples e implementan las semánticas deseadas correctamente. Mirarán los requisitos funcionales y comprobarán que se pueden implementar con las queries de los repositorios.
    - Los servicios están declarados como transaccionales, no hay miembros o atributos estáticos aparte de los repositorios y servicios autowired requeridos, ningún servicio gestiona un repositorio que no sea el que debe gestionar, y las reglas de negocio se implementan correctamente. Revisarán los requisitos funcionales para comprobar que los servicios permiten una implementación correcta.
    - Hemos escrito tests para cada servicio y método, y tienen sentido. Esperan que escribamos tests positivos y negativos.
    - Hemos comprobado que no se pueda hacer GET o POST hacking en la página principal.
    - Hemos añadido los métodos adecuados para reconstruir objetos de dominio a partir de formularios u objetos borrados y usamos validadores correctamente.
    - Hemos definido índices apropiados para las entidades de dominio de acuerdo con las queries.
- Con respecto a las vistas, comprobarán que:
    - Hemos actualizado el modelo Java con anotaciones @DataTimeFormat.
    - Hemos seguido las guías para implementar vistas.
    - Hemos reutilizado vistas apropiadamente.
    - Los ficheros i18n&l10n son correctos.
    - La configuración Apache Tiles combina las vistas con la página master apropiadamente.
    - Hemos hecho nuestro mejor intento de prevenir POST hacking usando formularios y borrado de objetos adecuado.
    - Hemos usado tags para hacer las vistas más compactas y menos dadas a error.
    - Los formularios no pueden sufrir cross scripting.
- Con respecto a los controladores, comprobarán que:

    - Los controladores dependen de los servicios apropiados e implementan bien los patrones de listado y edición.
    - Los ficheros de configuración de seguridad están bien configurados.
    - Los controladores manejan formularios y borrado de objetos propiamente.
- Con respecto al sistema, comprobarán que:
    - Usamos las credenciales del tipo “authority1/authority1”, “authority2/authority2” y así para loguear, donde “authority” se refiere a cualquier autoridad del sistema. Las únicas excepciones son las de tipo “admin/admin”, que se usarán para acceder al sistema como administrador.
    - Todas las vistas funcionan en inglés y español.
    - Ningún formulario permite meter datos inválidos, p.ej. dejando campos en blanco que corresponden a atributos no opcionales, o metiendo fechas o precios inválidos.
    - Si un formulario tiene errores de validación y pulsamos “save”, la información permanece en el formulario.
    - Los formularios funcionan bien tras meter datos inválidos, es decir, los botones “save”, “delete” y “cancel” funcionan.
    - El botón “cancel” funciona bien en todos los formularios.
    - Los enlaces de paginación funcionan correctamente. Es muy importante que PopulateDatabase.xml provee los suficientes objetos para que la lista supere el límite de paginación. Suponen que pondremos 5 objetos por página por defecto.
    - No es posible hackear las URLs. Intentarán editar datos que pertenecen a otros usuarios. También intentarán acceder a URLs no permitidas.
    - Todos los requisitos de información, funcionales y no funcionales están implementados correctamente.
- Con respecto al despliegue, comprobarán que:
    - El script crea la base de datos. Prestarán especial atención a que se ejecuta en el contexto de una transacción, los usuarios MySQL adecuados se crean y se asignan privilegios apropiados, el script no mete ningun dato superfluo ni de pruebas, pero sí los datos necesarios para crear una cuenta “admin/admin” predefinida con privilegios y cualquier otra información que el sistema necesita.
    - El script para borrar la base de datos. Se prestará especial atención a comprobar que los usuarios MySQL y sus permisos se borran completamente.
    - El artefacto war. Prestarán especial atención a que no incluya el código fuente Java.
    - Ejecutarán el script para crear la base de datos en el entorno de pre-producción.
    - Subirán el artefacto war al servicio Tomcat en el entorno de pre-producción.

    - Los requisitos funcionales descritos en el proyecto funcionan como se espera y no hay problemas cuando la aplicación corre en el dominio "www.acme.com".
- Con respecto al testing funcional, comprobarán que:
    - Hemos seguidos las guías sobre cómo organizar casos de test, clases de test y suites.
    - Ejecutarán la suite y comprobarán que JUnit no informa de excepciones, es decir, debe mostrar una barra verde.
    - Los casos de test están propiamente documentados.
    - Los tests son completamente automáticos, es decir, no imprimen información en consola, y las comprobaciones se hacen con los assert adecuados.
    - La cobertura es lo bastante buena
- Con respecto al testing de rendimiento, comprobarán que:
    - Hay al menos un test de rendimiento por cada caso de uso.
    - Los scripts no tienen entradas falsas.
    - Hemos añadido temporizadores para simular retrasos humanos cuando una persona comienza una interacción, no el navegador.
    - El informe con respecto a la máxima capacidad que el sistema puede soportar no muestra ningún error HTTP.
- Con respecto al testing de aceptación, comprobarán que:
    - Hemos seguido las guías para diseñar comprobaciones apropiadas para cada caso de uso
    - Los bugs intencionales tienen sentido.