Entrega: Los días del 3 al 8 de Abril a las 20:00. Los grupos del Lunes 04/04 entregan el 03/04, los grupos del Martes 05/04 entregan el 04/04 y así sucesivamente.

Esta práctica se podrá realizar en equipos de hasta 2 alumnos y su entrega es obligatoria.

Se ha marcado como "tachado" aquello partes suprimidas de la práctica 1 y como subrayado aquello nuevo en esta práctica.

Índice de contenidos

Entrega: Los días del 3 al 8 de Abril a las 20:00. Los grupos del Lunes 04/04 entregan el 03/04, los grupos del Martes 05/04 entregan el 04/04 y así sucesivamente.

Casos de uso

Basándose en el enunciado de referencia de "Share my trip", implementar los siguientes casos de uso siguiendo los patrones arquitectónicos N-capas y MVC al estilo de lo visto en clase en la sesión de laboratorio 6. La implementación del patrón MVC deberá realizarse empleando obligatoriamente JSF 2.2 y Primefaces (última versión). Se deberán implementar los siguientes casos de uso:

Público: registrarse como usuario

Para poder promover o participar en los viajes, el usuario debe estar registrado e identificado.

El sistema le pedirá un identificador (login) que no se podrá repetir en el sistema y datos personales como nombre, apellidos, email y su password (que deberá repetir dos veces).

Público: iniciar sesión

Suministrando su identificador y contraseña, un usuario podrá autenticarse ante el sistema. Sólo los usuarios que proporcionen correctamente su identificador podrán iniciar sesión con éxito. En el caso de que el inicio de sesión tenga éxito, el usuario tendrá acceso a la funcionalidad descrita. En caso de que el inicio de sesión fracase, será necesario mostrar un mensaje de error que dé alguna indicación sobre la naturaleza del problema.

Público / usuario registrado: Consultar viajes

La visualización de los viajes ofertados y sus detalles deberá estar accesible sin necesidad de identificación en el sistema. El sistema mostrará un listado de los viajes activos: que no se han realizado aún, que aún está abierto el plazo abierto y que disponen de plazas. Sí el usuario está registrado podrá ver la identidad~~, puntuación y comentarios sobre el~~ del promotor del viaje y del resto de participantes confirmados del viaje.

• Opción Filtrado1: El usuario podrá filtrar el listado anterior por los campos combinados Origen y Destino al menos para localizar rápidamente los viajes de su interés. Cada nueva tecla que el usuario pulse en el campo de búsqueda deberá de regenerar el listado.

• Opción Ordenacion1: El usuario podrá seleccionar el tipo de ordenación para los viajes: origen, destino, fecha, promotor, etc. Se valorará ordenación combinada de campos.

• Opción Paginación1: El usuario podrá seleccionar el número de filas visualizables.

~~Usuario registrado: modificar datos personales~~

~~Todo usuario registrado podrá modificar sus datos de registro (excepto su identificador) y cambiar su contraseña. Para esta última, el sistema le pedirá que introduzca la antigua y la nueva (esta última, por duplicado).~~

Usuario registrado: Solicitar plaza en un viaje

Una vez que el usuario ha seleccionado un viaje en el que está interesado (Caso de uso "Consultar viajes") podrá solicitar plaza en él. El viaje le aparecerá entonces en el listado de viajes en los que ha tenido alguna implicación (como promotor, participante o interesado).

El usuario podrá cancelar su solicitud de plaza en los viajes aún no cerrados (FechaCierre) haya sido confirmado o no.

• Opción Filtrado2: El usuario podrá filtrar el listado anterior por los campos combinados Origen y Destino al menos para localizar rápidamente los viajes de su interés. Cada nueva tecla que el usuario pulse en el campo de búsqueda deberá de regenerar el listado.

• Opción Ordenacion2: El usuario podrá seleccionar el tipo de ordenación para los viajes: origen, destino, fecha, promotor, etc. Se valorará ordenación combinada de campos.

• Opción Paginación2: El usuario podrá seleccionar el número de filas visualizables

Por cada viaje aparecerá una entrada en el listado que especificará: el viaje, la fecha y el estado de su relación con el viaje (PROMOTOR, PENDIENTE, ADMITIDO, SIN_PLAZA, EXCLUIDO).

~~*El usuario podrá cancelar su solicitud de plaza en los viajes aún pendientes haya sido confirmado o no.*~~

Usuario registrado, promotor: Confirmar pasajeros

Desde el listado de viajes del usuario, para aquellos en los que figure como promotor, y que aún no se han realizado, podrá acceder a un listado de solicitudes en el que se le ofrecerá la posibilidad de confirmar la plaza al solicitante.

Podrá repetir la operación tantas veces como necesite hasta agotar el cupo de plazas disponible para el viaje. Los usuarios que queden confirmados pasarán a ser ADMITIDO (y así lo verán en su listado particular de viajes), el resto de usuarios solicitantes seguirán en estado PENDIENTE hasta que finalmente se agote el cupo; en ese momento automáticamente pasarán a estado SIN_PLAZA.

El usuario promotor del viaje también tendrá la posibilidad de cancelar la participación de un usuario en el viaje (pasará a estado EXCLUIDO), siempre antes de que finalice el límite para apuntarse.

Una vez haya pasado la fecha del viaje el sistema no ofrecerá la posibilidad de hacer más modificaciones a la lista de participantes.

Usuario registrado, promotor: Registrar viaje

El usuario va a realizar un viaje y ofrece las plazas que tenga disponibles en su vehículo. El sistema le ofrecerá la posibilidad de registrar, modificar y cancelar viajes[1]. Los datos a aportar serán:

- Lugar de salida (dirección completa: calle, ciudad, provincia, país, código postal y opcionalmente coordenadas GPS)
- Fecha y hora de salida.
- Destino (dirección completa, como lugar de salida)
- Fecha y hora de llegada estimada.
- Fecha límite para apuntarse.
- Coste estimado del viaje en total (se entiende que éste se repartirá entre los participantes)
- Descripción o comentarios que el promotor desea dar a conocer.
- Número de plazas máximo y plazas disponibles en ese momento.

Opción autocompletado: Los campos Ciudad origen y destino deberán poder seleccionarse de la lista de provincias considerando cada provincia como una ciudad2. La selección deberá realizarse empleando el mecanismo de autocompletado de primefaces.

Precarga de datos

En el formulario de registro de viaje deberá habilitarse un botón (check, radio, ….) que permita cargar una plantilla de valores válidos en el todos los campos. Estos valores pueden ser siempre los mismos.

Cancelación de viajes

La cancelación de viajes deberá poder ser múltiple. Se sugiere el uso de controles de tipo Check.

Cuando un promotor cancele un viaje el estado de los pasajeros pasará directamente a estado "CANCELADO".

Un viaje será cancelable hasta su fecha de cierre.

Usuario registrado: cerrar sesión

Cualquier usuario con la sesión iniciada podrá solicitar la finalización de la sesión.

Pruebas automatizadas

Se deberá suministrar un proyecto Java JUnit con un mínimo de pruebas unitarias empleando el framework Selenium (Se suministrará en el campus virtual un proyecto plantilla de ejemplo).

Las clases de equivalencia mínimas (válidas e invalidas) que deberán realizarse serán:

1. [RegVal] Registro de Usuario con datos válidos.
2. [RegInval] Registro de Usuario con datos inválidos (contraseñas diferentes).

[1] Cancelar viaje no implica borrarlo del sistema, pasa a estado CANCELADO.

[2] https://github.com/alombarte/utilities/blob/master/sql/spain_provincias.sql

3. [IdVal] Identificación de Usuario registrado con datos válidos.
4. [IdInval] Identificación de usuario registrado con datos inválidos.
5. [AccInval] Intento de acceso con URL desde un usuario no público (no identificado). Intento de acceso a vistas de acceso privado.
6. [RegViajeVal] Registro de un viaje nuevo con datos válidos.
7. [RegViajeInVal] Registro de un viaje nuevo con datos inválidos.
8. [EditViajeVal] Edición de viaje existente con datos válidos.
9. [EditViajeInVal] Edición de viaje existente con datos inválidos.
10. [CancelViajeVal] Cancelación de un viaje existente por un promotor.
11. [CancelMulViajeVal] Cancelación de múltiples viajes existentes por un promotor.
12. [Ins1ViajeAceptVal] Inscribir en un viaje un solo usuario y ser admitido por el promotor.
13. [Ins2ViajeAceptVal] Inscribir en un viaje dos usuarios y ser admitidos los dos por el promotor.
14. [Ins3ViajeAceptInval] Inscribir en un viaje (2 plazas máximo) dos usuarios y ser admitidos los dos y que un tercero intente inscribirse en ese mismo viaje pero ya no pueda por falta de plazas.
15. [CancelNoPromotorVal] Un usuario no promotor Cancela plaza.
16. [Rech1ViajeVal] Inscribir en un viaje un usuario que será admitido y después rechazarlo por el promotor.
17. [i18N1] Cambio del idioma por defecto a un segundo idioma. (Probar algunas vistas)
18. [i18N2] Cambio del idioma por defecto a un segundo idioma y vuelta al idioma por defecto. (Probar algunas vistas)
19. [OpFiltrado] Prueba para el filtrado opcional.
20. [OpOrden] Prueba para la ordenación opcional.
21. [OpPag] Prueba para la paginación opcional.
22. [OpMante] Prueba del mantenimiento programado opcional.

Para cada una de estas pruebas se debe añadir al menos un caso de prueba y en caso querer añadir más casos se numerará la prueba (por ejemplo, Rech1ViajeVal, Rech1ViajeVal2, RechViajeVal3….).

En caso de desear incluir más se deberán incluir al final de la clase de pruebas JUnit.

Todas las pruebas deben incluir el click en las opciones de menú correspondientes, etc, etc, igual que lo haría un usuario real.

El protocolo de prueba

Para probar cada proyecto el profesor realizará los siguientes pasos:

a) Descomprimirá el zip suministrado por el alumno.

b) Copiará el archivo war a la carpeta deployments del servidor wildflyext (sólo el archivo war).

c) Lanzará la base de datos del proyecto suministrado por el alumno (sdi2-n/data/startup.bat).

d) Importará el proyecto sdi2-test en eclipse.

e) Ejecutará el proyecto sdi2-test.

Opcional: Mantenimiento programado de la BASE DE DATOS

Dado que un vez un viaje supera su fecha/hora de cierre puede que haya usuarios que queden en estado PENDIENTE cuando debería ser SIN_PLAZA, se deberá crear un proceso programado que actualice la base de datos en este sentido con la mayor eficiencia posible. Como tecnología a emplear se sugiere el uso de la clase Timer de Java. Se recomienda una frecuencia de ejecución de un máximo de **5** segundos con el fin de poder hacer pruebas. Así mismo se sugiere la generación de un mensaje de log de tipo INFO que informe cada vez que se ejecute el proceso de mantenimiento de la base de datos realizando la tarea de mantenimiento de forma efectiva. Cuando no haya registros que actualizar no se mostrará ningún mensaje de log.

Aspecto Generales

Seguridad

Deberán tenerse en cuenta los siguientes aspectos de seguridad:

- Emplear la técnica de autentificación/autorización más adecuada a este contexto.
- Se recomienda el uso de un URL inicial diferente para cada rol.
- Es obligatorio utilizar la librería de logging para registrar el funcionamiento interno de la aplicación.

Arquitectura

La aplicación deberá estar diseñada siguiente un patrón arquitectónico de N-Capas hibridado con el patrón MVC en la línea de lo visto en las sesión de la laboratorio sobre Diseño Arquitectónico.

Capa de Presentación

Deberán emplearse los siguientes recursos JSF como mínimo:

- Managed Beans: Emplear face-config.xml.
- Reglas de navegación. Emplear face-config.xml
- Internacionalización (i18n) en dos idiomas de todos los textos posibles.
- Uso del método de validación adecuado en formularios.
- Empleo coherente de plantillas en orden a reducir la duplicidad de código JSF.
- Uso de etiquetas Primefaces siempre que sea posible.

Capa de Negocio

En la capa de negocio deberá seguirse un diseño basado en interfaces de servicios empleando los patrones Facade y Factory.

Persistencia

Es obligatorio el empleo del SGBD Hypersonic y por lo tanto la creación y rellenado de la base de datos de nuestra aplicación a través del archivo LocalDB.script de la carpeta data del proyecto.

Sí es obligatorio que la aplicación se entregue con algún mecanismo de inicialización (un script SQL) con cierta cantidad de datos de partida disponibles para el usuario final:

- Varios usuarios registrados. Al menos 3 usuarios:
  - Todos los usuarios tendrán como "login" usuario1, usuario2, usuario3, etc.
  - Se hará coincidir la contraseña con el "login" en todos los usuarios.
- Cada usuario-i será promotor de al menos 10 viajes (se pueden generar automáticamente con datos ficticios), con plazas entre 1 y 4.

Se da la posibilidad de emplear JPA para la implementación de esta capa.

En caso de no emplear JPA será obligatorio el empleo del patrón DAO/DTO para el diseño del acceso de la base de datos así como el uso de consultas SQL empleando jdbc. Así mismo, deberá emplearse un patrón Factory para desacoplar el patrón DAO de su forma de obtención. Se deberán definir como transaccionales aquellos conjuntos de operaciones SQL que así lo requieran en orden a mantener la consistencia en la base de datos (**control de concurrencia optimista = uso de commit/rollback**).

Entrega

A cada equipo de trabajo se asignará un número n en el campus virtual. Según ese numero se deberá crear los proyectos eclipse de la aplicación web primefaces y las pruebas unitarias con los nombres **sdi2-n** y **sdi2-test-n** respectivamente (ambos en minúsculas). Se deberá subir a la tarea correspondiente un archivo ZIP (usando el formato ZIP) con el nombre sdi2-n.zip (en minúsculas) y que deberá contener en su raíz:

- Un documento PDF (con el mismo nombre que el proyecto sdi2-n.pdf) que contenga:
  - El mapa de navegación de la aplicación, al estilo de los vistos en clase.
  - Una descripción textual de cada una de las acciones asociadas a transiciones en el mapa de pantallas.
  - Una descripción clara y detallada de las partes opcionales implementadas.
  - Un catálogo de las pruebas unitarias realizadas y descripción sencilla de cada una
  - Cualquier otra información necesaria para una descripción razonablemente detallada de lo entregado y su correcto despliegue y ejecución.
- El proyecto Primefaces eclipse exportado en formato carpeta (no comprimido) con el nombre .**sdi2-n** (File/export/General/File System).
- El proyecto Java eclipse exportado en formato carpeta (no comprimido) con el nombre **.\sdi2-test-n** (File/export/General/File System).
- El archivo war correspondiente a tu proyecto primefaces exportado desde eclipse con el nombre sdi2-n.war (File/export/WAR File).
- En resumen el zip deberá contener en su raiz:

- sdi2-n.pdf
- sdi2-n
- sdi2-test-n
- sdi2-n.war

- Por otro lado se indica que el archivo sdi2-n.war entregado será el elemento de corrección tanto para las pruebas automatizadas como la posible prueba manual y **se probará de la siguiente forma**:
  - Se tomará el archivo war entregado y se depositará en la carpeta S:\wildflyext\standalone\deployments
  - No se admite la inclusión de ningún otro elemento a copiar en la carpeta anterior.
  - Caso de NO poder realizar la prueba funcional la práctica quedará automáticamente invalidada, por lo tanto se recomienda realizar la prueba anterior sobre el archivo war que se vaya entregar en su última versión.

Fecha máxima de entrega

La semana del 4 al 9 de Abril a las 20:00, coincidiendo con el día **ANTERIOR** en que se imparte el grupo de laboratorio al que pertenece el alumno. Por ejemplo, el grupo del Lunes 4 de Abril deberá entregar el 3 Abril a las 20.00 y así sucesivamente.

Ampliaciones

Se proponen como posibles ampliaciones a la especificación mínima anterior:

1. Filtrado1 Filtrado2 (hasta 0.5 puntos por el total de los dos filtrados).
2. Ordenación1 Ordenación2 (Hasta 0.5 puntos por el total de las dos ordenaciones).
3. Paginación1 Paginación2 (hasta 0.5 por el total de las dos paginaciones).
4. Validación avanzada usando custom validators (usuarios repetido, ya existe email, …) (hasta 0.5 puntos)
5. I18n. Generalización de la interfaz para n-idiomas. (hasta 1.0 puntos). Esta opción consiste en preparar el proyecto para que la inclusión de un nuevo idioma consista en colocar un archivo de propiedades con un nuevo idioma en la carpeta src y que al desplegar el proyecto de nuevo automáticamente se incluya el nuevo idioma. El texto
6. Uso de JPA en la capa de persistencia. (1.0 puntos)
7. Ajaxificación de aquellas pantallas/formularios donde sea oportuno (hasta 0.5 puntos).
8. Autocompletado de los campos del formulario de Alta de viaje. (hasta 0.5 puntos)
9. Mantenimiento programado de la base de datos. (hasta 0.5 puntos)
10. Implementación de un pool de conexiones (hasta 0.5 puntos).

La obtención de los 6.0 puntos opcionales se corresponden con 10.0 puntos en la variable Parte Opcional de la fórmula Nota final.

Evaluación

La implementación de la parte obligatoria supondrá el 60% de la nota total. El 10% de la nota total se corresponde con la evaluación de la documentación entregada. El resto de la nota se obtendrá sumando las calificaciones obtenidas en las partes opcionales implementadas.

Nota final: 0.6*Parte Obligatoria + 0.5*Parte Opcional+0.1*Documentación.

La calificación máxima que se puede obtener será de 12 puntos.

Se penalizará:

- La escasa claridad del código.
- Que el código Java NO se ajuste a las *Java Code Conventions*.
- Que la compilación del código genere "*warnings*".
- Que se presenten problemas durante el despliegue.