INFORME SDI2-13

INTEGRANTES DEL GRUPO

- Sergio Moradas Quintana - UO240772 – 9816446T
- Carlos Villa Blanco - UO236851– 71741051A

DESCRIPCIÓN DE LOS BEANS

BeanTrip

Este bean sirve para aceptar y eliminar las solicitudes de un viaje. También, se utiliza para obtener dichas solicitudes de la base de datos.

BeanTrips

Este bean se utiliza para almacenar los viajes existentes y los viajes en los que tiene algún tipo de implicación el usuario logueado. Además, se utiliza para filtrar los viajes que mostrar en los distintos listados, así como para apuntarnos a dichos viajes.

BeanRegisterTrip

En este bean se almacenar todos los datos necesarios para registrar un viaje mediante el formulario de registro de viajes. Además, su principal función es registrar dicho viaje en la base de datos.

BeanModifyTrip

Este bean obtiene los datos del viaje que se quiere modificar y posteriormente almacenar los datos modificados para actualizar el viaje en la base de datos.

BeanCancelarTrip

Este bean almacena aquellos viajes que el promotor ha seleccionado para cancelar. Por ello, su principal función es cambiar el estado de los viajes a cancelado.

BeanLogin

La principal función de este bean es almacenar el la sesión el usuario que está logueado, después de autenticar a dicho usuario comprobando la contraseña y usuario.

BeanUser

Bean utilizado para almacenar los datos del usuario para a continuación utilizarlo el BeanRegistro para almacenar el usuario en la BBDD.

BeanRegistro

Este Bean es el encargado de registrar el usuario en la aplicación. Si las validaciones se realizan correctamente, este Bean encripta la contraseña del usuario y después almacenar todos sus datos en la BBDD.

DESCRIPCIÓN DE LAS TRANSICIONES DE PANTALLAS

La primera página que se mostrará al usuario será la de login (login.xhtml). En ella, en caso de que el usuario ya esté registrado, éste podrá iniciar sesión introduciendo su login y contraseña o acceder a otras páginas a través del menú de navegación.

A través del menú de navegación, se podrá acceder a una página para observar la lista de viajes existentes en estos momentos (listaViajesPublicos.xhtml), podrá acceder al formulario de registro de nuevos usuarios (registrarse.xhtml) y, por último, podrá cambiar el idioma de la aplicación seleccionando elidioma que desee.

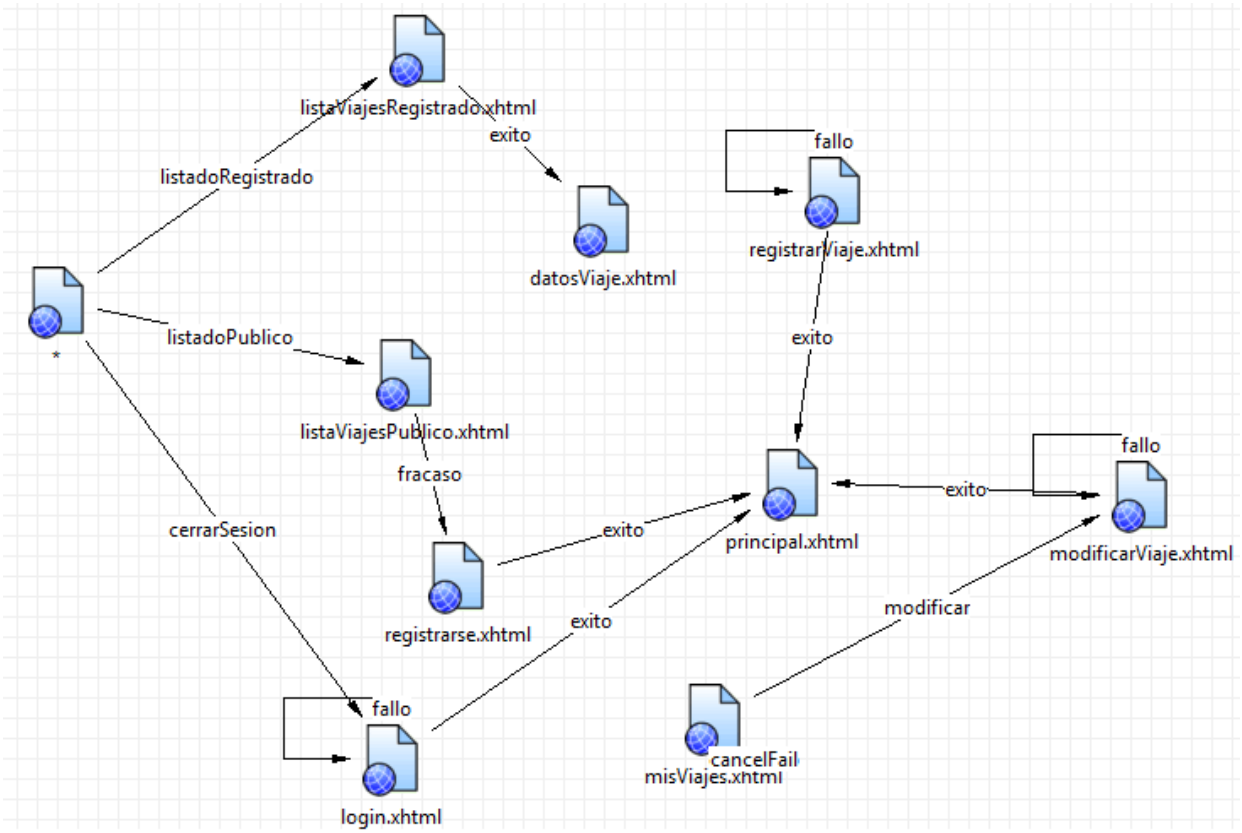
Una vez logueado se le mostrará al usuario la página principal (principal.xhtml). Desde ella, podrá visitar distintas páginas a través del menú de navegación. En primer lugar, podrá visualizar todos los viajes a los que se puede apuntar (listadoViajesRegistrado.xhtml).

En segundo lugar, podrá visualizar todos los viajes en los que tiene algún tipo de implicación (misViajes.xhtml), cancelar los viajes creados por él, aceptar solicitudes a sus viajes (solicitudesViaje.xhtml) o modificar dichos viajes (modificarViaje.xhtml).

Como tercera opción tenemos la de registrar viaje, donde se mostrará un formulario que hay que rellenar para registrar correctamente un viaje (registrarViaje.xhtml).

Por último, habrá un opción para cerrar la sesión y para cambiar el idioma de la aplicación.

A continuación se muestra la navegación entre páginas. Hay algunas transiciones que no se muestran debido a que el paso de una página a otra es constante y si hace falta la ejecución de un método de un Bean se utiliza un actionListener. Ese es el caso de algunas transiciones realizadas desde la barra de navegación:



DESCRIPCIÓN DE LAS PARTES OPCIONALES

1. *Ordenación1 Ordenación2*

```
<p:dataTable var="viaje" value="#{trips.viajes}" border="1"
    sortMode="multiple">

    <p:column headerText="#{msgs.ciudadOrigen}"
        sortBy="#{viaje.departure.city}">
```

Esta ampliación consistió en añadir a la dataTable la opción sortMode y para cada uno de las columnas que se quieran ordenar la opción sortBy con el parámetro de ordenación.

Para realizar la ordenación combinada de campos tenemos que seleccionar los campos por los que deseemos ordenar a la vez que pulsamos la tecla ‘ctrl’.

2. *Ajaxificación de aquellas pantallas/formularios donde sea oportuno*

La ajaxificación se utiliza en casi todos los formularios, es decir, en el formulario de registro, en el formulario de login y el formulario de registrar y modificar viaje.

```
<h:panelGrid columns="2" style="margin-left:60%" cellpadding="5">
    <h:outputText value="#{msgs.trip_form_preCharge}" />
    <p:selectBooleanCheckbox id="checkDatos">
        <p:ajax update="@form"  listener="#{registerTrip.datosPorDefecto}" />
    </p:selectBooleanCheckbox>
</h:panelGrid>
```

3. *Validación avanzada usando custom validators*

Se ha creado un validador propio para validar las contraseñas introducidas al registrarse como usuario y ver que ambas coinciden.

```
@FacesValidator(value = "passwordValidator")
public class PasswordValidator implements Validator {

    @Override
    public void validate(FacesContext fc, UIComponent component, Object value)
            throws ValidatorException {
        String attribute = (String) component.getAttributes().get("password");
        if (!value.equals(attribute)) {
            FacesMessage message = new FacesMessage();

            FacesContext facesContext = FacesContext.getCurrentInstance();
              ResourceBundle bundle =
               facesContext.getApplication().getResourceBundle(facesContext, "msgs");


            message.setSummary(bundle.getString("passNoCoincide"));
            message.setSeverity(FacesMessage.SEVERITY_ERROR);
            throw new ValidatorException(message);
        }
    }
}
```

4. *Mantenimiento programado de la base de datos.*

La realización de esta ampliación se basa en la creación de una nueva clase dentro del paquete com.sdi.utilidades.Timers. Dicha clase contiene un método estático que contiene un método que realizará dicho mantenimiento cada 5000 ms. Este método comprobará que los las viajes y los asientos de los viajes están en el estado correcto.

5. *Filtrado 1*

Se da la posibilidad de filtrar los viajes por ciudad de salida y ciudad de destino del listado de viajes público y del listado para usuario registrado.

6. *Paginación 2*

El listado de viajes 'Mis viajes' se muestra por páginas. El usuario puede seleccionar el número de viajes a mostrar por página utilizando un 'Select'.

Adicionalmente hemos realizado el siguiente punto:

7. *Encriptación de contraseñas*

Se almacenan las contraseñas de los usuarios utilizando el algoritmo de encriptación MD5.

DESCRIPCIÓN DE LAS PRUEBAS REALIZADAS

1. [RegVal] Registro de Usuario con datos válidos.

Primero se accede al formulario de registro. Luego, se rellenan los campos con datos válidos y se comprueba que el registro ha sido satisfactorio.

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

Igual que el anterior pero en este caso introducimos datos inválidos. En nuestro caso introducimos contraseñas diferentes para comprobar que nos muestra dicho error.

3. [IdVal] Identificación de Usuario registrado con datos válidos.

En este test únicamente iniciamos sesión con un usuario existente en la base de datos e introduciendo correctamente la contraseña.

4. [IdInval] Identificación de usuario registrado con datos inválidos.

Es el caso contrario al test anterior, pues en este caso intentamos con un usuario existente pero introduciendo una contraseña errónea.

5. [AccInval] Intento de acceso con URL desde un usuario no público (no identificado). Intento de acceso a vistas de acceso privado.

En este test intentamos acceder directamente a la página principal y a la página de misViajes sin logearnos. Todo esto tiene como objetivo comprobar que no se puede acceder sin estar autentificado.

6. [RegViajeVal] Registro de un viaje nuevo con datos válidos.

Este test consiste en loguearnos correctamente con un usuario, acceder la página de registrar viaje y registrar correctamente un nuevo viaje.

7. [RegViajeInVal] Registro de un viaje nuevo con datos inválidos.

Exactamente igual que el test anterior pero en este caso introducimos datos inválidos para comprobar que nos muestran los errores correctamente. En nuestro caso, introducimos datos incorrectos en el campo el precio del viaje y del número máximo de plazas.

8. [EditViajeVal] Edición de viaje existente con datos válidos.

En este test, después de loguearnos correctamente, accedemos a la página de misViajes y seleccionamos uno que queremos modificar. Una vez que estamos en la pantalla correspondiente al formulario de modificar viaje, introducimos un cambio y comprobamos que dicha operación se ha realizado con éxito.

9. [EditViajeInVal] Edición de viaje existente con datos inválidos.

Igual que el test anterior pero con la diferencia de que se modifica un dato con un valor inválido para comprobar que nos muestra correctamente dicho error.

10. [CancelViajeVal] Cancelación de un viaje existente por un promotor.

En este test, tras loguearnos y acceder a la página de mis viajes, se selecciona un viaje que deseamos cancelar y pulsamos el botón para que se produzca dicha cancelación. Comprobamos que todos los pasos se han realizado con éxito.

11. [CancelMulViajeVal] Cancelación de múltiples viajes existentes por un promotor.

Igual que el anterior pero en vez de seleccionar un solo viaje seleccionamos varios.

12. [Ins1ViajeAceptVal] Inscribir en un viaje un solo usuario y ser admitido por el promotor.

Primero borramos todos los viajes de la base de datos para evitar cualquier interferencia de estos con el correcto funcionamiento del test. A continuación, creamos un nuevo viaje con el usuario1 al cual el usuario2 envía una solicitud de plaza. Finalmente el usuario1 admite al usuario2 y volvemos a iniciar sesión con este último usuario para comprobar que realmente fue admitido.

13. [Ins2ViajeAceptVal] Inscribir en un viaje dos usuarios y ser admitidos los dos por el promotor.

En este test se llevaron unos pasos similares a los del test anterior. Pero en este se mandan dos solicitudes a un viaje por dos usuarios diferentes y el usuario promotor del viaje los admite. Finalmente se accede con los dos usuarios admitidos y se comprueba en la lista 'Mis Viajes' que realmente fueron admitidos.

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.

En primer lugar, nos logueamos y creamos un viaje con dos plazas disponibles. Posteriormente, iniciamos sesión con otros dos usuarios distintos y solicitamos plaza en dicho viaje. Ahora, iniciamos sesión con usuario promotor del viaje y confirmamos dichas plazas a los dos solicitantes anteriores. Para finalizar, iniciamos sesión con otro usuario distinto y nos intentamos apuntar pero vemos que no puede pues no quedan plazas.

15. [CancelNoPromotorVal] Un usuario no promotor Cancela plaza.

Realizamos los mismos pasos que en el anterior, es decir, registramos un viaje con un usuario y nos apuntamos a él con otro usuario. Sin embargo, en este test, posteriormente a que el promotor haya aceptado la plaza del usuario, vamos a iniciar sesión con el usuario y acceder la pantalla de mis Viajes para cancelar la participación en dicho viaje.

16. [Rech1ViajeVal] Inscribir en un viaje un usuario que será admitido y después rechazarlo por el promotor.

En este caso iniciamos sesión con un usuario y creamos un viaje. Posteriormente, iniciamos sesión con otro usuario y nos apuntamos a ese viaje. Por último, volvemos a iniciar sesión con el promotor del viaje y aceptamos la solicitud del usuario y, seguidamente, la cancelamos.

17. [i18N1] Cambio del idioma por defecto a un segundo idioma. (Probar algunas vistas)

Se comprueba que el idioma cambia de español a inglés y se revisa que algunos mensajes se muestran en este último idioma en las pantallas de login, registro y listado de viajes.

18. [i18N2] Cambio del idioma por defecto a un segundo idioma y vuelta al idioma por defecto. (Probar algunas vistas)

Se comprueba en las páginas de login y registro que el cambio de idioma a ingles y después de vuelta a español funciona perfectamente

19. [OpFiltrado] Prueba para el filtrado opcional.

Insertamos en la base de datos 5 viajes donde uno de los cuales tiene como ciudad de inicio Oviedo. Filtramos por ciudad de inicio introduciendo la cadena 'Ovi' y se comprueba que solo haya un viaje.

20. [OpOrden] Prueba para la ordenación opcional.

Se inicia sesión con un usuario y se registran cinco viajes. Luego, se accede a la página misViajes y se ordenan por ciudad de salida. Una vez hecho esto, comprobamos que están ordenados correctamente.

21. [OpPag] Prueba para la paginación opcional.

Se realiza un filtrado por ciudad de salida en la lista de viajes pública. En ella filtramos por la cadena 'Ovi' y comprobamos que solo se obtiene un viaje con salida desde Oviedo.

22. [OpMante] Prueba del mantenimiento programado opcional

Insertamos con un script un viaje cuya fecha de cierre ya pasó junto a una petición del usuario2. La fecha de salida y llegada se establecen en una fecha muy lejana para comprobar que el estado del viaje pasa a cerrado. A continuación iniciamos sesión con el usuario2 y comprobamos en el listado 'Mis Viajes' que aparece un viaje con estado cerrado y cuya relación con usuario2 es 'Sin plaza'.

DESCRIPCIÓN PARA CORRECTO DESPLIEGUE Y EJECUCIÓN

Para una correcta ejecución de la aplicación primero se deben ejecutar los tests, ya que cuando estos terminen dejan las base de datos con las información pedida (3 usuarios cuyo login coincide con la contraseña y 10 viajes cada uno).