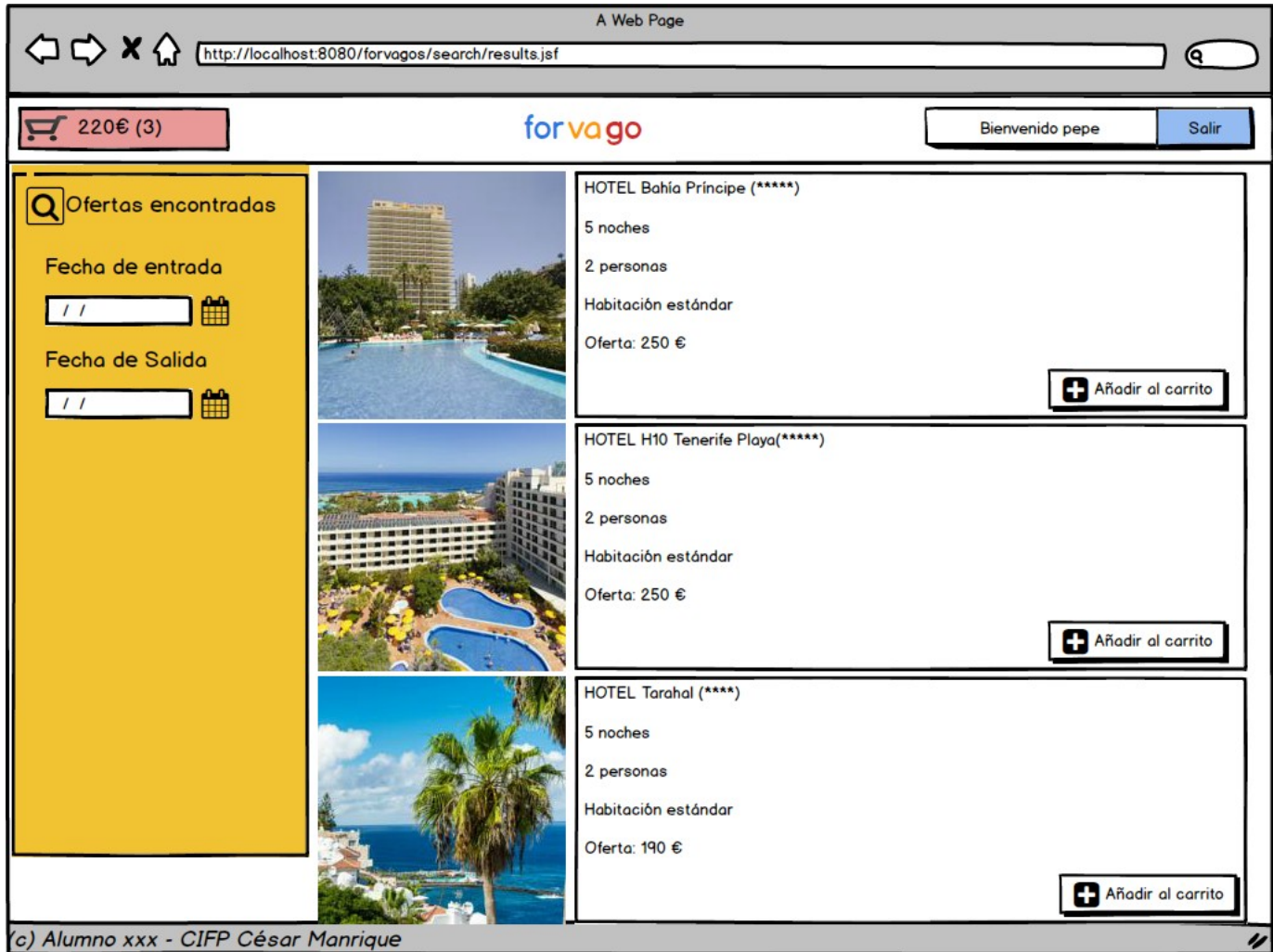




AUT06_03 Proyecto ForVago-Fase III

1. Creación de la lista de resultados de búsqueda y carrito de compra

Tendremos una clase que genere la lista de ofertas, vamos a mostrarla y permitir añadir ofertas al carrito.





2. Clases Java

Vamos a crear una serie de clases (luego ya la reemplazaréis con las vuestras, ya correctas). Todo el código es orientativo y no necesariamente funciona bajo cualquier circunstancia.

2.1 HotelOffer Clase que almacena ofertas de hoteles

Esta clase generará una lista de ofertas. El código sería:

```
public class HotelOffer implements Serializable {

    private String hotel;
    private BigDecimal price;
    public String hotelPicture;

    public HotelOffer() {
    }

    public HotelOffer(String hotel, BigDecimal price, String hotelPicture) {
        this.hotel = hotel;
        this.price = price;
        this.hotelPicture = hotelPicture;
    }

    public String getHotel() {
        return hotel;
    }

    public void setHotel(String hotel) {
        this.hotel = hotel;
    }

    public BigDecimal getPrice() {
        return price;
    }

    public void setPrice(BigDecimal price) {
        this.price = price;
    }

    public String getHotelPicture() {
        return hotelPicture;
    }

    public void setHotelPicture(String hotelPicture) {
        this.hotelPicture = hotelPicture;
    }

}
```



2.2 Interfaz HotelOfferDao

```
package es.cifpcm.inma_forvagos12.data;

import es.cifpcm.inma_forvagos12.model.hotel.HotelOffer;
import java.util.List;

/**
 *
 * @author inmav
 */
public interface HotelOfferDao {
    public List<HotelOffer> selectAll();
    public List<HotelOffer> selectByCriteria(Integer entero);
    public HotelOffer insert();
    public HotelOffer update();
    public HotelOffer delete();
}
```

2.3 ManagedBean HotelResultsBean

De ámbito de sesión (no es el más adecuado, pero vale para el ejemplo), es el bean que almacenaría las ofertas, cuando se ejecutara desde el formulario de búsqueda.

A este bean accederemos desde la página **searchResults.xhtml**.

2.4 Clase ShoppingCart

Representa un carrito de compra básico, con un método para obtener el total.

```
public class ShoppingCart implements Serializable {

    private final List<HotelOffer> offers = new ArrayList<>();

    public ShoppingCart() {
    }

    public void addOffer(HotelOffer offer) {
        this.offers.add(offer);
    }

    public void removeOffer(HotelOffer offer) {
        this.offers.remove(offer);
    }

    public BigDecimal getTotal() {

        return this.offers.stream().map(o -> o.getPrice()).reduce(BigDecimal.ZERO,
        BigDecimal::add);
    }
}
```



2.5 ManagedBean ShoppingCartBean

Extiende a la clase anterior, para facilitar la comunicación con JSF.

```
@Named(value = "shoppingCartBean")
@SessionScoped
public class ShoppingCartBean extends ShoppingCart implements Serializable {

    /**
     * Creates a new instance of ShoppingCartBean
     */
    public ShoppingCartBean() {
    }


    public String getTotalAsString() {
        return NumberFormat.getCurrencyInstance().format(getTotal());
    }
}
```

3. Páginas JSF

Vamos a ver, fundamentalmente, la página que muestra las ofertas encontradas.

3.1 Página searchResults.xhtml (facelets template client)

Esta página hace uso del shoppingCartBean para mostrar el total reservado y del hotelResultsBean para mostrar la lista de ofertas.

El  0,00 €

forvago

#{shoppingCartBean.totalAsString}

**<p:commandButton value="Añadir">
<p:ajax listener="#{shoppingCartBean.addOffer(offer)}" update="|Total"/>
</p:commandButton>**

Hotel: Club Tarahal
Precio: 70
Añadir

Hotel: Hotel Botanico y Oriental Spa Garden
Precio: 70
Añadir

Hotel: Bahía Príncipe San Felipe
Precio: 70
Añadir

<p:repeat value="#{hotelResultsBean.offers}" var="offer" offset="0" size="5" step="1" varStatus="status">

código está basado en el ejemplo de PrimeFaces repeat:

<http://www.primefaces.org/showcase/ui/data/repeat.xhtml>



3.1.1 Código de la página

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE composition PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<ui:composition xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
    template="../../WEB-INF/templates/publicTemplate.xhtml"
    xmlns:h="http://xmlns.jcp.org/jsf/html"
    xmlns:p="http://primefaces.org/ui"
    xmlns:f="http://xmlns.jcp.org/jsf/core">

    <ui:define name="content">
        <h:form id="form">
            <p:repeat value="#{hotelResultsBean.offers}" var="offer" offset="0" size="5" step="1"
varStatus="status">

                <h:panelGrid columns="2" style="width:100%" columnClasses="logo,detail">
                    <p:commandLink update=":form:offerDetail" oncomplete="PF('offerDialog').show()">
                        <f:setPropertyActionListener value="#{offer}"
target="#{hotelResultsBean.selectedOffer}" />
                        <p:graphicImage library="images" name="hotels/#{offer.hotelPicture}" />
                    </p:commandLink>

                    <p:outputPanel>
                        <h:panelGrid columns="2" cellpadding="5">
                            <h:outputText value="Hotel:" />
                            <h:outputText value="#{offer.hotel}" style="font-weight: bold"/>

                            <h:outputText value="Precio:" />
                            <h:outputText value="#{offer.price}" style="font-weight: bold"/>
                            <p:commandButton value="Añadir">
                                <p:ajax listener="#{shoppingCartBean.addOffer(offer)}"
update="lblTotal"/>
                            </p:commandButton>
                        </h:panelGrid>
                    </p:outputPanel>
                </h:panelGrid>
            </p:repeat>

            <p:dialog header="Información de la oferta" widgetVar="offerDialog" modal="true" showEffect="blind"
hideEffect="explode" resizable="false">
                <p:outputPanel id="offerDetail" style="text-align:center;">
                    <p:panelGrid columns="2" rendered="#{not empty hotelResultsBean.selectedOffer}"
columnClasses="label,value">
                        <f:facet name="header">
                            <p:graphicImage library="images" value="hotels/#{offer.hotelPicture}"
/>
                        </f:facet>

                        <h:outputText value="Hotel:" />
                        <h:outputText value="#{offer.hotel}" style="font-weight: bold"/>
                    </p:panelGrid>
                </p:outputPanel>
            </p:dialog>
        </h:form>
    </ui:define>
</ui:composition>
```



```
        <h:outputText value="Precio:" />
        <h:outputText value="#{offer.price}" style="font-weight: bold"/>

        </p:panelGrid>
    </p:outputPanel>
</p:dialog>
</h:form>
</ui:define>
</ui:composition>
```

3.1.2 Puntos clave

Hemos añadido un `commandButton` con un listener Ajax. Cuando se pulsa el botón se invoca vía Ajax el método `shoppingCartBean.addOffer(offer)`

Al retorno se actualiza el elemento con id **lblTotal**, que es un `outputText` situado en la plantilla, en la cabecera, y muestra la información del `shoppingCartBean`

```
<p:commandButton value="Añadir">
    <p:ajax listener="#{shoppingCartBean.addOffer(offer)}"
update="lblTotal"/>
</p:commandButton>
```

En la cabecera (en el template) hemos añadido:

```
<div style="float: left; width: 20%;">
    <h:graphicImage library="images" name="empty-cart-dark_small.png"/>
    <h:outputText id="lblTotal" value="#{shoppingCartBean.totalAsString}"/>
</div>
```