

كلية الهندسة المعلوماتية

جامعة البعث

قسم هندسة البرمجيات

مشروع مادة التجارة الكترونية

DaKakeen

إشراف المهندس:

عبدو دربولي

تقديم:

إيفان الدياب

ورد عطاالله

٢٠١٥/١/٥

الهدف من المشروع:

يتناول المشروع موقع تسويق الكتروني اسمه dakakeen خاص ببيع الإلكترونيات و جميع ملحقاتها من اكسسوارات و إضافات باستخدام تقنية JSF

بعض الصفحات الموجودة بالموقع:

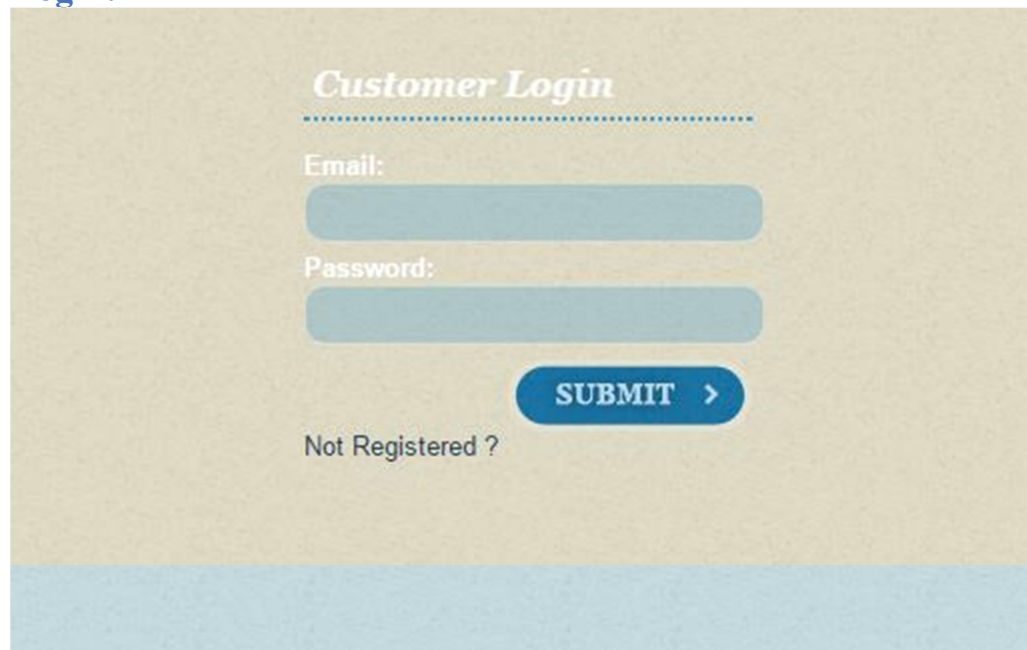
في البداية يقوم الزائر بالدخول للموقع فيعرض أمامه ال categories التي سنقوم بتسويقها و في أعلى الصفحة تبويبات هي : login—checkout—view cart

Bar:



حيث أن تبويبة login تأخذنا لصفحة تسجيل دخول للزبون..حيث عليه إدخال اسم مستخدم و كلمة مرور...

Login:



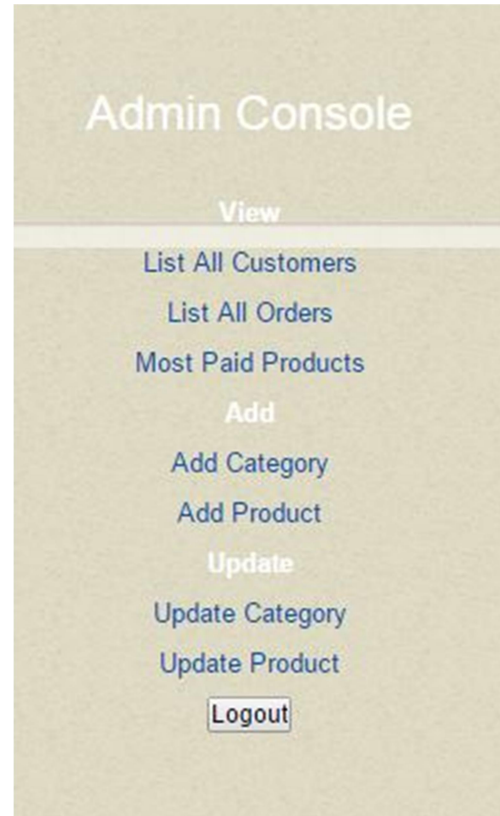
تم تحقيق تسجيل دخول مستخدم admin عبر رابط خاص هو:

<http://localhost:8080/Dakakeen/faces/admin/index.xhtml>

حيث يتم إدخال اسم مستخدم و كلمة مرور أيضا وتصبح تبويبة login هي panel نشرح عنها لاحقا"...

ويتم الدخول إلى صفحة admin :

Admin console:



والروابط الموضحة في هذه الصفحة تتيح للمدير معرفة كل عمليات الشراء التي يقوم بها المستخدمين إضافة إلى إضافة المنتجات و الأصناف.



وتحديث المنتجات و الأصناف.

وكل صفحة هي عبارة عن فورم إدخال.

عند اختيار احد الاصناف يتم تحديث الصفحة و تظهر فيها المنتجات التابعة للصنف المختار.

مثلا" قمنا بشراء منتج ننتقل لصفحة : view cart

Cart:

	satalite	3.49	1	<input type="button" value="Update"/>
	LG	2.29	1	<input type="button" value="Update"/>

يمكن في هذه الصفحة تعديل العدد المطلوب شراؤه من المنتج إضافة لإمكانية مسح جميع المشتريات الخاصة بهذا الزبون من خلال رابط clear cart ويعرض السعر الاجمالي لجميع المشتريات

checkout page:

Customer Name	evan
Email	evan.diab@yahoo.com
Phone	0932418562
Address	Homs
Region	SY
Credit Card Number	1324
<input type="button" value="submit purchase"/>	

يتم فيها ادخال بيانات المستخدم و طبقنا عليها ال validation

وقد قمنا بتحقيق نوعي التنقل الضمني والصريح من خلال إضافة قواعد التنقل الصريح الى الملف
faces configure

Navigation:

```
<navigation-rule>
  <from-view-id>*/</from-view-id>
  <navigation-case>
    <from-outcome>login</from-outcome>
    <to-view-id>/login.xhtml</to-view-id>
    <redirect/>
  </navigation-case>
  <navigation-case>
    <from-outcome>admin</from-outcome>
    <to-view-id>/admin/admin.xhtml</to-view-id>
    <redirect/>
  </navigation-case>
  <navigation-case>
    <from-outcome>cart</from-outcome>
    <to-view-id>/cart.xhtml</to-view-id>
    <redirect/>
  </navigation-case>
  <navigation-case>
    <from-outcome>checkout</from-outcome>
    <to-view-id>/checkout.xhtml</to-view-id>
    <redirect/>
  </navigation-case>
  <navigation-case>
    <from-outcome>index</from-outcome>
    <to-view-id>/index.xhtml</to-view-id>
    <redirect/>
  </navigation-case>
</navigation-rule>
```

basic code snapshots explaining the main functionality including shopping card and admin console:

category controller:

```
public Category getSelectedCategory() {
    FacesContext fc = FacesContext.getCurrentInstance();
    Map<String, String> params
        = fc.getExternalContext().getRequestParameterMap();
    String gID = params.get("category");
    selectedCategory = categoryFacade.find(Short.parseShort(gID));
    this.selectedCategoryName = selectedCategory.getName();
    return (selectedCategory);
}
```

Check out page:

```
<h:form>
  <table id="checkoutTable">
    <tr><td><label for="name">Customer Name </label></td>
      <td class="inputField"><h:outputText id="name" value="#{loginController.customer.name}" /></td>
    </tr>
    <tr><td><label for="email">Email</label></td><td class="inputField"><h:outputText id="email"
      value="#{loginController.customer.email}" /></td>
    </tr>
    <tr><td><label for="phone">Phone:</label></td> <td class="inputField">
      <h:outputText
        id="phone" value="#{loginController.customer.phone}" /></td>
    </tr>
    <tr><td><label for="address">Address</label></td> <td class="inputField">
      <h:outputText
        id="address" value="#{loginController.customer.address}" /></td>
    </tr>
    <tr><td><label for="creditcard">Region</label></td><td class="inputField">
      <h:outputText
        id="cityRegion" value="#{loginController.customer.cityRegion}" /></td>
    </tr>
    <tr><td><label for="creditcard">Credit Card Number</label></td><td class="inputField">
      <h:outputText
        id="ccNumber"
        value="#{loginController.customer.ccNumber}" /></td>
    </tr>
    <tr><td colspan="2">
      <h:commandButton value="submit purchase"
        action="#{customerOrderController.placeOrder(loginController.customer, cartController.cart)}" /></td>
    </tr></table>
</h:form>
```

update category:

```
public synchronized String updateFile(Category c){

    if(file.getSize()!=0 ){
        if (file.getSize() >1024*200){
            System.out.println("Size is more than 200 KB");
        }
        else {
            try {
                upload(c.getName());
                wait(3500);
                return ("updCat");
            } catch (InterruptedException ex) {
                Logger.getLogger(CategoryController.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
    }
    file=null;

    return null;
}
```



```

public synchronized String updateName(Category c){

    if (!newCat.equals("")){
        if(!renameFile(c, newCat))
            return null;
        try{
            c.setName(newCat);
            categoryFacade.edit(c);
            wait(1000);
            newCat="";
            return null;
        } catch (InterruptedException ex) {
            Logger.getLogger(CategoryController.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    newCat="";

    return null;
}

```

Update product:

```

public synchronized String updateFile(Product pr){

    if(file.getSize()!=0){

        try {
            upload(pr.getName());
            wait(5000);
            return ("updPro");
        } catch (InterruptedException ex) {
            Logger.getLogger(CategoryController.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    file=null;

    return null;
}

public synchronized String updateName(Product pr){

    if (!name.equals("")){
        if(!renameFile(pr, name))
            return null;
        try{
            pr.setName(name);
            productFacade.edit(pr);
            wait(1000);
            name="";
            return ("updPro");
        } catch (InterruptedException ex) {
            Logger.getLogger(CategoryController.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    name="";

    return null;
}

```

```

public synchronized String updatePrice(Product pr){

    if (price!=null){

        try{
            pr.setPrice(price);
            productFacade.edit(pr);
            wait(1000);
            price=null;
            return ("updPro");
        } catch (InterruptedException ex) {
            Logger.getLogger(CategoryController.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}

```

Using ajax and validation:

تم استخدام تقنية ajax في كل من ال update product,category و ال validation

Customer Name	<input type="text" value="user"/>	You must enter a name between 5 and 15 chars
Email	<input type="text"/>	You must enter your Email
Phone:	<input type="text" value="1234652"/>	
Address	<input type="text" value="place"/>	
Region	<input type="text" value="NYE"/>	You must enter a value less or equals 2 chars
Credit Card Number	<input type="text" value="1325"/>	
Password	<input type="password"/>	
Confirm Password	<input type="password"/>	
<input type="button" value="Submit Register"/>		

References:

****The Java EE 5 Tutorial.**

<http://docs.oracle.com/javaee/5/tutorial/doc/jvaeetutorial5.pdf>

[1]. docs.oracle.com/javaee/5/tutorial/doc/

[2]. <http://netbeans.org/kb/docs/javaee/ecommerce>

[3]. Enterprise JavaBeans 3.1, sixth edition, by Andrew Lee Rubinger, Bill Burke, O'Reilly Media.

[4]. Distributed Systems Principles and Paradigms, by A. Tanenbaum and M. Van Steen.