Correction Bug QualDev_00_ASBANK2023

Contents

Correction Bug Non-Bloquant

Bug N1 FAVICON :

Le favicon du site ne s'affichait pas malgré la présence de la ligne suivante :

html

Copier le code

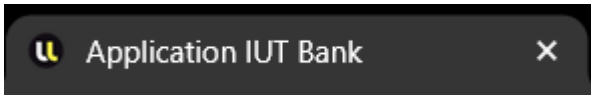
<link href="/_00_ASBank2023/style/favicon.ico" rel="icon" />

Cela était dû à l'absence du fichier favicon.ico dans le dossier : _00_ASBank2023/style/.

Pour résoudre ce problème, nous avons récupéré le logo de l'université au format PNG (240×240). Afin de l'adapter au format requis, nous avons utilisé le site convertio.co pour convertir le fichier .png en .ico.

Dans un premier temps, nous avions tenté d'utiliser le fichier .png comme favicon. Cependant, nous avons constaté que ce format est moins bien pris en charge par certains navigateurs. À l'inverse, le format .ico garantit un affichage optimal sur un plus grand nombre de navigateurs.

Ainsi, le favicon s'affiche désormais correctement :



Bug N2 Icon :

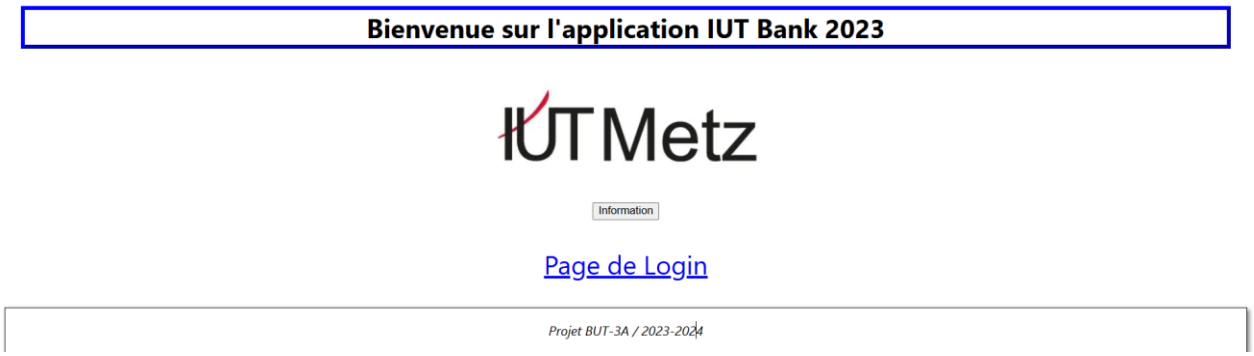
Afin d'afficher le logo du site, il suffisait de changer le lien :

En :

```
<img
    src="https://imgs.search.brave.com/P23fBts9D6Ds8ARzf6vORrTmA21cwWe_3LWyN2eqDu8/rs:fit:860:0:0:0/g:ce/aHR0cHM6Ly9mb3Jt/
    YXRpb25zLnVuaXYt/bG9ycmFpbmUuZnIv/aW1nL3N1LzQ1Lmpw/Zw" alt="logo" />
```

L'ancien lien pointait probablement sur une route protégée ou inexistante et donc une redirection sur la page d'accueil du site de l'université était réalisée.

Actuellement nous avons l'affichage :



Bug N2 Formulaire Création Utilisateur :

Le message d'erreur était le même qu'importe si le format du code utilisateur ou numéro du client soit dans le mauvais format.

Ainsi il a fallut modifier le type d'erreur « throw » par le setter de client pour le modele user, donc dans l'entité Client :

```java
/* Correction Formulaire Création Utilisateur Bug -- BUG_Non_bloquant_Nr_3 */
/**
 * {@inheritDoc}
 *
 * @throws IllegalArgumentException
 */
@Override
public void setUserId(String userId) throws IllegalArgumentException {
    if (!Client.checkFormatUserIdClient(userId)) {
        throw new IllegalArgumentException(s:"L'identifiant n'est pas au bon format.");
    }
    super.setUserId(userId);
}
/* Fin Correction Formulaire Création Utilisateur Bug -- BUG_Non_bloquant_Nr_3 */
```

Donc IllegalFormatException modifié en IllegalArgumentException mais également dans celui de utilisateur :

```java
/* Correction Formulaire Création Utilisateur Bug -- BUG_Non_bloquant_Nr_3 */
/**
 * @param userId
 *            : l'identifiant de l'utilisateur
 * @throws IllegalArgumentException
 */
public void setUserId(String userId) throws IllegalArgumentException {
    this.userId = userId;
}
/* Fin Correction Formulaire Création Utilisateur Bug -- BUG_Non_bloquant_Nr_3 */
```

Correction Bug Bloquant

Bug N1 Suppression Utilisateur Vide et Compte Vide :

Le bug empêche la suppression d'un utilisateur ne possédant pas de compte bancaire, mais également la suppression de compte bancaire vide et sans déficit.

Ainsi ce bug est un bug bloquant. Pour le corriger il a fallut créer une fonction afin d'obtenir le client avec son id :

_00_ASBank2023\src\main\java\com\iut\banque\facade\BanqueFacade.java

```
@@ -99,6 +99,22 @@ public class BanqueFacade {
                return banqueManager.getAllClients();
        }

+       /* Correction de BUG Bloquant Edit et Delete de compte par gestionnaire impossible
+       sur compte vide et user sans compte -- BUG_Bloquant_Nr_1 */
+       /**
+        * Méthode pour récupérer un client à l'aide de son Id
+        *
+        * @return Client
+        */
+       public Client getClientById(String idClient) {
+               Utilisateur user = banqueManager.getUserById(idClient);
+               if( user instanceof Client) {
+                       return (Client)user;
+               }
+               return null;
+       }
+       /* Fin Correction BUG Bloquant Nr_1 */
+
```

Puis utiliser cette fonction dans le deleteUser :

```
         */
        public String deleteUser() {
                try {
+                       client = banque.getClientById(idClient);
+                       if (client == null){
+                               throw new TechnicalException("Client introuvable ou gestionnaire");
+                       }
                        setUserInfo(client.getIdentity());
                        banque.deleteUser(client);
                        return "SUCCESS";
@@ -155,7 +205,11 @@ public class ListeCompteManager extends ActionSupport {
         */
        public String deleteAccount() {
                try {
-                       setCompteInfo(compte.getNumeroCompte());
+                       compte = banque.getCompte(idCompte);
+                       if(compte == null) {
+                               throw new TechnicalException("Compte introuvable");
+                       }
+                       setCompteInfo(idCompte);
                        banque.deleteAccount(compte);
                        return "SUCCESS";
                } catch (IllegalOperationException e) {
@@ -166,4 +220,5 @@ public class ListeCompteManager extends ActionSupport {
                        return "ERROR";
                }
        }
```

Tout en injectant le idClient dans le fichier JSP :

```
        <!-- Correction Bug Bloquant N1 -->
<td><s:url action="deleteAccount" var="deleteAccount">
                <s:param name="compte">
                <s:param name="idCompte">
                        <s:property value="value.numeroCompte" /
                </s:param>
                <s:param name="client">
                <s:param name="idClient">
                        <s:property value="value.owner.userId" /
                </s:param>
        <!-- Fin Correction Bug Bloquant N1 -->
        </s:url> <s:a href="%{deleteAccount}" onclick="return cc
```

L'action deleteUser a un bug de la même nature, on injecte également dans celui-ci idClient :

```
<td><s:url action="deleteUser" var="deleteUser">
                <s:param name="client">
                <!-- Correction Bug Bloquant N1 -->
                <s:param name="idClient">
                        <s:property value="value.userId" />
                </s:param>
                <!-- Fin Correction Bug Bloquant N1 -->
```

Bug N2 Ajout Compte à un Utilisateur :

L'ajout de compte a un utilisateur retourne une erreur de Struts, l'ajout ne peut donc pas se faire.

Ainsi ce bug est bloquant, pour le régler nous commençons par injecter l'idClient et non le client depuis le JSP :

```
<s:if test="(!aDecouvert)">
        <td><s:url action="urlAddAccount" var="addAccount">
                <s:param name="client">
            <!-- Correction Bug Bloquant N2 -->
                <s:param name="idClient">
            <!-- Fin Correction Bug Bloquant N2 -->
                        <s:property value="value.userId" />
```

Puis nous l'utilisons dans le JSP de CreerCompte.jsp

```
<body>
    <h1>Créer un nouveau compte</h1>
-
+       <!-- Correction Bug Bloquant N2 -->
    <p>
        Client choisi :
-       <s:property value="client" />
+       <s:property value="idClient" />
    </p>
+   <!-- Fin Correction Bug Bloquant N2 -->
    <s:form id="myForm" name="myForm" action="addAccount" method="POST">
-       <input type="hidden" name="client"
-           value="<s:property value='client.userId' />">
+       <input type="hidden" name="idClient"
+           value="<s:property value='idClient' />">
        <s:textfield label="Numéro de compte" name="numeroCompte" />
        <s:radio label="Type" name="avecDecouvert"
            list="#{false:'Compte sans découvert',true:'Compte avec découvert'}"
```

On modifie également créationCompte qui est l'action utilisée lors de la création de compte, pour utiliser l'idClient :

```
   // Correction Bug Bloquant N2
   /**
    * Action créant un compte client ou gestionnaire.
    *
92,12 +211,17 @@ public class CreerCompte extends ActionSupport {
    */
   public String creationCompte() {
           try {
                   if (avecDecouvert) {
                           banque.createAccount(numeroCompte, client, decouvertAutorise);
                   client = banque.getClientById(idClient);
                   if(client != null) {
                           if (avecDecouvert) {
                                   banque.createAccount(numeroCompte, client, decouvertAutorise);
                           } else {
                                   banque.createAccount(numeroCompte, client);
                           }
                           this.compte = banque.getCompte(numeroCompte);
                   } else {
                           banque.createAccount(numeroCompte, client);
                           throw new TechnicalException("Client introuvable");
                   }
                   this.compte = banque.getCompte(numeroCompte);
                   return "SUCCESS";
           } catch (IllegalOperationException e) {
                   e.printStackTrace();
08,4 +232,5 @@ public class CreerCompte extends ActionSupport {
                   return "INVALIDFORMAT";
           }
   }
   // Fin Correction Bug Bloquant N2
```

on oublie pas de throw une Exception dans le cas client est null.

Bug N3 Déconnection Manager, visionnage des données User sans être connecté :

Lorsqu'un gestionnaire se déconnecte les données des users restent stockés dans les classes de l'application.
Ainsi après la déconnexion une personne mal intentionné pourrait obtenir ces données, pour corriger ca nous créons une fonction pour clean ces données et nous l'appelons dans le logout.

Dans le banque Manager :

```
// Correction Bug Bloquant N3
        /**
 * Méthode pour vider les clients chargés
 */
public void cleanClients() {
        bank.setClients(null);
}
// Fin Correction Bug Bloquant N3
```

Et dans

Le log out dans la Banque Façade :

```
    // Correction BUG Bloquant N3
    /**
     * Méthode pour déconnecter l'utilisateur.
     */
    public void logout() {
            banqueManager.cleanClients();
            loginManager.logout();
    }
    // Fin Correctio Bug Bloquant
```