

<b>Liste des bugs découverts et résolus :</b>	<b>2</b>
BUG NON BLOQUANT :	2
BUG_Non_bloquant_Nr_1 :	2
Problème :	2
Résolution :	2
Code impacté pour l'image "modifier" :	3
Commit :	3
Code impacté pour le logo :	3
Commit :	4
Tests :	4
BUG_Non_bloquant_Nr_2 :	4
Problème :	4
Résolution :	4
Code impacté :	4
Commit :	5
Tests :	5
BUG_Non_bloquant_Nr_3 :	5
Problème :	5
Solution :	5
Code impacté :	6
Commit :	6
Test :	6
BUG BLOQUANT :	7
BUG_bloquant_Nr_1 :	7
Problème :	7
Solution :	7
Code impactés :	7
Commit :	8
Test :	8
BUG_bloquant_Nr_2 :	8
Problème :	8
Solution :	8
Code impactés :	9
Commit :	9
Test :	9
BUG_bloquant_Nr_3 :	10
Problème :	10
Solution :	10
Code impactés :	10
Commit :	11
Test :	11
<b>Synthèse de ce qui a été mis en place durant le projet :</b>	<b>12</b>
Situation avant les modifications :	12
Situation après les modifications :	12

## Liste des bugs découverts et résolus :

### BUG NON BLOQUANT :

#### BUG\_Non\_bloquant\_Nr\_1 :



Logo de la page d'accueil et image du bouton "Modifier" sur le tableau de bord administrateur

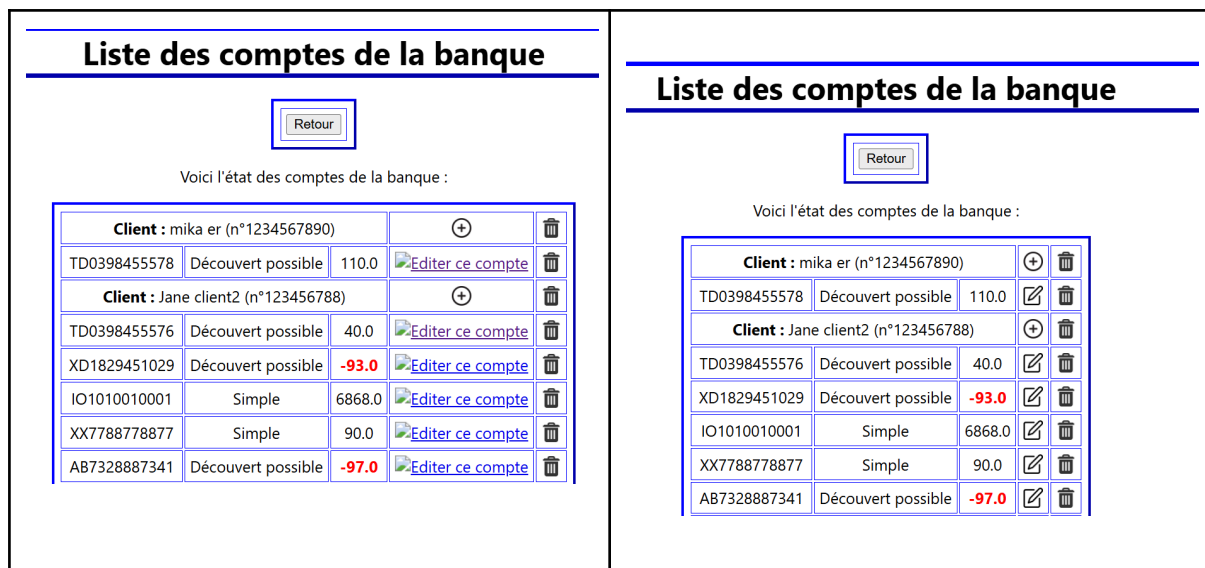
#### Problème :

Le logo de la page d'accueil et l'image du bouton "Modifier" ne s'affichaient pas sur le tableau de bord de l'administrateur. Cela était dû à des liens d'images obsolètes qui pointaient vers des ressources non disponibles. Ce problème affectait l'expérience utilisateur, bien qu'il n'empêchait pas le bon fonctionnement global de l'application.

#### Résolution :

Pour résoudre ce problème, nous avons simplement identifié les fichiers contenant les liens obsolètes et nous les avons remplacés par des liens corrects renvoyant bien une image disponible.

Avant	Après
<p><b>Bienvenue sur l'application IUT Bank 2023</b></p> <p> logo</p> <p>Information</p> <p><a href="#">Page de Login</a></p>	<p><b>Bienvenue sur l'application IUT Bank 2023</b></p> <p> UNIVERSITÉ DE LORRAINE</p> <p>Information</p> <p><a href="#">Page de Login</a></p>



Code impacté pour l'image "modifier" :

- Image modifiée sur la page administrateur remplacement de la ligne 100 dans le fichier ListeCompteManager.jsp  
src="<http://freeflaticons.com/wp-content/uploads/2014/10/write-copy-14138051958gn4k.png>" par la ligne src="<https://cdn-icons-png.flaticon.com/512/860/860814.png>".

Commit :

- correction : image modifiée compte dans la page ListeCompteManager

Code impacté pour le logo :

- Logo modifié sur la page de login remplacement de la ligne 23-24  
src="[https://www.iut-metz.univ-lorraine.fr/images/AdminSite/Logos/Logo\\_IUT\\_Metz.U.L.small.png](https://www.iut-metz.univ-lorraine.fr/images/AdminSite/Logos/Logo_IUT_Metz.U.L.small.png)" alt="logo" /> par la ligne  
src="[https://upload.wikimedia.org/wikipedia/commons/thumb/5/5b/Logo\\_Universit%C3%A9\\_de\\_Lorraine.svg/1200px-Logo\\_Universit%C3%A9\\_de\\_Lorraine.svg.png](https://upload.wikimedia.org/wikipedia/commons/thumb/5/5b/Logo_Universit%C3%A9_de_Lorraine.svg/1200px-Logo_Universit%C3%A9_de_Lorraine.svg.png)" alt="logo" style="width: 240px; height: 83px;"

Commit :

- Fixed: Cryptage du mot de passe lors de la création d'un nouvel utilisateur + Nouveau logo

Tests :

Un test visuel simple a été réalisé pour vérifier que l'image du bouton "Modifier" s'affiche correctement sur le tableau de bord administrateur. Et que le logo s'affiche correctement sur la page de connexion après modification des liens.

BUG\_Non\_bloquant\_Nr\_2 :

Accès à n'importe quel compte en modifiant l'url.

Problème :

Le numéro de compte bancaire d'un utilisateur apparaissait dans l'url lorsque celui-ci consultait un de ses comptes. La cause était l'utilisation d'une méthode GET au lieu de POST pour transmettre les données sensibles. Le problème lié à cela était l'exposition d'informations sensibles dans l'url et l'accès à n'importe quel compte en modifiant l'url.

Résolution :

Nous avons identifié la méthode GET utilisée dans le fichier Connect.jsp et nous avons modifié le code pour utiliser une méthode POST.

Code impacté :

- Dans le fichier connect.jsp ligne 32 à 37

```
<td><s:url action="urlDetail" var="urlDetail">
    <s:param name="compte"><s:property value="key" /></s:param>
    <!-- <s:param name="idCompte"><s:property value="key" /></s:param> --%>
</s:url> <s:a href="%{urlDetail}">
    <s:property value="key" />
</s:a></td>
```

remplacé par

```
<td>
    <s:form action="urlDetail" method="POST">
        <s:hidden name="compte" value="%{key}" />
        <s:submit value="%{key}" />
    </s:form>
</td>
```

Commit :

- Correction de sécurité : suppression du numéro de compte dans l'URL pour éviter l'accès non autorisé à d'autres comptes.

Tests :

-

Affichage avant correction du bug

← → ↺ 🏠 localhost:8080/\_00\_ASBank2023/urlDetail.action?compte=MD8694030938 🔍 ☆

Affichage après correction du bug

← → ↺ 🏠 localhost:8080/\_00\_ASBank2023/urlDetail.action 🔍 ☆

BUG\_Non\_bloquant\_Nr\_3 :

Cryptage des mots de passe dans la base de données et ajout des fonctions de décryptage dans l'application

Problème :

Les mots de passe des utilisateurs en base de données étaient stockés en clair représentant un risque pour la sécurité de l'application. Pour résoudre ce problème, un cryptage des mots de passe a été mis en place. Cependant, l'ajout de cette sécurité a également généré un bug bloquant lors de la création d'un nouvel utilisateur (voir **BUG\_Bloquant\_Nr\_2**).

Solution :

Nous avons crypté les mots de passe existant avec la fonction SQL :

UPDATE Utilisateur SET userPwd=MD5(userPwd)

Avant le cryptage en DB :

<input type="checkbox"/>	✎ Éditer	📋 Copier	🗑 Supprimer	s.s122144	knoffel	xavier	1 rue du saulcy	test123	1	CLIENT	1223344551
--------------------------	----------	----------	-------------	-----------	---------	--------	-----------------	---------	---	--------	------------

Après le cryptage en DB :

<input type="checkbox"/>	✎ Éditer	📋 Copier	🗑 Supprimer	s.s122144	knoffel	xavier	1 rue du saulcy	cc03e747a6afbcbf8be7668acfebee5	1	CLIENT	1223344551
--------------------------	----------	----------	-------------	-----------	---------	--------	-----------------	---------------------------------	---	--------	------------

Puis, une fonction a été ajoutée dans l'application pour crypter le mot de passe saisi par l'utilisateur au moment de la connexion. Le mot de passe crypté est ensuite comparé à celui stocké dans la base de données. Pour connecter l'utilisateur.

Code impacté :

- Ajout de la fonction crypterMD5 dans le fichier DaoHibernate.java

```
261 +     public String crypterMD5(String motDePasse) {
262 +         try {
263 +             MessageDigest md = MessageDigest.getInstance("MD5");
264 +             // Conversion du mot de passe en tableau de bytes et mise à jour du MessageDigest
265 +             md.update(motDePasse.getBytes());
266 +
267 +             // jsp ce que ça fait
268 +             byte[] digest = md.digest();
269 +             // Construction d'une chaîne hexadécimale à partir des bytes
270 +             StringBuilder sb = new StringBuilder();
271 +             for (byte b : digest) {
272 +                 // Conversion de chaque byte en hexadécimal
273 +                 sb.append(String.format("%02x", b));
274 +             }
275 +             return sb.toString();
276 +         } catch (NoSuchAlgorithmException e) {
277 +             throw new RuntimeException("Erreur : algorithme MD5 non trouvé.", e);
278 +         }
279 +     }
```

- appel de la fonction crypterMD5 dans la fonction isUserAllowed

```
204 -         return false;
205 -     }
206 -     return (userPwd.equals(user.getUserPwd()));
204 +         return false;}
205 +         String encryptedPwd = crypterMD5(userPwd);
206 +         return encryptedPwd.equals(user.getUserPwd());
207 +
208 +
```

Commit :

- Implémentation de la fonction de cryptage du mot de passe

Test :

- Mise en place d'un test unitaire sur la fonction crypterMD5

```

@Test new *
public void testCrypterMD5() {
    DaoHibernate daoHibernate = new DaoHibernate();

    try {

        String motDePasse = "azerty123";
        String mdpCrypte = daoHibernate.crypterMD5(motDePasse);

        assertEquals( expected: "882baf28143fb700b388a87ef561a6e5", mdpCrypte);

    } catch (IllegalArgumentException e) {
        fail("Le mot de passe devrait être chiffré et rendre le bon résultat");
        e.printStackTrace();
    }
}

```

## BUG BLOQUANT :

### BUG\_bloquant\_Nr\_1 :

Action débiter qui ne fonctionnait pas :

Problème :

Lorsqu'un utilisateur voulait débiter de l'argent sur son compte le bouton "débiter" créditait en réalité. La cause était que la configuration ne permettait pas d'identifier plusieurs actions dans un même formulaire.

Solution :

Ajout d'une ligne de code pour activer le mapping des actions multiples

Code impactés :

- Dans le fichier struts.xml :

```

14      14          <constant name="struts.devMode" value="true" />
15      +          <constant name="struts.mapper.action.prefix.enabled" value="true" />
16      +
15      17

```

Commit :

- Fixed: Correction de l'action de débiter un compte

Test :

- Des tests d'intégration ont été réalisés en utilisant l'application pour effectuer des opérations de débit et de crédit sur un compte. Ensuite, une vérification a été effectuée dans la base de données pour s'assurer que les données avaient été mises à jour correctement.

BUG\_bloquant\_Nr\_2 :

Le mot de passe n'était pas crypté à la création d'un nouvel utilisateur qui rendait la connexion impossible.

Problème :

Lors de la création d'un nouvel utilisateur par un administrateur, le mot de passe n'était pas crypté avant d'être sauvegardé dans la base de données. Cela entraînait une erreur lors de la connexion, car l'application tentait de comparer un mot de passe crypté (saisi au moment du login) avec un mot de passe stocké en clair en base de données.

Solution :

Import de la fonction crypterMD5 provenant de DaoHibernate dans le fichier CreerUtilisateur.java afin d'éviter la duplication de code et permettant d'appeler la fonction pour crypter le mot de passe lors de la création d'un utilisateur par l'administrateur.



## Code impactés :

```
29 30 private String result;
31 + private DaoHibernate daoHibernate;
30 32
31 33
32 34 /**
***
↓
@@ -153,32 +155,13 @@
↑
***
153 155 * Constructeur sans paramètre de CreerUtilisateur
154 156 */
155 157 public CreerUtilisateur() {
158 + this.daoHibernate = new DaoHibernate();
156 159 System.out.println("In Constructor from CreerUtilisateur class ");
157 160 ApplicationContext context = WebApplicationContextUtils
158 161 .getRequiredWebApplicationContext(ServletActionContext.getSe
159 162 this.banque = (BanqueFacade) context.getBean("banqueFacade");

226 209 public String creationUtilisateur() {
227 210 try {
228 - String cryptPwd = crypterMD5(userPwd);
211 + String cryptPwd = this.daoHibernate.crypterMD5(userPwd);
229 212
230 213 if (cryptPwd != null) {
```

## Commit :

- Fixed: Suppression de duplication de code

## Test :

- Mise en place d'un test unitaire sur la fonction CreateUser pour tester que le mot de passe a bien été crypté.

```
public void testCreateUserCrypter() { new {
String cryptPwd = daoHibernate.crypterMD5( motDePasse: "userPwd");
try {
try {
daoHibernate.createUser( nom: "NOM", prenom: "PRENOM", adresse: "ADRESSE", male: true, userId: "c.new1", cryptPwd, manager: false, numClient: "5544554455");
} catch (IllegalArgumentException e) {
fail("Il ne devrait pas y avoir d'exception ici");
} catch (IllegalFormatException e) {
fail("Il ne devrait pas y avoir d'exception ici");
}
Utilisateur user = daoHibernate.getUserById("c.new1");
assertEquals( expected: "NOM", user.getNom());
assertEquals( expected: "PRENOM", user.getPrenom());
assertEquals( expected: "ADRESSE", user.getAdresse());
assertEquals( expected: "c.new1", user.getUserId());
assertEquals(cryptPwd, user.getUserPwd());
assertTrue(user.isMale());
} catch (TechnicalException he) {
fail("L'utilisateur aurait du être créé.");
}
}
}
```

## BUG\_bloquant\_Nr\_3 :

Redirection sur la page login non fonctionnelle sur l'application dockerisée

Problème :

Lorsque l'image de l'application était lancée, la redirection vers la page de login à l'aide du lien "Page Login" ne fonctionnait pas. En effet, l'url enlevait la partie "localhost:{port}" pour ne garder que "http://jsp/redirectionLogin.action". Donc on arrivait sur une page inexistante et inaccessible.

Solution :

Dans le fichier web.xml, on retrouve cette redirection. On peut noter le caractère "/" au début du lien. Ce simple caractère enlève la partie initiale "localhost/{port}", ce qui empêche d'accéder à la page. En l'enlevant, le chemin indiqué dans le fichier devient relatif au contexte de l'application et non pas absolu.

Code impactés :

```
WebContent/WEB-INF/web.xml
@@ -10,7 +10,7 @@
10     <url-pattern>/*</url-pattern>
11 </filter-mapping>
12 <welcome-file-list>
13 -    <welcome-file>/JSP/Index.jsp</welcome-file>
14 </welcome-file-list>
15 <context-param>
16     <param-name>contextConfigLocation</param-name>
```

```

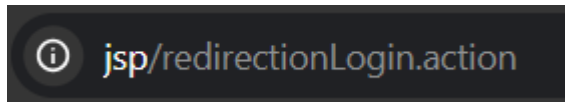
10     <url-pattern>/*</url-pattern>
11 </filter-mapping>
12 <welcome-file-list>
13 +    <welcome-file>JSP/Index.jsp</welcome-file>
14 </welcome-file-list>
15 <context-param>
16     <param-name>contextConfigLocation</param-name>
```

Commit :

- Fixed : Bug redirection du login sur l'app dockerisé + Test fonction CrypterMD5

Test :

- Avant la correction :



### Ce site est inaccessible

Impossible de trouver l'adresse IP du serveur de **jsp**.

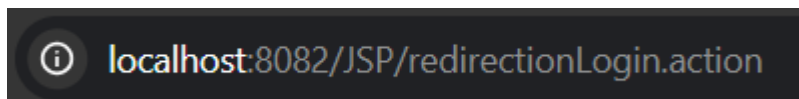
Voici quelques conseils :

- Vérifier la connexion
- [Vérifier les configurations du proxy, du pare-feu et du DNS](#)
- [Exécutez les diagnostics réseau de Windows](#)

ERR\_NAME\_NOT\_RESOLVED

Actualiser

- Après la correction :



## Login :

Code user:

Password:

Submit

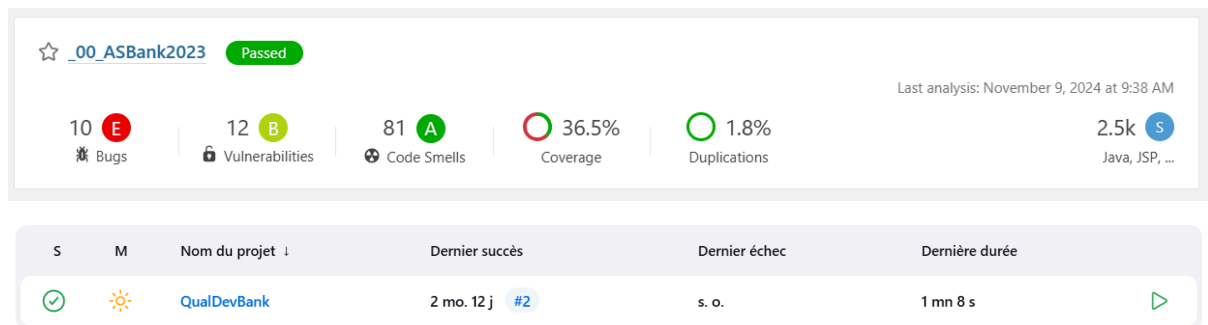
Retour à l'accueil

Projet BUT-3A / 2023-2024

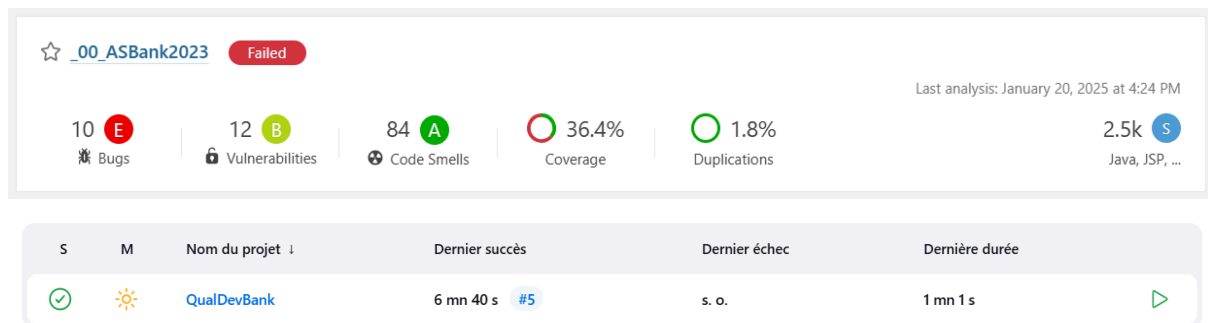
# Captures d'écran de SonarQube et Jenkins :

Nous rappelons que lors du lancement de Jenkins, un build de l'application est automatiquement effectué avec "mvn clean install", et cela lance automatiquement une analyse SonarQube.

Avant :



Après :



## Synthèse de ce qui a été mis en place durant le projet :

Situation avant les modifications :

- Absence de documentation technique d'installation
- Aucun outil de qualité de code (Sonar) mis en place
- Plusieurs bugs bloquants et non bloquants
  - Absence de cryptage des mots de passe dans la base de donnée
  - Le logo et l'image du bouton "modifier" qui ne s'affiche pas
  - Bouton "débit" qui crédite
  - etc ...

## Situation après les modifications :

- Mise en place d'un Github et d'un Trello
- Rédaction d'une documentation d'installation claire
- Création de fiches détaillées pour chaque bug
- Utilisation de Sonar cloud pour analyser le code
- Dockerisation de l'application et de la DB
- Correction des bugs bloquants et non bloquants
- Ajout de tests unitaires, d'intégrations, visuels pour garantir le bon fonctionnement des fonctionnalités ajoutées.
- Jenkins

En conclusion, la phase de DevOps sur ce projet a permis de l'améliorer considérablement, tant en termes de qualité que de fiabilité. Grâce aux outils et pratiques mis en place, nous avons transformé un projet initialement peu maintenable en une base solide et professionnelle.

La documentation claire et détaillée simplifie l'installation et la prise en main du projet, même pour de nouveaux arrivants. L'utilisation d'outils comme GitHub, Trello et Sonar Cloud a renforcé la gestion du projet, la collaboration et la surveillance de la qualité du code. De plus, la dockerisation assure un déploiement plus rapide et homogène, réduisant considérablement les erreurs liées à l'environnement.

Les corrections des bugs, l'ajout de tests unitaires, d'intégration et visuels, ainsi que l'automatisation via Jenkins, garantissent un fonctionnement stable et une régression minimale lors de futures évolutions.

Ce projet est désormais non seulement fiable, mais également conçu pour être maintenu et évoluer efficacement par une autre équipe.