

# Compte Rendu : Résolution des Bugs

## Contexte

Ce document détaille les actions menées pour identifier, résoudre, et valider les bugs rencontrés dans l'application Java. Les bugs sont classés en fonction de leur impact (bloquant ou non bloquant). Chaque section présente :

1. Une description du problème / Ajout sur Trello.
2. Les étapes de résolution (+ identification dans le code + commit).
3. Les tests effectués pour validation.

---

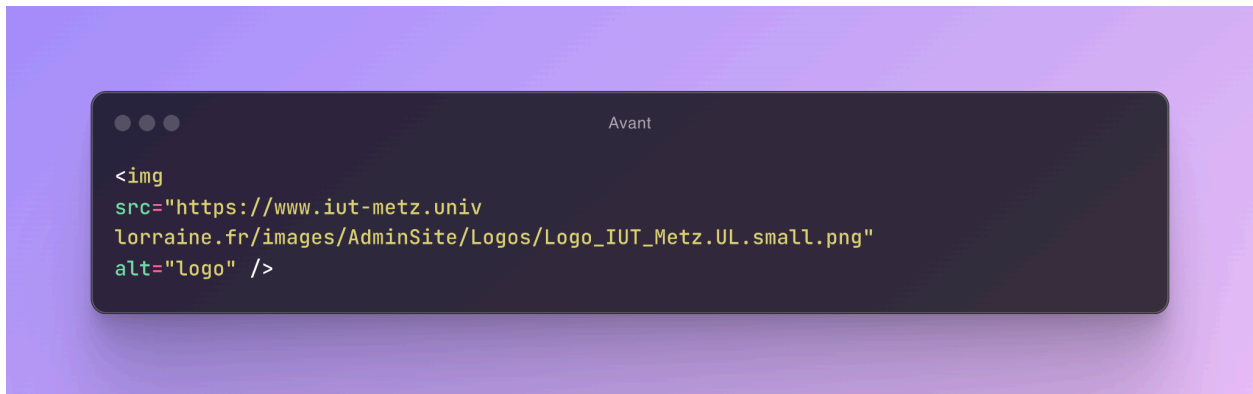
<b>Compte Rendu : Résolution des Bugs</b>	<b>1</b>
Bug N°1 : Le logo ne se charge pas dans l'application	2
Résolution	2
Tests	3
Bug N°2 : Faille de sécurité, cryptage des mots de passes	4
Résolution	4
Tests	5
Bug N°3 : Créditer, Débiter	6
Résolution	6
Tests	6
Bug N°4 : Redirection vers la page de login	8
Résolution	8
Tests	9
Bilan du projet	10

## Bug N°1 : Le logo ne se charge pas dans l'application

- Type : Non bloquant
- Description :
  - Le chemin d'accès au logo pointait vers une ressource hébergée sur un serveur externe.
  - Cela cause le non chargement de l'image sur l'application
  - Impact : Affichage d'une image manquante à la place du logo.

## Résolution

- Étapes effectuées :
  - Création d'un dossier local `images` dans le projet.
  - Déplacement du fichier de logo dans ce dossier.
  - Modification du code pour pointer vers le logo local.
- Commit Git :
  - Message : Résolution du bug N°2 : Passage à un chemin local pour le logo.
  - ID du commit : 0fa7ddc



## Tests

1. Test Unitaire (TU) : Chargement correct de l'image via le nouveau chemin local.
2. Test de Non-Régression (TR) : Aucun impact négatif détecté sur d'autres fonctionnalités.

**Bienvenue sur l'application IUT Bank 2023**



Information

[Page de Login](#)

Projet BUT-3A / 2023-2024

## Bug N°2 : Faille de sécurité, cryptage des mots de passes

- Type : non bloquant
- Description :
  - Les mots de passe des utilisateurs étaient enregistrés en clair dans la base de données, ce qui constitue une faille de sécurité majeure.
  - A la connexion ou la création d'un utilisateur le mot de passe est enregistré en clair dans la base de donnée
  - Impacte : Danger pour la sécurité des données de l'utilisateur

### Résolution

- Étapes effectuées :
  - Création d'une fonction hash qui utilise l'algorithme SHA-256
  - Intégration de cette fonction dans les classes BanqueManager et LoginManager pour :

Hacher les mots de passe lors de la création d'un nouvel utilisateur.

Comparer les mots de passe hachés à la connexion.
  - Mise à jour des mots de passe existants dans la base de données sous forme de hash sécurisé.
- Commit Git :
  - Message : maj hash final v2.
  - ID du commit : 258b423


```
static public String hashPassword(String password) {
    try {
        MessageDigest digest = MessageDigest.getInstance("SHA-256");
        byte[] encodedHash = digest.digest(password.getBytes());

        StringBuilder hexString = new StringBuilder();
        for (byte b : encodedHash) {
            String hex = Integer.toHexString(0xff & b);
            if (hex.length() == 1) hexString.append('0');
            hexString.append(hex);
        }
        return hexString.toString();
    } catch (NoSuchAlgorithmException e) {
        throw new RuntimeException(e);
    }
}
```


## Tests

1. Test Unitaire (TU) : Vérification que tous les mots de passe existants sont désormais stockés sous forme de hash sécurisé.
2. Test Fonctionnel (TU) : Création d'un utilisateur avec un mot de passe : vérification que le mot de passe est stocké en hash dans la base de données.

Connexion avec le mot de passe haché : confirmation que la validation fonctionne correctement.

	 <b>userId</b> varchar(50) ▲	<b>nom</b> varchar(45) ▲	<b>prenom</b> varchar(45) ▲	<b>adresse</b> varchar(100) ▲	<b>userPwd</b> varchar(250) ▲
1	a	a	a	a	a
2	admin	Smith	Joe	123, grande rue, Metz	adminpass
3	client1	client1	Jane	45, grand boulevard, ...	clientpass1
4	client2	client2	Jane	45, grand boulevard, ...	clientpass2

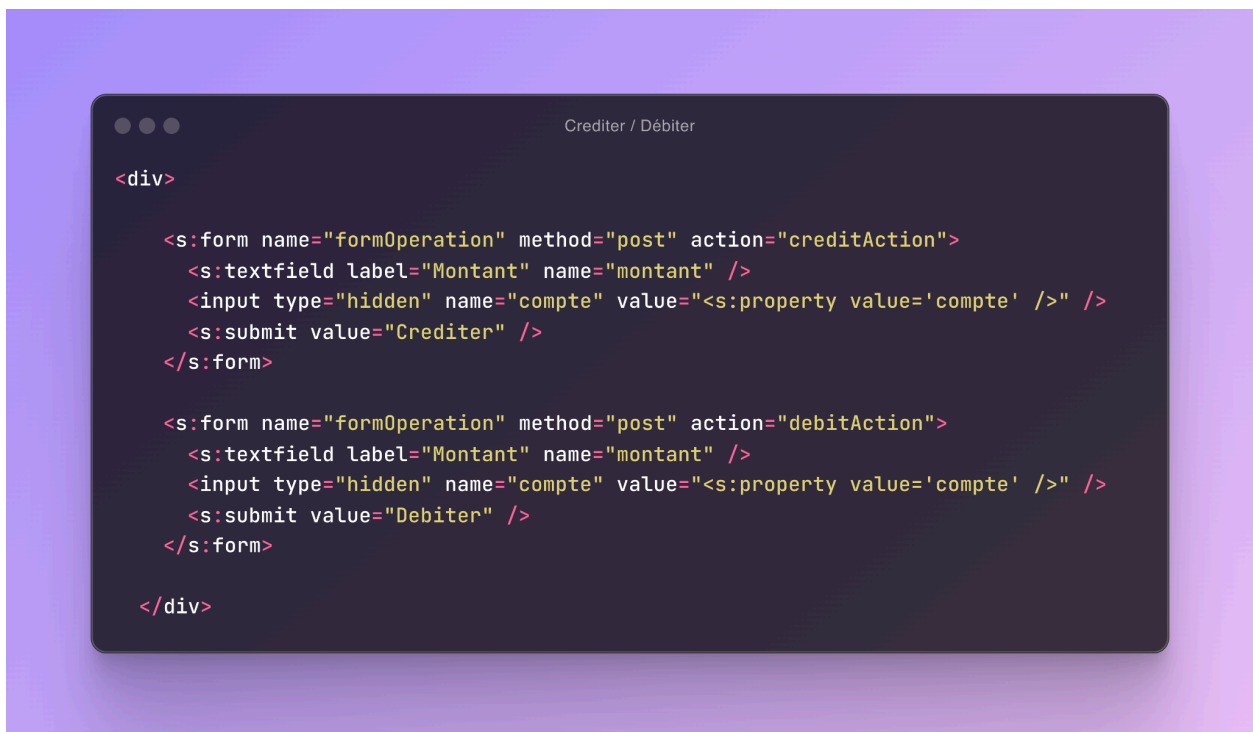
	 <b>userId</b> varchar(50) ▲	<b>nom</b> varchar(45) ▲	<b>prenom</b> varchar(45) ▲	<b>adresse</b> varchar(100) ▲	<b>userPwd</b> varchar(250) ▲
1	a	a	a	a	ca978112ca1bbdcf...
2	admin	Smith	Joe	123, grande rue, Metz	713bfda78870bf9d1...
3	client1	client1	Jane	45, grand boulevard, ...	11fa5aade33a4f8ea1...
4	client2	client2	Jane	45, grand boulevard, ...	699a3eec2c9f94989...

## Bug N°3 : Créditer, Débiter

- Type : Bloquant
- Description :
  - Les méthodes créditer et débiter de l'application effectuaient la même action, ce qui empêchait les opérations financières de fonctionner correctement. Ce bug se trouve du côté client mais également du côté administrateur, lorsqu'il édite un compte.

## Résolution

- Étapes effectuées :
  - Ajout de deux champs distincts (deux forms) dans le code pour gérer séparément les opérations de crédit et de débit.
- Commit Git :
  - Message : correction bug crediter/debiter
  - ID du commit : 7fdcc81



```
Crediter / Débiter

<div>

  <s:form name="formOperation" method="post" action="creditAction">
    <s:textfield label="Montant" name="montant" />
    <input type="hidden" name="compte" value="<s:property value='compte' />" />
    <s:submit value="Crediter" />
  </s:form>

  <s:form name="formOperation" method="post" action="debitAction">
    <s:textfield label="Montant" name="montant" />
    <input type="hidden" name="compte" value="<s:property value='compte' />" />
    <s:submit value="Debiter" />
  </s:form>

</div>
```

## Tests

1. Test Fonctionnel (TU) : Débit d'argent sur un compte administrateur ainsi que client.
2. Test de Non-Régression (TR) : Aucun impact négatif détecté sur d'autres fonctionnalités.

## Détail du Compte FF5050500202

Logout

Retour

Type de compte : Simple  
Solde : 805.0

Montant:	<input type="text" value="50"/>
	<input type="button" value="Crediter"/>

Montant:	<input type="text" value="50"/>
	<input type="button" value="Debiter"/>

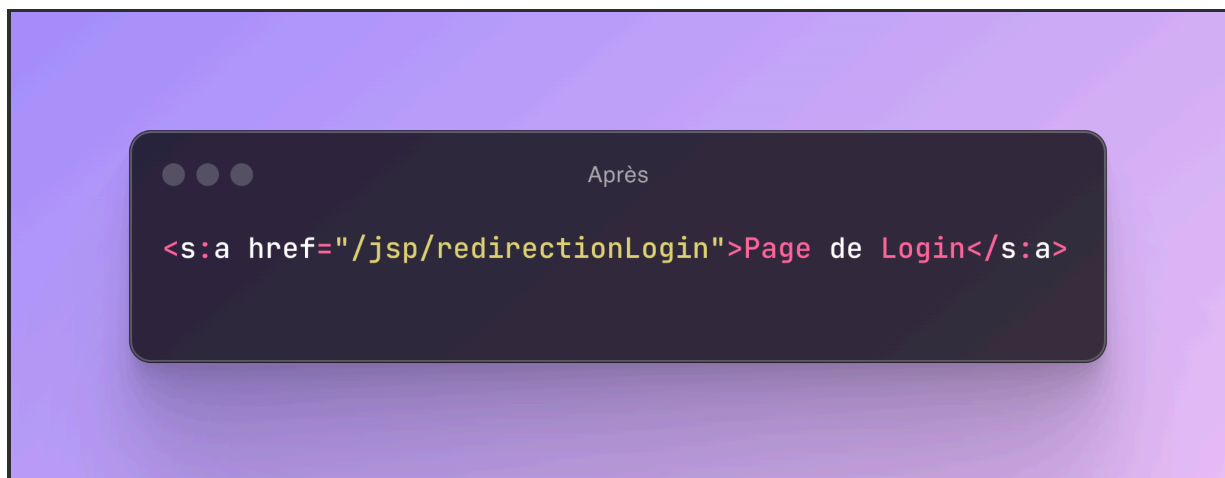
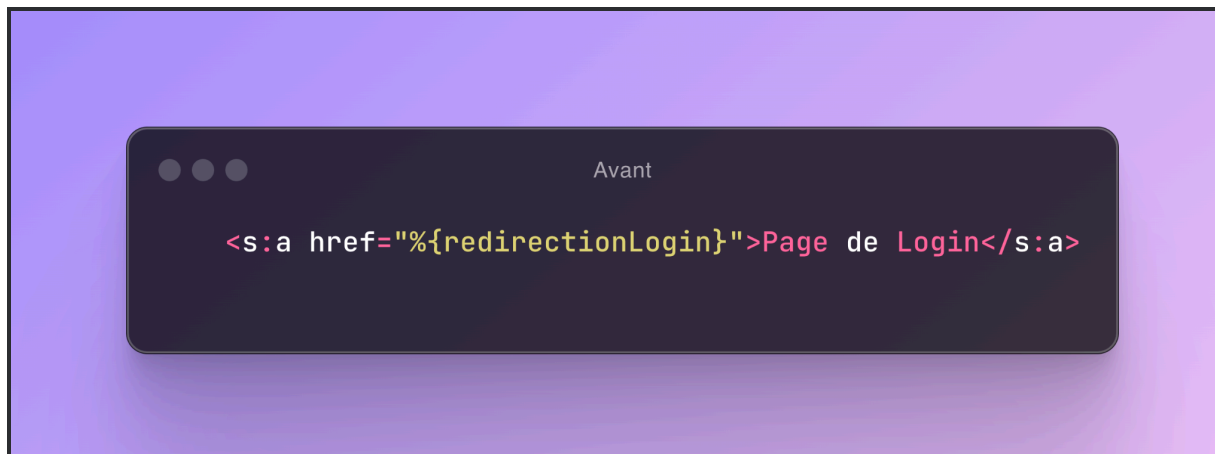
Projet BUT-3A / 2023-2024

## Bug N°4 : Redirection vers la page de login

- Type : Bloquant
- Description :
  - Lors de la mise en production de l'application Dockerisée sur le serveur Tomcat, la redirection vers la page de login échouait. Contrairement au serveur local, l'application ne possédait pas de point d'entrée pour gérer la redirection correcte vers localhost:8083/jsp/redirectionLogin. Lors de la tentative d'accès à la page de login, l'application renvoyait une URL relative (jsp/redirectionLogin), ce qui empêchait l'accès à la page de login en raison de l'absence de chemin d'accès complet.
  - Cela engendrer le blocage de l'accès à la page de connexion, rendant l'application inutilisable pour les utilisateurs.

### Résolution

- Étapes effectuées :
  - Modification du **href** dans le lien de la page index.jsp pour inclure l'URL complète avec le port et le chemin correct.

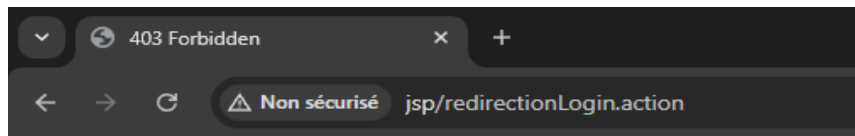




- Commit Git :
  - Message : login redirection fix
  - ID du commit : f878d0e

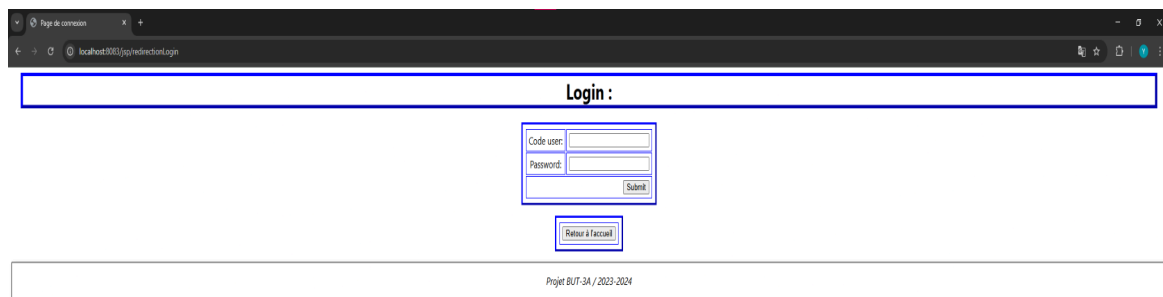
## Tests

1. Test Unitaire (TU) : Vérification que la redirection vers la page de login fonctionne correctement après la modification du lien.
2. Test de Non-Régression (TR) : Confirmation qu'aucune autre fonctionnalité de l'application n'est impactée par cette modification, en particulier la navigation et la gestion des pages.



# Forbidden

You don't have permission to access /redirectionLogin.action on this server.



## Bilan du projet

Voici les améliorations et les actions réalisées pour moderniser et optimiser l'application :

Documentation complète :

- Création d'un guide d'installation détaillant les étapes nécessaires pour déployer l'application sur une nouvelle machine (Mise en place serveur tomcat, configuration, installation base de donnée ,...) Cela a permis de réduire considérablement le temps de mise en place du projet.

Compréhension du code :

- Élaboration d'un schéma UML de la classe métier afin de faciliter la compréhension du fonctionnement global de l'application.

Sécurité renforcée :

- Implémentation d'une fonction de hachage des mots de passe pour sécuriser les données des utilisateurs.

Suivi de projet :

- Mise en place d'un tableau Trello pour organiser et suivre les tâches liées au développement, cela permet une vision plus clair du projet au global

Outils de qualité et d'intégration continue :

- Installation et configuration de SonarCloud pour analyser la qualité du code et de suivre différentes métriques comme le coverage, la duplication de code ,...
- Mise en place de Jenkins pour l'intégration et le déploiement continus.

Automatisation :

- Génération automatique d'une image Docker pour faciliter le déploiement de l'application.

Tests unitaires et couverture de code :

- Rédaction de tests unitaires supplémentaires avec JUnit, afin d'augmenter le pourcentage de couverture de code dans SonarCloud, contribuant ainsi à améliorer la qualité et la fiabilité du projet.

Corrections de bugs :

- Résolution de deux bugs bloquants et de deux bugs non bloquants, contribuant à la stabilité de l'application.