Guide d’utilisation de l’application “IUT BANK”

Réalisé par Melosso Tom, Idoux Clément, Riese Tom, Sabatier Romain

I) Introduction

Ce projet consiste à maintenir et améliorer l'application bancaire du département informatique de l'IUT de Metz. L'application actuelle contient des bugs et n'a pas de documentation.

Le but est donc de :

- Faire fonctionner l'application dans un environnement de développement.
- Corriger les bugs et ajouter les fonctionnalités demandées.
- Fournir une documentation claire pour l'installation, l'utilisation et la maintenance.
- Mettre en place des tests et des outils pour assurer la qualité du code.

Cette documentation a donc pour objectif de guider les futurs utilisateurs et développeurs dans l'installation, l'utilisation et la maintenance de l'application, tout en fournissant une vision complète de sa structure et de ses fonctionnalités.

II) Installation du projet

1) Prérequis

Avant de commencer, assurez-vous d'avoir installé :

- **Java JDK 11**
- **Apache Tomcat 9**
- **IntelliJ IDEA Ultimate**
- **Wamp**

2) Installation et configuration

1. Installer IntelliJ IDEA

- Télécharger IntelliJ IDEA : lien officiel

2. Cloner le projet

- Avec Git :

```
git clone https://github.com/TomRiese57/StickSpin.git
```

- Ou directement via le bouton **"Clone repository"** dans IntelliJ IDEA

2. Mise en place de la base de données

- **Lancer Wamp et connecter vous à PHPMyAdmin : ici**
- **Créer la première base de données "bank_iut_prod", puis importer "dumpSQL.sql" (disponible dans le répertoire script du projet)**
- **Créer la seconde base de données "bank_iut_test", puis importer "dumpSQL_JUnitTest.sql"**

(N'oubliez pas de garder Wamp ouvert lors de l'utilisation du site)

3. Installer le JDK

- Aller dans **File → Project Structure → SDK**
- Installer **JDK 11**

4. Compiler le projet en .war

- Aller dans **Run → Edit Configurations → New → Maven [Clean install]**
- Exécuter **Maven [clean install]**

5. Télécharger TomCat 9.0.10

- Télécharger TomCat : lien officiel

6. Configurer TomCat et lancer le serveur

- Aller dans **Run** → **Edit Configurations** → **New** → **Tomcat Server (local)**
- **Before launch** → ajouter **Build Artifact** → sélectionner **_00_ASBank2023.war** *
- **Deployment** → ajouter **Artifact** → sélectionner **_00_ASBank2023.war exploded**
- En dessous, dans **“Application context”,** ajouter **: “/_00_ASBank2023”**
- Run → **TomCat**

***(Si, quand vous voulez ajouter un artefact à build, aucun n'apparaît, suivez les instructions suivantes :**

- **File → Project Structure → Artifact → + → Web Application: Exploded > From Module → Selectionner le dossier du projet**
- **Puis, de nouveau, + → Web Application:Archive > From _00_ASBANK2023:war Exploded**

)

III) Test unitaires

Voici l’ensemble des tests unitaires réalisés au lancement de l'application, qui vérifient son bon fonctionnement :

1. Mot de passe :

- Hash non nul : vérifie que le hash généré n’est jamais vide.
- Longueur du hash : vérifie que le hash SHA-256 fait bien 64 caractères.
- Vérification mot de passe correct : le mot de passe initial correspond au hash.
- Vérification mot de passe incorrect : un mot de passe différent est rejeté.

2. Base de données :

- Récupération de compte : vérifie qu’un compte existant peut être récupéré et qu’un compte inexistant renvoie null.
- Type de compte : teste que les comptes récupérés sont bien CompteSansDecouvert ou CompteAvecDecouvert.
- Création de compte : teste la création de comptes avec ou sans découvert, et que la création échoue si l’ID existe déjà.
- Suppression de compte : vérifie qu’un compte peut être supprimé et n’est plus récupérable après suppression.
- Récupération d’utilisateur : teste la récupération d’un utilisateur existant ou inexistant, et la distinction entre Client et Gestionnaire.
- Création d’utilisateur : vérifie la création de clients et de gestionnaires, et que la création échoue si l’ID existe déjà.
- Suppression d’utilisateur : vérifie qu’un utilisateur peut être supprimé et n’est plus récupérable après suppression.

- Vérification des droits : teste la méthode isUserAllowed pour différents cas : login/mot de passe correct, mot de passe incorrect, utilisateur inexistant, login ou mot de passe vide ou null.

3. Manager de la banque:
   - Création de client : vérifie qu'un client peut être créé correctement et que la création échoue si un compte avec le même numéro existe.
   - Suppression de compte avec découvert : teste la suppression réussie si le solde est zéro, et l'échec si le solde est différent de zéro.
   - Suppression de compte sans découvert : teste la suppression réussie si le solde est zéro, et l'échec si le solde est différent de zéro.
   - Suppression d'utilisateur : vérifie que la suppression réussit si l'utilisateur n'a pas de comptes ou si ses comptes ont un solde nul.
   - Suppression interdite : teste que la suppression du dernier gestionnaire ou d'un client avec comptes non nuls échoue correctement.

4. Client :
   - Vérification du format d'ID client : teste différents cas valides et invalides pour l'identifiant utilisateur (userId), comme la présence de lettres, chiffres, points, caractères spéciaux, ou chaînes vides.
   - Vérification du format du numéro client : teste que le numéro client contient exactement 10 chiffres et ne comporte ni lettres ni caractères spéciaux.
   - Possède des comptes à découvert : vérifie que la méthode possedeComptesADecouvert renvoie correctement true ou false selon que le client possède des comptes avec ou sans découvert.
   - Comptes avec solde non nul : teste que getComptesAvecSoldeNonNul retourne uniquement les comptes dont le solde est supérieur à zéro, en gérant différents types de comptes et combinaisons.

5. Compte :
   - Méthode créditer : teste que le solde est correctement incrémenté avec un montant positif et que la méthode lève une exception pour un montant négatif.
   - Constructeur de compte : vérifie que le constructeur lève une exception si le numéro de compte est au mauvais format.
   - Vérification du format du numéro de compte : teste plusieurs cas valides et invalides, incluant le nombre de lettres au début, la présence de lettres au milieu ou à la fin, et le nombre exact de chiffres.

6. Compte avec découvert :
   - Vérification de la classe : confirme que l'objet est bien de type CompteAvecDecouvert.
   - Méthode debiter : teste que la méthode lève une exception pour un montant négatif, permet un débit réalisable selon le découvert autorisé, et renvoie une exception si le débit dépasse le découvert autorisé.

7. Compte sans découvert :
   - Vérification de la classe : confirme que l'objet est bien de type CompteSansDecouvert.

- Méthode debiter : teste que la méthode lève une exception pour un montant négatif, permet un débit réalisable selon le solde disponible, et renvoie une exception si le débit dépasse le solde.

8. Contrôleur de Réinitialisation de mot de passe (ResetPasswordAction) :
   - Vérification de la page : test de la redirection correcte vers la page de réinitialisation.
   - Vérification des champs : tests de rejet lorsque l'identifiant, le nouveau mot de passe ou la confirmation sont nuls ou vides.
   - Vérification de la correspondance : test vérifiant que le nouveau mot de passe et sa confirmation correspondent.
   - Vérification de la longueur : test de sécurité pour empêcher un mot de passe trop court.
   - Vérification de l'utilisateur : test d'erreur si l'utilisateur renseigné n'existe pas.
   - Vérification de succès : test du bon changement de mot de passe et du nettoyage des champs après l'opération.
   - Vérification de la gestion des erreurs : test d'une exception lors de la sauvegarde (ex: erreur de la base de données).

9. Contrôleur de Changement de mot de passe (ChangePasswordAction) :
   - Vérification de l'état connecté : empêche le changement de mot de passe sans être connecté.
   - Vérification de l'ancien mot de passe : l'opération est bloquée si l'ancien mot de passe fourni est incorrect, nul ou vide.
   - Vérification du nouveau mot de passe : test de la non-acceptation de champs vides, de mots de passe trop courts, ou d'une mauvaise correspondance avec la confirmation.
   - Vérification de l'historique : test pour éviter que le nouveau mot de passe soit identique à l'ancien.
   - Vérification de succès et erreurs : tests pour le succès de la mise à jour et la gestion des exceptions.

10. DAO Hibernate (Unitaires avec DaoHibernateUnit) :
    - Vérification du rechargement d'utilisateur (reloadUser) : tests pour un utilisateur inexistant, un gestionnaire, et un client avec ses comptes.
    - Vérification de la déconnexion : test pour s'assurer que l'opération ne lève pas d'exception de base de données.
    - Vérification de récupération des comptes (getAccountsByClientId) : vérifie le retour d'une liste selon la validité de l'identifiant.
    - Vérification de la mise à jour : tests validant l'appel des méthodes de base de données pour "updateAccount" et "updateUser".
    - Vérification des accès : test de la solidité de la méthode "isUserAllowed" face à des champs contenant des espaces.
    - Vérification de suppressions : tests validant "deleteAccount" et "deleteUser", et s'assurant que les références nulles sont rejetées.

IV) Nouvelles fonctionnalités

Lors de la maintenance et l'évolution de l'application, les fonctionnalités suivantes ont été ajoutées ou modifiées :

1. Fonctionnalité de changement de mot de passe : les utilisateurs connectés ont désormais un moyen sécurisé de modifier leur mot de passe depuis leur compte, incluant une vérification de leur mot de passe actuel.

2. Fonctionnalité de réinitialisation de mot de passe : en cas d'oubli, un contrôleur "ResetPasswordAction" permet aux utilisateurs de redéfinir leur mot de passe depuis la page de connexion, sans avoir besoin de se connecter, grâce à leur identifiant utilisateur.

3. Ajout de la fonctionnalité de gestion de la carte bancaire : possibilité de création d’une carte bancaire, de gérer son contenue et son plafond, la bloquer et gérer les montants différés

4. Modification d'outils de base de l'application :
- Ajout de la méthode "getUserById" dans "BanqueFacade" et la DAO, offrant un accès optimisé à la recherche d'utilisateur par identifiant.
- Correction d'une erreur qui affichait un message technique vide après une réinitialisation de mot de passe réussie.