

Documentation d'installation et de configuration

Projet Bank IUT

Table des matières

I. Introduction	3
II. Prérequis	5
III. Récupération du projet	6
IV. Maven	7
V. TOMCAT	9
VI. Configuration de la base de données	10
VII. Lancement de l'application	12
VIII. Lancer le build et la compilation sans l'option DSKIPTTEST:	13
IX. Migration des Versions:	14
X. Hachage du Mot de passe:	16
XI. Indicateurs:	19

I. Introduction

Bienvenue dans la documentation d'installation du projet Bank IUT ! Nous sommes ravis que vous ayez choisi notre application pour votre gestion financière personnelle.

Cette documentation a pour objectif de vous guider à travers le processus d'installation de Bank IUT à partir de notre référentiel GIT, afin que vous puissiez rapidement commencer à profiter de ses fonctionnalités. Bank IUT est un projet informatique conçu pour simplifier la gestion de vos finances, en vous offrant un moyen pratique de suivre vos comptes, vos dépenses et vos revenus.

Bank IUT est un projet développé en Java par notre équipe d'étudiants de l'IUT de Metz. Ils ont été créés dans le cadre de notre cursus académique pour répondre aux besoins en gestion financière, tant personnelle que professionnelle. L'application est conçue pour être facile à utiliser, tout en offrant des fonctionnalités puissantes pour vous aider à mieux comprendre et gérer vos finances.

Dans cette documentation, nous vous expliquerons en détail comment installer Bank IUT à partir de notre référentiel GIT. Cette méthode d'installation vous permettra d'obtenir la version la plus récente du projet et de bénéficier des mises à jours ultérieures.

Nous vous guiderons pas à pas à travers le processus d'installation, en vous fournissant des instructions claires, des captures d'écran et des exemples de commandes pour simplifier le processus. Même si vous n'avez aucune

expérience préalable en programmation, vous pourrez suivre ces instructions avec facilité.

Nous vous encourageons également à explorer la documentation complète du projet, qui détaille les fonctionnalités de Bank IUT et vous fournit des conseils pour une utilisation optimale.

Nous vous remercions de faire confiance à Bank IUT pour votre gestion financière, et nous espérons que cette documentation rendra votre expérience d'installation aussi fluide que possible.

Prêt à commencer ? Suivez simplement les étapes décrites dans cette documentation, et vous serez opérationnel en un rien de temps.

II. Prérequis

Avant de commencer le processus d'installation de Bank IUT, assurez vous que votre environnement de développement répond aux pré requis suivants :

1. Java

Bank IUT est une application Java, ce qui signifie que vous devez disposer d'une version de Java à jour sur votre ordinateur. Si vous n'avez pas encore installé Java, vous pouvez l'installer à partir du site officiel d'Oracle ou d'une autre source fiable.

Pour vérifier si Java est déjà installé, ouvrez votre terminal et exécutez la commande suivante :

```
java -version
```

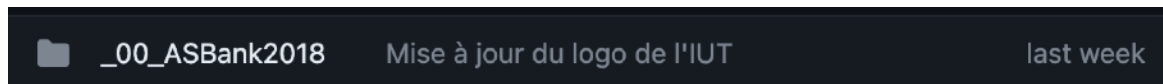
Si Java est correctement installé, vous verrez des informations sur la version de Java installée.

2. Serveur apache Tomcat

Ban IUT nécessite un serveur apache 9.0 pour exécuter l'application. Assurez-vous d'avoir téléchargé un serveur Tomcat avant de procéder à l'installation de l'application. Vous pouvez ce serveur à partir du site officiel d'Apache.

III. Récupération du projet

Sur le Git que nous vous transmis, veuillez récupérer l'entièreté du dossier _00_ASBank2018 (dossier comprenant le code de l'application).

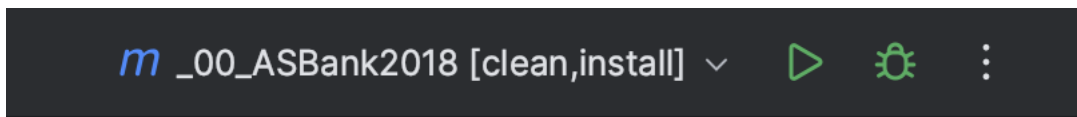


Veuillez ensuite ouvrir le dossier dans l'IDE IntelliJ IDEA.

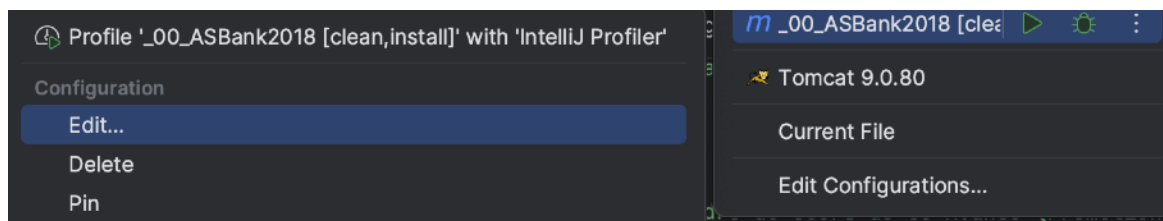
IV. Maven

Maven est l'outil qui nous permettra de build notre application. Pour cela, nous allons devoir le configurer.

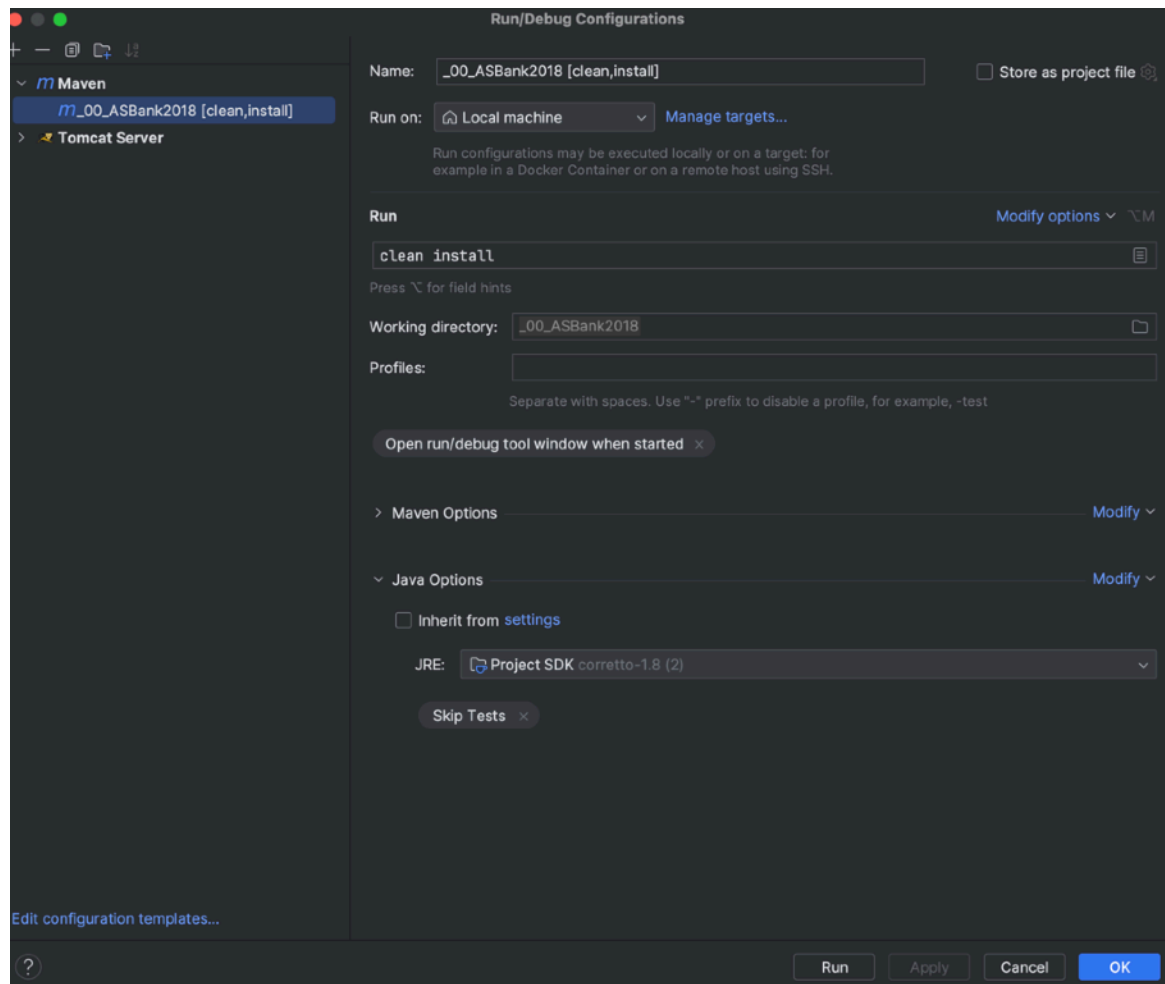
Pour configurer Maven, cliquez sur les 3 points à côté du logo Maven :



Cliquez ensuite sur édit :



Reprenez bien les mêmes paramètres de configuration que la capture d'écran suivante :

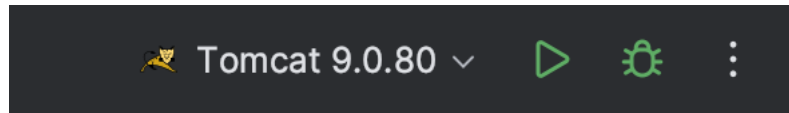


Pour cette version d'installation nous lancerons l'application sans les tests. Veuillez donc à ne pas oublier l'option « Skip Tests ».

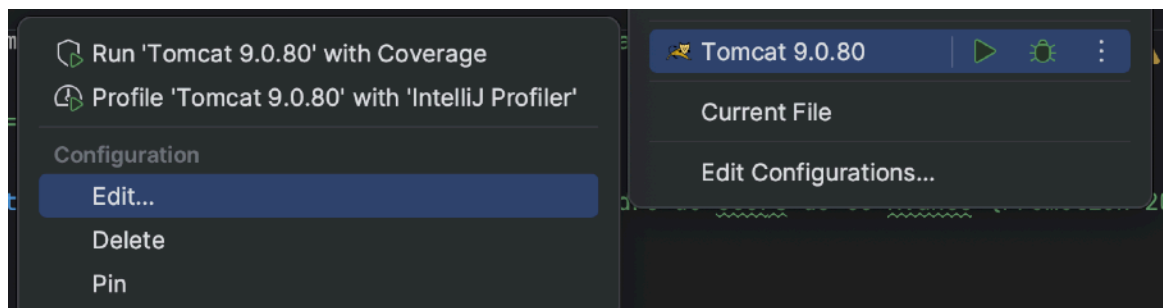
V. TOMCAT

Tomcat est le serveur qui nous permettra de lancer l'application.

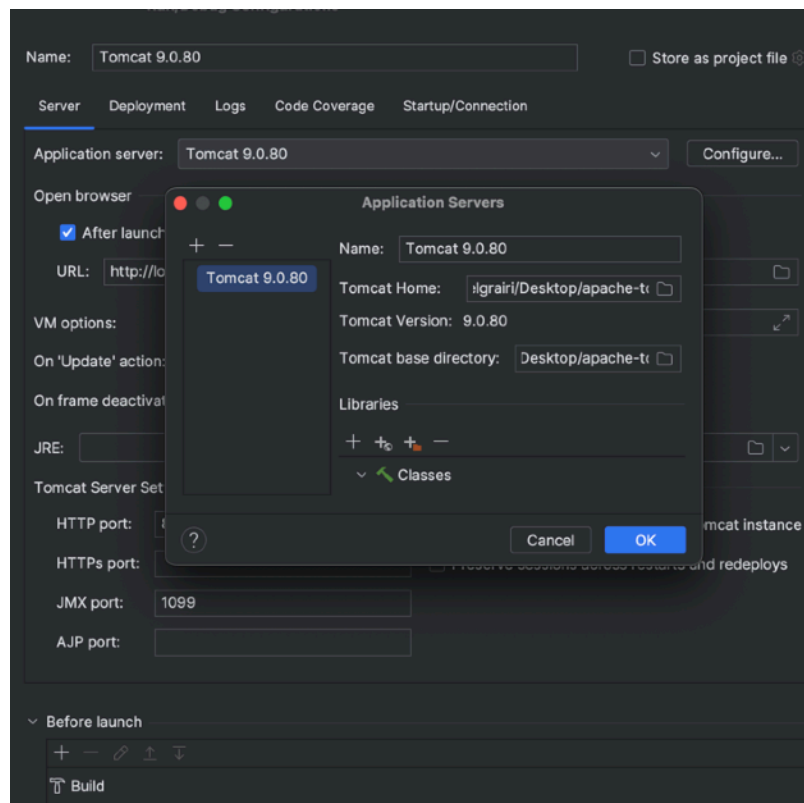
Pour configurer Tomcat, cliquez sur les 3 points à côté du logo Tomcat :



Puis cliquez sur edit :



C'est maintenant que vous pouvez configurer le serveur que vous avez téléchargé dans les prérequis en l'insérant comme ci-dessous :



VI. Configuration de la base de données

Pour configurer la base de données, il faut que vous possédiez un serveur local vous permettant d'utiliser un SGBD. Dans notre cas, nous utiliserons XAMPP mais vous pouvez utiliser celui que vous souhaitez.

Une fois votre SGBD lancé, rendez vous sur la page vous permettant de lancer vos bases de données (chez nous phpMyAdmin).

Cliquez sur « Nouvelle base de données » et créez-la en l'appelant « bankiut » :

Bases de données

Création d'une base de données

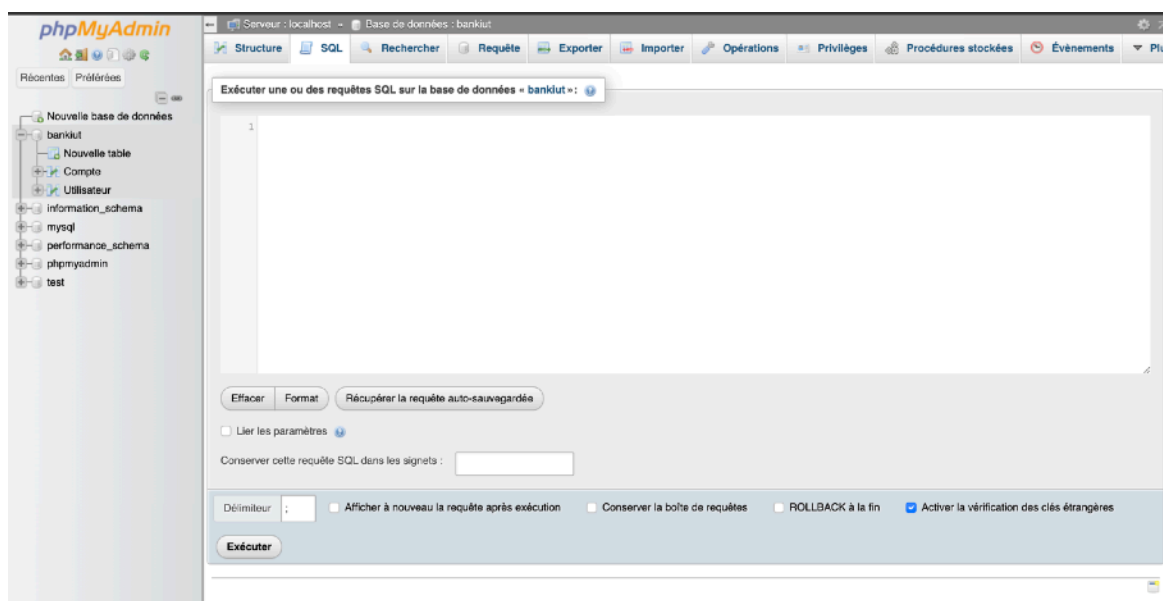
Créer

☐ Tout cocher Supprimer

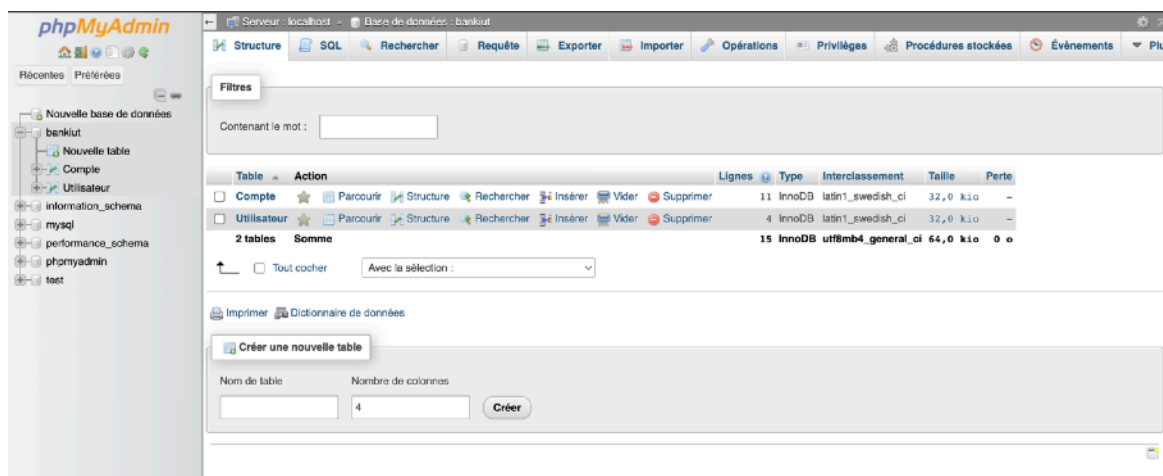
	Base de données	Interclassement	Action
<input type="checkbox"/>	bankiut	utf8mb4_general_ci	 Vérifier les privilèges
<input type="checkbox"/>	information_schema	utf8_general_ci	 Vérifier les privilèges
<input type="checkbox"/>	mysql	utf8mb4_general_ci	 Vérifier les privilèges
<input type="checkbox"/>	performance_schema	utf8_general_ci	 Vérifier les privilèges
<input type="checkbox"/>	phpmyadmin	utf8_bin	 Vérifier les privilèges
<input type="checkbox"/>	test	utf8mb4_general_ci	 Vérifier les privilèges

Total : 6

Une fois ceci fait, allez sur le fichier « dumpSQL.sql » et copiez collez le contenu de ce dernier dans l'onglet script:

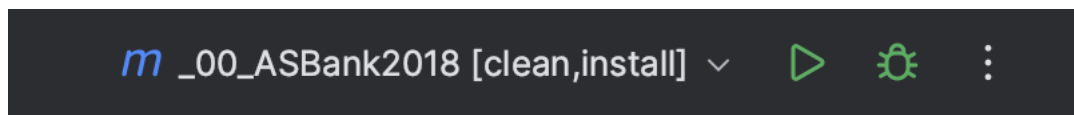


Une fois ce dernier exécuté, vous obtiendrez ceci. :

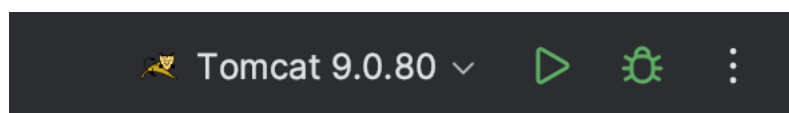


VII. Lancement de l'application

Maintenant que l'intégralité du projet est configuré, vous pouvez lancer le projet. Pour ce faire builder le projet dans votre IDE grâce à Maven en cliquant sur ce bouton :



Une fois le build terminé, vous pouvez maintenant lancer l'application grâce au serveur Tomcat :



Quand le serveur sera lancé, il ouvrira une page sur votre navigateur qui sera l'application comme ceci :



VIII. Lancer le build et la compilation sans l'option

DSKIPTTEST:

Le message d'erreur que nous avons reçu lors de l'exécution de la commande `mvn clean install` indique un échec dans la phase de nettoyage du projet Maven.

Plus précisément, le problème se situe lors de la suppression d'un fichier spécifique :

```
[ERROR] Failed to execute goal org.apache.maven.plugins:maven-clean-plugin:3.2.0:clean (default-clean) on project _00_ASBank2023:
```

```
Failed to clean project: Failed to delete C:\Users\HP\Desktop\Projet_jv\_00_ASBank2023\target\_00_ASBank2023-0.0.1-SNAPSHOT\WEB-INF\lib\struts2-core-2.3.37.jar
```

Ce que nous avons constaté ici, c'est que Maven a échoué à supprimer le fichier `struts2-core-2.3.37.jar` situé dans le répertoire `WEB-INF\lib` du dossier de destination de la construction du projet (`target`).

Les raisons possibles de cet échec peuvent être les suivantes :

Permissions Insuffisantes : nous n'avons peut-être pas les autorisations nécessaires pour supprimer le fichier en question.

Problème de Plugin ou de Configuration : Le problème pourrait être lié à la version du plugin Maven Clean ou à sa configuration.

IX. Migration des Versions:

Nous avons entrepris la migration des versions dans le cadre du projet _00_ASBank2018. Cette mise à jour vise à adopter des versions plus récentes de Java et des dépendances pour garantir une compatibilité optimale et profiter des dernières fonctionnalités.

1. Mise à Jour de Java

Nous avons mis à jour la version de Java de 1.8 à 11 dans le fichier `pom.xml`. Cette migration permet de bénéficier des améliorations de performance, des fonctionnalités et des correctifs de sécurité offerts par les versions plus récentes de Java.

```
``xml
<!-- Avant la mise à jour -->
<source>1.8</source>
<target>1.8</target>
<----->
<!-- Après la mise à jour -->
<source>11</source>
<target>11</target>
```

2. Mise à Jour des Dépendances

Nous avons également mis à jour les versions des dépendances pour suivre les meilleures pratiques et garantir la stabilité du projet. Voici un aperçu des principales mises à jour effectuées :

Plugins Maven :

Dans le premier pom.xml, le plugin Maven Compiler a l'ID de groupe maven-compiler-plugin, tandis que dans le deuxième, il a l'ID de groupe org.apache.maven.plugins. De plus, les versions des plugins sont différentes.

Le plugin Maven War a également une différence de version entre les deux versions.

Version de Java :

Dans le premier pom.xml, la version de Java est définie à 1.8 pour la compilation, tandis que dans le deuxième, elle est mise à jour à 11.

Dépendances :

La version d'Hibernate Core a été mise à jour de 5.1.17.Final à 5.3.20.Final.

La version du connecteur MySQL a été mise à jour de 5.1.44 à 8.0.33.

La version de Struts2 Core a été mise à jour de 2.1.6 à 2.3.37.

La version de JUnit a été mise à jour de 4.12 à 4.13.1.

Nouvelles Dépendances :

Dans le deuxième pom.xml, des dépendances supplémentaires ont été ajoutées, notamment spring-security-crypto, junit-jupiter, et junit-jupiter-api.

Scope des Dépendances :

Dans le deuxième pom.xml, certaines dépendances, comme javax.servlet-api, ont un scope de provided, ce qui signifie qu'elles sont attendues d'être fournies par l'environnement d'exécution.

Nouvelle Configuration de Sécurité :

Dans le deuxième pom.xml, une dépendance à org.springframework.security:spring-security-crypto:6.2.0 a été ajoutée, indiquant probablement l'introduction d'une fonctionnalité de sécurité.

Nouvelles Dépendances pour JUnit Jupiter :

Dans le deuxième pom.xml, des dépendances pour JUnit Jupiter ont été ajoutées pour les tests, indiquant une éventuelle mise à jour des tests unitaires vers JUnit 5.

X. Hachage du Mot de passe:

Cette partie explique la manière dont le hachage des mots de passe est implémenté dans notre application. Le hachage des mots de passe est une pratique de sécurité essentielle pour protéger les informations sensibles des utilisateurs, en garantissant que même en cas d'accès non autorisé, les mots de passe réels ne sont pas exposés.

Implementation :

Dans notre application, nous avons suivi les meilleures pratiques en matière de sécurité pour hacher les mots de passe en utilisant des algorithmes de hachage forts. Voici comment nous avons réalisé cela :

Choix de l'Algorithme de Hachage

Nous avons sélectionné un algorithme de hachage robuste et sécurisé, tel que BCrypt, SHA-256 ou SHA-3. Ces algorithmes sont réputés pour leur résistance aux attaques par force brute et aux attaques par dictionnaire.

Utilisation d'une Fonction de Hachage Salée

Le hachage salé est une technique qui consiste à ajouter une valeur aléatoire unique à chaque mot de passe avant de le hacher. Cela renforce la sécurité en évitant que les attaquants utilisent des tables arc-en-ciel (rainbow tables) pour décoder les mots de passe hachés.

Mise en Œuvre dans le Code

Dans le code source de notre application, le hachage des mots de passe est généralement effectué au moment de l'inscription d'un utilisateur ou lorsqu'un utilisateur met à jour son mot de passe. Voici le code utilisé pour hacher le mot de passe en BCrypt :

Comparaison lors de l'Authentification


```

    */
    // creer par Mamadou le 20.11.2023
    public void setUserPwd(String userPwd) {
        // Utilisation de BCrypt pour hacher le mot de passe
        BCryptPasswordEncoder passwordEncoder = new BCryptPasswordEncoder();
        this.userPwd = passwordEncoder.encode(userPwd);
    }

    // Méthode pour vérifier un mot de passe haché
    2 usages
    public boolean checkPassword(String plainTextPassword) {
        BCryptPasswordEncoder passwordEncoder = new BCryptPasswordEncoder();
        return passwordEncoder.matches(plainTextPassword, this.userPwd);
    }
}
////////////////////////////////////

```

Lorsqu'un utilisateur tente de se connecter, le système compare le hachage du mot de passe fourni avec le hachage stocké dans la base de données. Cela est généralement réalisé à l'aide de bibliothèques de gestion des mots de passe, telles que Spring Security, qui gèrent automatiquement la vérification du mot de passe.

Résultat du Hachage du mot de passe:

Voici le formulaire après l'ajout de notre utilisateur:

Code utilisateur:		<input type="text"/>
Nom:		<input type="text"/>
Prenom:		<input type="text"/>
Adresse:		<input type="text"/>
Password:		<input type="password"/>
Sexe:		<input checked="" type="radio"/> Homme <input type="radio"/> Femme
Type:		<input checked="" type="radio"/> Client <input type="radio"/> Manager
Numéro de client:		<input type="text"/>
<input type="button" value="Submit"/>		
<input type="button" value="Retour"/>		
Le nouvel utilisateur avec le user id 'AB' a bien été crée.		
Evolution du Projet CO Avancée 2017-2018		

Et voila le résultat dans la base de données:

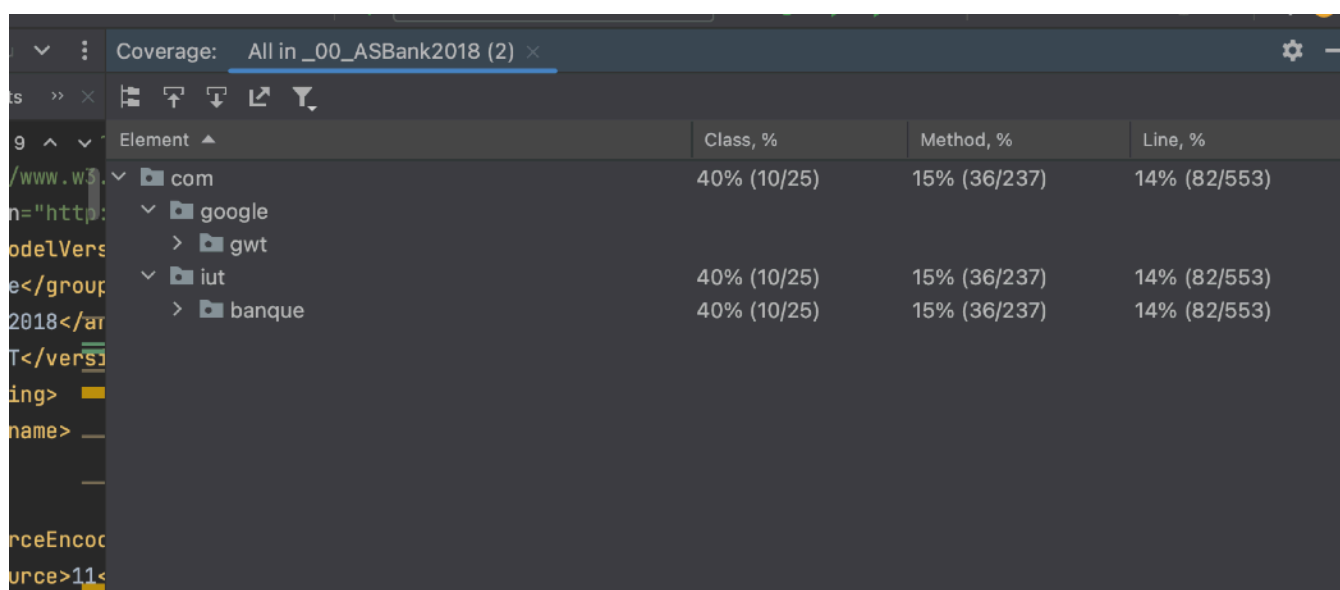
WHERE ORDER BY userPad

userId	nom	prenom	adresse	userPad	male	type
1	mama	mama	1 rue de mama 4556 metz	\$2a\$10\$1Nd3kUcxCJXdxXrrLP908.thK5rtArt/ZMsHI/RTDsPx4ow9.DePa	• true	MANAGER
2	a	a	a	a	• true	MANAGER
admin	Smith	Joe	123, grande rue, Metz	adminpass	• true	MANAGER
client1	client1	Jane	45, grand boulevard, Brest	clientpass1	• true	CLIENT
client2	client2	Jane	45, grand boulevard, Brest	clientpass2	• true	CLIENT

XI. Indicateurs:




















Cette partie vous illustre les différences entre les indicateurs de test et ef Cette section présente une évaluation détaillée de la qualité du code source basée sur l'analyse statique effectuée à l'aide de l'outil SonarLint. Les indicateurs fournis par SonarLint permettent d'identifier des aspects spécifiques du code qui pourraient nécessiter une attention particulière

Indicateurs de Départ:

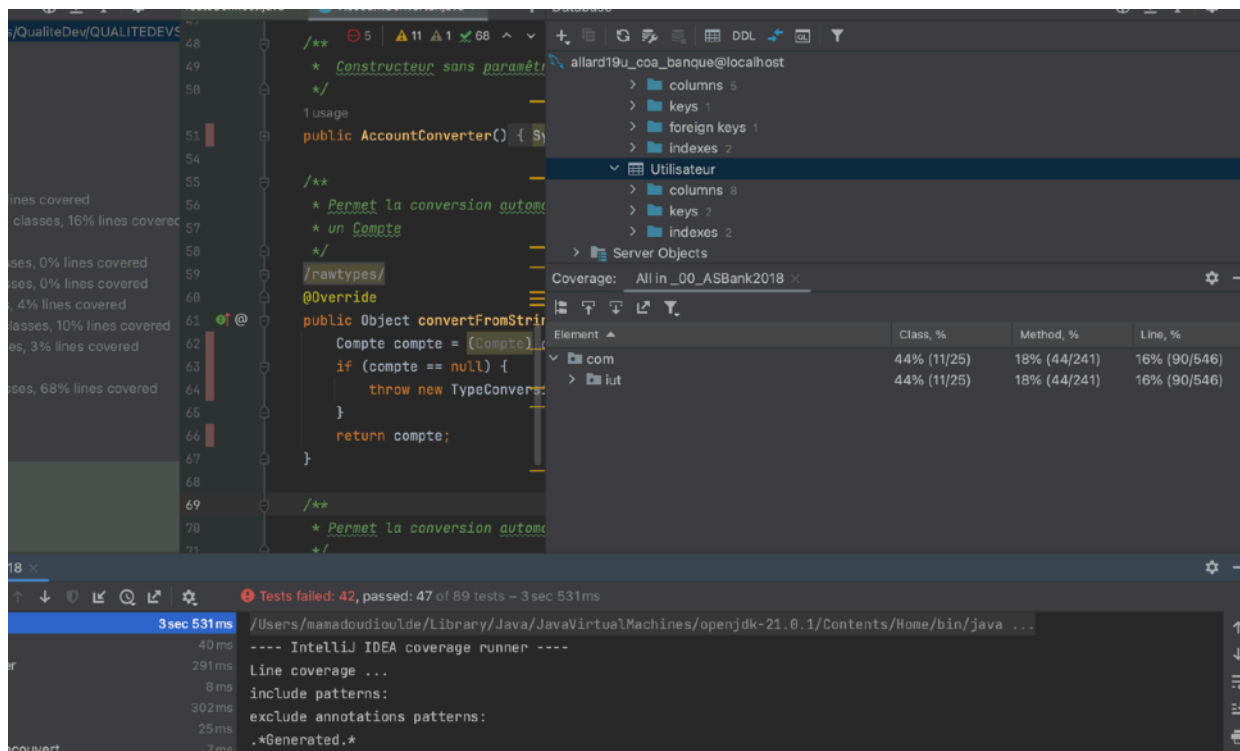


The screenshot shows the SonarLint Coverage window for the project 'All in _00_ASBank2018 (2)'. The table displays coverage metrics for different elements in the project structure.

Element	Class, %	Method, %	Line, %
com	40% (10/25)	15% (36/237)	14% (82/553)
google			
> gwt			
> iut	40% (10/25)	15% (36/237)	14% (82/553)
> banque	40% (10/25)	15% (36/237)	14% (82/553)

- >  Compte.java (3 issues)
- >  JSP Connect-Manager.jsp (1 issue)
- >  JSP Connect-error.jsp (1 issue)
- >  Connect.java (9 issues)
- >  JSP Connect.jsp (3 issues)
- >  CreerCompte.java (6 issues)
- >  JSP CreerCompte.jsp (2 issues)
- >  CreerUtilisateur.java (3 issues)
- >  JSP CreerUtilisateur.jsp (2 issues)
- >  DaoHibernate.java (9 issues)
- >  DetailCompte.java (7 issues)
- >  JSP DetailCompte.jsp (1 issue)
- >  DetailCompteEdit.java (2 issues)
- >  JSP DetailCompteEdit.jsp (1 issue)
- >  IDao.java (1 issue)
- >  JSP Index.jsp (1 issue)
- >  ListeCompteManager.java (4 issues)
- >  JSP ListeCompteManager.jsp (4 issues)
- >  JSP Login.jsp (2 issues)

Amélioration:



The screenshot displays the IntelliJ IDEA IDE with the following components:

- Code Editor:** Shows the `AccountConverter.java` file. The code includes a constructor `AccountConverter()` and a method `convertFromStr`. A coverage bar on the left indicates the percentage of lines covered for each line of code.
- Server Objects:** A tree view on the right showing the project structure, including `Utilisateur` and `Server Objects`.
- Coverage Table:** A table at the bottom right showing coverage statistics for the current run.

Element	Class, %	Method, %	Line, %
com	44% (11/25)	18% (44/241)	16% (90/546)
iut	44% (11/25)	18% (44/241)	16% (90/546)
- Status Bar:** At the bottom, it shows "Tests failed: 42, passed: 47 of 89 tests - 3 sec 531ms".