

Ce document présente la réalisation de
l'application V3.1.0

Réalisation

Vinci Thermo Green

GUILET Julien

Page de service

Référence : Vinci Thermo Green

Plan de classement : stadium-technic-maquette-thermo-green

Niveau de confidentialité : confidential

Mises à jour

Version	Date	Auteur	Description du changement
1.0.0	21/11/2020	GUILET Julien	Création Networking Inc.

Table des matières

Page de service.....	1
Objet du document	2
Architecture.....	2
Implémentation des classes métiers.....	3
1. Base de données.....	3
a) Demande de connexion.....	3
b) Récupérer le nom de l'utilisateur	4
c) Récupérer le rôle de l'utilisateur	4
d) Créer un nouveau compte dans la base de données	5
2. La vue appGUI	6
a) Constructeur de la vue	6
b) Méthode de la vue	7
3. La vue Login	8
a) Constructeur.....	8
b) Les méthodes	9
4. La vue CreateUser	10
a) Constructeur.....	10
b) Ajouter un compte	12
5. Le contrôleur	13
a) Afficher le login.....	13
b) Demande de connexion.....	13
c) Connexion réussie	13
d) Demande de déconnexion	14
e) Affiche la vue createUser	14
f) Créer un comte.....	14

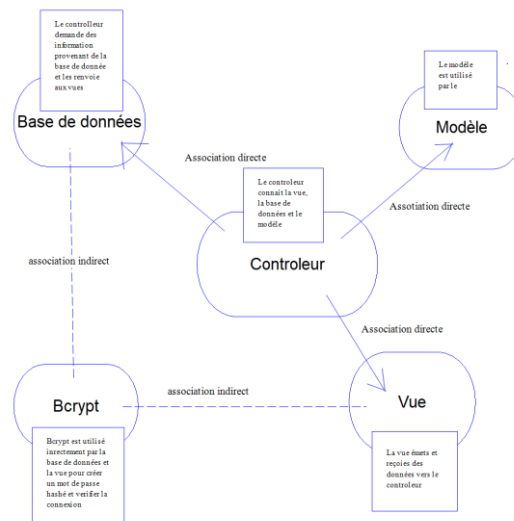
Objet du document

Ce document décrit l'implémentation Java du projet Vinci Thermo Green.

Ce projet vise à modifier la version 3.0.0 de l'application Vinci Thermo Green afin d'y introduire un système de connexion et d'ajout de compte.

Architecture

Voici l'architecture globale de la version 3.1.0 de l'application Vinci Thermo Green.



La conception de la v.3.1.0 nécessite une nouvelle architecture avec pour centre un contrôleur qui fera la jonction avec la base de données, les vues et modèles. Le contrôleur recevra des informations et demandes de la part de l'utilisateur à travers les différentes vues.

Dans cette version de l'application les demandes venant des vues sont principalement des demandes de requêtes de sélection et d'insertion dans la base de données, ces requêtes seront formulées et envoyées par le contrôleur vers la base de données.

La base de données répondra au contrôleur en effectuant ces différentes demandes. Pour certaines de ces demandes, la base de données aura besoin de l'utilisation de Bcrypt d'où cette association indirecte de Bcrypt vers la base de données.

Le contrôleur a aussi besoin d'une association directe vers Modèle afin de pouvoir traiter les demandes venant des différentes vues.

Implémentation des classes métiers

1. Base de données

Comparer à la dernière version de l'application de nouvelles méthodes son nécessaire pour mettre en place le système d'authentification.

a) Demande de connexion

La première méthode est celle de la demande de connexion. Celle-ci aura pour but de vérifier l'existence du compte mis en paramètre et de vérifier si le mot de passe inscrit coïncide avec celui enregistré en base de données.

Pour ce faire une première requête de sélection est faite avec les informations mises en paramètres. Une fois cette requête effectuer nous essayons de récupérer le mot de passe en base de données associer au login.

Si nous la récupération du mot de passe ne fonctionne pas cela signifie qu'il n'existe pas de compte possédant les informations mises en paramètres, nous renvoyons donc un booléen false.

Si dans le cas contraire un mot de passe (hashé) a été trouvé alors nous utilisons la méthode de Bcrypt « checkpw() » pour vérifier si le mot de passe de la BDD correspond bien avec celui mis en paramètre, si c'est le cas alors nous renvoyons un booléen true.

Voici la requête :

```
public boolean demandeConnexion(String login, String mdp) throws SQLException {
    ResultSet resultset = null;
    resultset = statement.executeQuery(
        "select login, password from user where login = '"
+ login + "';");
    resultset.next();
    try {
        String mdp2 = resultset.getString("password");
        if (BCrypt.checkpw(mdp, mdp2) == true) {
            return true;
        } else {
            return false;
        }
    } catch (java.sql.SQLException e) {
        return false;
    }
}
```

b) Récupérer le nom de l'utilisateur

La seconde méthode va permettre de récupérer le nom et prénom de l'utilisateur connecter à l'aide de son login. Cette méthode sera utilisée lorsque la connexion aura réussi et que ces informations devront être affichées en haut de l'application de consultation des mesures des stades.

Pour cela, nous effectuons une requête de sélection vers la base de données à l'aide du login mis en paramètre et récupérerons le nom et prénom de l'utilisateur. Une fois fait, nous retournons l'association de son nom et prénom séparés par un espace. À noter qu'ici il n'est pas nécessaire de vérifier si login est existant dans la base de données étant donné que l'utilisateur est déjà connecté et que la vérification s'est faite au moment de sa connexion.

Voici la requête :

```
public String getNomUtilisateur(String login) throws SQLException {  
    ResultSet resultset = null;  
    resultset = statement.executeQuery("select nom, prenom from user  
where login = '" + login + "';");  
    resultset.next();  
    return resultset.getString("nom") + ' ' +  
resultset.getString("prenom");  
}
```

c) Récupérer le rôle de l'utilisateur

La prochaine méthode va permettre de récupérer le rôle du compte de l'utilisateur connecter à l'aide du login. Cette méthode sera utilisée lorsque la connexion aura réussi et que ces informations devront être affichées en haut de l'application de consultation des mesures des stades.

Cette requête ne diffère en rien à la requête précédente hormis la requête de sélection faite vers la BDD.

Voici la requête :

```
public String getRoleUtilisateur(String login) throws SQLException {  
    ResultSet resultset = null;  
    resultset = statement.executeQuery("select type from user where  
login = '" + login + "';");  
    resultset.next();  
    return resultset.getString("type");  
}
```

d) Créer un nouveau compte dans la base de données

Cette méthode permet d'ajouter un nouveau compte dans la BDD. Cette méthode interviendra lorsqu'un utilisateur administrateur aura rempli le formulaire et cliquer sur le bouton valider de l'application.

Pour faire cette méthode, nous devons avoir dans les paramètres toutes les informations concernant le nouveau compte à ajouter. Une fois récupérer, nous effectuons une requête d'insertion vers la BDD avec tous les paramètres. Nous retournons true si aucune erreur n'est apparue lors de l'insertion et retournons false lorsque que cela est le cas.

Voici la méthode :

```
public boolean ajoutUser(String identifiant, String nom, String prenom, String mdp,
String role)
    throws SQLException {
    try {
        boolean t = statement.execute("insert into user
values('"+identifiant+"','"+mdp+"','"+nom+"','"+prenom+"','"+role+"');");
        return true;
    } catch (SQLException e) {
        System.out.println(e);
        return false;
    }
}
```

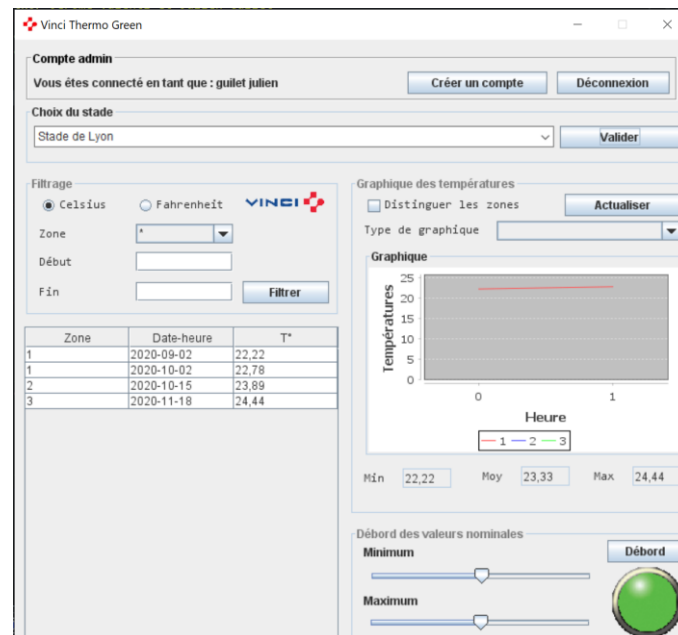
2. La vue appGUI

a) Constructeur de la vue

AppGUI fait référence à la vue originale controlerGUI. Nous avons renommé cette vue en appGUI car étant donné que dans notre architecture nous avons un nouveau contrôleur au centre de tout, nous préférons séparer la vue du contrôleur pour plus de lisibilité.

Cette vue appGUI correspond à la maquette de l'application GUI (voir ThermoGreen_maquette_V3.1.0). Cette vue est une modification de la précédente c'est pour cela que nous détaillerons uniquement les changements apportés de la version 3.0.0 à la version 3.1.0.

Voici la nouvelle vue :



Comme vous pouvez le voir comparé à la version 3.0.0 seule, la rubrique « Compte » a été ajoutée.

Voici les ajouts supplémentaires :

```
JPanel panelCompte = new JPanel();
    panelCompte.setBorder(new TitledBorder(new
EtchedBorder(EtchedBorder.LOWERED, new Color(255, 255, 255), new Color(160,
160, 160)), "Compte " + role, TitledBorder.LEADING, TitledBorder.TOP, null,
new Color(0, 0, 0)));
    panelCompte.setBounds(10, 11, 675, 54);
    getContentPane().add(panelCompte);
    panelCompte.setLayout(null);

    JLabel labelCompte = new JLabel("Vous \u00Eates
connect\u00E9 en tant que : " + this.nom);
    labelCompte.setBounds(10, 22, 375, 22);
    panelCompte.add(labelCompte);

    JButton btnNewButton = new JButton("D\u00E9connexion");
    btnNewButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
```

```
deconnection();  
    }  
});  
btnNewButton.setBounds(549, 22, 116, 23);  
panelCompte.add(btnNewButton);  
  
String a ="admin";  
if (a.equals(role)) {  
    JButton btnNewButton_1 = new JButton("Cr\u00E9er un  
compte");  
    btnNewButton_1.addActionListener(new  
ActionListener() {  
        public void actionPerformed(ActionEvent arg0)  
        {  
            control.afficheCreateUser();  
        }  
    });  
    btnNewButton_1.setBounds(395, 22, 144, 23);  
    panelCompte.add(btnNewButton_1);  
}
```

Pour apporter ces modifications, nous avons ajouté un JPanel sur lequel, nous avons inscrit un titre (ce titre change en fonction du rôle de l'utilisateur). Ensuite, il y a un label contenant le texte « Vous êtes connecté en tant que : » suivi du nom et prénom de l'utilisateur connecté.

À droite de ce texte, deux boutons sont disposés, le premier est « Créer un compte » celui apparaît uniquement si le rôle de l'utilisateur est un admin dans le cas contraire rien n'est affiché. Lorsque ce bouton est appuyé par l'administrateur il fait appelle à la méthode « afficheCreateUser() » du contrôleur qui envoie l'administrateur sur le formulaire d'ajout de compte. Le second bouton est un bouton de déconnexion celui-ci s'affiche obligatoirement et permet lorsqu'il est enclenché il d'utiliser la méthode « deconnection() » de cette vue.

b) Méthode de la vue

Cette possède une seule nouvelle méthode appeler « donnection() ». Cette méthode permet uniquement de faire appelle à une autre méthode venant du contrôleur appelé « demandeDeconnexion() ».

Voici la méthode deconnection() :

```
public void deconnection() {  
    control.demandeDeconnexion();  
}
```


3. La vue Login

La vue login fait référence à la maquette « Connexion » (voir ThermoGreen_maquette_V3.1.0).

Voici la vue Login :



a) Constructeur

Cette vue est composée d'un label principal qui contient le texte « Connexion ». Deux autres labels qui sont accompagné de deux champs, le premier correspond à l'identifiant que devra rentrer l'utilisateur s'il souhaite se connecter.

Le second correspond au mot de passe indispensable pour se connecter. Il y a également 2 boutons, le premier permet de mettre la visibilité de la fenêtre à false ce qui ferme l'application alors que le second permet d'utiliser une méthode de cette vue appelle « demandeLogin() ».

Voici le constructeur :

```
getContentPane().setMinimumSize(new Dimension(300, 300));
setMinimumSize(new Dimension(500, 500));
setLocation(new Point(1000, 300));
setTitle("Page de connexion");
getContentPane().setLayout(null);

setIconImage(Toolkit.getDefaultToolkit().getImage("img\\vinci_ico.jpg"));
JButton ButtonValider = new JButton("Valider");
ButtonValider.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent arg0) {
        try {
            demandeLogin();
        } catch (SQLException | ParseException e) {
            e.printStackTrace();
        }
    }
});
ButtonValider.setBounds(296, 251, 89, 23);
getContentPane().add(ButtonValider);

JButton btnNewButton = new JButton("Fermer");
btnNewButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        setVisible(false);
    }
});
btnNewButton.setBounds(184, 251, 89, 23);
```

```

getContentPane().add(btnNewButton);

txtLogin = new JTextField();
txtLogin.setBounds(184, 141, 201, 17);
getContentPane().add(txtLogin);
txtLogin.setColumns(10);

JLabel lblNewLabel = new JLabel("Connexion");
lblNewLabel.setFont(new Font("Tahoma", Font.PLAIN, 25));
lblNewLabel.setBounds(184, 53, 178, 54);
getContentPane().add(lblNewLabel);

JLabel lblNewLabel_1 = new JLabel("Identifiant");
lblNewLabel_1.setBounds(80, 141, 107, 14);
getContentPane().add(lblNewLabel_1);

JLabel lblNewLabel_2 = new JLabel("Mot de passe");
lblNewLabel_2.setBounds(80, 195, 107, 14);
getContentPane().add(lblNewLabel_2);

txtMotDePasse = new JPasswordField();
txtMotDePasse.setBounds(184, 192, 201, 20);
getContentPane().add(txtMotDePasse);

```

b) Les méthodes

Cette possède une seule méthode : demandeLogin().

Celle-ci permet de récupérer l'identifiant et le mot de passe mis dans les champs de la vue. Cela permet par la suite d'utiliser la méthode « demandeConn() » du contrôleur. Celle-ci permet de vérifier les informations de l'utilisateur. Si cette fonction retourne true alors l'utilisateur est envoyé sur 'application appGUI dans le cas contraire un message s'affiche lui indiquant que la connexion a échoué.

La voici :

```

public void demandeLogin() throws SQLException, ParseException {
    String login;
    String mdp;
    login = this.txtLogin.getText();
    mdp = this.txtMotDePasse.getText();

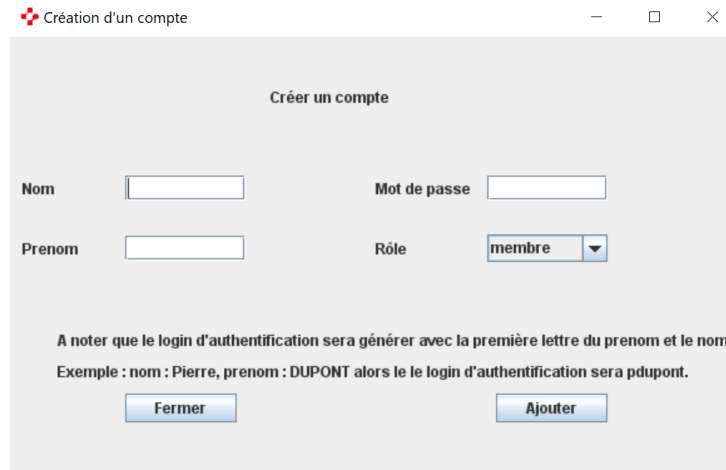
    if (control.demandeConn(login, mdp) == true) {
        JOptionPane.showMessageDialog(null, "Connexion réussi");
        control.connectionReussi(login);
    } else {
        JOptionPane.showMessageDialog(null, "Connexion échoué");
    }
}

```

4. La vue CreateUser

La vue CreateUser fait référence à la maquette « Créer un compte » (voir ThermoGreen_maquette_V3.1.0).

Voici la vue CreateUser :



a) Constructeur

Cette vue est composée de quatre champs à compléter afin de saisir des informations relatives au nouveau compte. Les trois premiers permettant de saisir le nom, prénom et mot de passe du compte. Le dernier champ est un choix parmi deux proposition membre ou admin. Pour terminer, cette vue possède deux boutons, le premier est fermé et lorsque l'administrateur clique dessus la méthode du contrôleur fermeCreationUser() est appelée.

Le second bouton lui permet utiliser la méthode de la vue ajoutUser().

Voici le code :

```
public CreateUser(Controller conn) {
    setTitle("Cr\u00E9ation d'un compte");

    this.control = conn;
    getContentPane().setMinimumSize(new Dimension(300, 300));
    setMinimumSize(new Dimension(600, 400));
    setLocation(new Point(1000, 300));
    getContentPane().setLayout(null);
    setIconImage(Toolkit.getDefaultToolkit().getImage("img\\vinci_ico.jpg"));
    JLabel lblNewLabel = new JLabel("Cr\u00E9er un compte");
    lblNewLabel.setBounds(208, 41, 115, 14);
    getContentPane().add(lblNewLabel);

    textNom = new JTextField();
    textNom.setBounds(92, 111, 96, 20);
    getContentPane().add(textNom);
    textNom.setColumns(10);

    textPrenom = new JTextField();
    textPrenom.setBounds(92, 159, 96, 20);
    getContentPane().add(textPrenom);
    textPrenom.setColumns(10);

    JComboBox comboBoxRole = new JComboBox();
    this.comboBoxRole = comboBoxRole;
    this.comboBoxRole.setBounds(381, 158, 96, 22);
    getContentPane().add(this.comboBoxRole);

    JLabel lblNewLabel_1 = new JLabel("Nom");
    lblNewLabel_1.setBounds(10, 114, 72, 14);
    getContentPane().add(lblNewLabel_1);
}
```

```

JLabel lblNewLabel_2 = new JLabel("Prenom");
lblNewLabel_2.setBounds(10, 162, 72, 14);
getContentPane().add(lblNewLabel_2);

JLabel lblNewLabel_3 = new JLabel("Mot de passe");
lblNewLabel_3.setBounds(292, 114, 79, 14);
getContentPane().add(lblNewLabel_3);

passwordField = new JPasswordField();
passwordField.setBounds(381, 111, 96, 20);
getContentPane().add(passwordField);

JLabel lblNewLabel_4 = new JLabel("R\u00F4le");
lblNewLabel_4.setBounds(292, 162, 47, 14);
getContentPane().add(lblNewLabel_4);

JButton buttonAjouter = new JButton("Ajouter");
buttonAjouter.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        try {
            ajoutUser();
        } catch (SQLException e1) {
            e1.printStackTrace();
            JOptionPane.showMessageDialog(null, "Probl\u00e8me
avec la Base de donn\u00e9e");
        }
    }
});
buttonAjouter.setBounds(388, 285, 89, 23);
getContentPane().add(buttonAjouter);

JButton buttonFermer = new JButton("Fermer");
buttonFermer.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent arg0) {
        control.fermeCreationUser();
    }
});
buttonFermer.setBounds(92, 285, 89, 23);
getContentPane().add(buttonFermer);

JLabel lblNewLabel_5 = new JLabel("A noter que le login d'authentification
sera g\u00E9n\u00E9r\u00E9 avec la premi\u00e8re lettre du prenom et le nom.");
lblNewLabel_5.setBounds(38, 235, 629, 14);
getContentPane().add(lblNewLabel_5);

JLabel lblNewLabel_6 = new JLabel("Exemple : nom : Pierre, prenom : DUPONT
alors le le login d'authentification sera pdupont.");
lblNewLabel_6.setBounds(38, 260, 629, 14);
getContentPane().add(lblNewLabel_6);

this.comboBoxRole.addItem("membre");
this.comboBoxRole.addItem("admin");

```

b) Ajouter un compte

Cette vue ne possède qu'une méthode « ajoutuser() ». Cette méthode récupère les informations présentes dans vue et vérifie que la méthode « createuser() » du contrôleur renvoie true. Si elle renvoie, true, cela signifie que l'ajout de l'utilisateur s'est bien passé et donc nous affichons un message indiquant que l'ajout a réussi. Dans le cas contraire, si la fonction renvoie false alors nous renvoyons un message indiquant que l'ajout a échoué.

Voici la méthode :

```
public void ajoutUser() throws SQLException {
    try {
        String l= this.textPrenom.getText().substring(0,1);
        String identifiant = l + this.textNom.getText();
        String mdp = this.passwordField.getText();
        String nom = this.textNom.getText();
        String prenom = this.textPrenom.getText();
        int role = this.comboBoxRole.getSelectedIndex();
        String Role;
        if (role == 1) {
            Role ="admin";
        } else {
            Role ="membre";
        }
        String mdphash = BCrypt.hashpw(mdp, BCrypt.gensalt(10));
        //System.out.println("identifiant: "+identifiant+" mdp:
        "+mdphash+" nom: "+nom+" prenom: "+prenom+" role: "+Role);
        if (this.control.createUser(identifiant, nom, prenom,
        mdphash, Role) == true) {
            JOptionPane.showMessageDialog(null, "Ajout réussi
            !");
        } else {
            JOptionPane.showMessageDialog(null, "Ajout raté
            !");
        }
    } catch (Exception ex) {
        JOptionPane.showMessageDialog(null, "Ajout raté !");
    }
}
```

5. Le contrôleur

Dans cette version, il a fallu ajouter des méthodes afin de pouvoir mettre en place la nouvelle version de l'application.

a) Afficher le login

Cette méthode permet de créer une vue Login et de l'afficher à l'écran.

Voici le code :

```
public void afficheLogin(Controller control) {
    Login log = new Login(control);
    this.login = log;
    this.login.setVisible(true);
}
```

b) Demande de connexion

Cette méthode fait le lien entre la vue et la BDD. En effet, cette méthode est appelée dans la vue Login et renvoie le résultat de la fonction « demandeConnexion() » de la base de données.

Voici le code :

```
public boolean demandeConn(String login, String mdp) throws SQLException{
    return BDD.demandeConnexion(login, mdp);
}
```

c) Connexion réussie

Cette fonction intervient lorsque la connexion est réussie. Elle permet d'enlever la vue Login, puis récupère les différentes informations de l'utilisateur grâce à son login en paramètre et aux fonctions « getNomUtilisateur() » et « getRoleUtilisateur » de la base de données. Une fois, récupérer elle affiche la vue appGUI avec ces informations.

Voici le code :

```
public void connexionReussi(String login) throws ParseException, SQLException {
    this.login.setVisible(false);
    String nom = BDD.getNomUtilisateur(login);
    String role = BDD.getRoleUtilisateur(login);
    AppGUI app = new AppGUI(this, nom, role);
    this.appGUI = app;
    this.appGUI.setVisible(true);
}
```

d) Demande de déconnexion

Cette fonction intervient lorsque le bouton « déconnexion » de la vue appGUI est activé. Cette fonction détruit l'appGUI en cours d'utilisation et renvoie l'utilisateur sur l'écran de connexion.

Voici le code :

```
public void demandeDeconnexion() {
    appGUI.setVisible(false);
    this.appGUI = null;
    this.login = null;
    Login log = new Login(this);
    this.login = log;
    this.login.setVisible(true);
}
```

e) Affiche la vue createUser

Cette fonction intervient lorsqu'un administrateur clique sur le bouton « Créer un compte » de la vue appGUI. Elle permet de fermer la vue appGUI et de créer puis d'ouvrir une vue CreateUser permettant d'ajouter un utilisateur.

Voici le code :

```
public void afficheCreateUser() {
    this.appGUI.setVisible(false);
    CreateUser u = new CreateUser(this);
    this.createUser = u;
    this.createUser.setVisible(true);
}
```

f) Créer un compte

Cette fonction utilise la méthode « ajoutUser() » de la base de données afin de créer un utilisateur. Elle renvoie un booléen en fonction du résultat de « ajoutUser() » de la base de données. Elle est appelée lorsqu'un administrateur appuie sur le bouton de validation de création d'un compte de la vue createUser.

Voici le code :

```
public boolean creationUser(String identifiant, String nom, String prenom, String
mdp, String role) throws SQLException {
    boolean b = this.BDD.ajoutUser(identifiant, nom, prenom, mdp,
role);
    return b;
}
```