

Dossier Réalisation Vinci Thermo Green

Version : v3.1.0

Mise à jour

Version	Date	Auteur	Description changement
3.0.0	02/11/2020	GROUSSET Sylvain	Implémentation multi-sites
3.1.0	23/11/2020	GROUSSET	Identification & JBCrypt
3.2.0	14/12/2020	GROUSSET	Alerte SMS via API

Table des matières

1.	Architecture	1
2.	Implémentation des classes métier	2
3.	Implémentation de la base de données.....	4
4.	Base de données	5
5.	Connexion à la base de données.....	6
6.	Cryptage avec JBCrypt	6
7.	Implémentation API Twilio.....	8
8.	JSlider, temp_min, temp_max	10

Objet du document

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

Il présentera l'application réalisé à partir de l'implémentation du diagramme des classes métier et du diagramme de séquence objet présenté dans le dossier d'Analyse-Conception.

Il montrera les points majeurs, importants et complexes de l'application nécessitant documentation.

1. Architecture

L'architecture de l'application a été réalisée avec le pattern de MVC (Modèle Vue Controller). Ce pattern est structuré en 3 couches :

- Modèle
- Vue
- Controller

Chaque couche a un rôle bien spécifique :

- La vue s'occupe de tous les éléments visuels de l'application.
- Le contrôleur s'occupe du traitement des données entre le modèle et la vue
- Et donc le modèle traite les données envoyées par le contrôleur

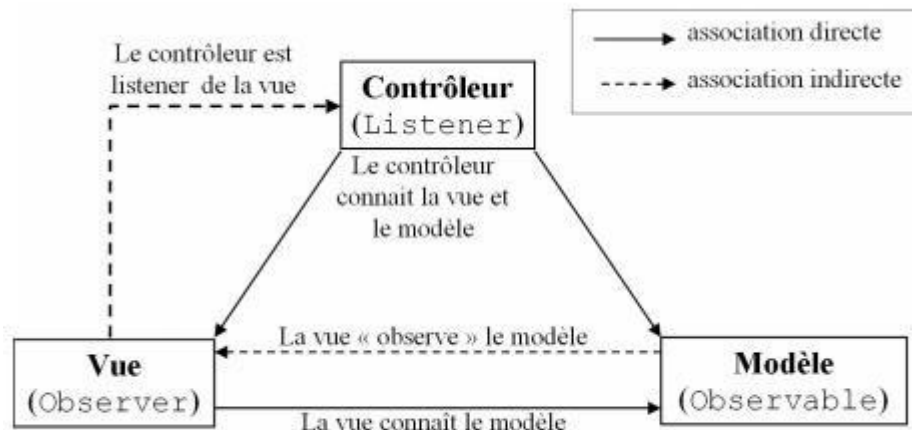


Figure 1 : principe de fonctionnement du modèle MVC

2. Implémentation des classes métier

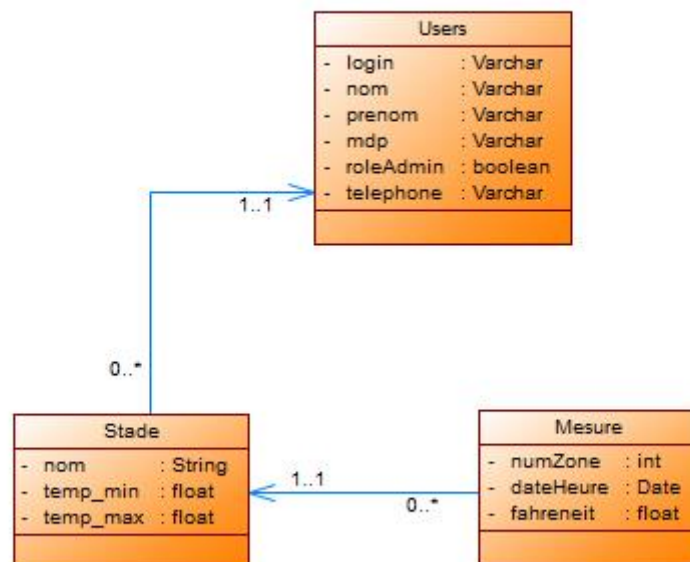


Figure 2 : Diagramme des classes métiers

L'analyse de la version v3.2.0 a permis d'établir le diagramme des classes ci-dessus.

Une classe Users sera créée au niveau de la base de données pour permettre de gérer les utilisateurs.

La classe `Mesure` implémentée en java permettra de stocker les températures en Fahrenheit.

Une méthode **`getCelsius`** permet de convertir une température en Fahrenheit à Celsius par la formule : $(\text{fahrenheit} - 32.0f) / 1.8f$

```
package model; import
java.util.Date;

public class Mesure {
    /**
     * <p>numZone contient le numéro de la zone mesurée</p>
     */
    private String numZone;
    /**
     * <p>horoDate contient la date et l'heure de la mesure au format aa-mm-jj hh:mm</p>
     */
    private Date horoDate;
    /**
     * <p>valFahrenheit contient la valeur de la température mesurée en degré
     Fahrenheit</p>
     */
    private float fahrenheit;

    public Mesure() {
        this.numZone = new String();
        this.horoDate = new Date();
        this.fahrenheit = 0.0f;
    }

    public Mesure String pZone, Date pDate, float pFahrenheit) {

        this.numZone = pZone;
        this.horoDate = pDate;
        this.fahrenheit = pFahrenheit;
    }

    public String getNumZone() {
        return numZone;
    }

    public void setNumZone String numZone) {
        this.numZone = numZone;
    }

    public Date getHoroDate() {
        return horoDate;
    }

    public void setHoroDate Date horoDate) {
        this.horoDate = horoDate;
    }

    public float getFahrenheit() {
        return fahrenheit;
    }

    public void setFahrenheit(float valFahrenheit) {
        this.fahrenheit = valFahrenheit;
    }
}
```

```
/**
 * <p>Convertit Fahrenheit en °Celsius</p>
 * @since 2.0.0
 * @return float t°Celsius
 */
public float getCelsius() {
    //return (float) (valFahrenheit - 32) / 1.8)
    return (fahrenheit - 32.0f) / 1.8f;
}
```

3. Implémentation de la base de données

Avant de pouvoir utiliser l'application, il faut que la base de données soit implémentée. Nous utilisons une base de données sous MySQL.

Prérequis :

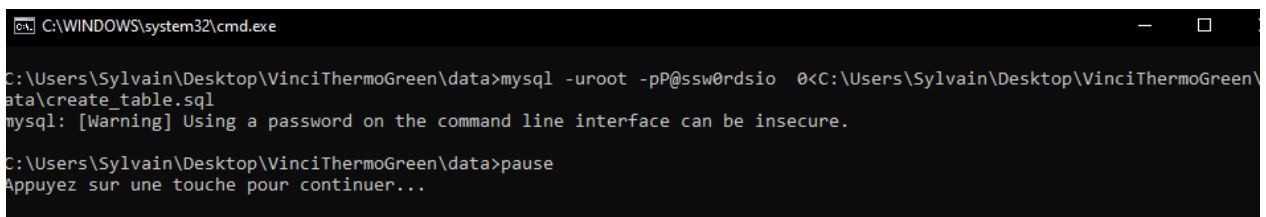
- MySQL Server installé
- &
- MySQL Server ajouté au PATH

Un fichier "**database.bat**" se trouvant dans le dossier "**data**" du build, permet la création automatique de la base de données, si vous avez les prérequis ci dessus. Avant de lancer le script il faut faire attention à certaines choses :

- Bien avoir **MySQL ajouté au PATH** sinon le script ne fonctionnera jamais.
- Ne pas avoir de base de données ayant comme nom "**vinci**"

Dans le cas où vous n'arriveriez pas à ajouter MySQL au path, référez vous à ce document pour implémenter la base de données d'une autre manière.

L'exécution de ce script donne lieu à cette fenêtre :



```
C:\WINDOWS\system32\cmd.exe
C:\Users\Sylvain\Desktop\VinciThermoGreen\data>mysql -uroot -p@ssw0rdsio -C:C:\Users\Sylvain\Desktop\VinciThermoGreen\data\create_table.sql
mysql: [Warning] Using a password on the command line interface can be insecure.
C:\Users\Sylvain\Desktop\VinciThermoGreen\data>pause
Appuyez sur une touche pour continuer...
```

Ce qui signifie que la base de données s'est importée avec succès.

Attention 1, le script ne fonctionne qu'une seule fois. Si vous tentez de le lancer 2 fois pour x raisons, il vous affichera une erreur au niveau du **CREATE USER** avec une erreur **1396 (HY000)**. Pour pallier ce problème il faut mettre un nom d'utilisateur différent de celui créé précédemment.

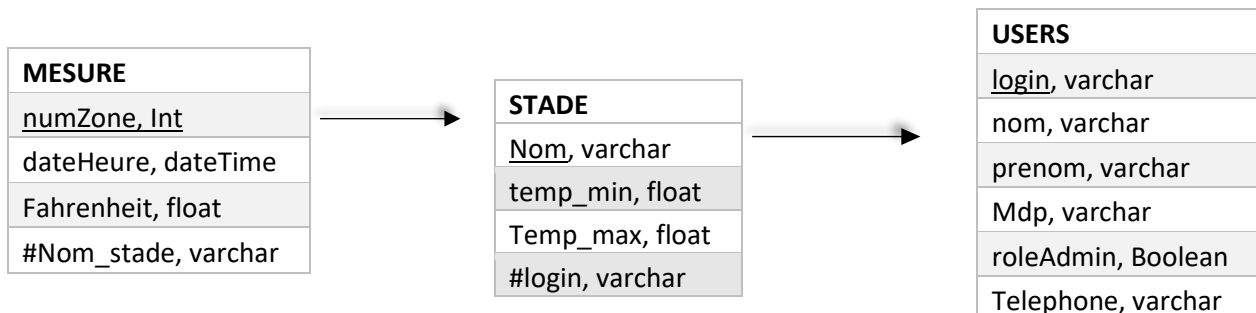
Si vous êtes amenés à changer ce nom d'utilisateur, des modifications dans le code sont nécessaires. Il vous faudra appeler notre service d'aide.

Attention 2, il faut bien faire attention à avoir MySQL ajouté au PATH, sinon le script database.bat ne pourra fonctionner ! Si vous n'arrivez pas à ajouter MySQL au path référez-vous à ce document d'aide pour implementer la base de données sans ce script .bat.

4. Base de données

La base de données est constituée de 3 tables :

- STADE
- MESURE
- USERS



- Les tables **Stade** et **Mesure** sont en relations : Une mesure concerne 1 et 1 seul stade et un stade peut avoir 0 ou plusieurs Mesures. Donc en plus de l'identifiant numZone dans la table Mesure, il y a une référence (clés étrangère) du nom du stade dans Mesure (attribut Nomstade).
- Un stade est identifié par son nom. Il possède deux attributs : temp_min et temp_max pour stocker les températures minimum et maximum d'un stade.
Il y a aussi une clés étrangère du login de l'utilisateur faisant donc référence à au login de la table **USERS**.
- La table **USERS** a pour identifiant le login de l'utilisateur. Cette table est en relation avec la table STADE : Un stade est géré par 1 et 1 seul USERS, un USERS peut gérer 0 ou plusieurs stades.

Un script d'insertion de base est fourni avec le script de création des tables pour que l'application soit un minimum peuplée lors de son implémentation.

Ce script comprend : L'insertions de 3 stades avec des températures pour chacun des stades ainsi que de 3 utilisateurs :

Utilisateur	EGUEISSAZ	SGROUSSET	JGUILLET
Droits admin ?	Oui	Oui	Non

Stades gérés	Groupama Stadium	Parc des Princes Velodrome	Aucun
--------------	------------------	----------------------------	-------

Le mot de passe pour les utilisateurs est "bpsen".

Les températures ont été modifiées pour que l'envoi d'un SMS se fasse lorsque : L'utilisateur EGUEISSAZ clique sur le bouton VALIDER pour voir les températures du stade Groupama Stadium, ce stade possédant une température à 40 degrés, ce qui est au dessus de sa température maximum.

Idem pour le stade Velodrome, géré par l'utilisateur SGROUSSET, celui-ci possédant également une température à 40 degrés, ce qui est bien au dessus de la température maximum.

5. Connexion à la base de données.

La connexion à la base de données se fait dans la classe **données** via la méthode « openDatabase » mais est exécutée depuis le contrôleur.

```
public void openDatabase() {  
  
    String url = "jdbc:mysql://localhost:3306/vinci?serverTimezone=UTC";  
    String username = "adminVinci";  
    String password = "vinciThermogreen";  
  
    try {  
  
        Connection conn = DriverManager.getConnection(url, username, password);  
        System.out.println("Connexion base OK !");  
        setConn(conn);  
  
    } catch (SQLException e) {  
  
        throw new IllegalStateException("Cannot connect database !", e);  
  
    }  
}
```

Cette méthode appelle la méthode « getConnection » de DriverManager avec en paramètre les informations pour se connecter à la base de données : l'URL, l'username et le mot de passe de l'username.

Puis le tout est encapsulé dans un try/catch pour attraper les éventuelles erreurs.

6. Cryptage avec JBCrypt

Les mots de passe des utilisateurs sont cryptés via la librairie JBCrypt.

Les méthodes concernant le cryptage sont appelées à 2 endroits dans le programme :

- Lorsque l'utilisateur Admin crée un compte (le MDP est crypté)
- Lorsque l'utilisateur se connecte

Lorsque l'utilisateur ADMIN crée un compte :

```

public void actionPerformed(ActionEvent arg0) {

    try {
        control = new Controller();
    } catch (ParseException | SQLException e) {
        e.printStackTrace();
    }

    @SuppressWarnings("deprecation")
    String mdp = BCrypt.hashpw(passwordField.getText(),
BCrypt.gensalt(10));

    String nom = textFieldNom.getText();
    String prenom = textFieldPrenom.getText();
    String login = prenom.charAt(0)+nom;
    boolean adminCheckBox = chckbxAdmin.isSelected();

    try {
        control.createAccount(login, nom, prenom, mdp,
adminCheckBox);
        JOptionPane.showMessageDialog(null, "Compte ajouté avec
succès !");

        textFieldNom.setText("");
        textFieldPrenom.setText("");
        passwordField.setText("");

    } catch (SQLException e) {
        e.printStackTrace();
    }
}

```

Ce bout de code correspond à lorsque l'utilisateur clique sur le bouton Valider pour créer un compte. Le bout de code utilisé pour crypter le mot de passe est le suivant :

```
String mdp = BCrypt.hashpw(passwordField.getText(), BCrypt.gensalt(10));
```

La méthode BCrypt.hashpw prend 2 paramètres : Le String à hacher, et du salage.

Ici, on lui fournit en 1^{er} paramètre le passwordField.getText() (le mot de passe entré dans le passwordField) qui est la valeur à hacher, et en second paramètre une méthode de BCrypt (genSalt) qui permet de générer du sel.

Lorsque l'utilisateur se connecte :

```

public boolean connexion(String login, String mdp throws SQLException {

    stmt = conn.createStatement();

    ResultSet rs = stmt.executeQuery("SELECT USERS.MDP FROM USERS WHERE login =
 '"+login+"'");

    while (rs.next()) {

        if(BCrypt.checkpw(mdp, rs.getString("mdp"))) == true){
            return true;
        }

    }

    return false;
}

```

Voici le bout de code permettant de vérifier si le mot de passe et le login entré par l'utilisateur est correct.

Cette méthode est dans la classe données et prend en paramètre le login et le MDP entré dans le formulaire de connexion Login.

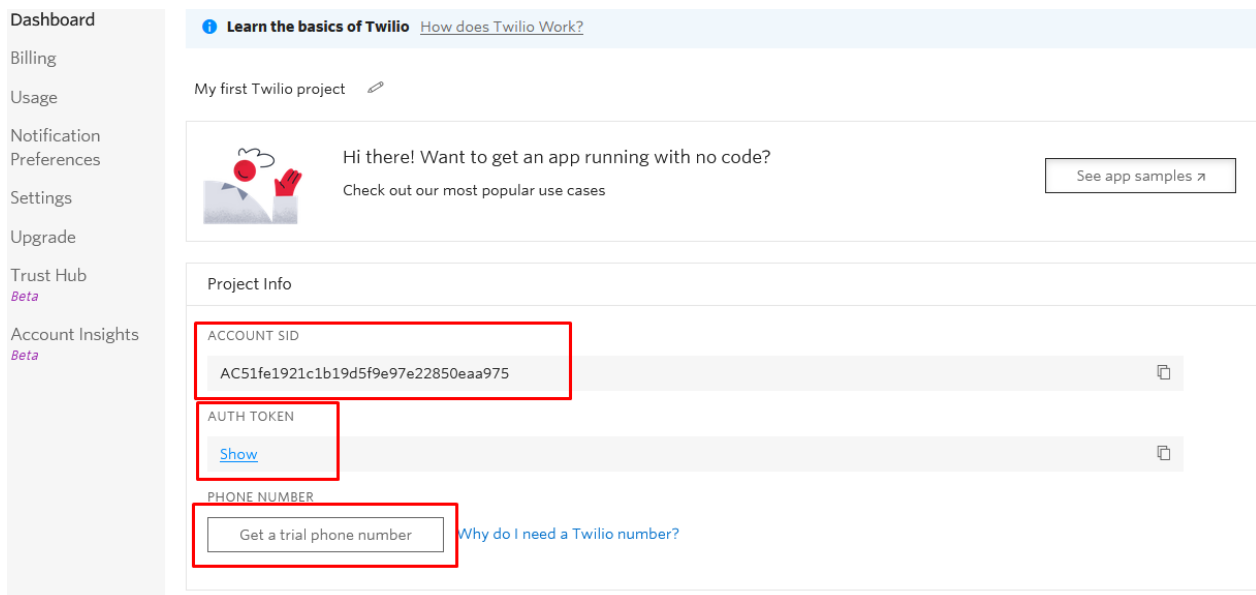
Une requête SQL est exécutée pour récupérer le mot de passe haché correspondant à l'utilisateur donné en paramètre.

Puis la méthode **checkpw** de **Bcrypt** est appelée. Cette méthode prend 2 paramètres : Un string classique et un String qui est le résultat d'un hachage effectué précédemment.

Si cette méthode renvoie **true**, cela veut dire que le mot de passe entré par l'utilisateur et le mot de passe haché récupéré depuis la base de données correspondant au login entré par l'utilisateur est correct. La méthode renvoie **true** au **contrôleur** et affiche l'application via l'instanciation d'un nouvel objet ConsoleGUI.

7. Implémentation API Twilio

L'API Twilio nécessite de créer un compte sur <https://www.twilio.com/>



The screenshot shows the Twilio dashboard interface. On the left is a sidebar menu with options: Dashboard, Billing, Usage, Notification Preferences, Settings, Upgrade, Trust Hub (Beta), and Account Insights (Beta). The main content area has a header with a link to 'Learn the basics of Twilio' and 'How does Twilio Work?'. Below this is a section titled 'My first Twilio project' with a sub-header 'Project Info'. Under 'Project Info', there are three fields: 'ACCOUNT SID' with the value 'AC51fe1921c1b19d5f9e97e22850eaa975', 'AUTH TOKEN' with a 'Show' link, and 'PHONE NUMBER' with a 'Get a trial phone number' button and a link 'Why do I need a Twilio number?'. Red boxes highlight the ACCOUNT SID, AUTH TOKEN, and PHONE NUMBER sections.

Sur le menu de Twilio, vous avez votre Account SID et l'AUTH TOKEN. Ses deux informations seront utiles pour pouvoir envoyer des SMS, nous y reviendrons plus tard.

Maintenant il faut obtenir un trial phone number, cliquer sur "Get a trial phone number" puis sur "Choose this number"

Your first Twilio Phone Number

(336) 585-7189

Don't like this one? [Search for a different number](#)

 This United States phone number has the following capabilities:

 **Voice:** This number can receive incoming calls and make outgoing calls.

 **SMS:** This number can send and receive text messages to and from mobile numbers.

 **MMS:** This number can send and receive multi media messages to and from mobile numbers.

Cancel

Choose this Number

Vous avez maintenant un numéro attribué. C'est depuis ce numéro que seront envoyés les SMS.

Congratulations!

Your new Phone Number is **+13365857189**

For help building your Twilio application, check out the resources on the getting started page.
Once you've built your application, you can configure this phone number to send and receive calls and messages.

Done

Maintenant que nous avons notre **AUTH TOKEN**, le **SID Account** et le **numéro de telephone** de l'envoyeur, nous pouvons commencer l'implémentation de l'API dans le code.

Pour ce faire, il faut chercher sur internet le .jar de Twilio et l'implémenter dans le projet.

Ensuite note choix a été de créer un nouveau package nommé "SMS" avec une classe "TwilioSMS".

```
package sms;

import com.twilio.Twilio;
import com.twilio.rest.api.v2010.account.Message;
import com.twilio.type.PhoneNumber;

public class TwilioSMS {
    // Find your Account Sid and Auth Token at twilio.com/console
    public static final String ACCOUNT_SID =
        "AC04ed9ac9aebc5311614215380e938431";
    public static final String AUTH_TOKEN =
        "ad70d14a3e48b7711d9cbfd0a33fe825";

    public void envoiSMS(String noTel) {
        Twilio.init(ACCOUNT_SID, AUTH_TOKEN);
    }
}
```

GROUSSET Sylvain

```

        Message message = Message
            .creator(new PhoneNumber(noTel), // to
                    new PhoneNumber("+12055091076"), // from
                    "Il y a une zone qui dépasse ou est en dessous de
l'intervalle !")
            .create();
    }
}

```

La méthode “*envoiSMS(String noTel)*” est la méthode qui va envoyer le SMS au noTel passé en paramètre qui sera le numéro de téléphone de la personne gérant le stade auquel une hausse ou baisse de température a été observée.

8. JSlider, temp_min, temp_max

L'implémentation des 2 JSlider se fait comme ceci :

```

JSlider slider_mini = new JSlider();
slider_mini.setMinimum -20 ;
slider_mini.setMaximum 50 ;
slider_mini.setBounds 16, 40, 231, 25 ;
pnlBounds.add(slider_mini);
JSlider slider_maxi = new JSlider();
slider_maxi.setMinimum -20 ;
slider_maxi.setMaximum 50 ;
slider_maxi.setBounds 16, 76, 231, 25 ;
pnlBounds.add(slider_maxi);

```

A la consultation des températures pour un stade, les valeurs des températures min et max pour ce stade sont automatiquement récupérées et affichées sur les 2 JSlider. L'utilisateur peut éventuellement décider de changer ses deux valeurs :

```

JButton btnDebord = new JButton("D\u00E9bord");
btnDebord.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent arg0) {
        try {
            if(slider_maxi.getValue() < slider_mini.getValue()) {
                JOptionPane.showMessageDialog(null, "La valeur
maxi ne peut pas être inférieure à la valeur minimum !");
            }else {
                if(control.updateMinMax(choixStadeUtilisateur,
slider_maxi.getValue(), slider_mini.getValue()) == true) {
                    JOptionPane.showMessageDialog(null,
"Températures modifiées avec succès !");
                }
            }
        }
    }
});

```

```
        }  
    }  
    } catch (SQLException e) {  
        // TODO Auto-generated catch block  
        e.printStackTrace();  
    }  
    }  
});
```

Ce bout de code ci dessus est un action Listener sur le bouton “Débord”, ce bouton permet de mettre à jour les températures min et max d’un stade en bougeant les curseurs sur les sliders.

Ce bout de code vérifie d’abord si la température min n’est pas supérieur à la température max.

Et envoie ensuite dans une méthode au **contrôleur** avec en parameter les **valeurs min, max** des curseurs des JSlider avec le **nom du stade** et le **contrôleur** va envoyer tout ça dans une méthode de la classe **Donnees** pour faire la requête d’update sur la base de données.