

Projet Java/SQL Messagerie

Nous sommes l'équipe 1, composé de Quentin Rodrigues et Sébastien Roche

[Revenir en arrière](#)

Contexte du projet

L'entreprise **IsTisNot** souhaite développer **une messagerie interne**, celle-ci doit pouvoir utiliser **une base de données MySQL**. Le langage de programmation qui a été choisi est le **langage Java**. L'objectif est de mettre en œuvre un connecteur permettant de directement agir avec le serveur de la base de données **MySQL** et ce grâce à l'**API JDBC**.

Objectifs de projet

L'application Message est sensé regrouper ses trois fonctionnalités :

- **Connexion d'un utilisateur**
- **Gestion des utilisateurs (uniquement pour les utilisateurs dont le rôle est admin)**
- **Gestion des messages**

La **connexion des utilisateurs** et la **gestion des utilisateurs** a déjà été réalisé, ils ne nous restent que la **gestion des messages**.

Pour réaliser ce projet, nous devons nous appuyer sur le **travail déjà réaliser**, qui nous permet de visualiser l'**interface en mode console**, et que nous devons **étudier**.

Nous devons donc développés ces deux classes suivantes :



- **GestionMessage.java** : la classe gère les messages (réponse, suppression, et affichage)
- **JdbcMessage.java** : la classe devra être développée selon les mêmes principes que **JdbcUsers.java**

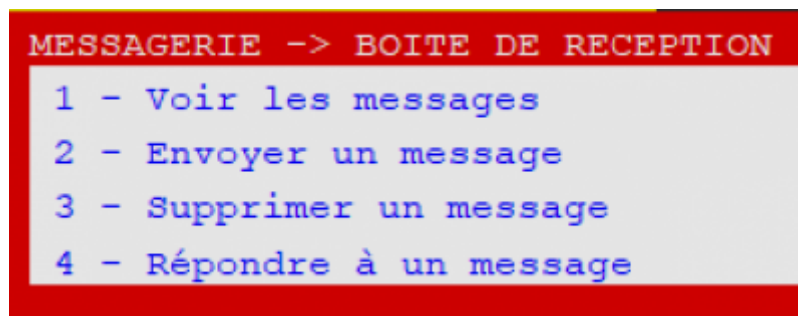
Voici les classes métiers du projet :

- **MainMessagerie.java** : contient le main() de l'application, permet de démarrer celle-ci.
- **Connexion.java** : la classe gère la connexion à l'application
 - les utilisateurs user sont dirigés automatiquement vers GestionMessage
 - les utilisateurs admin sont dirigés automatiquement vers GestionUsers
- **GestionUsers.java** : classe gérant les utilisateurs
 - Un utilisateur (user ou admin) ne voit que ses messages (envoyés ou reçus)
- **JdbcUsers.java** : la classe se connecte à **MySQL** et possède les méthodes permettant de lire, modifier, supprimer les données de la table **users**.

Voici les classes techniques du projet :

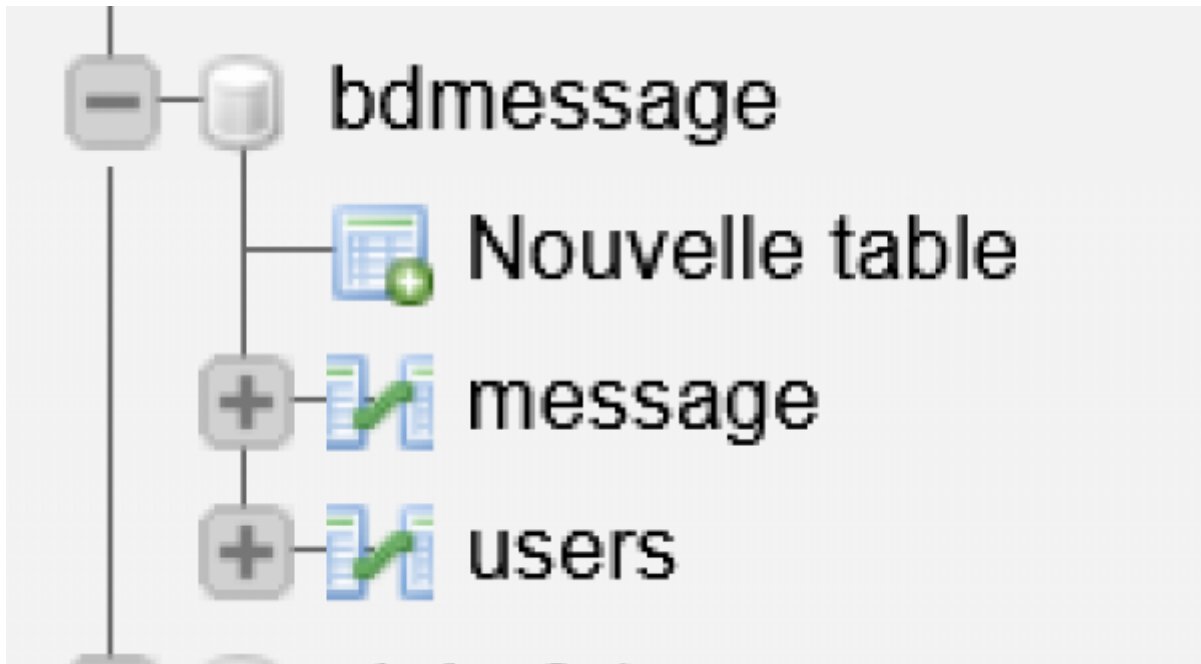
- **Color.java** : classe contenant les codes couleur pour l'affichage des informations sur la console
- **GForm.java** : classe permettant d'afficher un formulaire en mode console
- **GMenu.java** : classe permettant d'afficher et gérer un menu en mode console

Gestion des messages



Connexion & messages

Pour commencer, nous devons utiliser certaines ressources mises à disposition comme par exemple notre base de données sur PHPmyadmin :



Cette **base de donnée** est composé de **2 tables** : la table **message** et **user**. Nous devons donc effectuer la **liaison** entre le **Java** et la **base de donnée SQL** grâce à la classe **JdbcMessage()** (cette connexion à la base de donnée était donnée).

```
public JdbcMessage() {
    try {
        // create our mysql database connection
        String myDriver = "com.mysql.cj.jdbc.Driver";
        Class.forName(myDriver);

        // déclaration URL base de données... réglage du TIMESTAMP
        String myUrl = "jdbc:mysql://localhost/bdmessage"
            + "?useUnicode=true&useJDBCCompliantTimezoneShift=true"
            + "&useLegacyDatetimeCode=false&serverTimezone=UTC";

        //Connection à la base bdmessage sous MYSQL
        conn = DriverManager.getConnection(myUrl, "root", "root");

    } catch (Exception e) {
        System.err.println("Got an exception! ");
        System.err.println(e.getMessage());
    }
}
```

Possibilité d'afficher des messages

Voici ci-dessous le **case 1**, c'est à dire le **choix numéro 1** sur le **menu de messagerie**. Il

appelle toutes les **fonctions** nécessaire à l'affichage des données comme **getAllUtilisateurs()** ou **getAllMessages()**.

case 1:

```
ResultSet rs = jdbc.getAllMessages();
ResultSet rso = jdbc.getAllUtilisateurs();
String nom[] = new String[20];
int Id[] = new int[20];
int i = 0;
while (rso.next()) {
    String NomUsers = rso.getString("identifiant");
    int IdUsers = rso.getInt("id");
    nom[i] = NomUsers;
    Id[i] = IdUsers;
    i++;
}

while (rs.next()) {
    String oUsers = "";
    String dUsers = "";
    int id = rs.getInt("id");
    int origineUsers = rs.getInt("origineUsers");
    int destinataireUsers =
rs.getInt("destinataireUsers");
    String objet = rs.getString("objet");
    String message = rs.getString("message");
    Date dateEnvoi = rs.getDate("dateEnvoi");
    String etat = rs.getString("etat");

    for (i = 0; i < Id.length; i++) {
        if (origineUsers == Id[i]) {
            oUsers = nom[i];
        }
    }

    for (i = 0; i < Id.length; i++) {
        if (destinataireUsers == Id[i]) {
            dUsers = nom[i];
        }
    }

    System.out.format("%s\n%s\n%s\n%s\n%s\n%s\n\n",
        "Objet: " + objet,
        "Expéditeur : " + oUsers,
        "a " + dUsers,
        "le " + dateEnvoi,
        "Message :",
        message);
}
```

```
break;
```

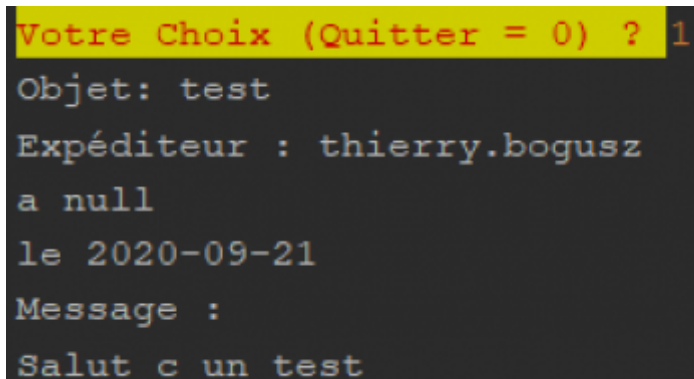
La méthode **getAllMessages()** (ci dessous) permet d'aller récupérer **tous les messages** de la base de donnée avec notre **numéro d'id** (ici l'identifiant numéro 4).

```
public ResultSet getAllMessages() {
    String query = "SELECT * FROM message WHERE `message`.`origineUsers`
= 4 or `message`.`destinataireUsers` = 4 ";
    ResultSet rs = null;
    try {
        // create the java statement
        Statement st = conn.createStatement();

        // execute the query, and get a java resultset
        rs = st.executeQuery(query);
    } catch (SQLException e) {
        System.err.print(e);
    }

    return rs;
}
```

Voici la liste des messages présent dans la base de donnée (sur cette image, il n'y en a qu'un seul).



```
Votre Choix (Quitter = 0) ? 1
Objet: test
Expéditeur : thierry.bogusz
a null
le 2020-09-21
Message :
Salut c un test
```

Possibilité d'envoyer des messages

Ici le **case 2** permet d'**envoyer un message**, il appelle donc les méthodes pour savoir quel est le **destinataire**, mais il appelle surtout à la fin, la **procédure envoieMessage()**:

```
case 2:
    reponse = GForm.show("ENVOYER UN MESSAGE", champ, null);
    int ori = 0;
```

```
        rs = jdbc.getUtilisateurWithIdentifiant(identifiant);
        while (rs.next()) {
            ori = rs.getInt("id");
        }
        String s = String.valueOf(ori);
        rs = jdbc.getUtilisateurWithIdentifiant(reponse[0]);
        int des = 0;
        while (rs.next()) {
            des = rs.getInt("id");
        }

        String p = String.valueOf(des);

        if (reponse != null) {
            String envoyer[] = {s, p, reponse[1], reponse[2],
reponse[3], "non lu"};
            jdbc.envoieMessage(envoyer);
        }
        break;
```

L'envoi des messages est géré par la procédure **envoieMessage()** ci dessous où elle envoi les messages dans la table **message** via la **requête préparée**.

```
public void envoieMessage(String[] données) {
    String sql = "INSERT into message "
        + "(origineUsers,"
        + "destinataireUsers,"
        + "objet,"
        + "message,"
        + "dateEnvoi,"
        + "etat)"
        + " VALUES(?,?,?,?,?,?,?)";

    try {
        PreparedStatement prepare;
        prepare = conn.prepareStatement(sql);
        prepare.setString(1, données[0]);
        prepare.setString(2, données[1]);
        prepare.setString(3, données[2]);
        prepare.setString(4, données[3]);
        prepare.setString(5, données[4]);
        prepare.setString(6, données[5]);

        int r = prepare.executeUpdate();
        prepare.close();
    } catch (SQLException e) {
        System.err.println(e);
    }
}
```

Possibilité de supprimer des messages

Le **case 3** reprend les **mêmes principes** que l'**affichage des messages** mais avec une option pour **supprimer les messages** supplémentaire. Il appelle donc la procédure **supprimerMessage()**.

case 3:

```
rs = jdbc.getAllMessages();
rso = jdbc.getAllUtilisateurs();
String nom3[] = new String[20];
int Id3[] = new int[20];
i = 0;
while (rso.next()) {
    String NomUsers = rso.getString("identifiant");
    int IdUsers = rso.getInt("id");
    nom3[i] = NomUsers;
    Id3[i] = IdUsers;
    i++;
}
int numero = 0;
while (rs.next()) {
    String oUsers = "";
    String dUsers = "";
    numero++;
    int id = rs.getInt("id");
    int origineUsers = rs.getInt("origineUsers");
    int destinataireUsers =
rs.getInt("destinataireUsers");
    String objet = rs.getString("objet");
    String message = rs.getString("message");
    Date dateEnvoi = rs.getDate("dateEnvoi");
    String etat = rs.getString("etat");
    // print the results

    for (i = 0; i < Id3.length; i++) {
        if (origineUsers == Id3[i]) {
            oUsers = nom3[i];
        }
    }

    for (i = 0; i < Id3.length; i++) {
        if (destinataireUsers == Id3[i]) {
            dUsers = nom3[i];
        }
    }
}
```

```
    }
    System.out.format("----- %s%s %s %s%s%s -----
- ",
                      "Objet: " + objet,
                      "Message de " + oUsers,
                      "À " + dUsers,
                      "Le " + dateEnvoi,
                      "Texte:",
                      message);
    }

    String[] champSup = {"Id"};
    String[] usersup = GForm.show("SUPPRIMER UN MESSAGE ->
Saisir le numero du message", champSup, null);

    ResultSet rss = jdbc.getMessageByID(usersup[0]);

    if (rss.next() == false) {
        GForm.message("Ce numéro existe pas dans la base de
donnée.");
        break;
    }
    String confirm[] = {"Supprimer ? (Oui / Non)"};

    reponse = GForm.show("SUPPRIMER UN MESSAGE ->
VERIFICATION", confirm, null);

    if (reponse != null && reponse[0].equals("OUI")) {
        jdbc.supprimerMessage(rss.getInt("id"));
        GForm.message("Message supprimé.");
    } else {
        GForm.message("Annulation.");
    }
    break;
}
```

Voici la procédure **supprimerMessage()** :

```
public void supprimerMessage(int id) {
    String sql = "DELETE FROM message"
        + " where id=?";

    try {
        conn.setAutoCommit(false);

        PreparedStatement prepare = conn.prepareStatement(sql);

        prepare.setInt(1, id);

        int r = prepare.executeUpdate();
        conn.commit();
    }
```



```
    } catch (SQLException e) {  
        System.err.print(e);  
    }  
}
```

Possibilité de répondre aux messages

Et voici donc **le case 4**, qui est exactement le même principe que la **suppression** mais avec la **procédure repondreMessage()** :

```
case 4:  
  
    rs = jdbc.getAllMessages();  
    rso = jdbc.getAllUtilisateurs();  
    String nom2[] = new String[20];  
    int Id2[] = new int[20];  
    i = 0;  
    while (rso.next()) {  
        String NomUsers = rso.getString("identifiant");  
        int IdUsers = rso.getInt("id");  
        nom2[i] = NomUsers;  
        Id2[i] = IdUsers;  
        i++;  
    }  
    numero = 0;  
    while (rs.next()) {  
        String oUsers = "";  
        String dUsers = "";  
        numero++;  
        int id = rs.getInt("id");  
        int origineUsers = rs.getInt("origineUsers");  
        int destinataireUsers =  
rs.getInt("destinataireUsers");  
        String objet = rs.getString("objet");  
        String message = rs.getString("message");  
        Date dateEnvoi = rs.getDate("dateEnvoi");  
        String etat = rs.getString("etat");  
        // print the results  
  
        for (i = 0; i < Id2.length; i++) {  
            if (origineUsers == Id2[i]) {  
                oUsers = nom2[i];  
            }  
        }  
    }  
}
```

```
        for (i = 0; i < Id2.length; i++) {
            if (destinataireUsers == Id2[i]) {
                dUsers = nom2[i];
            }
        }

        System.out.format("%s\n%s\n%s\n%s\n%s\n%s\n%s\n\n",
            "Numéro:" + id,
            "Objet: " + objet,
            "Message de " + oUsers,
            "À " + dUsers,
            "Le " + dateEnvoi,
            "Texte:",
            message);
    }

    reponse = GForm.show("REPONDRE A UN MESSAGE",
champReponse, null);

    ResultSet rsm = jdbc.getMessageByID(reponse[0]);

    if (rsm.next() == false) {
        GForm.message("Message inconnu...");
        break;
    }
    String origine = rsm.getString("origineUsers");
    String destinataire =
rsm.getString("destinataireUsers");
    String objet = rsm.getString("objet");
    reponse = GForm.show("REPONDRE A UN MESSAGE",
champMessage, null);

    String ReMessage[] = {destinataire, origine, objet,
reponse[0], reponse[1], "non lu"};
    if (reponse != null) {
        jdbc.repondreMessage(ReMessage);
    }
    break;
}
```

Ceci est la **procédure repondreMessage()** :

```
public void repondreMessage(String[] données) {
    String sql = "INSERT into message "
        + "(origineUsers,"
        + "destinataireUsers,"
        + "objet,"
        + "message,"
        + "dateEnvoi,"
        + "etat)"
        + " VALUES(?,?,?,?,?,?,?)";
}
```

```
try {  
    PreparedStatement prepare;  
    prepare = conn.prepareStatement(sql);  
    prepare.setString(1, données[0]);  
    prepare.setString(2, données[1]);  
    prepare.setString(3, données[2]);  
    prepare.setString(4, données[3]);  
    prepare.setString(5, données[4]);  
    prepare.setString(6, données[5]);  
  
    int r = prepare.executeUpdate();  
    prepare.close();  
} catch (SQLException e) {  
    System.err.println(e);  
}  
}
```

From:

<https://ppe.boonum.fr/> - **AP.SIO**

Permanent link:

<https://ppe.boonum.fr/doku.php?id=slam:ws:2021:sio:ap2.3:equipe1:accueil>

Last update: **2021/09/21 21:52**

