Projet MESSAGERIE

Contexte

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.

Mise en œuvre

- Travail possible en binôme
- Développement avec **NetBeans**
- SGBD **MySql**
- Cloner votre dépôt **Git** et rendre le projet en mettant à jour ce même dépôt

JDBC

JDBC est l'acronyme de Java DataBase Connectivity et désigne une l'API permettant un accès

aux bases de données avec Java.

https://www.jmdoudoux.fr/java/dej/chap-jdbc.htm#jdbc-4

Fonctionnalités de l'application

Le chef de projet a fixé les fonctionnalités suivantes à l'application **Messagerie** :

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

Travaux déjà réalisés

Une première équipe a réalisé :

- La connexion des utilisateurs
- La gestion des utilisateurs

Organisation du projet

Base de données

Pour installer votre base de données **MySQL** vous devez utiliser sur votre poste de travail *EasyPHP* ou un logiciel équivalent.

bdmessage

- Le fichier SQL de création de la base de données et des tables se trouve dans _**ressources/bdmessage.sql**

Pour le bon fonctionnement de l'application **Messagerie**, vous devez démarrer *EasyPHP*

Classes déjà developpées

Classes métiers

- **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**

Classes techniques

- **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

Travail à faire

Le chef de projet vous demande de terminer le travail, en développant la **gestion des messages**.

Pour cela **vous devez vous appuyez sur le travail déjà réalisé** qui doit vous permettre :

- de visualiser l'interface en mode Console
- de programmez la **gestion des messages** en étudiant le code déjà développé

Classes à développer

Après avoir étudier le code *des classes* ***métiers*** *déjà développées vous* ***devez coder les deux classes suivantes*** :

- **GestionMessage.java** : la classe gère les messages
- **JdbcMessage.java** : la classe devra être développée selon les mêmes principes que ***JdbcUsers.java***

JdbcMessage.java

Cette procédure permet de récupérer tous les messages de la base de données en fonction de l'utilisateur.

```
public ResultSet getAllMessage() {

        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;

    }
```

Cette procédure permet d'insérer un nouveau message dans la base de données.

```
public void insertMessage(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);
        }
```

Cette procédure permet de créer un nouveau message.

```
public void ReponseMessage(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);

        } }
```

Cette procédure supprime un message dans la base de données.

```
public void deleteMessage(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);

        }

    }
```

Cette procédure permet récupérer un message à partir de son id .

```
 public ResultSet getMessage(String id) {

        String query = "SELECT * FROM message where id='" + id + "'";

        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;

    }
```

GestionMessage.java

case 1 : Le case 1 affiche tous les messages si l'utilisateur est destinataire ou expéditeur d'un message.

```
ResultSet rs = jdbc.getAllMessage();

                    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,
                            "Message de " + oUsers,
                            "À :" + dUsers,
                            "Le " + dateEnvoi,
                            "Texte:",
                            message);
                }
```

case 2 : le case 2 permet d'envoyer un message.

```
reponse = GForm.show("AJOUTER UN MESSAGE", champ, null);
                    int ori = 0;
                    rs = jdbc.getUtilisateur2(identifiant);
                    while (rs.next()) {
                        ori = rs.getInt("id");
                    }
                    String s = String.valueOf(ori);
                    rs = jdbc.getUtilisateur2(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.insertMessage(envoyer);
                    }
                    break;
```

case 3 : le case 3 permet d'envoyer un message.

```
rs = jdbc.getAllMessage();
                    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\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);


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


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


                    if (rss.next() == false) {
                        GForm.message("Numero inconnu...");
                        break;
                    }
```

```
                    String confirm[] = {"CONFIRMEZ LA SUPPRESSION (Tapez
OUI)"};



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



                    if (reponse != null && reponse[0].equals("OUI")) {

                        jdbc.deleteMessage(rss.getInt("id"));

                        GForm.message("Le message a été supprimé");

                    } else {

                        GForm.message("AUCUNE SUPPRESSION...");

                    }

                    break;
```

case 4 : le case 4 permet de répondre à un message.

```
rs = jdbc.getAllMessage();

                    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.getMessage(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.ReponseMessage(ReMessage);

                    }

                    break;



            }

        } while (rep != 0);



    }
```