


Projet Messagerie

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.



Présentation de l'équipe et de la mission

 Je suis l'équipe n°9.
Notre mission doit répondre à 3 objectifs.


► Connexion d'un utilisateur

► Gestion des utilisateurs

► Gestion des messages

Les deux premiers objectifs ont déjà été réalisés. Ce qui nous vient à réaliser la **Gestion des messages**.

La documentation technique

 La documentation technique présentera les logiciels permettant la réalisation de notre projet.

Prérequis

► Nous allons présenter les prérequis nécessaires à la création de notre sites en différentes sous parties :

- MAMPP : Serveur Local
- NetBeans
- Module de connexion a la BDD avec Jdbc
- Notre BDD : PHP MY ADMIN
- Le Modèle-vue-contrôleur

MAMPP : Serveur Local

MAMP un ensemble de logiciels MACOS permettant de mettre en place un serveur Web local, un serveur FTP et un serveur de messagerie électronique.



NetBeans

NetBeans est un environnement de développement intégré, placé en open source par Sun en juin 2000 sous licence CDDL et GPLv2. En plus de Java, NetBeans permet la prise en charge native de divers langages tels le C, le C++, le JavaScript, le XML, le Groovy, le PHP et le HTML, ou d'autres par l'ajout de greffons



MODULE DE CONNEXION A LA BDD : Jdbc

Nous avons créé une classe publique « JdbcMessage » pour avoir accès à la base de données. Nous indiquons dans les propriétés de Jdbc les paramètres de la base de données.

Nous avons ensuite créé le module de connexion :

```
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");

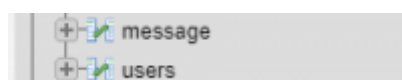
} catch (Exception e) {
    System.err.println("Got an exception! ");
    System.err.println(e.getMessage());
}
}
```

Nous utilisons l'instruction `try catch` qui exécute notre programme et définit une réponse si l'une de ces instructions provoque une exception.



Notre BDD : PHP MY ADMIN

Nous avons également besoin d'une base de données afin de répertorier les messages et les utilisateurs.



Nous avons donc répertorié 2 tables.

bdmessage message	bdmessage users
id : int(10)	id : int(10) unsigned
origineUsers : int(10)	identifiant : varchar(30)
destinataireUsers : int(10)	motDePasse : varchar(12)
objet : varchar(100)	service : varchar(50)
message : text	nom : varchar(32)
dateEnvoi : date	prenom : varchar(32)
etat : varchar(10)	date_created : timestamp
	role : varchar(5)

Notre base de données est stockée en local grâce à PhpMyAdmin.

PERMETTRE D'ENVOYER UN MESSAGE

Procédure développée : insertMessage()

```
public ResultSet insertMessage(String destinataire, String objet, String message) {
    int origineUsers = Connexion.getIdentifiant();
    ResultSet rse = null;
    int idDest = -1;
    JdbcUsers jdbc = new JdbcUsers();
    rse = jdbc.getUtilisateur(destinataire);
    LocalDate dateActuelle = LocalDate.now();

    String sql = "INSERT into message
(origineUsers,destinataireUsers,objet,message,DateEnvoi) VALUES(?,?,?,?,?)";

    try {
        if (!rse.next()) {
            GForm.message("erreur!!!");
        }

        idDest = rse.getInt("id");
        PreparedStatement prepare;
        prepare = conn.prepareStatement(sql);
        prepare.setInt(1, origineUsers);
        prepare.setInt(2, idDest);
        prepare.setString(3, objet);
        prepare.setString(4, message);
        prepare.setDate(5, java.sql.Date.valueOf(dateActuelle));
        int r = prepare.executeUpdate();
        prepare.close();
    } catch (SQLException e) {
        System.err.println(e);
    }

    return rse;
}
```

Détaillons dès à présent notre requête SQL :

```
INSERT into message (origineUsers,destinataireUsers,objet,message,DateEnvoi)
VALUES(?,?,?,?,?)
```

Nous insérons une enregistrement dans notre base de donnée, avec une requête INSERT.

Jeu d'essais :

```
Votre Choix (Quitter = 0) ? 2
MESSAGERIE -> ENVOYER UN MESSAGE
Destinataire ?      claude.pasqualini
Objet ?             Bonjour
Message ?           Comment se passe ta retraite ?
VALIDER (O/N) ? 0
```

Nous retrouvons bien notre enregistrement dans notre base de donnée :

id	origineUsers	destinataireUsers	objet	message	dateEnvoi	etat
22	4	5	Bonjour	Comment se passe ta retraite ?	2021-09-21	non lu

Dans notre case 1, nous récupérons tout les informations inscrites par l'utilisateur connecté :

```
String[] champ1 = {"Destinataire", "Objet", "Message"};
reponse = GForm.show("ENVOYER UN MESSAGE", champ1, null);
```

Pour exploiter les données et le transmettre à la fonction **insertMessage()** nous récupérons sous forme de tableau les informations entrés (exemple : reponse[0], reponse[1]).

Petit aperçu en cas d'erreur du destinataire :

```
Votre Choix (Quitter = 0) ? 2
MESSAGERIE -> ENVOYER UN MESSAGE
Destinataire ?      nicolas.jacques
Objet ?             Salut
Message ?           Tu es qui ?
VALIDER (O/N) ? 0
!!!Le destinaire n'a pas été trouvé !!!!!!
```

Retour

From: <http://ppe.boonum.fr/~AP310>
Permalink: "pdp"
http://ppe.boonum.fr/doku.php?id=siam:ws:2021:sio:ap2.3:equipe9:permettre_d_envoyer_un_messag
Last update: 2021/09/21 21:55



PERMETTRE DE SUPPRIMER UN MESSAGE

Procédure développée : deleteMessage()

```
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);

        prepare.executeUpdate();
        prepare.close();

        conn.commit();

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

Détaillons dès à présent notre requête SQL :

```
DELETE FROM message where id=?
```

Nous supprimons le message dont l'id est celui inséré par l'utilisateur.

Dans notre case 3 : nous affichons tout les messages et leur ID, puis nous demandons l'id du message à supprimer:

```
if (reponse != null && reponse[0].equals("OUI")) {
    jdbc.deleteMessage(idMessageASupprimee);
    GForm.message("Le message dont l'id est " + idMessageASupprimee + " a été supprimé");
} else {
    GForm.message("La suppression a été annulé");
}
```

Jeu d'essais :

```
Votre Choix (Quitter = 0) ? 3
ID : 21 | De : agnes.bourgeois | Objet : Bonsoir | message : Demain il y a frites au self | Date d'envoi : 2021-09-21
ID : 20 | De : claude.pasqualini | Objet : Bonjour | message : Tout ce passe ? | Date d'envoi : 2021-09-21
MESSAGERIE -> SUPPRIMER UN MESSAGE
Numéro du message ? 21
MESSAGERIE -> SUPPRIMER UN MESSAGE -> CONFIRMATION
Pour confirmer la suppression entrez OUI ? OUI
!!Le message dont l'id est 21 a été supprimé!!
```

PERMETTRE DE REPONDRE A UN MESSAGE

Autre procédure n'a été développée, outre le fait que nous avons créé une procédure qui récupère les informations du message reçu afin de lui répondre.

Procédure développée : getMessageById()

```
public ResultSet getMessageById(int idMessage) {

    String query = "SELECT * FROM message INNER JOIN users ON
message.origineUsers = users.id where message.id='" + idMessage + "'";
    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;
}
```

Comme précédemment nous affichons tout les messages et nous en profitons pour les stocker dans un ArrayList, que nous contrôlerons afin de vérifier que l'utilisateur va bien répondre à un message qui lui a été destiné.

```
messageListe.add(rsrep.getInt("message.id"));
```

Afin de contrôler cette ArrayList nous utilisons la méthode contains, pour nous recherchons l'expéditeur d'origine afin de le placer en tant de receveur de la futur réponse.

```
if (messageListe.contains(idMessageARepondre)) {
    ResultSet rsenvo = jdbc.getMessageById(idMessageARepondre);
}
```

Puis nous changeons l'objet du message en lui concaténant "Re-" : `<code java> String nouveauObjetMessage = "Re-" + objetMessage; </java>`

Jeu d'essais :

Session : thierry.bogusz

```
Votre Choix (Quitter = 0) ? 4
ID : 20 | De : claude.pasqualini | Objet : Bonjour | message : Tout ce passe ? | Date d'envoi : 2021-09-21
MESSAGERIE -> REPONDRE A UN MESSAGE
  Numéro du message ?      20
MESSAGERIE -> REPONDRE A CE MESSAGE
  Message ?                Oui tout ce passe bien et toi ?
```

Session : claude.pasqualini

ID : 23 | De : thierry.bogusz | Objet : Re-Bonjour | message : Oui tout ce passe bien et toi ? | Date d'envoi : 2021-09-21

Testons si l'utilisateur veut répondre à un message qui n'est pas le siens :

MESSAGERIE -> REPONDRE A UN MESSAGE

Numéro du message ? 6

Vous ne pouvez pas répondre à ce message !

Petit tour au niveau de la base de donnée :

<input type="checkbox"/>	 Éditer	 Copier	 Supprimer	23	4	5	Re-Bonjour	Oui tout ce passe bien et toi ?	2021-09-21	lu
<input type="checkbox"/>	 Éditer	 Copier	 Supprimer	20	5	4	Bonjour	Tout ce passe ?	2021-09-21	lu

Retour

From:
http://ppe.boonum.fr/ - AP-SIO

Permanent link:
http://ppe.boonum.fr/doku.php?id=slam:ws:2021:sio:ap2.3:equipe9:permettre_de_repondre_a_un_message

Last update: 2021/09/21 22:06

