Rapport De Projet

Plateforme de Réseaux sociaux combinés

Rapport De Projet

Plateforme de Réseaux sociaux combinés

Introduction

L'objectif de cette application Web est de centraliser l'ensemble des réseaux sociaux d'un utilisateur (première Facebook puis Twitter) sur une seule et même application : Pluss. Et pouvoir à partir de cette application Web, publier sur l'ensemble des réseaux sociaux voulus de l'utilisateur d'un seul clic de souris. La fonctionnalité secondaire est de récupérer l'ensemble des flux voulus des divers réseaux sociaux de l'utilisateurs sur la page d'accueil du site Web Pluss.

I. Choix réalisés et solutions apportées face aux problèmes

a. Architecture du projet et diagramme de classe

Nous avons décidé d'agencer notre projet Web autour de deux servlets principales. La servlet première, MainServlet, est celle liée à l'ensemble des actions possibles et permises par notre application Web, la publication, la navigation interne au site, les filtres de restriction, etc. La servlet secondaire est celle qui est en charge des actions liées à l'utilisateur en tant qu'entité, elle permet donc la connexion, l'inscription, la déconnexion, la suppression de compte, et toute autre modification ou interaction entre l'utilisateur et le site Web Pluss.

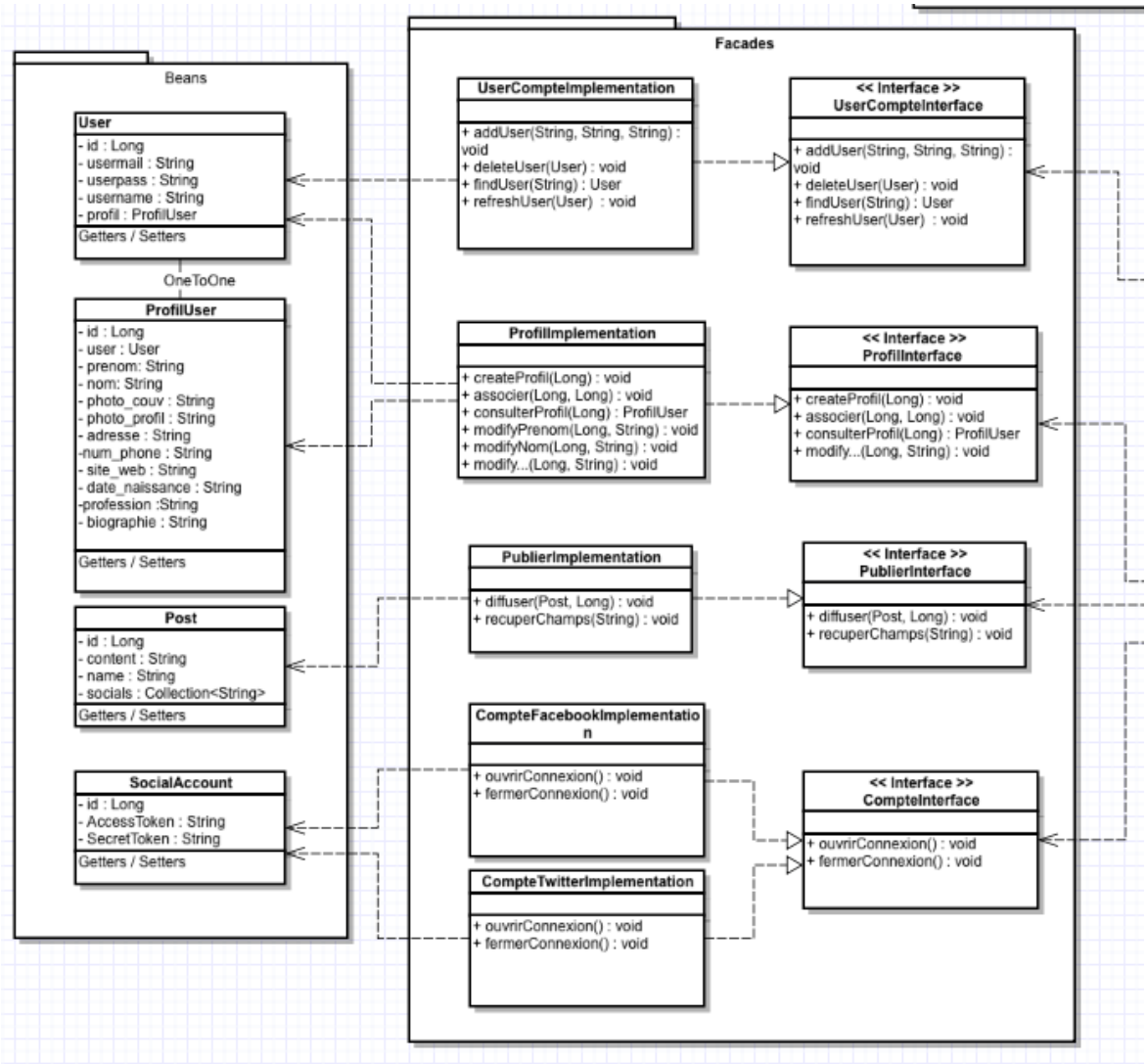

figure *Diagramme de classe revu : état actuel.*

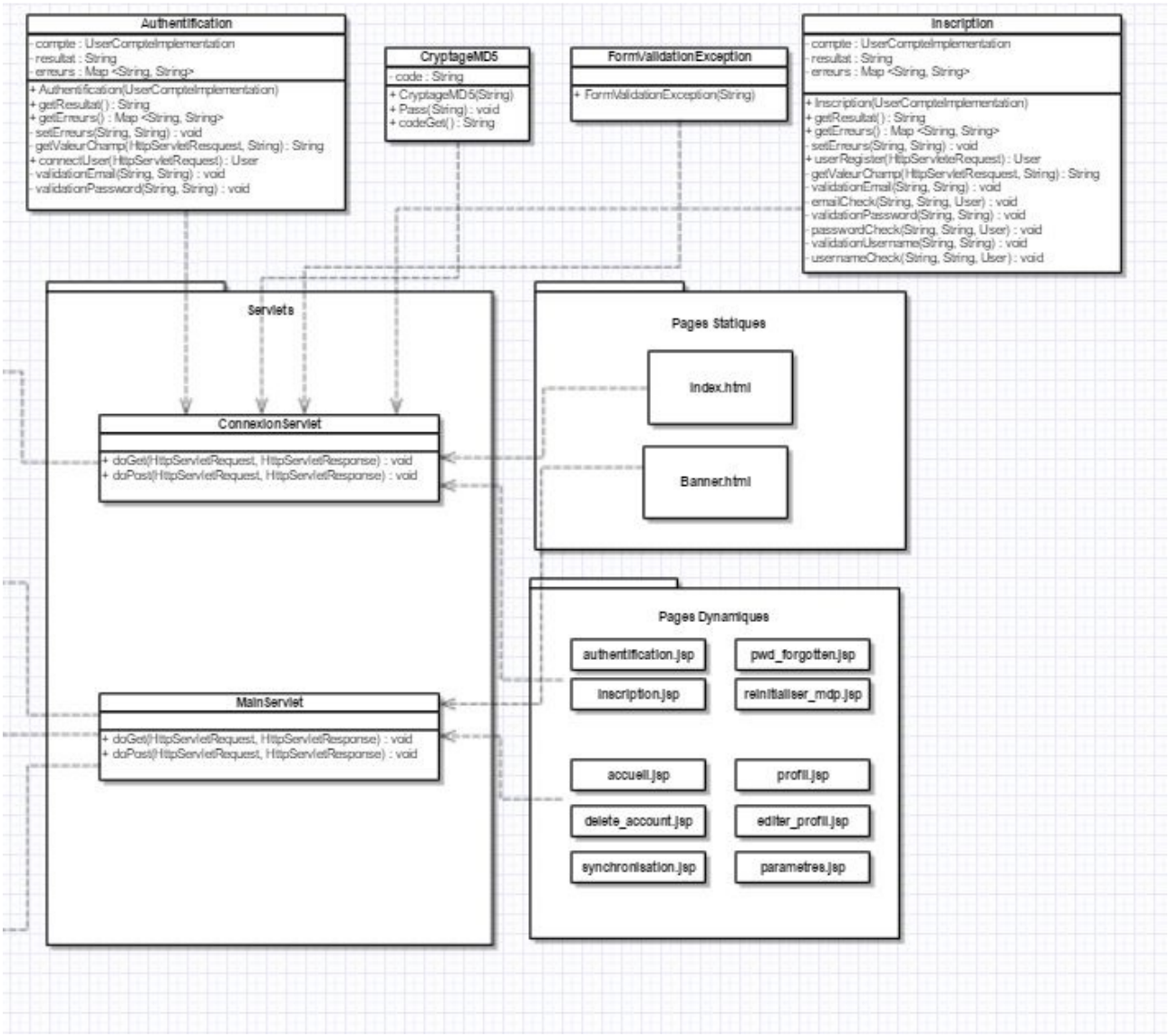

figure *Diagramme de classe revu : état actuel. (Suite)*

b. Principaux choix réalisés

Nous avons choisis par soucis d'efficacité, de diviser le travail suivant des taches. Avantages et inconvénients, il y en a qui ont réalisés une servlet entière privant donc les autres de la compréhension du fonctionnement de cette servlet mais permettant en outre d'avancer et d'aider par la suite.

Ensuite pour revenir à l'architecture, nous avons décidé par soucis toujours de légèreté et de fluidité d'utiliser des classes utilitaires. Ainsi nous n'avons que du traitement de requêtes et d'affichage de vue dans les servlets, on s'en tient donc en terme de visuel aux fonctions premières d'une servlet. Apparait donc un package form (formulaire) par exemple qui traite tout les formulaires.

```java
public class Inscription {
    public static final String CHAMP_EMAIL = "usermail";
    public static final String CHAMP_EMAIL_CONFIRM = "usermailconfirm";
    public static final String CHAMP_PASS = "userpass";
    public static final String CHAMP_PASS_CONF = "userpassconfirm";
    public static final String CHAMP_USERNAME = "username";

    private UserCompteImplementation compte;
    private String resultat;
    private Map<String, String> erreurs = new HashMap<String, String>();

    public Inscription( UserCompteImplementation compte ) {
        this.compte = compte;
    }

    public Inscription() {
    }

    public String getResultat() {
        return resultat;
    }

    public Map<String, String> getErreurs() {
        return erreurs;
    }

    /*
     * Ajoute un message correspondant au champ spécifié à la map des erreurs.
     */
    private void setErreur( String champ, String message ) {
        erreurs.put( champ, message );
    }

    /* Enregistrement d'un nouvel utilisateur */
    public User userRegister( HttpServletRequest request ) {
        String usermail = getValeurChamp( request, CHAMP_EMAIL );
        String usermailconfirm = getValeurChamp( request, CHAMP_EMAIL_CONFIRM );
        String userpass = getValeurChamp( request, CHAMP_PASS );
        String userpassconfirm = getValeurChamp( request, CHAMP_PASS_CONF );
        String username = getValeurChamp( request, CHAMP_USERNAME );

        User user = new User();

        emailCheck(usermail, usermailconfirm, user);
        passwordCheck (userpass, userpassconfirm, user);
        usernameCheck (username, user);

        if ( erreurs.isEmpty() ) {
```

figure *Exemple de classe form.Inscription*

```
60          if ( erreurs.isEmpty() ) {
61              resultat = "You're now registred on Pluss.";
62          } else {
63              resultat = "Inscription failed.";
64          }
65
66          return user;
67      }
68
69      /*
70       * Méthode utilitaire qui retourne null si un champ est vide, et son contenu
71       * sinon.
72       */
73      private static String getValeurChamp( HttpServletRequest request, String nomChamp ) {
74          String valeur = request.getParameter( nomChamp );
75          if ( valeur == null || valeur.trim().length() == 0 ) {
76              return null;
77          } else {
78              return valeur;
79          }
80      }
81
82      /**
83       * Valide l'adresse mail saisie.
84       */
85      private void validationEmail( String usermail, String usermailconfirm )
86          throws FormValidationException {
87          if ( usermail != null && usermail.trim().length() != 0
88              && usermailconfirm != null && usermailconfirm.trim().length() != 0) {
89              if (!usermail.equals(usermailconfirm)){
90                  throw new FormValidationException("The two email adresses are different");
91              } /*else if ( this.compte.findUser( usermail ) != null ) {
92                  throw new FormValidationException( "This Email adress is already used" );
93              }*/
94          } else {
95              throw new FormValidationException( "Please, give and confirm your email " +
96              "adress if you want to register" );
97          }
98      }
99
100     /**
101      * Traitement de l'adresse mail saisie et enregistrement.
102      */
103     private void emailCheck (String usermail, String usermailconfirm, User user){
104         try {
105             validationEmail( usermail, usermailconfirm );
106         } catch ( FormValidationException e ) {
107             setErreur( CHAMP_EMAIL, e.getMessage() );
108         }
109         user.setEmail( usermail );
110     }
```

figure *Exemple de classe form.Inscription (suite 1)*

```
112     /**
113      * Valide si les password saisis sont utilisables.
114      */
115     private void validationPassword( String userpass, String userpassconfirm )
116         throws Exception{
117         if (userpass != null && userpass.trim().length() != 0 && userpassconfirm != null
118             && userpassconfirm.trim().length() != 0) {
119             if (!userpass.equals(userpassconfirm)) {
120                 throw new Exception("The two given passwords are different. Try again");
121             } else if (userpass.trim().length() < 6) {
122                 throw new Exception("Your password should be at least 6 letters length");
123             }
124         } else {
125             throw new Exception("Thanks to enter and confirm your password");
126         }
127     }
128
129     /**
130      * Traitement du password saisi et enregistrement.
131      */
132     private void passwordCheck (String userpass, String userpassconfirm, User user){
133         String userpassCrypted = userpass;
134         try {
135             validationPassword( userpass, userpassconfirm );
136             CryptageMD5 crypteur = new CryptageMD5(userpass);
137             userpassCrypted = crypteur.codeGet();
138         } catch ( Exception e ) {
139             setErreur( CHAMP_PASS, e.getMessage() );
140             setErreur( CHAMP_PASS_CONF, null );
141         }
142
143         user.setPassword( userpassCrypted );
144     }
145
146     /**
147      * Valide le nom d'utilisateur saisi.
148      */
149     private void validationUsername( String username ) throws Exception {
150         if ( username != null && username.trim().length() < 5 ) {
151             throw new Exception("Your username should be at least 5 letters long");
152         }
153     }
154
155     /**
156      * Traitement de l'id saisi et enregistrement.
157      */
158     private void usernameCheck (String username, User user){
159         try {
160             validationUsername( username );
161         } catch ( Exception e ) {
162             setErreur( CHAMP_USERNAME, e.getMessage() );
```

figure *Exemple de classe form.Inscription (suite 2)*

II. Difficultés rencontrées

a. Difficultés rencontrées ou pensées pendant ou avant l'étape de codage

Initialement nous avons opté de manière intuitive pour plusieurs servlets à la place d'une servlet unique pour la connexion, l'inscription, le mot de passe perdu, etc. Il nous a été conseillé de passer sur une servlet unique, et nous avons du faire un peu plus que du copier-coller pour que ça fonctionne. Les problèmes rencontrés ont majoritairement été du a une méconnaissance des technologies utilisées. Par exemple, le sendRedirect qui se fait une fois l'accolade fermante atteinte et uniquement pour des pages statiques. Contre le getRequestDispatcher qui s'execute automatiquement, avec des pages dynamiques aussi, il nous a donc fallu comprendre la nécessité de mettre un return après l'appel de cette méthode.

```
18  @WebServlet("/Connexion")
19  public class ConnexionServlet extends HttpServlet {
20
21      @EJB
22      UserCompteImplementation compte;
23
24      private static final long serialVersionUID = 1L;
25      public static final String VUE_PREMIERE     = "/WEB-INF/authentification.jsp";
26      public static final String VUE_INSCRIPTION  = "/WEB-INF/inscription.jsp";
27      public static final String VUE_PWD          = "/WEB-INF/pwd_forgotten.jsp";
28      public static final String VUE_REINITIALISER = "/WEB-INF/reinitialiser_mdp.jsp";
29      public static final String ATT_USER         = "user";
30      public static final String ATT_FORM         = "formulaire";
31      public static final String ATT_SESSION_USER = "userSession";
32      public static final String VUE_ACCUEIL  = "/WEB-INF/accueil.jsp";
33
34
35      protected void doGet(HttpServletRequest request, HttpServletResponse response)
36          throws ServletException, IOException {
37          String operation =  request.getParameter("op");
38          if (operation != null){
39          if (operation.equals("Deconnexion")) {
40              HttpSession session = request.getSession();
41              session.invalidate();
42              request.getRequestDispatcher( VUE_PREMIERE ).forward(request, response);
43              return;
44          }else if (operation.equals("DeleteAccountOUI")) {
45              HttpSession session = request.getSession();
46              User user = (User) session.getAttribute( ATT_SESSION_USER );
47              System.out.println(user.getEmail());
48              compte.deleteUser(user);
49              session.invalidate();
50              request.getRequestDispatcher( VUE_PREMIERE ).forward(request, response);
51              return;
52          }else if (operation.equals("DeleteAccountNON")) {
53              request.getRequestDispatcher( VUE_ACCUEIL ).forward(request, response);
54              return;
55          }
56          doPost(request, response);
57          }
58      }
59
60      protected void doPost(HttpServletRequest request, HttpServletResponse response)
61          throws ServletException, IOException {
62          String operation =  request.getParameter("op");
63
64          if (operation.equals("Authentification")) {
65              Authentification formulaire = new Authentification(compte);
66              User user = formulaire.connectUser(request);
67
68              if (user != null){
```

figure *Exemple de la servlet de Connexion 1*

```
68              if (user != null){
69                  /* Récupération de la session depuis la requête */
70                  HttpSession session = request.getSession();
71                  /**
72                   * Si aucune erreur de validation n'a eu lieu, alors ajout du bean
73                   * Utilisateur à la session, sinon suppression du bean de la session.
74                   */
75                  if ( formulaire.getErreurs().isEmpty() ) {
76                      session.setAttribute( ATT_SESSION_USER, user );
77                      compte.setId(user); //set le ID de notre côté
78                      request.getRequestDispatcher(VUE_ACCUEIL).forward(request, response);
79                      return;
80                  } else {
81                      request.setAttribute( ATT_SESSION_USER, null ); //detruire session inv
82                  }
83              }
84              this.getServletContext().getRequestDispatcher( VUE_PREMIERE ).forward(request,
85
86          }else if (operation.equals("Inscription")){
87              Inscription formulaire = new Inscription(compte);
88              User user = formulaire.userRegister(request);
89              if ( formulaire.getErreurs().isEmpty() && user != null) {
90                  compte.addUser(user);
91                  request.setAttribute( ATT_FORM, formulaire );
92                  request.setAttribute( ATT_USER, user );
93                  HttpSession session = request.getSession();
94                  session.setAttribute( ATT_SESSION_USER, user );
95                  System.out.println(user.getId());
96                  request.getRequestDispatcher(VUE_ACCUEIL).forward(request, response);
97                  return;
98              } else {
99                  /*try {
100                     Thread.sleep(1000);
101                 } catch (InterruptedException e) {
102                     Thread.currentThread().interrupt();
103                 }*/
104                 request.setAttribute( ATT_FORM, formulaire );
105             }
106             this.getServletContext().getRequestDispatcher( VUE_INSCRIPTION ).forward( requ
107
108
109         }else if (operation.equals("pwdforgotten")) {
110             this.getServletContext().getRequestDispatcher( VUE_PWD ).forward( request, res
111         }else if (operation.equals("reinitialiserMdp")) {
112             ReinitialiserMdp reiMdp = new ReinitialiserMdp(compte);
113             User user = reiMdp.userPassword(request);
114             compte.refreshUser(user);
115             request.getRequestDispatcher( VUE_ACCUEIL ).forward(request, response);
116         }
117     }
118
```

figure *Exemple de la servlet de Connexion 2*

```
83              }
84              this.getServletContext().getRequestDispatcher( VUE_PREMIERE ).forward(request,
85
86          }else if (operation.equals("Inscription")){
87              Inscription formulaire = new Inscription(compte);
88              User user = formulaire.userRegister(request);
89              if ( formulaire.getErreurs().isEmpty() && user != null) {
90                  compte.addUser(user);
91                  request.setAttribute( ATT_FORM, formulaire );
92                  request.setAttribute( ATT_USER, user );
93                  HttpSession session = request.getSession();
94                  session.setAttribute( ATT_SESSION_USER, user );
95                  System.out.println(user.getId());
96                  request.getRequestDispatcher(VUE_ACCUEIL).forward(request, response);
97                  return;
98              } else {
99                  /*try {
100                     Thread.sleep(1000);
101                 } catch (InterruptedException e) {
102                     Thread.currentThread().interrupt();
103                 }*/
104                 request.setAttribute( ATT_FORM, formulaire );
105             }
106             this.getServletContext().getRequestDispatcher( VUE_INSCRIPTION ).forward( reque
107
108
109         }else if (operation.equals("pwdforgotten")) {
110             this.getServletContext().getRequestDispatcher( VUE_PWD ).forward( request, resp
111         }else if (operation.equals("reinitialiserMdp")) {
112             ReinitialiserMdp reiMdp = new ReinitialiserMdp(compte);
113             User user = reiMdp.userPassword(request);
114             compte.refreshUser(user);
115             request.getRequestDispatcher( VUE_ACCUEIL ).forward(request, response);
116         }
117     }
118
119     protected boolean CheckSessionPublic(HttpServletRequest request,
120          HttpServletResponse response, String VUE) throws IOException, ServletException {
121         HttpSession session = request.getSession();
122         boolean sortie = false;
123         if ( session.getAttribute( ATT_SESSION_USER ) != null ) {
124             /* Redirection vers la page accueil */
125             System.out.println("OUI");
126             request.getRequestDispatcher( VUE ).forward( request, response );
127             sortie = true;
128         }
129         return sortie;
130     }
131
132 }
```

figure *Exemple de la servlet de Connexion 3*

Toujours dans cette lignée d'une méconnaissance des technologies utilisées. Nous avions crée originellement un modèle DAO (avec des ouverture de connexion, des statement, etc, à la différence de JPA où tout cela est traité implicitement) lié à une BDD en MySQL. En réalité, il nous a été moins difficile de crée cette BDD MySQL que de réaliser le passage à H2. En revanche nous avons aujourd'hui un code beaucoup plus léger, lisible et manipulable. par tous et non pas seulement par celui qui a crée le modèle DAO.

```java
package com.pluss.dao;

import com.pluss.beans.User;
import java.sql.*;

import static com.pluss.dao.DAOUtilitaire.*;

public class UserDaoImpl implements UserDao {

    private DAOFactory daoFactory;

    private static final String SQL_SELECT_WITH_EMAIL =
    "SELECT id, usermail, username, userpass, register_date FROM User WHERE usermail = ?";

    private static final String SQL_INSERT =
    "INSERT INTO User (usermail, userpass, username, register_date) VALUES (?, ?, ?, NOW()

    UserDaoImpl( DAOFactory daoFactory ) {
        this.daoFactory = daoFactory;
    }

    /* Implémentation de la méthode find() définie dans l'interface UtilisateurDao */
    @Override
    public User find( String email ) throws DAOException {
        Connection connexion = null;
        PreparedStatement preparedStatement = null;
        ResultSet resultSet = null;
        User user = null;

        try {
            /* Récupération d'une connexion depuis la Factory */
            connexion = daoFactory.getConnection();
            preparedStatement = initialisationRequetePreparee(connexion,
            SQL_SELECT_WITH_EMAIL, false, email );
            resultSet = preparedStatement.executeQuery();
            /* Parcours de la ligne de données de l'éventuel ResulSet retourné */
            if ( resultSet.next() ) {
                user = map( resultSet );
            }
        } catch ( SQLException e ) {
            throw new DAOException( e );
        } finally {
            fermeturesSilencieuses( resultSet, preparedStatement, connexion );
        }

        return user;
    }

    /* Implémentation de la méthode create() définie dans l'interface UtilisateurDao */
    @Override
    public void create( User user ) throws IllegalArgumentException, DAOException {
```

figure *Exemple de l'implémentation réalisé en modèle DAO*

```java
    }

    /* Implémentation de la méthode create() définie dans l'interface UtilisateurDao */
    @Override
    public void create( User user ) throws IllegalArgumentException, DAOException {
        Connection connexion = null;
        PreparedStatement preparedStatement = null;
        ResultSet valeursAutoGenerees = null;

        try {
            /* Récupération d'une connexion depuis la Factory */
            connexion = daoFactory.getConnection();
            preparedStatement = initialisationRequetePreparee(connexion, SQL_INSERT, true,
             user.getEmail(), user.getPassword(), user.getUsername() );
            int statut = preparedStatement.executeUpdate();
            /* Analyse du statut retourné par la requête d'insertion */
            if ( statut == 0 ) {
                throw new DAOException( "Failure in creating the user, no lines were added
            }
            /* Récupération de l'id auto-généré par la requête d'insertion */
            valeursAutoGenerees = preparedStatement.getGeneratedKeys();
            if ( valeursAutoGenerees.next() ) {
                /* Puis initialisation de la propriété id du bean Utilisateur avec sa vale
                user.setId( valeursAutoGenerees.getLong( 1 ) );
            } else {
                throw new DAOException( "Failure in creating the user in base, no auto-gen
            }
        } catch ( SQLException e ) {
            throw new DAOException( e );
        } finally {
            fermeturesSilencieuses( valeursAutoGenerees, preparedStatement, connexion );
        }
    }

    /******* Mapping d'un ResultSet dans un User (bean) ********/
    /*
     * Simple méthode utilitaire permettant de faire la correspondance (le
     * mapping) entre une ligne issue de la table des utilisateurs (un
     * ResultSet) et un bean Utilisateur.
     */
    private static User map( ResultSet resultSet ) throws SQLException {
        User user = new User();
        user.setId( resultSet.getLong( "id" ) );
        user.setEmail( resultSet.getString( "usermail" ) );
        user.setPassword( resultSet.getString( "userpass" ) );
        user.setUsername( resultSet.getString( "username" ) );
        user.setRegisterDate( resultSet.getTimestamp( "register_date" ) );
        return user;
    }
}
```

figure *Exemple de l'implémentation réalisé en modèle DAO*

Le second type de problème est lié entièrement au fait que nous utilisons des API. Nous dépendons de ces API, donc des contraintes de sécurités premièrement mais aussi et surtout de toute modification de conditions d'accès au API. En cas de changement des API nous devrions certainement revoir notre code.

Enfin toujours lié à ces API nous avons eu le problème d'accès au API. Toutes n'étant pas proposées en Java, il nous a fallu utiliser différentes méthodes de plusieurs bibliothèques différentes.

Ensuite du côté mise en page statique nous avons mis du temps à nous familiariser avec les technologies. Sachant notamment que notre application se veut responsive, c'est à dire avec une mise en page s'adaptant au support de l'utilisateur : tablettes, Iphone, Android, Pc, etc. Nous avons aussi une petite partie en JavaScript.

Pour faire le lien avec la partie suivante des améliorations possibles, nombreuses étaient pensées et prévues à la création du projet. On pourrait par exemple penser à un serveur mail permettant de vérifier l'authenticité, utile à l'inscription et la perte du mot de passe par exemple. Mais en réalité, le nombre de personne effective dans le groupe étant réduit, cette fonctionnalité n'a pas pu être implémentée en temps et en heure.

b. Améliorations possibles

Les améliorations dans notre code sont multiples et variées, comme dit précédemment. On peut par exemple inclure un système de messagerie privée, permettant aux utilisateurs Pluss de communiquer entre eux. Enfin idée présente à la genèse du projet, on aurait voulu intégrer plus de réseaux sociaux ; LinkedIn, Instagram, Google+, etc

III. Batterie de tests réalisés

Un ensemble de test a partiellement été réalisé dans une classe spécifique au moyen de JUnit. Les raisons de la création de cette classe sont les suivantes :

- Vérifier l'execution des méthodes permettant la connexion, l'inscription, la suppression et la ré-initialisaiton.
- Permettre à l'ensemble des personnes codant l'application d'avoir une idée de la structure globale de l'application et de l'appel des différentes méthodes. L'ordre d'appel ou la nécessité d'appeler une méthode avant une autre (par soucis de logique) étant primordial, les tests permettent à cette personne de se référer à un modèle.

Enfin nous avons bien-sûr utilisé une méthode de décodage dure, au travers de la console. En affichant dans le terminal des éléments lors de l'exécution des méthodes. Comme par exemple pour comprendre comment utiliser le merge() ou le refresh() lors d'un changement de paramètres. Et voir ce changement effectif dans la BDD.